UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS

NUBE PRIVADA A TRAVÉS DE KOLLA-OPENSTACK

CASO: INSTITUTO DE CIENCIAS NUCLEARES

TESINA
QUE PARA OPTAR POR EL GRADO DE
ESPECIALIZACIÓN EN CÓMPUTO DE ALTO RENDIMIENTO

PRESENTA:
LETICIA ROJAS NAVA

TUTORES:
DR. LUKAS NELLEN FILLA
M. EN I. JUAN LUCIANO DÍAZ GONZÁLEZ
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

CIUDAD UNIVERSITARIA, CD. MX. JUNIO 2021

Resumen

En el ICN se cuenta con un cluster HPC con más de 5 petabytes de información y genera una cantidad de ambientes productivos y de pruebas en apoyo a la investigación del instituto. Se requiere tener una administración de dicha infraestructura muy flexible y dinámica que permita cambiar ágilmente los requerimientos técnicos desde cualquier parte del mundo.

Es por esto, que se realiza la implementación de la distribución Kolla de Openstack para administrar una nube privada utilizada orquestar un cluster HPC, a fin de proporcionar un evaluación técnica en dos ambientes de laboratorio para brindar las herramientas para administrar esta solución, adicionalmente generar el conocimiento del uso de esta plataforma en el Instituto.

Agradecimientos:

Mi más sincero agradecimiento a los profesores que sin su apoyo, motivación, corrección y guía no hubiera sido posible presentar esta tesina. En primer lugar al Dr. Lukas Nellen por su capacidad de sembrar en el alumno la curiosidad en temas de innovación tecnológica. También de manera muy especial al Mtro. Luciano Díaz que con firme y constante apoyo fue pilar en la creación de este laboratorio durante la pandemia. A mi maestra Yolanda quién nos dio nuestros primeros pasos en el mundo del supercómputo.

Quiero agradecer a mi compañero de desveladas y colega Ing. Roberto Angeles, así como a mis compañeros de la especialidad.

De igual manera a la Universidad Nacional Autónoma de México, en especial al IIMAS por tener los mejores catedráticos dentro del programa de Posgrado, que generan proyectos como lo es: **UNAM PAPIIT IN110621,** el cual apoya a los alumnos como yo.

Finalmente a mi esposo quién me brindó la oportunidad de seguir superándome día con día.

A los Ing. David Islas, Ing. Noé Cruz Marín quienes figuran como mis maestros de vida.

ÍNDICE

Capítulo 1

Introducción

Actualmente, la demanda de una infraestructura de alto rendimiento dependerá de las necesidades que defina el tipo de dato que genere la investigación, por tanto, puede ser tan vasto y diverso como el campo de conocimiento que abarca todo el Instituto de Ciencias Nucleares (ICN)enry considerando que tiene el más grande clúster de almacenamiento en Latinoamérica el reto es aún mayor.

Ante esta necesidad se requiere tener una infraestructura flexible, ágil, disponible y muy eficiente para cubrir la demanda de recursos de cómputo, ya sea para construir modelos de prueba de forma muy rápida, o bien, para mantener una producción con la adecuada gestión de proyectos.

El tener esa agilidad implica tiempo, personal altamente especializado y reducción de costos sin perder la capacidad de procesamiento y almacenamiento que se requiere en el cómputo científico. En este punto, convergen algunas características de la plataforma de Nube (Cloud) y HPC (High Performance Computing). Es decir, la orquestación, automatización y virtualización son mecanismos que permiten crear portales de auto aprovisionamiento para los investigadores (nacionales o internacionales) sin dejar de administrar los recursos de cómputo disponibles en el ICN.

Por otro lado, el crecimiento de los sistemas distribuidos, el desarrollo de algoritmos paralelos, el cómputo con GPU, la nube, la ciencia de datos, son campos de investigación que demandan tener un crecimiento horizontal y vertical en la infraestructura de alta disponibilidad.

1.1 Problema

El Instituto ha logrado generar diferentes soluciones desarrolladas localmente con la finalidad de agilizar y automatizar algunas tareas para brindar recursos de cómputo científico a los investigadores, sin embargo, esto ha provocado que no se cuente con una plataforma homologada y heterogénea para el multiprocesamiento, teniendo silos de trabajo.

Hoy se tiene automatizada la creación de máquinas virtuales en diversos servidores de diferentes modelos y fabricantes, esto permite avanzar con algunas tareas administrativas, sin embargo, para la migración de hardware o versión de sistema operativo, se pierde la agilidad y flexibilidad ganada.

1.2 Propuesta de Solución

Ante esta situación, es necesario revisar y evaluar las ventajas y desventajas que ofrece la nube ante la convergencia con el cómputo científico (HPC), para poder tener una infraestructura que permita la integración y uso colectivo de los recursos entre diferentes universidades o red colaborativa con las que trabaja el Instituto de Ciencias Nucleares (ICN).

Este trabajo tiene el propósito de evaluar una de las distribuciones del sistema operativo de nube Openstack: KOLLA. Dicho proyecto proporciona herramientas de desarrollo y genera contenedores para la producción de varias imágenes de sistemas operativos, ya sea en un ambiente virtualizado y/o en un hardware dedicado (*baremetal*).

Capítulo 2

Herramientas de gestión para la nube privada

2.1 Nube Privada

Una nube privada es aquel entorno que ofrece una plataforma para aplicaciones e infraestructura sobre dos o más componentes de hardware que permite tener una flexibilidad, escalabilidad y portabilidad para los servicios de aplicaciones e información dentro del centro de datos del cliente[11].

2.2 OpenStack

Openstack es un sistema operativo en la nube que controla grandes grupos de servidores, almacenamiento y recursos de red dentro de un centro de datos para administrar, aprovisionar diferentes APIs con un solo mecanismo de autenticación.

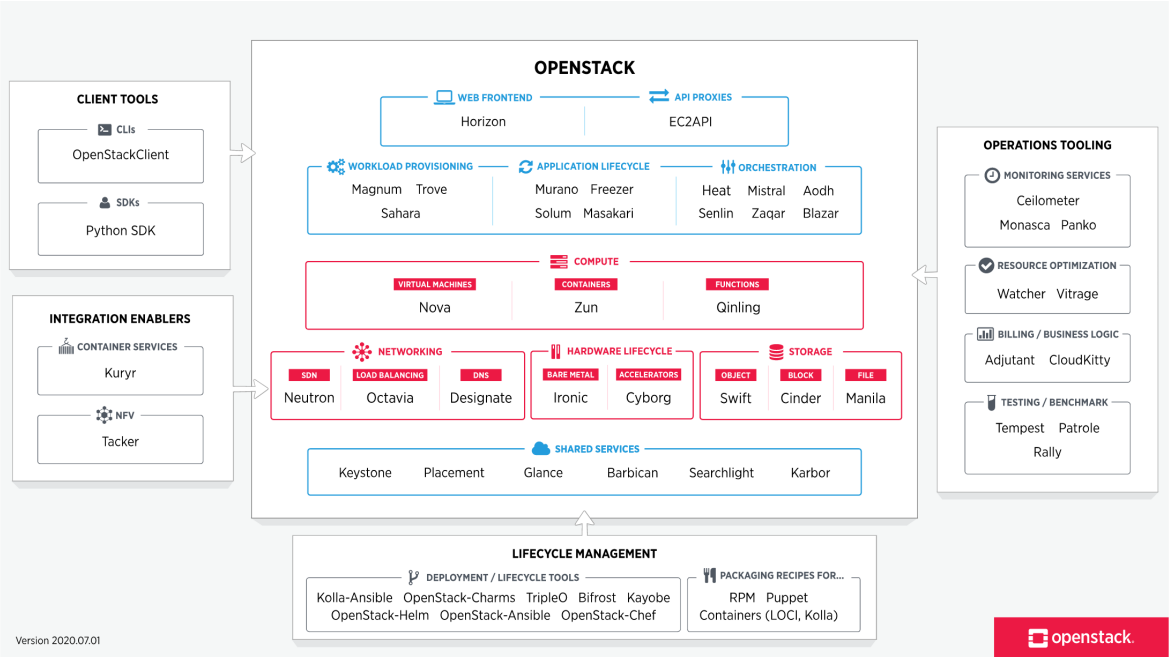


Figura 1: Componentes de OpenStack[5]

2.3 Ironic

Ironic es un proyecto de Openstack que permite realizar aprovisionamiento de servidores dedicados (conocidos como Baremetal) única y exclusivamente para un cliente y un solo propósito como parte de un componente de Nube. Administra el hardware a través de protocolos de administración remota como IPXE,REDFISH,PXE, IPMI a través de dos modos: manual y automático.

A continuación, se muestra el diagrama de estados que utiliza este proyecto para su etapa de aprovisionamiento del hardware.

Figura 2: Estados de aprovisionamiento del hardware a través de Ironic[14]

Ironic se compone de los siguientes módulos marcados en negro, adicionalmente se puede integrar con otros módulos de Openstack (marcados en gris); a través de herramienta como Ansible y Bifrost para su configuración, automatización y orquestación (marcados en café):

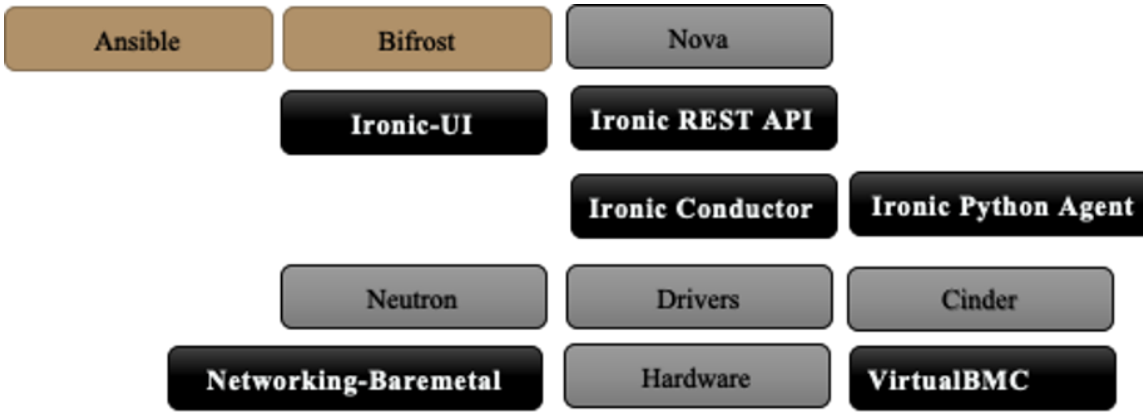


Figura 3: Componentes de Ironic e integración con otros módulos de OpenStack

A través de Ironic Conductor se administran los siguientes propiedades del hardware:

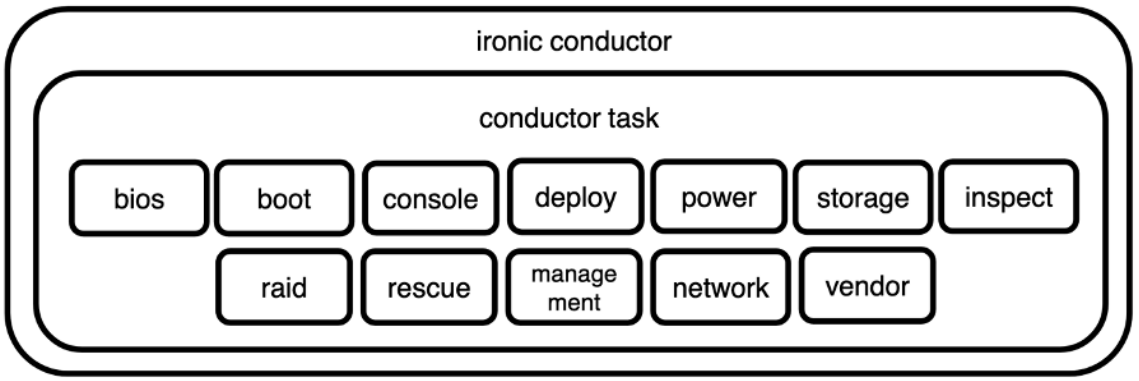


Figura 4: Componentes de Ironic Conductor

2.4 Ansible

Es una herramienta para la automatización de las tecnologías de la información que permite automatizar el aprovisionamiento de nube, administración de la configuración de sus módulos, orquestación de los servicios. Está desarrollado en el lenguaje YAML; el cual no requiere la instalación de un agente adicional en la infraestructura a administrar, ya que permite utilizar el protocolo de SSH para su comunicación y administración con los privilegios asignados al usuario de dicha conexión[1].

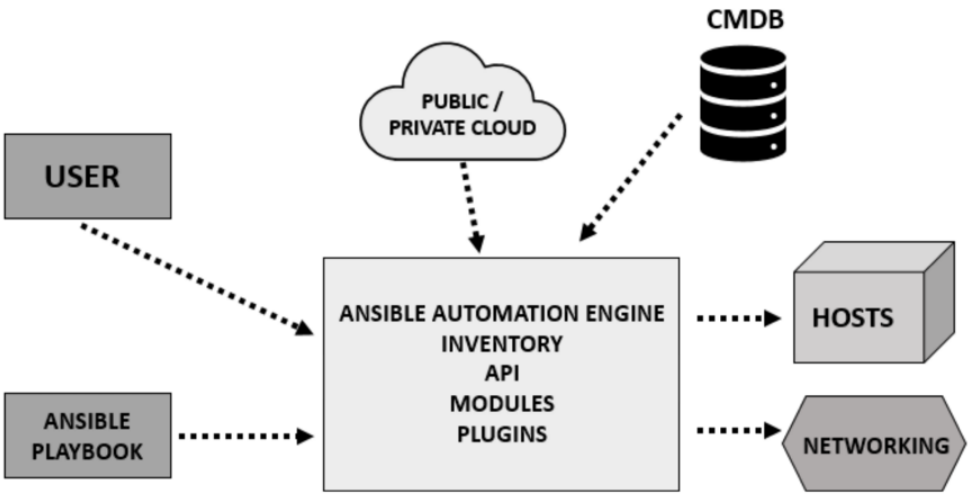


Figura 5: Arquitectura lógica de Ansible[5]

2.5 Bifrost

Esta herramienta permite automatizar tareas de aprovisionamiento para una imagen de sistema operativo ya sea dentro de un servidor *baremeta*l usando el módulo de Ironic, o bien un ambiente virtualizado.

Kolla utiliza bifrost (a través de un "*container"*) como un mecanismo para provisionar el *"boot"* a través de un panel de control de Openstack a un conjunto de servidores *baremetal*; mientras que kolla-ansible ofrece el conjunto de *"playbooks*" para configurar y crear este servicio en un *"container"*, así como la construcción de la imagen del sistema operativo y el aprovisionamiento de un servidor *baremetal*.

2.6 Virtual BMC

Es otra herramienta que ofrece openstack para poder administrar de forma virtual la conexión vía IPMI máquinas virtuales, para simular un servidor *baremetal,* se utiliza únicamente en ambientes de pruebas.[17]

Capítulo 3

Implementación

Con la distribución de Kolla, existen dos formas de realizar la instalación: *“all-in-one”* y *“multinode”*, esta primera modalidad es recomendada para ambientes de pruebas y la segunda es recomendada para ambientes productivos. La diferencia radica en la separación de los servicios dentro de los servidores (virtuales o dedicados).

Por ejemplo, el módulo encargado de la administración del disco, red o autenticación son separados e inclusive se puede diseñar una arquitectura con alta redundancia, esto implica realizar configuraciones con mayor detalle en los servicios de “cola de mensajes” (rabbitmq) o bases de datos (mariadb).El nivel de complejidad de la arquitectura lógica y física dependerá de las necesidades a cubrir para la creación de la nube privada.

El objetivo de la creación de este laboratorio de pruebas es generar la bitácora de la instalación en modo *“all-in one”* teniendo como base las distribuciones de linux: CentOS 8.3 y Ubuntu 18.04, las cuales se encuentran certificadas y soportadas en la distribución de kolla, a fin de generar el conocimiento técnico de los requisitos y factibilidad técnica que ofrecen ambas distribuciones.

3.2 Hardware

Los dos laboratorios se crearon a fin de implementar la solución que ofrece Kolla y kolla-ansible para la generación de una nube privada con Openstack. Para ello se utilizaron dos servidores bare metal con las siguientes características y recursos:

Servidor 1	**Laboratorio CentOS**	**ASUSTek RS920A-E6/RS8**
Recurso	Cantidad	Modelo
CPU	48	AMD Opteron(tm) Processor 6344 2.6GHz
Memoria	64 GB	DIMM DDR Kingston
Tarjetas de Red	1 con 4 puertos	Intel Corporation 82580 Gigabit Network
Disco	8 x 3TB	ATA Disk
Tarjetas SAS	2 con 4 puertos	Broadcom / LSI MegaRAID SAS 2208
Tarjetas Infiniband	1 con 2 puertos	Mellanox Technologies MT25208 InfiniHost III Ex

Servidor 2	**Laboratorio UBUNTU**	**ASUSTek** **RS920A-E6/RS8**
Recurso	Cantidad	Modelo
CPU	48	AMD Opteron(tm) Processor 6344 2.6GHz
Memoria	64 GB	DIMM DDR Kingston
Tarjetas de Red	1 con 4 puertos	Intel Corporation 82580 Gigabit Network
Disco	3 x 1TB	ATA Disk
Tarjetas SAS	2 con 4 puertos	Broadcom / LSI MegaRAID SAS 2108
Tarjetas Infiniband	1 con 2 puertos	Mellanox Technologies MT25208 InfiniHost III Ex

3.3 Arquitectura

El proyecto de kolla proporciona herramientas para el aprovisionamiento de instancias y servidores a través de contenedores para el sistema operativo de nube. Cabe mencionar que tiene subproyectos como kolla-ansible (automatización de tareas de administración a través de ansible) y Kayobe (a través ansible y bifrost se aprovisionan los servidores dedicados).

La arquitectura de Kolla define varios roles para poder administrar los recursos de la nube privada. Cabe destacar que al manejar el modo *“all-in one”* en este laboratorio dichos roles se consolidarán en el *controller*.

- **Controlador:** Encargado de administrar los proyectos,usuarios y recursos a través del portal de autoservicio para la nube privada.
- **Nodo de Red:** Es el encargado de administrar los diferentes dispositivos virtuales o físicos que proporciona la comunicación de la red por los diferentes protocolos soportados.
- **Nodo de Cómputo:** Es el elemento que ofrece recursos de procesamiento, memoria y espacio para almacenar las instancias que conforman la nube privada (puede ser físico o virtual). Se denomina instancia o servidor en el ámbito openstack.
- **Servidor dedicado (baremetal):** Se denomina como un servidor físico dedicado que se agrega se puede administrar como un hypervisor, o bien como un sistema operativo dedicado.
- **Servidor de almacenamiento:** Es el elemento que proporciona el espacio de almacenamiento, ya sea disco, volumen lógico o raid.

3.3.1 Distribución Lógica

En la figura 6 se presenta un diagrama de la arquitectura lógica de cómo está construido el laboratorio en modo *"all-in-one"* dentro de cada uno de los servidores físicos (baremetal), solo cambia el sistema operativo base utilizado en la máquina virtual donde se instala la distribución de Kolla, es decir, Ubuntu por CentOS en el controlador (kolla).

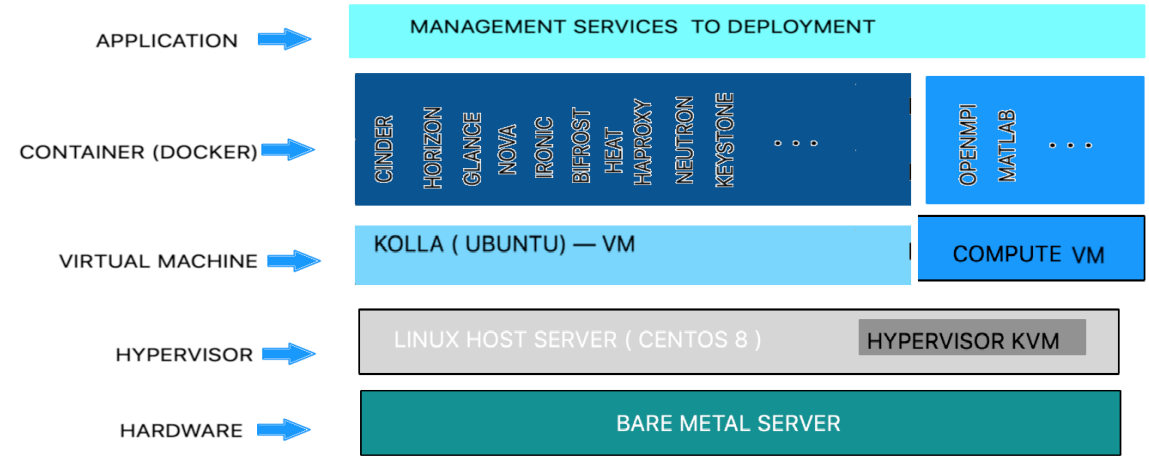


Figura 6 Arquitectura Lógica del laboratorio

3.3.2 Distribución Física

El laboratorio cuenta con dos servidores físicos, que se han estado utilizando en diferentes etapas como los muestra la siguiente tabla:

Etapa	Objetivo	Servidor - Sistema Operativo - Rol
1	Requisitos de Instalación Kolla (ver anexo A y B)	Servidor 1 - CentOS 8.3 - Controlador (kolla) Servidor 2- Ubuntu 20.04-Controlador (kolla)
2	Aprovisionamiento de Máquinas Virtuales (ver anexo XX)	Servidor 1- CentOS 8.3 -Controlador (kolla) Servidor 1- Compute Node (KVM)
3	Aprovisionamiento Baremetal (ver anexo XX)	Servidor 1- CentOS 8.3 -Controlador (kolla) Servidor 2- CentOS 8.3- *Baremetal* Node

En la figura 7, se muestran todos los roles de manera independiente que se cubrieron en las tres etapas de esta tesina. Adicionalmente, se muestra cómo se distribuye toda la comunicación tanto interna como externa en la red.

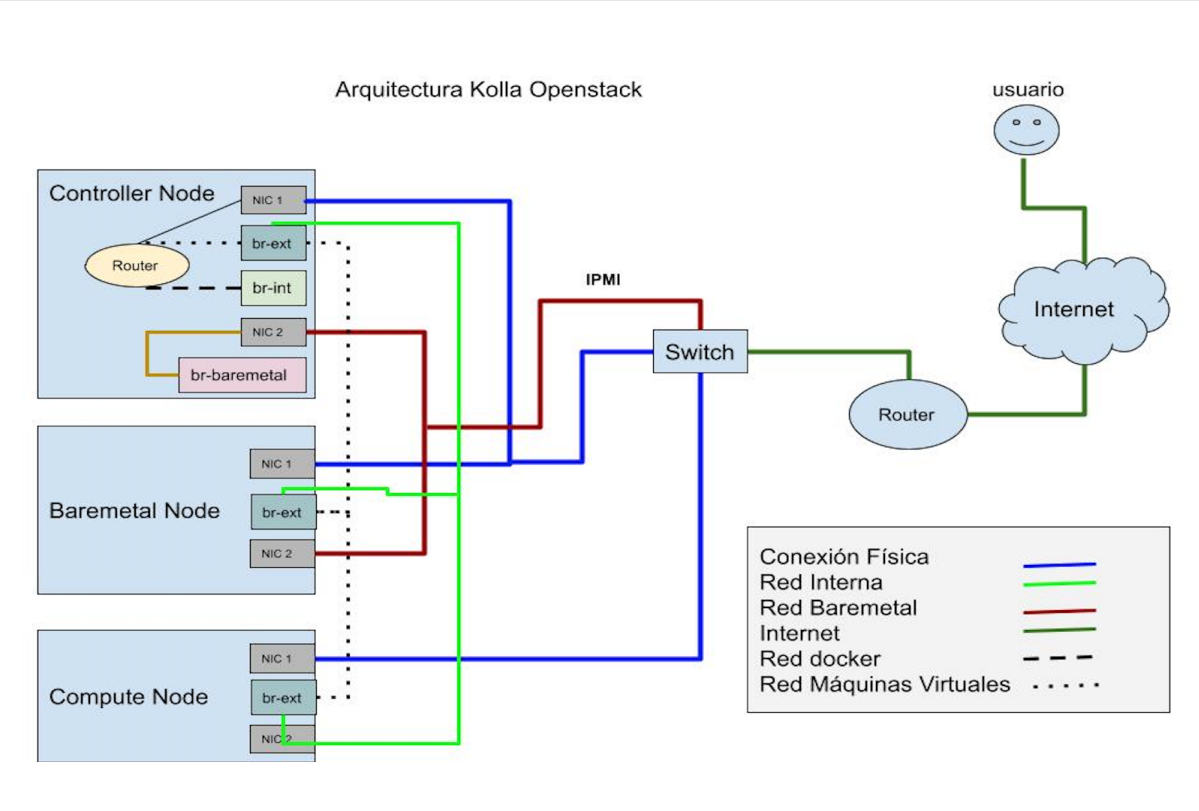


Figura 7 Arquitectura Física del Laboratorio Kolla

3.3.3 Distribución para Ironic

También es necesario presentar la distribución e interconexión de los diversos módulos de openstack que intervienen con el módulo de ironic para realizar el aprovisionamiento de servidores físicos *(baremetal)* de forma DIRECTA, que proporcionan los recursos y la comunicación para llevarlos a cabo.

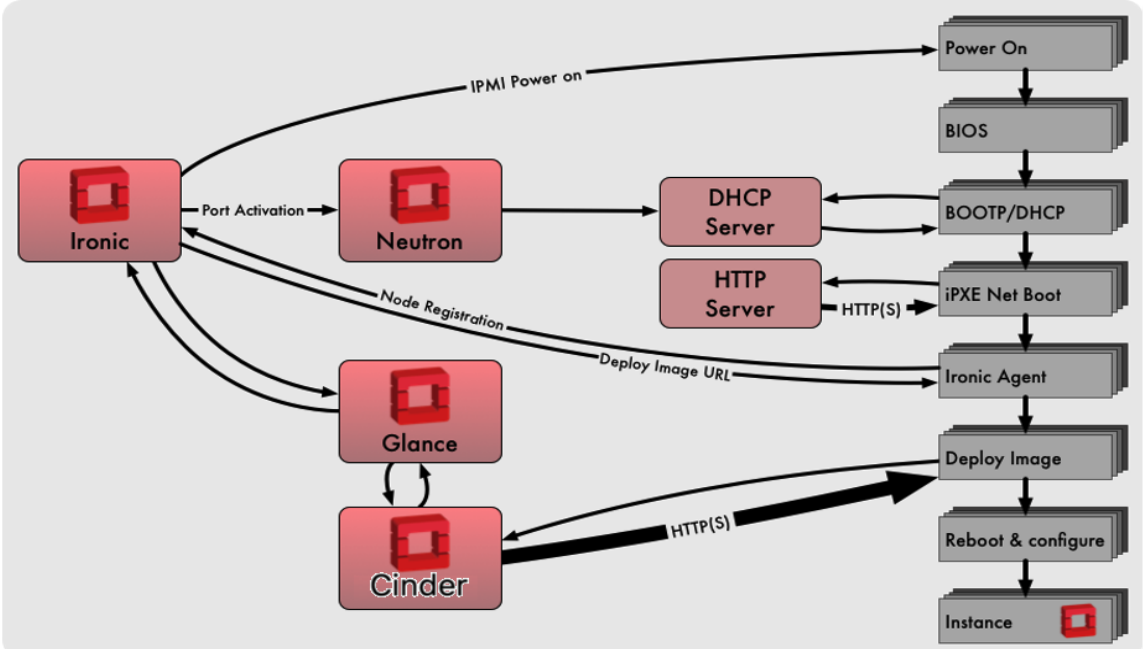


Figura 8 Arquitectura para el aprovisionamiento con módulo Ironic

3.4 Red

En la figura 9, se muestra un diagrama general de la conectividad que se realizó en el laboratorio de pruebas, a través de un switch ethernet a 1Gps sobre cable UTP con conector RJ45, esto solo muestra la capa física de comunicación.

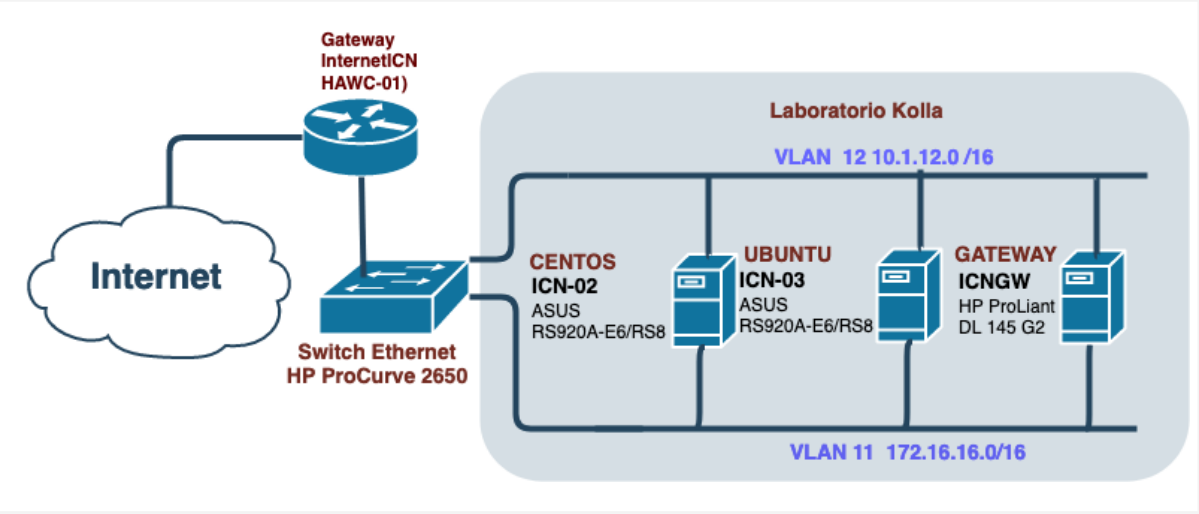


Figura 9 Diagrama de conexión física de red

Dada la limitante de puertos en los servidores y los estragos que se tuvieron durante la pandemia COVID, solo se tuvieron dos puertos físicos conectados para cubrir las necesidades de red en el laboratorio.

Se manejaron dos tipos de redes con los siguientes propósitos:

Propósito	Segmento	VLAN
Red Externa (tipo Flat)	172.16.16.0/16	11
IPMI / PXE (baremetal)	172.16.32.0/16	11
Red Publica (Gateway)	132.248.194.X	No definida
Red Administración (baremetal)	10.1.12.0/16	12
Red Interna (Tipo VxLAN)	10.1.0.0/16	No definida

A continuación se muestra la lista de direccionamiento de red asignado a cada servidor físico:

Servidor	Interfaz	Dirección IP	Netmask	Gateway
icnGW	enp3s0	132.248.A.B	255.255.255.128	132.248.A.D
icnGW	enp2s0	172.16.24.254	255.255.0.0	172.16.24.254
icnGW	enp2s0.12	10.1.12.253	255.255.0.0	10.1.12.254
icn02	enp2s0f1	10.1.12.2	255.255.0.0	172.16.24.254
icn02	IPMI	172.16.32.200	255.255.0.0	172.16.24.254
icn02	enp2s0f0	172.16.16.2	255.255.0.0	172.16.24.254
icn03	enp2s0f1	10.1.12.3	255.255.0.0	172.16.24.254
icn03	IPMI	192.168.24.3	255.255.0.0	172.16.24.254

icn03	enp2s0f0	172.16.28.21	255.255.0.0	172.16.24.254
supermicro	enp1s0	10.1.12.230	255.255.0.0	10.1.12.253
supermicro	ipmi	172.16.32.200	255.255.0.0	172.16.24.254

Ahora bien, la distribución interna de red que se realiza en la capa de virtualización se ejemplifica en el siguiente diagrama.

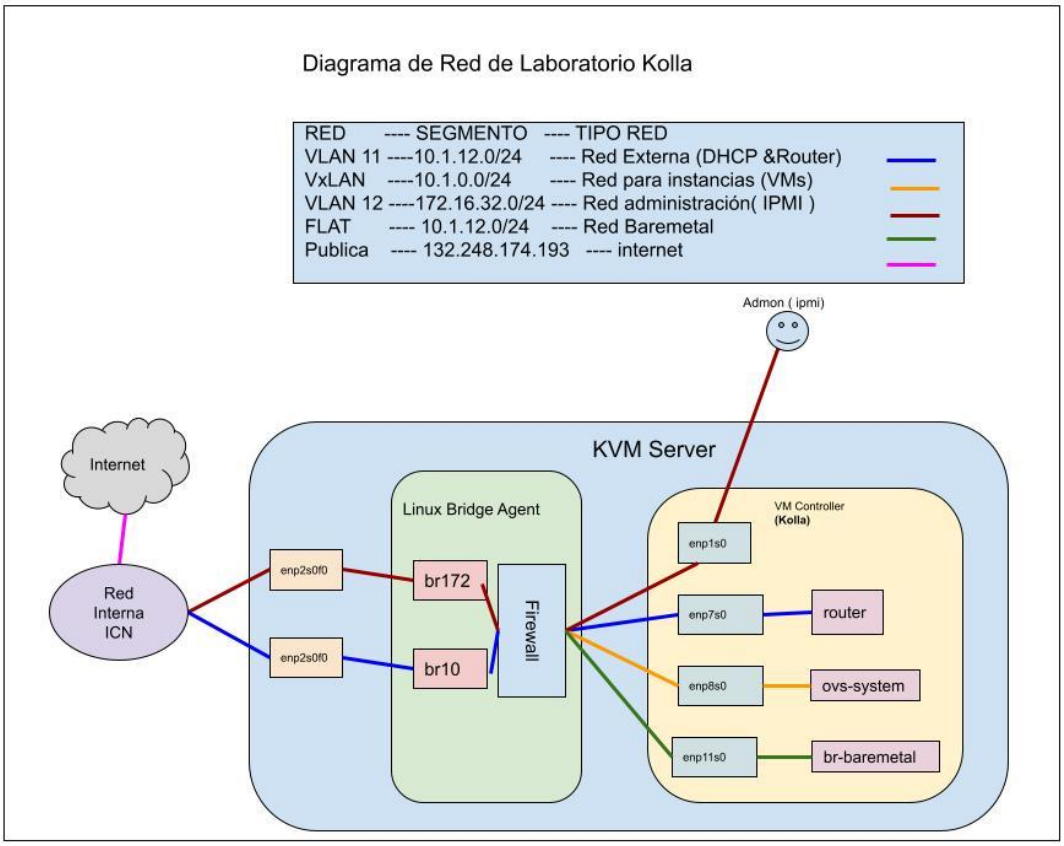


Figura 10 Diagrama distribución de interna red en el Controlador de Kolla

Una vez instalada la distribución de Kolla en el sistema operativo (para mayor detalle técnico revisar anexo A, B y C), ejecuta por única vez el script "/root/kolla-ansible/tools/init-runonce" se permite definir la red externa tipo "Flat" y la red interna tipo "VxLAN".Una vez que se termina la instalación de kolla, es importante definir los rangos y segmentos de red en el proceso de postinstalación que se puede consultar en los anexos A y B.

Es importante destacar la red interna que se usará para el aprovisionamiento de las máquinas virtuales se realizará sobre el mismo *hypervisor*, es decir, dicha red sólo se comunicará en el ámbito del *hypervisor*, por tanto, es necesario interconectar con otros hipervisores previamente a través del protocolo de red VXLAN para extender el ámbito de movilidad de las máquinas virtuales.

Posteriormente, el administrador de la nube creará la red *baremetal*. Se ha dedicado una red para la administración de los servidores *"baremetal"*, es decir para generar el aprovisionamiento del sistema operativo en nuestro caso, ha sido nombrada como *baremetal*-net.

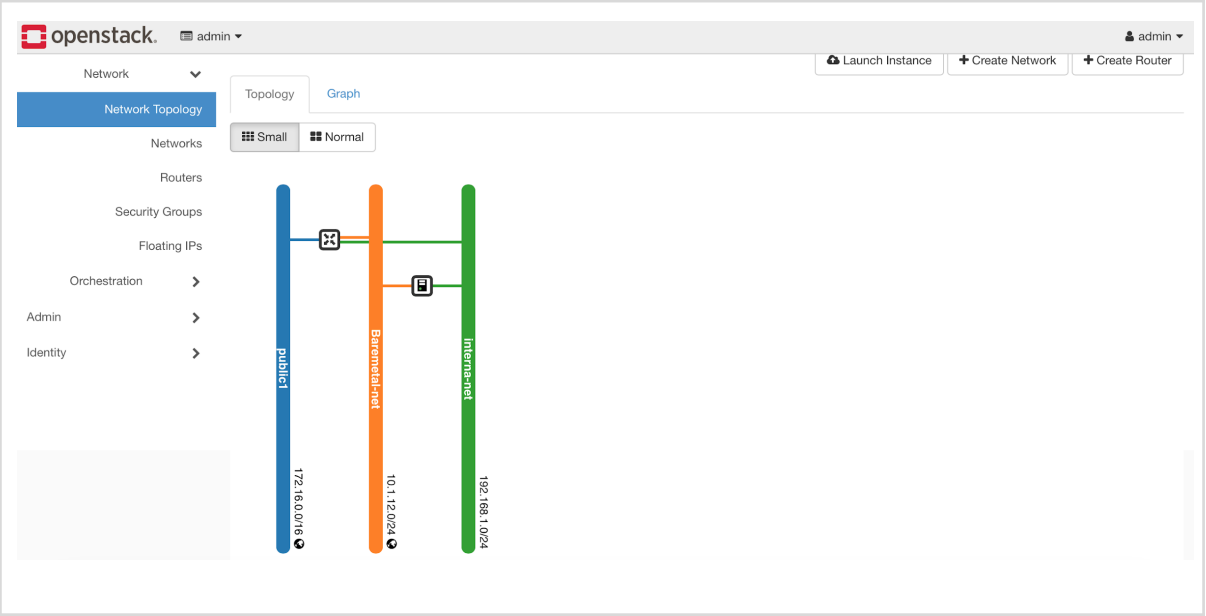


Figura 11 Red *Baremetal* en Horizon

La comunicación interna entre los contenedores se realiza a través de la red tipo *"host"* definida por docker, el cual realiza un *"tunneling"* por puerto sobre el mismo *Controller.*

El contenedor dentro de la distribución kolla que se encargará de la comunicación es neutron_server y junto con openvswitch (ovs-system) el cual tiene definido tres componentes:

- br-int (Comunicacion interna, tipo vxlan)
- br-ext (Red externa, tipo flat)
- br-tun (Red administrativa, tipo vxlan)
- br-*baremetal* (Red *Baremeta*l, tipo flat)

3.5 Configuración

Terminada la postinstalación de la distribución Kolla, es necesario comenzar a configurar diferentes recursos a través de los módulos que los administran. En la siguiente tabla se muestra la lista de servicios ofrecidos a través de contenedores que se configuraron en el archivo /etc/kolla/global.yml a fin de poder brindar un servicio básico de nube privada. Cabe destacar que existen más módulos, sin embargo, dado las limitaciones del hardware se instalaron los elementos mínimos en una arquitectura de nube.

Contenedor	Servicio en controlador	Descripción
fluentd	Bitacoras	Administra las bitácoras
kolla_toolbox		Proporciona el ambiente de entorno para los nodos a aprovisionar
cron	Cron	Servicio de cron para toda la nube
haproxy	Control para Alta disponibilidad	Proporciona el servicio de balance de cargas y un proxy para un esquema de alta disponibilidad.
keepalived	Control para Alta disponibilidad	Permite gestionar el enrutamiento de forma dinámica (protocolo VRRP)
chrony	Agente	Servicio de chrony para toda la nube
mariadb	Base de datos de control (opensource)	Contiene todos los objetos y componentes de la nube
memcached	Keystone	Aprovisiona una solución de caché para Keystone
keystone	Keystone	Módulo de autenticación, control de acceso y roles
keystone_ssh	Keystone	Gestión de llaves ssh
keystone_fernet	Keystone	Gestión de políticas de identidad para usuarios y permisos
rabbitmq	Manejador de cola de mensajes (opensource)	Encargado de la comunicación de todos los módulos
ironic_conductor	Ironic	Encargado de ejecutar el aprovisionamiento del servidor *baremetal*.
ironic_inspector	Ironic	Servicio auxiliar para descubrir las propiedades del hardware a administrar por ironic.
ironic_api	Ironic	Atiende todas las peticiones del usuario final del modulos ironic.
ironic_pxe	Ironic	Proporciona los mecanismos para realizar el boot (bios,legacy uefi) por medio de PXE.Ofrece el servicio tftpboot
ironic_ipxe	Ironic	Proporciona los mecanismos para realizar el boot(bios,legacy,uefi) por medio de IPXE. Ofrece el servicio httpboot
ironic_dnsmaq	Ironic	Servicio de dns para el servidor de ironic.
glance_api	Glance	Búsqueda y localización de imágenes para aprovisionamiento de la nube.
cinder_api	Cinder	Atiende peticiones del usuario final para el servicio de almacenamiento por bloque
cinder_scheduler	Cinder	Encargado de sincronizar los volúmenes
cinder_volume	Cinder	Administra los volúmenes de almacenamiento por bloques de la nube.
cinder_backup	Cinder	Gestiona los respaldos del almacenamiento definido para la nube
nova_scheduler	Nova	Encargado de seleccionar el nodo donde se ejecutará la instancia
nova_api	Nova	Atiende peticiones del usuario final hacia otros módulos de nova
nova_conductor	Nova	Proporciona coordinación entre las peticiones y la base de datos de Nova
nova_novnproxy	Nova	Ofrece el servicio de proxy para la comunicación middleware

nova_ssh	Nova	Ofrece la administracion y configuracion del servicio SSH entre los nodos
nova_libvirt	Nova	Encargado de administrar el *driver* de virtualización de QEMU.
nova_compute	Nova	Encargado de gestionar las instancias en el proyecto.
nova_compute_ironic	Nova	Gestiona el aprovisionamiento de nuevas servidores *baremetal* con las configuraciones realizadas en el módulo de ironic
neutron_server	Neutrón	Proporciona el servicio de la administración de la red
neutron_openswitch_agent	Neutrón	Gestiona los switches virtuales
neutron_dhcp_agent	Neutrón	Gestiona el servicio de dhcp
neutron_l3_agent	Neutrón	Gestiona los router virtuales
neutron_metadata_agent	Neutrón	Es un proxy de metadatos que se comunica a través de sockets
ironic_neutron_agent	Neutrón	Gestiona la red de comunicación para los servidores *baremetal*
heat_api	Heat	Atiende las peticiones del usuario final para orquestar la nube a través de templates.
heat_api_cfn	Heat	Permite la comunicación entre heat_engine a través de cola de mensajes RPC
heat_engine	Heat	Orquesta los recursos en la etapa de implementación
horizon	Dashboard	Interfaz gráfica para todos los módulos para ofrecer un servicio de nube.
bifrost	Bifrost	Automatización de tareas para el aprovisionamiento de servidores *baremetal* y máquinas virtuales

3.5.1 Configuración para módulo Ironic

Existen varias interfaces que proporcionan el módulo de ironic para el aprovisionamiento del sistema operativo para servidores *baremetal*: iSCSI, HTTP, Ansible, *Ramdisk* y *"Direct"*. Dado el diseño y recursos disponibles en el laboratorio, se utilizará la interfaz *"Direct"*.

Es necesario generar los siguientes prerrequisitos para el despliegue de un servicio *baremetal* a través del módulo ironic:

1. Creación de las redes para aprovisionar y limpiar el nuevo hardware una vez que se agreguen a la administración de Kolla-Openstack, a través de la interfaz, o bien, por línea de comando dentro del contenedor de neutron-server.
2. Una vez creadas las redes necesarias para administrar el hardware, se configura el módulo de Ironic.
3. Obtener el ID de cada una de las redes

```
[root@controller ~]# openstack network list
+--------------------------------------+---------------+--------------------------------------+
| ID                                   | Name          | Subnets                              |
+--------------------------------------+---------------+--------------------------------------+
| ad3f4118-3fec-44cf-928c-27544ecbf379 | interna-net   | 216b6e3e-b5ea-40e8-b34c-dfb3b7a368fd |
| b2cd80b4-c7b6-4ee7-9c9d-160699bd50a3 | Baremetal-net | 312099a1-823c-499e-8e94-dcee7e1bde35 |
| f5e2fb20-c8c4-47e4-8a09-d88324aa38d5 | public1       | 827a539b-f956-4880-a0a4-8ac3a3f8260b |
+--------------------------------------+---------------+--------------------------------------+
```

Figura 12 . Identificadores de Red a través de Openstack CLI

4. Editar en ambos archivos: /etc/kolla/ironic-api/ironic.conf y /etc/kolla/ironic-conductor/ironic.conf
    a. En la sección [neutron] , editar los parámetros cleaning_network y provisioning-network

```
[root@kolla6:/etc/kolla/ironic-api# grep b5af6776-47ff-4b6b-9783-73f095814662 ironic.conf
cleaning_network = b5af6776-47ff-4b6b-9783-73f095814662a
provisioning_network = b5af6776-47ff-4b6b-9783-73f095814662
```

b. Reiniciar los contenedores ironic-api e ironic-conductor

```
root@kolla6:/etc/kolla/ironic-api# docker stop ironic_api
ironic_api
root@kolla6:/etc/kolla/ironic-api# docker start ironic_conductor
ironic_conductor
root@kolla6:/etc/kolla/ironic-api# docker start ironic_api
ironic_api
```

c. Se necesita crear un puerto (VIF) en el cual se asignará para que el servicio de DHCP asigne una dirección IP al servidor *Baremetal* sobre la red de *Baremetal.*

```
#openstack baremetal port create $MAC_ADDRESS --node $NODE_UUID --physical-network ph-bare
```

d. Se asocia el puerto al servidor *Baremetal* para intercomunicar el módulo de Ironic conductor con el módulo de Neutrón.

Durante la instalación se generó un disco dedicado (volume group) dentro del controlador Kolla, para tener recursos de almacenamiento administrado por el módulo de cinder. A través de este módulo, se proporciona un volumen dentro del disco (Logical Volume) para realizar el aprovisionamiento de la imagen a instalar.

```
#pip install diskimage-builder
#disk-image-create ubuntu vm dhcp-all-interfaces -o ubuntu-image
```

Para cargar las imágenes *baremetal* para usuario final en la interfaz de *Horizon* es necesario asociarlas y cargarlas en la base de datos a través de openstack, el cual genera un ID. A continuación se da un ejemplo con CentOS 8.

```
=====Creación de VMLINUZ para usuario baremetal

#  CentOS8.vmlinuz --public  --disk-format aki --container-format aki --file CentOS8-image.vmlinuz
>ID=a8f1f44e-6f8b-4837-890e-ac0b9dd02180

=====Creación de KERNEL para usuario baremetal

#openstack image create CentOS8.initrd --public  --disk-format ari --container-format ari --file CentOS8-image.initrd
>ID=254a863-a8f7-499e-aa49-391a5fe0ec18

=====Creación de IMAGEN de instancia para usuario baremetal

#openstack image create CentOS8.qcow2 --public --disk-format qcow2 --container-format bare --file
CentOS-8-ec2-8.2.2004-20200611.2.x86_64.qcow2 --property kernel_id=a8f1f44e-6f8b-4837-890e-ac0b9dd02180
--property ramdisk_id=6254a863-a8f7-499e-aa49-391a5fe0ec18
>ID=6e175c84-5958-458a-a08f-a552c1f703f2

===== Comando openstack para la validación de IMAGEN en módulo glance
#openstack image list
```

Se requiere crear un “flavor” que describa los recursos con los que se cuenta en el servidor *baremetal.*

```
#openstack flavor create --public baremetal --id auto  --ram 256 --disk 0 --vcpus 1 --rxtx-factor 1
```

3.5.2 Creación de Imágenes

Para la creación de imágenes de algún sistema operativo existen varios formatos para poder realizar un aprovisionamiento de máquinas virtuales o de servidores *baremetal*. En dicho laboratorio se usaron imágenes para las siguientes distribuciones: CentOS 8, Fedora y Ubuntu.

Hay que distinguir diferentes formatos: qcow2, aki y ari, ya que se asocia un kernel y un initrd para poder realizar el boot de la instancia o de un servidor. Las diferentes distribuciones linux ofrecen la descarga de estas imágenes virtuales para desplegarse con el estándar openstack para una variedad de hipervisores y arquitecturas de hardware (x86_64, ARM, etc).

Se utilizará la herramienta libguestfs, la cual permitirá acceder y modificar estas imágenes virtuales (ver anexo G para su instalación).

3.5.2.1 Máquinas virtuales.

En el siguiente recuadro se muestra la modificación de una imagen cloud de CentOS 8, es decir, el cambio de la contraseña de root y el usuario con perfil administrador.

```
==== Descargando la imagen cloud
#wget
https://cloud.CentOS.org/CentOS/8/x86_64/images/CentOS-8-GenericCloud-8.2.2004-20200611.2.x86_64.qcow2

#cp CentOS-8-GenericCloud-8.2.2004-20200611.2.x86_64.qcow2 /tmp/CentOS-8.3.2011-20201204.qcow2
#export LIBGUESTFS_BACKEND=direct
===== Generando hash para la contraseña de root
#openssl passwd -1 admin1234
$1$U9/e3Bcu$zekMP6DJ3r/3EO9kwtVWz1
===== Modificando Imagen con herramienta
#guestfish --rw -a CentOS-8-GenericCloud-8.3.2011-20201204.2.x86_64.qcow2
>RUN
>list-filesystems
>mount /dev/sda1 /
>vi /etc/cloud/cloud.cfg
disable_root:0
ssh_pwauth: 1
lockpasswd: false
plain_text_passwd: "123456"  //contraseña usuario default CentOS
>vi /etc/shadow
root:$1$yf5HxL17$WIjQ1THmj3TuCEWc6K7yd0:18374:0:99999:7:::
// agregar el hash de openssl para root
>quit
=========
#virt-customize -a ubuntu-image.qcow2 --root-password password:Admin1234
```

3.5.2.2 Servidores *baremetal.*

El cambio para las imágenes baremetal, es importante, revisar las especificaciones del fabricante del hardware. Para esto, se requieren dos tipos de imágenes: para el aprovisionamiento del servidor y para la imagen del usuario.

La imagen del usuario (instancia) se creará con la herramienta disk-images-builder, el cual genera tres archivos (my-image.qcow2, my-image.vmlinuz y my-image.initrd) de la siguiente manera:

```
#disk-image-create ubuntu baremetal dhcp-all-interfaces grub2 -o ubuntu-image
#export DIB_LOCAL_IMAGE=CentOS-8.3.2011-20201204.qcow2
#disk-image-create CentOS7 baremetal dhcp-all-interfaces grub2 -o CentOS8-image
#disk-image-create fedora baremetal dhcp-all-interfaces grub2 -o fedora-image
```

Después de generar las imágenes del usuario, es necesario registrarlas en la base de datos del módulo de glance, a través del API de *openstack* para que esté disponible en el dashboard de Horizon.

```
#openstack image create --container-format ari --disk-format ari --public \
```

```
 --file /etc/kolla/config/ironic/ironic-agent.initramfs bm-deploy-ramdisk
 #openstack image create --container-format aki --disk-format aki --public  --file ./ironic-python-agent.kernel
bm-deploy-kernel
```

Para la creación de la imagen de aprovisionamiento DIRECTO a través de PXE Boot del servidor *baremetal* es necesario descargar los archivos CoreOS, los cuales proveen las distribuciones de kernel y el ramdisk (coreos_production_pxe.vmlinuz y coreos_production_pxe_image-oem.cpio.gz), es decir, bm-deploy-kernel y bm-deploy-ramdisk.

A continuación se describe cómo registrar las imágenes de aprovisionamiento para sistema operativo en el módulo de glance.

```
#openstack image create public --disk-format aki --container aki deploy-vmlinuz < coreos_production_pxe.vmlinuz
#openstack image create  --public --disk-format ari --container-format ari deploy-initrd<
coreos_production_pxe_image-oem.cpio.gz
```

3.5.3 Aprovisionamiento con Ironic

Para agregar un nuevo servidor *baremeta*l, es necesario agregar el hardware nuevo a través de la configuración IPMI, configuración de los drivers necesarios para su boot a través de la red, así la imagen del sistema operativo a instalar.

1.- Configuración de parámetros para la red *baremetal* en el módulo de ironic_api

2.- Configuración de privilegios IPMI en el servidor *baremetal*

Se requiere tener los privilegios suficientes para el usuario administrador de driver IPMI del servidor.

```
[root@icn02 ~]# ipmitool -I lanplus -H 172.16.32.200  -U ADMIN -P ADMIN user list
ID  Name             Callin  Link Auth  IPMI Msg   Channel Priv Limit
1                    true    false      false      Unknown (0x00)
2   ADMIN            true    true       true       ADMINISTRATOR
3                    true    false      false      Unknown (0x00)
```

En caso de no tenerlo, es necesario activarlos con el siguiente comando:

```
#ipmitool -I lanplus -H 172.16.32.200  -U ADMIN -P ADMIN channel setaccess 1 2 link=on
#ipmitool -I lanplus -H 172.16.32.200  -U ADMIN -P ADMIN channel setaccess 1 2 callin=on
```

3.- Enrollar el servidor físico en el controlador, para ello es necesario crear el nodo en la base de datos de openstack con la siguiente información:

```
#openstack baremetal node create --name supermicro --driver ipmi
```

4.- Se modifica el servidor registrado en el módulo de Ironic, de tal manera que se asocian los drivers a usar.

```
#openstack baremetal node set --name supermicro \
  --driver-info ipmi_address=172.16.32.200 \
  --driver-info ipmi_username=ADMIN \
  --driver-info ipmi_password=MKO=)IJN·$
  --driver ipmi --deploy-interface direct
  --boot-interface pxe
 --resource-class baremetal
```

5.- Posteriormente se definen algunas propiedades del servidor *baremetal*.

```
#openstack baremetal node set --name supermicro \
  --property capabilities="boot_option_local" \
  --property root_device='{"size":128}' \
  --property cpu=$CPU' \
  --property memory=$RAM_MB \
  --property local_gb=$DISK_GB \
  --property cpu_arch=$ARCH \
  --property capabilities='{"disk_label":"gpt"}'
```

6.- Se genera la firma de la imagen de usuario para el aprovisionamiento del servidor *baremetal.*

```
#md5sum image.qcow2
 >CHECK=de0a6e2085105f7fb38efd7288b76643
# openstack baremetal node set $NODE \
--instance-info image_checksum=$CHECK
```

7.- Se asocian las imágenes para el aprovisionamiento y las imágenes de usuario.

```
#openstack baremetal node set --name supermicro \
  --driver-info deploy_ramdisk=http://10.1.12.21:8089/deploy-initrd \
  --driver-info deploy_kernel=http://10.1.12.21:8089/deploy-vmlinuz \
 --instance-info ramdisk=http://10.1.12.21:8089/bare-CentOS.initrd \
--instance-info kernel=http://10.1.12.21:8089/bare-CentOS.vmlinuz \
--instance-info image_source=http://10.1.12.21:8089/bare-CentOS.qcow2 \
--instance-info capabilities='{"boot_option": "local"}' \
--instance-info capabilities='{"disk_label"="gpt"}' \
--instance-info root_gb=40 \
--driver-info image_download_source=local
```

8.- Asocia el servidor a la red en la que estará administrando la etapa de “clean”

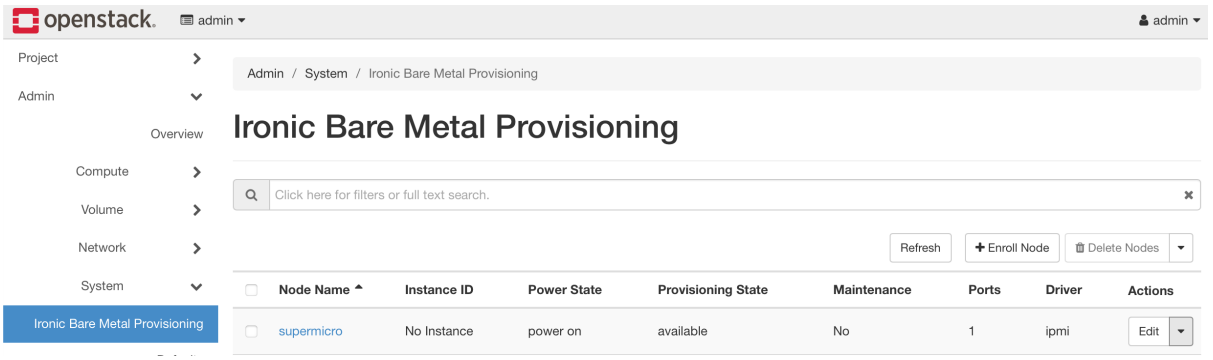
```
openstack baremetal node set  --name supermicro \
> --driver-info cleaning_network=02b54fe1-552b-4a72-abad-e32306b43399 \
>--driver-info provisioning_network=02b54fe1-552b-4a72-abad-e32306b43399
```

9.- Comunicar el servidor con la red definida previamente en el módulo neutron

```
===== Validar los puertos asociados al servidor
#openstack baremetal node vif list
===== Asociar el puerto previamente creado al servidor baremetal
 #openstack baremetal node vif attach $NODE $VIF
===== Validar el puerto asociado al nodo.
 # openstack baremetal node vif list $NODE
```

10.- Una vez registrado el nodo *baremetal*, inicia la etapa de “enroll” y lo cambia a la etapa de “manageable” (previamente se realiza un proceso de validación de parámetros), para finalmente avanzar a la etapa de “deploy” hasta tener el estado de “available”.

```
===== Validar los prerrequisitos para el nodo baremetal
#openstack baremetal node validate $NODE
===== Asociar el puerto previamente creado al servidor baremetal
 #openstack baremetal node manage $NODE
===== Validar el puerto asociado al nodo.
 # openstack baremetal node provide $NODE
===== Validar
# openstack baremetal node deploy $NODE
```



3.5.4 Registro de un nuevo Hipervisor

Es importante mencionar que para dar de alta un hipervisor adicional al nodo controlador se requiere instalar agentes de Nova-compute y Neutrón previamente a fin de poder comunicarse con el mismo.

```
#yum install CentOS-release-openstack-ussuri
# yum config-manager --set-enabled PowerTools
# yum install openstack-neutron openstack-neutron-ml2
openstack-neutron-openvswitch
#yum install python3-openstackclient
#yum -y install openstack-nova-compute
```

Se realiza la configuración de cada uno de los agentes de Nova-compute, para ello se edita el archivo nova.conf ubicado en /etc/nova. Los parámetros que se configuraron son:

```
[DEFAULT]
internal_service_availability_zone=internal
default_availability_zone=nova
compute_driver=libvirt.LibvirtDriver
my_ip=172.16.16.3
transport_url=rabbit://openstack:ITwtuCr3fuNZytPA0OUrHIvqkiFZH4M8fEg1UDoy@172.1
6.32.13:5672//
[api]
auth_strategy=keystone
[keystone_authtoken]
www_authenticate_uri = http://172.16.32.101:5000
auth_url = http://172.16.32.101:35357
auth_type = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = VmCBdUMIoOVwzfynpyMAMqVdrNK3NBzURbQo3kTD
[libvirt]
virt_type = qemu
[oslo_concurrency]
lock_path=/var/lib/nova/tmp
[placement]
auth_type = password
auth_url = http://172.16.32.101:35357
username = placement
password = MIQpkFOfqUjx2LvqfrV94plouDivlX1pWxhRPgQY
user_domain_name = Default
project_name = service
project_domain_name = Default
region_name = RegionOne
[vnc]
server_listen=0.0.0.0
server_proxyclient_address=0.0.0.0
novncproxy_base_url=http://172.16.32.13:6080/vnc_auto.html
```

Posteriormente se inicia el servicio de nova en el hipervisor y se valida la conexión con el nodo Controlador.

Dentro del hypervisor ejecutar:
#systemctl start openstack-nova-compute
Dentro del Controlador:
#docker exec -it nova_api bash
(nova-api)#nova-manage cell_v2 discover_hosts --verbose

Capítulo 4 Solución

4.1 Resultados

Se crea una nube tipo “Infraestructure as a Service” que permite a los administradores e investigadores tener una API para que puedan ingresar y administrar sus propios ambientes de pruebas con un grupo de recursos de cómputo previamente definido.

Permitió reducir el tiempo que los administradores dedican a una de las tantas actividades que tiene asignada: la agilidad del despliegue de los ambientes de pruebas centralizando la administración desde el controller.

4.2 Conclusiones

La flexibilidad que da la distribución de kolla para el aprovisionamiento de una nube privada, nos agiliza la integración y configuración de los módulos necesarios para el servicio.

Con respecto a las diferencias encontradas entre Ubuntu Bionic y CentOS 8 nos permitió evaluar la factibilidad técnica para definir qué versión poner en producción el controlador.

Además de conocer, revisar y evaluar el funcionamiento de cada uno de los módulos que contiene la distribución Kolla.

Una vez definida la arquitectura en base a la distribución y soporte de la red,la plataforma ofrece una forma sencilla y ágil de aprovisionar el servicio directamente desplegado por los investigadores en base a sus necesidades, sin tener conocimientos especializados de la plataforma, generando sus propios entornos de pruebas. Esto permite a los administradores de la infraestructura poder contabilizar y optimizar el uso de los recursos de cómputo por cada proyecto definido en dicha plataforma.

Una vez implementado la arquitectura de la nube privada, la reducción en tiempos de entrega, impactan directamente en los costos del proyecto (optimización de recursos, productividad,reducción de tiempo de atención en soporte técnico,etc).

Dado que Openstack se ha definido como un estándar, ofrece una buena portabilidad y flexibilidad que permite la integración con algún servicio de nube pública, o bien con alguna otra institución educativa a nivel mundial.

Por la limitación de recursos en el laboratorio, se consolidaron todos los módulos de Openstack en un solo servidor, sin embargo, la distribución de Kolla permite tener el modo “multinode” a fin de distribuir , o bien, integrar si ya existen dichos servicios, con tan solo instalar los agentes de Openstack, esto permitirá la adopción rápida en el ambiente de producción de servicios ya existentes (servidores de almacenamiento

distribuido) evitando tener silos, para unificar poder unificar los recursos disponibles en la nube.

4.3 Mejoras

Se requiere profundizar en el módulo de neutron para poder definir una arquitectura más completa para el ambiente de producción en el diseño y distribución de la red, es decir, incluir la red infiniband.

En este laboratorio, no se incluyeron la creación de zonas desmilitarizadas y firewall, sin embargo, para un ambiente de producción es necesario incluir esquemas de seguridad propios del sistema operativo. Tampoco se incluyó un servicio de monitoreo de la infraestructura como el módulo de Manila.

En el ambiente de producción es necesario integrar el módulo de Cinder, con otro módulo con Ceph o swift, ya que este tiene la limitante de solo manejar almacenamiento por bloque, falta agregar el almacenamiento de objetos.

Otro aspecto importante, es la alta disponibilidad del servicio del controlador que ya ofrece la distribución de kolla, En el ambiente de producción, el manejo de cluster aumenta la capacidad de recuperación ante alguna contingencia en el área de producción.

Anexo Técnico A

Preparando Laboratorio en CentOS 8

1. Instalación de KVM en CentOS 8

El comando para realizar la instalación del hipervisor KVM es:

```
#sudo yum install -y qemu-kvm qemu-img virt-manager libvirt libvirt-python libvirt-client virt-install virt-viewer
#dnf install cockpit-machines virt-viewer virt-install virt-who libguestfs-tools-c
```

Deshabilitar el servicio de Selinux

```
#sudo getenforce 0
#vim /etc/selinux/config
>SELINUX=disabled
#reboot
```

Deshabilitar el servicio de firewalld

```
#systemctl disable firewalld
#systemctl stop firewalld
```

Es necesario agregar en el /etc/hosts las direcciones IP de las máquinas virtuales para montar el laboratorio.

```
#vim /etc/hosts
#Virtuales CentOS 8
10.1.12.16          controller8
```

2. Es necesario crear el usuario stack con un perfil de administrador

```
# groupadd stack wheel
# useradd stack -m /home/stack -d -c " Usuario admin" -g stack
# passwd stack
```

3. Conexión por medio de llaves desde el controller hacia los diferentes nodos del laboratorio

```
[stack@overcloud ~]$ ssh-keygen
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/home/stack/.ssh/id_rsa):
/home/stack/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/stack/.ssh/id_rsa.
Your public key has been saved in /home/stack/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:TN3RPfY1QvAU6+F64dpUAkb4BMRMar2nWOI+0Xzyv0U stack@overcloud
The key's randomart image is:
+---[RSA 3072]----+
|          =+oo+=.. |
|          o=.oooo+o|
|          o..+o.+o =|
|          .o  o.+ . .|
|          .So . = E |
|          ..++o.o =  |
|          o..+. + . |
|          ..   .= .  |
|          ..  ..+.  |
+----[SHA256]-----+
[stack@overcloud ~]$ ssh-copy-id 172.16.32.6
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/stack/.ssh/id_rsa.pub"
The authenticity of host '172.16.32.6 (172.16.32.6)' can't be established.
ECDSA key fingerprint is SHA256:6dV3/6aH5x+TgxqSE8aQLfLH/l4mPDebPKFH3vAqpaE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
install the new keys
stack@172.16.32.6's password:
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh '172.16.32.6'"
and check to make sure that only the key(s) you wanted were added.
[stack@overcloud ~]$ exit
logout
```

4. Configuración de privilegios con visudo para el usuario stack

```
## Allows people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL
## Same thing without a password
 %wheel ALL=(ALL)     NOPASSWD: ALL
```

5. Creación de Bridge en CentOS 8

Para la configuración de la red, se requiere tener dos VLANs una con acceso a internet y la otra para una red Interna., como se ejemplifica en el siguiente diagrama:

En el diagrama se crea el bridge br172 para asociarlo al puerto de red VLAN 11, mientras que el bridge br10 se asocia al puerto de red de la VLAN 12. Idealmente se requiere otra red para la administración del servicio PXE, pero dado los recursos que se tiene se usará la red interna.

Existen varias formas de generar un bridge en CentOS 8, en esta bitácora se presentará la opción que se hace a través de línea de comandos utilizando la utilería de Network Manager que ofrece el sistema operativo para la versión CentOS 8, ya esto cambia con la versión 7.

Se requiere tener un acceso adicional , ya que se realizará un borrado de la interfaz de red y se perderá momentáneamente dicho acceso.

```
[root@icn02 ~]# sudo nmcli con show
NAME     UUID                                  TYPE        DEVICE
enp2s0f1  0ff8880e-97c8-95d5-45d4-064bf1f84ea2  ethernet  enp2s0f1
br172    ae5214d9-9f5f-9544-e039-938ddab94fc9  bridge        br172
enp2s0f0  b48971eb-621a-99c1-8ef3-319130930ee5  ethernet  enp2s0f0
[root@icn02 ~]# sudo nmcli connection add type bridge autoconnect yes con-name br10
ifname br10
Connection 'br10' (dcf50005-6c2b-40d9-90d9-22132f05bff7) successfully added.
[root@icn02 ~]# sudo nmcli conn modify br10 ipv4.addresses 10.1.12.2 ipv4.method manual
[root@icn02 ~]# sudo nmcli conn modify br10 ipv4.gateway 10.1.12.254
[root@icn02 ~]# nmcli conn show
NAME  UUID                                  TYPE    DEVICE
br10     dcf50005-6c2b-40d9-90d9-22132f05bff7  bridge        br10
enp2s0f1  0ff8880e-97c8-95d5-45d4-064bf1f84ea2  ethernet  enp2s0f1
br172    ae5214d9-9f5f-9544-e039-938ddab94fc9  bridge        br172
enp2s0f0  b48971eb-621a-99c1-8ef3-319130930ee5  ethernet  enp2s0f0
[root@icn02 ~]#  nmcli connection delete enp2s0f0  &&  nmcli connection add type
bridge-slave autoconnect yes con-name enp2s0f0 ifname enp2s0f0 master br10
[root@icn02 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp2s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br172
state UP group default qlen 1000
        link/ether 30:85:a9:06:ee:76 brd ff:ff:ff:ff:ff:ff
3: enp2s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br10
state UP group default qlen 1000
        link/ether 30:85:a9:06:ee:77 brd ff:ff:ff:ff:ff:ff
4: enp2s0f2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN
group default qlen 1000
        link/ether 30:85:a9:06:ee:78 brd ff:ff:ff:ff:ff:ff
5: enp2s0f3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN
group default qlen 1000
        link/ether 30:85:a9:06:ee:79 brd ff:ff:ff:ff:ff:ff
6: br172: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
        link/ether 30:85:a9:06:ee:76 brd ff:ff:ff:ff:ff:ff
        inet 172.16.16.2/16 brd 172.16.255.255 scope global noprefixroute br172
        valid_lft forever preferred_lft forever
        inet6 fe80::3285:a9ff:fe06:ee76/64 scope link
        valid_lft forever preferred_lft forever
7: br10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
```

```
link/ether 30:85:a9:06:ee:77 brd ff:ff:ff:ff:ff:ff
inet 10.1.12.2/24 brd 10.1.12.255 scope global noprefixroute br10
valid_lft forever preferred_lft forever
inet6 fe80::3285:a9ff:fe06:ee77/64 scope link
valid_lft forever preferred_lft forever
```

6. Instalación de máquinas virtuales con KVM

Para la creación de máquinas virtuales se hizo un script bash para agilizar la creación de dichas máquinas, a continuación se muestra el código:

```
# ./createVM.sh NombreVM
#!/bin/bash
VM=$1
VM_HDD=/VMs/HardDisk/$1.CentOS7.7.qcow2
echo "Numero de usuario: $USER - $1 - $VM_HDD"
virt-install  --name $VM \
--memory 8192 --vcpus 4 --cpu host \
--disk $VM_HDD,format=qcow2,device=disk,size=40,sparse=yes,cache=none,bus=virtio \
--network type=bridge,source=br172,model=virtio \
--network type=bridge,source=br10,model=virtio \
--os-type=linux \
--os-variant=rhel7.5 \
--graphics vnc \
--noautoconsole \
--cdrom=/home/icn02/ISOS/CentOS-8-x86_64-Everything-1908.iso
```

Es importante tener espacio y recursos suficientes para generar las máquinas virtuales, ya que el espacio mínimo que se le da es de 40GB, 8 cores y 16GB de memoria.
Se requieren al menos 3 máquinas virtuales para generar el laboratorio con los siguientes roles:

- Controller.- Será el encargado de tener los módulos de Openstack que proveen los recursos para la creación de máquinas virtuales y los servicios asociados a estas.
    - Se instala Kolla, Kolla-ansible (cinder, keystone, neutron, nova, heat, and horizon)

En el controller se requiere tener un segundo disco para la instalación de cinder:
Se agrega un disco de 40GB adicional en el hypervisor:
Se puede agregar en línea sin dar de baja la máquina virtual

```
#virsh attach-device --config Controller8 ~/NewStorage.xml
```

El contenido del archivo XML es:

```
<disk type='file' device='disk'>
<driver name='qemu' type='raw' cache='none'/>
  <source file='/VMs/HardDisk/Controller8.CentOS8.1.disk2.qcow2'/>
  <target dev='vdb'/>
</disk>
```

Se debe apagar la máquina para que quede el cambio de forma permanente:

```
#virsh edit Controller8
```

Se tiene que agregar estas líneas:

```
<disk type='file' device='disk'>
  <driver name='qemu' type='raw' cache='none'/>
  <source file='/VMs/HardDisk/Controller8.CentOS8.1.disk2.qcow2'/>
  <target dev='vdb'/>
</disk>
```

Se crea un Volumen Group dentro de la máquina virtual:

```
# fdisk -l
# fdisk /dev/vdb
# pvcreate /dev/vdb
# vgcreate cinder-volumes /dev/vdb
```

- Deployment.- Será el encargado de tener bifrost para poder desplegar el módulo de Ironic, así como los respectivos drivers para aprovisionar baremetal o hipervisores.

- Compute.- Esta máquina nos permitirá realizar pruebas para la conexión de PXE y poder aprovisionar el sistema operativo.

Anexo Técnico B

Instalación de Kolla y kolla-ansible en CentOS 8

1. Actualización del sistema Operativo

```
#dnf update -y
```

2. Instalación de conexión a consola serial de la máquina virtual hacia el hypervisor

```
#cp /usr/lib/systemd/system/serial-getty@.service
/etc/systemd/system/serial-getty@ttyS0.service
#vim /etc/systemd/system/serial-getty@ttyS0.service
#ln -s /etc/systemd/system/serial-getty@ttyS0.service
/etc/systemd/system/getty.target.wants/
#systemctl daemon-reload
#systemctl start serial-getty@ttyS0.service
#
```

3. Instalación de Repositorio EPEL

```
#dnf install epel-release
```

4. Instalación del repositorio para Docker

```
#dnf config-manager --add-repo=https://download.docker.com/linux/CentOS/docker-ce.repo
#dnf list docker-ce
```

5. Instalación de Ansible

```
#dnf install screen
#dnf install python36-devel libffi-devel gcc openssl-devel python3-libselinux
#dnf install ansible git
```

6. Configurando python3.6

```
#python3 -m pip --version
#python3 -m pip install --upgrade pip
#python3 -m pip install --upgrade pip setuptools Wheel
#python3 -m pip install kolla-ansible --ignore-installed
```

Esto es para que use Python 3.6 en CentOS 8:

```
#ln -s /usr/bin/python3 /usr/bin/python
#python –version
```

7. Instalando kolla-ansible

```
cd /root
#git clone https://github.com/openstack/kolla
#git clone https://github.com/openstack/kolla-ansible
#cp -r /usr/local/share/kolla-ansible/etc_examples/kolla /etc/
#cp /usr/local/share/kolla-ansible/ansible/inventory/* .
#python3 -m pip install -r kolla/requirements.txt
#python3 -m pip install -r kolla-ansible/requirements.txt
#ansible -i all-in-one all -m ping
#mkdir -p /etc/kolla
#cp -r kolla-ansible/etc/kolla/* /etc/kolla
#cp kolla-ansible/ansible/inventory/* .
```

8. Configuración de Kolla incluyendo el módulo de ironic.

```
vim /etc/kolla/globals.yml
# Must be defined with these values:
grep -v ^# /etc/kolla/globals.yml | grep -v ^$
---
config_strategy: "COPY_ALWAYS"
kolla_install_type: "binary"
openstack_release: "master"
node_custom_config: "/etc/kolla/config"
kolla_internal_vip_address: "10.1.12.100"
docker_namespace: "kolla"
network_interface: "ens4"
neutron_external_interface: "ens3"
neutron_plugin_agent: "openvswitch"
keepalived_virtual_router_id: "81"
enable_openstack_core: "yes"
enable_haproxy: "no"
enable_cinder: "yes"
enable_cinder_backup: "yes"
enable_cinder_backend_iscsi: "{{ enable_cinder_backend_lvm | bool or
enable_cinder_backend_zfssa_iscsi | bool }}"
enable_cinder_backend_lvm: "yes"
enable_heat: "{{ enable_openstack_core | bool }}"
enable_horizon_ironic: "{{ enable_ironic | bool }}"
enable_ironic: "yes"
enable_ironic_ipxe: "yes"
enable_ironic_neutron_agent: "{{ enable_neutron | bool and enable_ironic | bool }}"
enable_ironic_pxe_uefi: "yes"
```

```
enable_iscsid: "{{ (enable_cinder | bool and enable_cinder_backend_iscsi | bool) or
enable_ironic | bool }}"
nova_compute_virt_type: "qemu"
ironic_dnsmasq_interface: "{{ network_interface }}"
ironic_dnsmasq_dhcp_range: "172.16.32.20,172.16.32.30"
ironic_dnsmasq_boot_file: "pxelinux.0"
ironic_inspector_kernel_cmdline_extras: ['ipa-lldp-timeout=90.0','ipacollect-lldp=1']
```

9. Instalar los siguientes paquetes :

```
#python3 -m pip install --upgrade decorator python-dateutil
#python3 -m pip install openstack-heat --ignore-installed
```

10. Configurar el módulo de NOVA para que trabaje con el hypervisor KVM

```
#mkdir -p /etc/kolla/config/nova
#vim /etc/kolla/config/nova/nova-compute.conf
[libvirt]
virt_type=qemu
```

11. Generando contraseñas para ingreso al web

```
#kolla-ansible/tools/generate_passwords.py
#grep keystone_admin_password /etc/kolla/passwords.yml
# keystone_admin_password: FL5ullJrsGzdP3c1a3pdGot8Wcgzfd4q36rRfsqH
```

12. Generar respaldo de la máquina virtual

```
# virsh shutdown Controller8
# virt-clone  --original Controller8 –-name backup –auto-clone
```

13. Se realiza un backup en el hypervisor en este punto, por tanto, se apaga la máquina virtual "Controller8"
14. Para la instalación de Kolla y kolla-ansible se realiza el modo *"all in one":*

```
#cd /root
#kolla-ansible/tools/kolla-ansible -i all-in-one bootstrap-servers
```

15. Validar el archivo /etc/hosts editado por el *playbook*

```
cat /etc/hosts
# 127.0.0.1 localhost
# ::1     localhost localhost.localdomain localhost6 localhost6.localdomain6
# # BEGIN ANSIBLE GENERATED HOSTS
# 172.16.32.12 controller8
# # END ANSIBLE GENERATED HOSTS
```

16. En este punto NO debe de fallar nada en el *playbook* anterior, en caso de haber algún error se tiene que corregir antes de avanzar al *prechecks*.

```
#kolla-ansible/tools/kolla-ansible -i all-in-one prechecks
# screen
```

17. En este punto de debe de fallar nada en el *playbook* anterior, en caso de haber algún error previamente antes de avanzar al *deploy*.

```
kolla-ansible/tools/kolla-ansible -i all-in-one deploy
```

El deploy tarda aproximadamente 70 minutos y genera 43 *containers* en *docker*:

```
# docker ps -a
CONTAINER ID        IMAGE                                                COMMAND                  CREATED             STATUS                       PORTS               NAMES
84f4a4f2b6c1        kolla/CentOS-source-bifrost-deploy:3.0.1   "/sbin/init"             2 weeks ago         Up 2 weeks                                       bifrost
ff53b89ebd33        bf0696ec01a0                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       horizon
36c665ca39b9        587914b993eb                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       heat_engine
4655def96ca6        b1ef90d16469                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       heat_api_cfn
be6b488cd936        d18da38980ef                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       heat_api
9d37cb797c37        5874e346c9e9                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       ironic_neutron_agent
1562d1214208        927b87cd271e                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       neutron_metadata_agent
634dc2fc7e83        3b2d19b5e721                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       neutron_l3_agent
861a3e3b31ae        43c19676746c                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       neutron_dhcp_agent
df46b3af225f        2a9299f03582                                         "dumb-init --single-…"   4 weeks ago         Restarting (1) 26 seconds ago                    neutron_openvswitch_agent
960e3cc6e25b        6eaeb00afa1a                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       neutron_server
e17b1bdd6f2b        f7466f06e2e6                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       nova_compute_ironic
487e3476a2ba        f9fecacb4c98                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       nova_compute
8bf8e6c73660        46ccff88322f                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       nova_libvirt
f1524141111a        ef4f698d5c7b                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       nova_ssh
9c3c2193f7c3        252f47250a3d                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       nova_novncproxy
700fd873100d        51102fc3c577                                         "dumb-init --single-…"   4 weeks ago         Restarting (1) 17 seconds ago                    nova_conductor
6722d569396a        f366087521d3                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       nova_api
a0cc4a24fc86        a9a60532b0d3                                         "dumb-init --single-…"   4 weeks ago         Up 4 seconds                                     nova_scheduler
c024484f3bd9        85ac2f7acb05                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       cinder_backup
85b1e9b6edad        fe4d821f2628                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       cinder_volume
d10a7d204454        e428943dc827                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       cinder_scheduler
a6b1fc6b9a9b        f589b6b1ea15                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       cinder_api
5dde5987d958        82497e601568                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       glance_api
218f768fd364        7df1ae77677f                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       ironic_dnsmasq
d47db55459fb        8d6cba83a050                                         "dumb-init --single-…"   4 weeks ago         Up 3 weeks                                       ironic_ipxe
```

```
0cb9498a3632        8d6cba83a050                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          ironic_pxe
275a0b3ebe5c        76098a2aa20e                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          ironic_inspector
b9c8be36c8a5        1836d8505bb5                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          ironic_api
b3032bb96336        24f5a9445182                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          ironic_conductor
2c90d7f56139        dd7db8e8c729                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          rabbitmq
5a0ad086a14c        1a874bb05b0e                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          keystone_fernet
8b8ee71930ca        b5d10b4b67d7                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          keystone_ssh
03a605d470fd        e36b202fbe81                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          keystone
8e9a6638daf7        f43d63f11f64                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          memcached
adb36b16c40b        e94bcf728c53                "dumb-init -- kolla_..."   4 weeks ago         Up 3 weeks
          mariadb
5c637d487f48        d18b3b9aea2d                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          chrony
7213643a8d53        4baad9e32fb4                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          keepalived
0f2492abf10f        715e885ba6e7                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          haproxy
e2ecb6ae054c        ab426446703b                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          cron
d6033d5ff38b        cb1fa6f47d75                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          kolla_toolbox
e43d2ec691fa        74a405bdc316                "dumb-init --single-..."   4 weeks ago         Up 3 weeks
          fluentd
```

18. Post-instalación de kolla y kolla-ansible

```
kolla-ansible/tools/kolla-ansible -i all-in-one deploy
```

19. Instalación de paquetes openstack

```
#python3 -m pip install  python-openstackclient --ignore-installed python-glanceclient
python-neutronclient
#kolla-ansible/tools/kolla-ansible post-deploy
```

20. Ejecución de scripts adicionales

```
#vim kolla-ansible/tools/init-runonce
# EXT_NET_CIDR='172.16.0.0/16'
# EXT_NET_RANGE='start=172.16.32.151,end=172.16.32.200'
# EXT_NET_GATEWAY='172.16.24.254'    # Your network gateway


#source /etc/kolla/admin-openrc.sh
#kolla-ansible/tools/init-runonce
```

21. Validación del servicio web en horizon

```
URL: http://kolla_internal_vip_address:10.1.12.100/
# User: admin
# Password: FL5ullJrsGzdP3c1a3pdGot8Wcgzfd4q36rRfsqH
```

Anexo Técnico C

Instalación de Kolla y kolla-ansible en Ubuntu 20.04

1. Se requiere actualizar el sistema operativo y los últimos paquetes liberados

```
#apt-get  update
#apt-get upgrade
#reboot
```

2. A continuación se instalan paquetes de prerrequisitos

```
#apt-get install python3-dev libffi-dev gcc libssl-dev python3-pip
# apt-get install python-jinja2 python-pip libssl-dev
#apt install python3-dev python3-venv libffi-dev gcc libssl-dev git
```

3. Instalación de ansible a través de comando de sistema operativo y no del repositorio oficial de openstack

```
#apt-get install ansible
# vim /etc/ansible/ansible.cfg
[defaults]
host_key_checking=False
pipelining=True
forks=100
```

4. Instalación y actualización del comando pip

```
#ln -s /usr/bin/pip3 /usr/bin/pip
#pip install --upgrade pip setuptools wheel
#pip install -U 'ansible<2.10'
#pip install kolla-ansible
```

5. Descargando kolla-ansible y kolla

```
cd /root
#git clone https://github.com/openstack/kolla
#git clone https://github.com/openstack/kolla-ansible
#cp -r /usr/local/share/kolla-ansible/etc_examples/kolla /etc/
#cp /usr/local/share/kolla-ansible/ansible/inventory/* .
```

```
#python3 -m pip install -r kolla/requirements.txt
#python3 -m pip install -r kolla-ansible/requirements.txt
#ansible -i all-in-one all -m ping
#mkdir -p /etc/kolla
#cp -r kolla-ansible/etc/kolla/* /etc/kolla
#cp kolla-ansible/ansible/inventory/* .
```

6. Creación del volumen group para asignación del módulo de cinder

```
#pvcreate /dev/sdb
#vgcreate cinder-volumes /dev/sdb
```

7. Generación de password de kolla para el módulo de keystone

```
#kolla-genpwd
#grep keystone_admin_password /etc/kolla/passwords.yml
keystone_admin_password: amq3jRWyeOI7UOsrJsHygjdvY516iz58scUb4g9S
```

8. Instalación de paquete heat de openstack

```
#pip install openstack-heat --ignore-installed
```

9. Configuración de kolla

```
#cp -r /usr/local/share/kolla-ansible/etc_examples/kolla /etc/kolla
#vim /etc/kolla/globals.yml
grep -v ^# /etc/kolla/globals.yml | grep -v ^$
---
---
config_strategy: "COPY_ALWAYS"
kolla_base_distro: "ubuntu"
kolla_install_type: "source"
openstack_release: "master"
node_custom_config: "/etc/kolla/config"
kolla_internal_vip_address: "10.1.12.252"
docker_namespace: "kolla"
network_interface: "enp6s0"
neutron_external_interface: "enp3s0"
neutron_plugin_agent: "openvswitch"
keepalived_virtual_router_id: "151"
enable_haproxy: "no"
enable_cinder: "yes"
enable_cinder_backup: "yes"
enable_cinder_backend_lvm: "yes"
enable_fluentd: "yes"
enable_heat: "{{ enable_openstack_core | bool }}"
enable_horizon_heat: "{{ enable_heat | bool }}"
enable_horizon_ironic: "{{ enable_ironic | bool }}"
enable_ironic: "yes"
enable_ironic_ipxe: "yes"
enable_ironic_neutron_agent: "{{ enable_neutron | bool and enable_ironic | bool }}"
```

```
enable_ironic_pxe_uefi: "yes"
ironic_dnsmasq_interface: "{{ network_interface }}"
ironic_dnsmasq_dhcp_range: "10.1.12.240,10.1.12.250"
ironic_dnsmasq_boot_file: "pxelinux.0"
ironic_inspector_kernel_cmdline_extras: ['ipa-lldp-timeout=90.0','ipa-collect-lldp=1']
```

10. Configuración de Nova

```
#vim /etc/kolla/config/nova/nova-compute.conf
[libvirt]
virt_type=qemu
```

11. Configuración de Ironic

```
#mkdir -p /etc/kolla/config/ironic/
#curl
https://tarballs.openstack.org/ironic-python-agent/coreos/files/coreos_production_pxe.vmlinuz \
        -o /etc/kolla/config/ironic/ironic-agent.kernel
#curl
https://tarballs.openstack.org/ironic-python-agent/coreos/files/coreos_production_pxe_image-oem.cpio.gz \
        -o /etc/kolla/config/ironic/ironic-agent.initramfs
```

12. Instalación de paquetes adicionales de openstack

```
#pip install  python-openstackclient --ignore-installed python-glanceclient
python-neutronclient python-ironicclient
#apt-get install python3-diskimage-builder
```

13. Revisando prerrequisitos de Kolla

```
#kolla-ansible -i all-in-one all -m ping
```

14. Validando conexión a servidores

```
#kolla-ansible -i ./all-in-one bootstrap-servers
#kolla-ansible pull
#kolla-ansible -i ./all-in-one prechecks
```

15. Realizando la implementación de módulos en dockers de kolla

```
#kolla-ansible -i ./all-in-one deploy
```

16. Ejecutando el post-deploy de kolla

```
#kolla-ansible -i ./all-in-one post-deploy
```

17. Validación de los dockers

```
# docker ps -a
```

18. Ejecutando la post-instalación

```
#kolla-ansible -i ./all-in-one post-deploy
```

19. Corriendo el script para definir redes.

```
#vim kolla-ansible/tools/init-runonce
# EXT_NET_CIDR='172.16.0.0/16'
# EXT_NET_RANGE='start=172.16.32.151,end=172.16.32.200'
# EXT_NET_GATEWAY='172.16.24.254'    # Your network gateway


#source /etc/kolla/admin-openrc.sh
#kolla-ansible/tools/init-runonce
#grep keystone_admin_password /etc/kolla/passwords.yml
```

20. Validación del servicio web en horizon

```
URL: http://10.1.12.16/
# User: admin
# Password: FL5ullJrsGzdP3c1a3pdGot8Wcgzfd4q36rRfsqH
```

Anexo Técnico D

Instalación de Bifrost en un container

1. Hacer el deploy de bifrost solo se puede con CentOS, Ubuntu Oracle Linux (requiere mucho espacio en disco al menos unos 120GB adicionales)

```
#cd /root/kolla
# ./tools/build.py --base ubuntu --type source bifrost-deploy
#docker run -it --net=host -v /dev:/dev -d --privileged --name bifrost_deploy
kolla/ubuntu-source-bifrost-deploy:latest
```

2. Configurando bifrost y copiando en el container

```
#docker exec -it bifrost_deploy mkdir /etc/bifrost
#docker cp /etc/kolla/bifrost/servers.yml bifrost_deploy:/etc/bifrost/servers.yml
#docker cp /etc/kolla/bifrost/bifrost.yml bifrost_deploy:/etc/bifrost/bifrost.yml
#docker cp /etc/kolla/bifrost/dib.yml bifrost_deploy:/etc/bifrost/dib.yml
```

3. Ejecutando en el container los prerrequisitos para la instalación

```
#docker exec -it bifrost_deploy bash
#ssh-keygen
#cd /bifrost
#./scripts/env-setup.sh
#source /var/lib/kolla/venv/bin/activate
#pip install --upgrade pip
#pip install -r requirements.txt
#pip install -r test-requirements.txt
#cat > /etc/rabbitmq/rabbitmq-env.conf << EOF
   HOME=/var/lib/rabbitmq
   EOF
#./scripts/test_bifrost.sh
```

Anexo E

A continuación se lista un archivo XML para crear una máquina virtual a fin de simular la instalación de un servidor baremetal.

Es necesario crear previamente el disco virtual que usará esta máquina virtual.

```
#qemu create image -f format qcow2 -o size=200G virtual-baremetal.qcow2
```

Cabe resaltar que se tiene atributos como el UUID y la configuración de la terminal VNC para poder realizar dicha comunicación y el boot a través de la red por default.

```
<domain type='kvm'>
  <name>virtual-baremetal</name>
  <uuid>2a4cedd7-4912-4f96-ab04-a5c40e014c1c</uuid>
  <memory unit='KiB'>25165824</memory>
  <currentMemory unit='KiB'>25165824</currentMemory>
  <vcpu placement='static'>2</vcpu>
  <os>
     <type arch='x86_64' machine='pc-i440fx-rhel7.0.0'>hvm</type>
     <boot dev='network'/>
  </os>
  <features>
     <acpi/>
     <apic/>
  </features>
  <cpu mode='host-passthrough' check='none'/>
  <clock offset='utc'>
     <timer name='rtc' tickpolicy='catchup'/>
     <timer name='pit' tickpolicy='delay'/>
     <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <pm>
     <suspend-to-mem enabled='no'/>
     <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
     <emulator>/usr/libexec/qemu-kvm</emulator>
     <disk type='file' device='disk'>
     <driver name='qemu' type='qcow2'/>
     <source file='/home/VMs/HardDisk/virtual-baremetal.qcow2'/>
     <target dev='vda' bus='virtio'/>
     <address type='pci' domain='0x0000' bus='0x00' slot='0x09' function='0x0'/>
     </disk>
     <controller type='usb' index='0' model='ich9-ehci1'>
     <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x7'/>
     </controller>
     <controller type='usb' index='0' model='ich9-uhci1'>
     <master startport='0'/>
```

```
      <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x0'
multifunction='on'/>
      </controller>
      <controller type='usb' index='0' model='ich9-uhci2'>
      <master startport='2'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x1'/>
      </controller>
      <controller type='usb' index='0' model='ich9-uhci3'>
      <master startport='4'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x2'/>
      </controller>
      <controller type='pci' index='0' model='pci-root'/>
      <controller type='virtio-serial' index='0'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
      </controller>
      <interface type='bridge'>
      <mac address='52:54:00:ff:15:55'/>
      <source bridge='br172'/>
      <target dev='tap52'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
      </interface>
      <serial type='pty'>
      <target port='0'/>
      </serial>
      <console type='pty'>
      <target type='serial' port='0'/>
      </console>
      <channel type='unix'>
      <target type='virtio' name='org.qemu.guest_agent.0'/>
      <address type='virtio-serial' controller='0' bus='0' port='1'/>
      </channel>
      <input type='tablet' bus='usb'>
      <address type='usb' bus='0' port='1'/>
      </input>
      <input type='mouse' bus='ps2'/>
      <input type='keyboard' bus='ps2'/>
      <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'>
      <listen type='address' address='0.0.0.0'/>
      </graphics>
      <video>
      <model type='cirrus' vram='16384' heads='1' primary='yes'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
      </video>
      <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0'/>
      </memballoon>
  </devices>
  <seclabel type='none' model='none'/>
</domain>
```

Anexo F

Agregando un bridge nuevo dentro del contenedor de Open vSwitch:

```
# docker exec -it --user root neutron_openswitch_agent bash
>#ovs-vsctl add-br br-ironic
>#ovs-vsctl add-port br-ironic eth2
>#ovs-vsctl show
```

Anexo G

Instalación de herramienta libguestfs en CentOSOS

```
$ sudo dnf install libguestfs-tools
```

Instalación de herramienta libguestfs en Ubuntu

```
$ sudo apt-get install libguestfs-tools
```

Bibliografía

[1] Michael Heap, Ansible From Beginner to Pro, Apress, United Kingdom, 2016, ISBN-13 (pub): 978-1-4842-1660-6

[2] Anwar Osseyran & Merle Giles, Industrial Applications of High-Performance Computing, Best Global Practices. Illinois, USA, 2015, CRC Press, ISBN-13:978-1-4665-9681-8.

[3] Esteban Mocskos & Sergio Nesmachnow (Eds.), Communications in Computer and Information Science, High Performance Computing, Springer Nature, Argentina,2018, ISBN 978-3-319-73353-1. Library of Congress Control Number: 2017963753

[4] Pasupathinathan, Vijayakrishnan & Pieprzyk, Josef & Wang, Huaxiong & Cho, Joo. (2006). Formal analysis of card-based payment systems in mobile devices. 54. 213-220. 10.1145/1151828.1151853.

[5] Openstack, 2020, Disponible en web <https://www.openstack.org/software/>

[6] 2020 -EDUCBA (Corporate Bridge Consultancy Pvt Ltd) , Ansible Tutorial, Ansible Architecture, Disponible en web:< https://www.educba.com/ansible-architecture>

[7] OpenDev, 2021, Bifrost,consulta[agosto 2020],Disponible en web: <https://opendev.org/openstack/bifrost/>

[8] Openstack, ironic dev 11.1.5.dev17, Introduction,consulta [2021-02-04], Disponible en web:<https://docs.openstack.org/ironic/rocky/>

[9] Red Hat, Red Hat Ansible Automatization Platform, consulta [dic], Disponible en web: <https://www.redhat.com/es/technologies/management/ansible>

[10] Ansible project,Ansible-core 2.11, Ansible Core Documentation, consulta[agosto 2020, Disponible en web:<https://docs.ansible.com/ansible-core/devel/index.html>

[11] Red Hat, Cloud Computing,¿Qué es una nube?,consulta[diciembre 2020], Disponible en web:<https://www.redhat.com/es/topics/cloud-computing/what-is-private-cloud>

[12] Christine Hall, publicado en Agosto 6 del 2020, Renewed Interest in OpenStack Bare Metal Project Ironic, as Software Moves Closer to Hardware, DatacenterKnowledge,Disponible en web: <https://www.datacenterknowledge.com/open-source/renewed-interest-openstack-bare-metal-project-ironic-software-moves-closer-hardware>

[13] HPCSYSPROS19, System Professionals workshop, noviembre 22 2019,Denver CO, consulta [agosto 2020],Disponible en:<https://github.com/HPCSYSPROS/Workshop19>

[14] Mark Goddard, Bespoke Bare Metal: Ironic Deploy Templates, Publicado septiembre 23 de 2019, StackHPC Ltd, número:09938332, Disponible en web:< https://www.stackhpc.com/bespoke-bare-metal.html>

[15]Openstack, Bifrost Documentation, actualizado. 2020-6-07,kolla-ansible 6.2.4dev4, disponible en web: <https://docs.openstack.org/kolla-ansible/queens/reference/bifrost.html>

[16] Openstack, Bifrost Documentation, actualizado. 2020-6-07,kolla-ansible 6.2.4dev4, disponible en web:<https://docs.openstack.org/kolla-ansible/latest/reference/deployment-and-bootstrapping/bifrost.html>

[17] Openstack, Bifrost Documentation, actualizado. 2020-6-07,kolla-ansible 12.1.0dev103,Version Rocky, disponible en web:https://docs.openstack.org/project-deploy-guide/tripleo-docs/latest/environments/virtualbmc.html

[18] Rahul Sharma,blog, A new Beginning,Openstack and Container(project Kolla), Septiembre 20 2016, disponible en web:<https://rahulait.wordpress.com/category/networking/>

[19]Opentack,Bare Metal Service User Guide,ironic 18.0.1dev13,Junio 14 de 2021, version Ocata,disponible web:<https://docs.openstack.org/ironic/latest/user/index.html#conceptual-architecture>

[20] Red Hat, Red Hat OpenStack Platform 13, Baremetal Provisioning, actualizado 6 de Abril de 2021,pp49, disponible en web:<

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/pdf/bare_metal_provisioning/Red_Hat_OpenStack_Platform-13-Bare_Metal_Provisioning-en-US.pdf>

[21] Richard Jones's blog,libguestfs,Latest development version: 1.41.5 (released 2019-10-08).Library for accessing and modifying virtual machine disk images, disponible en web:<https://libguestfs.org/>

[22] Red Hat Inc, CoreOS Installer,version v0.9.1,disponible en web:<https://coreos.github.io/coreos-installer/>

[23] OpenStack, Ironic 9.1.8dev8, Create and add images to the Image service,updated: 2020-01-06,version pike,disponible en web:<https://docs.openstack.org/ironic/pike/install/configure-glance-images.html>

[24] Openstack, Openstack Cloud software, version ocata, Introduction to Ironic, Ironic 7.0.8dev7, disponible en web:<https://docs.openstack.org/ironic/ocata/deploy/user-guide.html>

[25]Openstack, VirtualBMC, TripleO 3.0.0,Actualizado 2021-06-09, disponible en web: <https://docs.openstack.org/project-deploy-guide/tripleo-docs/latest/environments/virtualbmc.html>