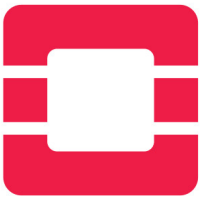
INSTALACIÓN Y CONFIGURACIÓN DE OPENSTACK CON MANILA



openstack®

MANILA

Introducción

En este proyecto haré una instalación de openstack y en especial estudiaré, instalaré y configuraré el componente de manila. A continuación explico un poco lo que es openstack y de que se compone y después explicaré el componente de manila y lo que nos proporciona.

¿Qué es openstack?

Openstack es un proyecto de cloud computing de código abierto programado en python. Openstack a diferencia de los programas que estamos acostumbrados a instalar, no es solo un programa, paquete o servicio, si no que mas bien esta formado por varios componentes. A continuación describiré brevemente los componentes esenciales de openstack.

Nova: Es el componente que se encarga de virtualizar dentro de openstack, por lo tanto tendremos que tener en cuenta que la máquina donde este dicho componente debe tener lo necesario para virtualizar como puede ser la extensión de virtualización del procesador.

Swift: Es el componente que se encarga del almacenamiento de objetos en openstack.

Cinder: Nos proporciona los volúmenes para poder utilizarlo en nuestros instancias.

Neutron: Gestiona todas las redes de openstack: ips flotantes, privadas, routers, redes privadas, externas…

Horizon: Nos proporciona el cliente web para openstack y así poder hacer más fácil la gestión de nuestros recursos en openstack.

Keystone: Se encarga de toda la autenticación en openstack. Toda la autenticación de usuarios también la gestiona él.

Glance: Nos proporciona el servicio de las imágenes en openstack. Hay que tener en cuenta que glance no acepta cualquier imagen hay que poner unas imágenes que las acepte.

Heat: El servicio de orquestación de openstack, gracias a heat podemos mediante una url o fichero crear un escenario automáticamente. Creando automáticamente todas las redes, instancias etc

Además de todos estos componentes openstack nos ofrece muchos más.
En la siguiente url podemos encontrar cada uno de ellos en la versión de pike: ***https://docs.openstack.org/pike/install/***

Pero como ya he dicho en este proyecto nos centraremos en el componente de Manila.

Componente de Manila

El componente de manila se encarga de establecer el servicio de ficheros compartidos en openstack. Esto una de las cosas que nos podría permitir por ejemplo es tener dos instancias y poder tener en las dos un volumen compartido y las dos pueden utilizarlo sin ningún problema.
Este componente podría confundirse con cinder y es normal ya que se baso en este al crearlo, pero una de las desventajas que tiene cinder es que no tiene la opción de tener un mismo volumen compartido entre varias instancias.
El componente de Manila soporta diferentes configuraciones de almacenamiento compartido como:

- NFS
-GlusterFS
-CephFS
-…

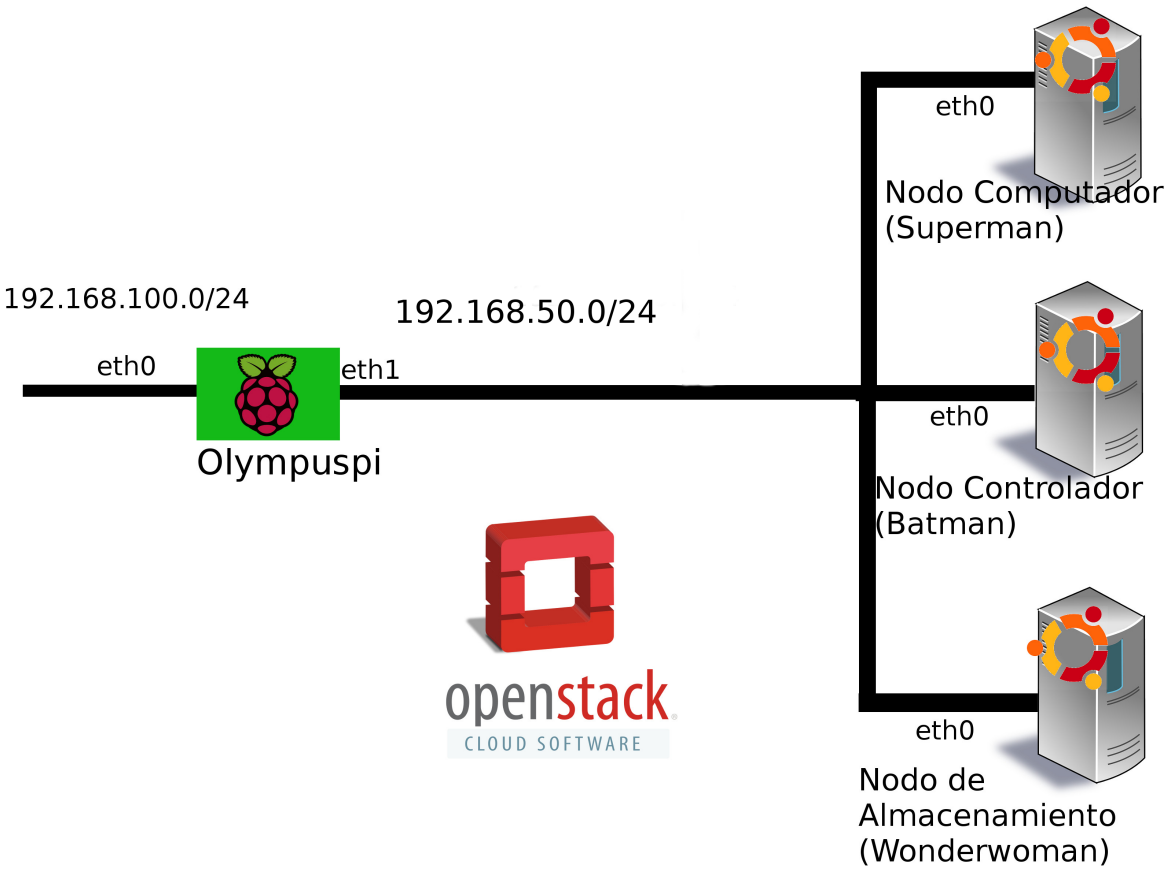
GlusterFS y CephFS es una buena opción para montarlo con manila ya que son sistemas de ficheros distribuidos y varias instancias podrían escribir a la vez en un mismo fichero.
Podemos ver todos los backends soportados por manila en la siguiente url: https://docs.openstack.org/manila/latest/admin/index.html

En conclusión para terminar con la descripción de este componente podemos decir que nos ayudará mucho para compartición de datos entre dos instancias y también para tratar el tema de alta disponibilidad en openstack.

En AmazonWebService tenemos un servicio que nos proporciona muy parecido a Manila llamado Amazon EFS. Podemos decir que Manila es la alternativa a este servicio en openstack(Para mas información de Amazon EFS: https://aws.amazon.com/es/efs/?nc2=h_m1).

Descripción del Escenario

A continuación adjunto un esquema del escenario que he creado:



Este escenario lo he creado con distintas máquinas que tengo en mi casa y he podido aprovecharlas para recrear el escenario. A continuación hago una descripción ligera del hardware que tiene:

-*Olympuspi*: Raspberry pi 2 con 1GB de ram
-*Superman*: Portátil con 8GB de RAM
-*Batman*: Ordenador de Sobremesa con 8GB de RAM
-*WonderWoman*: Portátil con 2GB de RAM

Como vemos tampoco es que haya unas máquinas muy potentes pero para hacer un “mini openstack” nos servirá.

Ahora describiré un poco que hará cada una de la máquinas que se ven en mi escenario:

-*Olympuspi*: Hará de router para separar cada una de las redes, también hará de servidor DNS para nuestra red interna y por último será un servidor VPN para poder conectarnos desde el exterior a nuestro openstack.

-*Superman*: Será el nodo computador de nuestro openstack, aquí estaría el componente de nova que como antes explicamos se encarga de virtualizar. Esta máquina por supuesto tiene la extensión de virtualización por hardware en el procesador.

-*Batman:* Es el nodo controlador, en este nodo estará componentes como glance, neutron, heat, entre otros. También estará el componente de horizon el encargado de proporcionarnos la api web, por lo que si quisiéramos entrar en la api web de openstack sería la ip de dicha máquina.
-*WonderWoman:* nodo de almacenamiento, en este nodo se encontrará el componente de cinder que nos proporciona los volúmenes para nuestras instancias. Para los volúmenes esta máquina tendrá conectado un dispositivo de bloques de 128Gb en lvm. Además en este nodo también actuará como nodo compartido por lo que también instalaremos manila en este nodo. Manila también utilizará otro dispositivo de bloques de 32 GB para los volúmenes compartidos.

Esta es la descripción de mi escenario ahora tendremos que configurar cada maquina para que haga su rol, hacer la instalación de openstack y por último la instalación y configuración de manila.

Configuración de Olympuspi

Como dije anteriormente olympuspi se encargará de hacer de router, dns y servidor VPN.
Instalaremos los siguientes paquetes necesarios, ya que como explicaré mas adelante tendremos que configurar las tarjetas de red de los nodos con vlan y para eso necesitaremos los paquetes necesarios.

```
root@olympuspi:/home/pi# apt install bridge-utils
root@olympuspi:/home/pi# apt install vlan
```

Y cargamos el módulo de vlan

```
echo "8021q" >> /etc/modules'
```

La configuración de las dos tarjetas de red de la raspberry es la siguiente:

```
auto eth0
iface eth0 inet static
address 192.168.100.14
netmask 255.255.255.0
gateway 192.168.100.1

auto eth1.10
iface eth1.10 inet manual
vlan-raw-device eth1

auto br-mgmt
iface br-mgmt inet static
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports eth1.10
address 192.168.50.1
netmask 255.255.255.0
```

Teniendo la configuración presente pasaremos a la instalación de los distintos servicios.

Configuración de router

Primero añadiremos la siguiente regla de iptables para que pueda pasar a distintas redes:

```
iptables --table nat --append POSTROUTING --jump MASQUERADE
```

También tendremos que activar el bit de forward para que la raspberry pueda hacer de router y enrutar hacia las dos redes.

```
root@olympuspi:/home/pi# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Esta configuración se iría si reiniciáramos la máquina para hacerla permanente haremos lo siguiente:

Para el bit de forward editamos la siguiente línea del fichero /etc/sysctl.conf

```
net.ipv4.ip_forward=1
```

Y para la regla de iptables la añadimos para que se ejecuten al arranque de la máquina en el fichero /etc/rc.local

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
/etc/init.d/ssh start
iptables --table nat --append POSTROUTING --jump MASQUERADE
exit 0
```

Y listo con esto nuestras máquinas de la red interna ya podrán acceder a internet y a la red externa.

Instalación y configuración de DNS.

¿Para qué instalo un DNS? quizás nos preguntemos. Básicamente es para poder acceder a nuestras distintas máquinas por su nombre ya que es más

fácil recordar los nombres de cada una que su ip. Para el servidor DNS utilizaremos bind9.

Instalación:

```
root@olympuspi:/home/pi# apt install bind9
```

Configuramos el fichero /etc/bind/named.conf.local en el que le indicaremos nuestra zona directa e inversa y los respectivos fichero de configuración de cada una.

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "vargax.com"
{
        type master;
        file "/var/cache/bind/db.vargax";
};

zone "50.168.192.in-addr.arpa"
{
     type master;
     file "/var/cache/bind/50.168.192.in-addr.arpa";
};
```

En el fichero de la zona directa(/var/cache/bind/db.vargax) tendremos la siguiente configuración:

```
; BIND reverse data file for empty rfc1918 zone
;
; DO NOT EDIT THIS FILE - it is used for multiple zones.
; Instead, copy it, edit named.conf, and use that copy.
;
$TTL 86400
@    IN   SOA  olympuspi.vargax.com. root.vargax.com. (
                     1        ; Serial
                604800        ; Refresh
                 86400        ; Retry
                2419200        ; Expire
                  86400 ) ; Negative Cache TTL
;
@    IN   NS   olympuspi.vargax.com.
@    IN   MX   10   mail.vargax.com.
$ORIGIN vargax.com.
olympuspi IN   A    192.168.50.1
olympuspi-ex   IN   A    192.168.100.14
batman    IN   A    192.168.50.10
superman  IN   A    192.168.50.12
wonderwoman    IN   A    192.168.50.14
tp-link   IN   A    192.168.100.60
huawei    IN   A    192.168.100.1
www  IN   CNAME     olympuspi-ex
ssh  IN   CNAME     olympuspi-ex
```

Cada una de las ip serán las designadas estáticamente para cada máquina. También he designado algunas máquinas como routers(tp-link, huawei) y el servicio de www y ssh aunque no son necesarios pero he aprovechado ya que iba a crear el DNS.
Y la configuración de la zona inversa de DNS

```
; BIND reverse data file for empty rfc1918 zone
;
; DO NOT EDIT THIS FILE - it is used for multiple zones.
; Instead, copy it, edit named.conf, and use that copy.
;
$TTL 86400
@    IN   SOA  olympuspi.vargax.com. root.vargax.com. (
                     1         ; Serial
                604800         ; Refresh
                 86400         ; Retry
                2419200         ; Expire
                  86400 ) ; Negative Cache TTL
;
@    IN   NS   olympuspi.vargax.com.
$ORIGIN 50.168.192.in-addr.arpa.
1    IN   PTR  olympuspi.vargax.com.
10   IN   PTR  batman.vargax.com.
12   IN   PTR  superman.vargax.com.
14   IN   PTR  wonderwoman.vargax.com.
```

Una vez configurado nuestro DNS ya podremos hacer consultas a nuestras zonas.

Instalación y configuración de VPN con Openswan

La configuración de este servicio nos servirá para poder acceder a nuestra red local desde cualquier lugar sin tener la necesidad de abrir demasiados puertos en nuestro router y dejar nuestra red insegura. Ahora pasaré a describir los pasos y configuraciones mas importantes de este servicio.

Instalamos los siguientes paquetes en nuestra máquina:

```
root@olympuspi:/home/pi# apt-get install openswan xl2tpd ppp lsof
```

Ahora añadiremos unas configuraciones en nuestra raspberry para que acepte redirect y las envíe. Al igual que cuando le dijimos que enrutara esta configuración la hacemos en el fichero /etc/systctl.conf. Y las dos siguientes líneas las descomentamos y la ponemos a 0.

```
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.send_redirects = 0
```

A continuación tendremos que hacer un bucle para rellenar con 0 todos los archivos de configuración de VPN que tengan redirects.

```
for vpn in /proc/sys/net/ipv4/conf/*; do echo 0 > $vpn/
accept_redirects; echo 0 > $vpn/send_redirects; done
```

Para que cada vez que reiniciemos no se pierda la configuración tendremos que añadir ese bucle al fichero /etc/rc.local

Una vez hecho estos pasos previos a la configuración de la VPN pasaremos ahora a configurarla.

Configuraremos primero ipsec. En el fichero /etc/ipsec.conf añadiremos la siguiente configuración.
Primero el bloque de setup donde configuraremos la configuración básica de ipsec.

```
config setup
        nat_traversal=yes
        protostack=netkey
        plutostderrlog=/var/log/log_error_ipsec.err
```

Y después configuraremos el bloque de L2TP-PSK para la configuración de L2TP.

```
conn L2TP-PSK
     authby=secret
     pfs=no
     auto=add
     rekey=yes
     ike=aes128-sha-modp1024!
     dpddelay=30
     dpdtimeout=120
     dpdaction=clear
     ikelifetime=8h
     keylife=1h
     type=tunnel
     left=192.168.100.14
     leftnexthop=%defaultroute
     leftprotoport=17/1701
     right=%any
     rightprotoport=17/%any
     rightsubnetwithin=0.0.0.0/0
```

Luego tendremos que configurar el secreto que será necesario para autenticarnos en la VPN, ya que la VPN necesita 3 cosas necesarias: usuario, contraseña y el secreto. El secreto se configura en el fichero /etc/ipsec.secrets y normalmente para el secreto se recomienda que utilicemos un frase basada en PSK.
En el fichero dicho añadimos la siguiente línea

```
%any %any: PSK "XXXXXXXX"
```

Por supuesto tendríamos que cambiar las X por nuestra PSK.
Ahora pasaremos a la configuración del servicio de xl2tpd. Nuestro primer fichero a configurar será /etc/xl2tpd/xl2tpd.conf
En ese fichero tendremos que configurar la parte global primero.

```
[global]
auth file = /etc/xl2tpd/l2tp-secrets
debug network = yes
debug tunnel = yes
```

Y la parte del lns por defecto la configuraremos de la siguiente manera:

```
[lns default]
ip range = 192.168.100.80-192.168.100.99
local ip = 192.168.100.14
require chap = yes
name = VPN_Vargax
ppp debug = yes
pppoptfile = /etc/ppp/options.xl2tpd
length bit = yes
```

Ahora configuraremos los atributos del túnel que se creará en el fichero /etc/ppp/options.xl2tpd añadiendo lo siguiente.

```
ipcp-accept-local
ipcp-accept-remote
ms-dns 192.168.100.14
noccp
auth
crtscts
idle 1800
mtu 1410
mru 1410
nodefaultroute
debug
lock
proxyarp
connect-delay 5000
```

Por último configuraremos los usuarios y contraseñas en el fichero /etc/ppp/chap-secrets con el siguiente formato:

```
# Secrets for authentication using CHAP
# client        server  secret                  IP
addresses
user * password *
```

Donde user es el nombre de usuario y password su contraseña.
Pues bien el servidor VPN ya estaría configurado. A continuación instalaremos un paquete en la máquina que genere números aleatorios para hacer el proceso de encriptación mas seguro.

```
root@olympuspi:/home/pi# apt install rng-tools
```

En el fichero /etc/modules le indicaremos que cargue el módulo instalado.

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
bcm2708-rng
```

Y en el fichero de configuración /etc/default/rng-tools indicaremos que utilice el dispositivo hwrng para generar números aleatorios descomentando la siguiente línea solo.

```
HRNGDEVICE=/dev/hwrng
```

Reiniciamos el servicio de rng-tools y ya estará listo.
Este servicio tendremos que ponerlo para que se inicie cada vez que la raspberry se inicie con el siguiente comando.

```
root@olympuspi:/home/pi# chkconfig rng-tools --level 2 on
```

Una vez hecho esto comprobaremos que el servicio de ipsec este bien configurado con la siguiente instrucción y nos tendrá que salir una salida parecida a la siguiente:

```
root@olympuspi:/home/pi# ipsec verify
Checking your system to see if IPsec got installed and started
correctly:
Version check and ipsec on-path                                 [OK]
Linux Openswan U2.6.38/K4.14.30-v7+ (netkey)
Checking for IPsec support in kernel                            [OK]
 SAref kernel support                                           [N/A]
 NETKEY:  Testing XFRM related proc values                      [OK]
     [OK]
     [OK]
Hardware RNG detected, testing if used properly                 [OK]
Checking that pluto is running                                  [OK]
 Pluto listening for IKE on udp 500                             [OK]
 Pluto listening for NAT-T on udp 4500                          [OK]
Checking for 'ip' command                                       [OK]
Checking /bin/sh is not /bin/dash
[WARNING]
Checking for 'iptables' command                                 [OK]
Opportunistic Encryption Support
[DISABLED]
```

Solo nos faltaría reiniciar los servicios de ipsec y xl2tpd y ya tendríamos la VPN funcionando.
Si queremos conectarnos desde el exterior tendremos que abrir los puertos en nuestro router 500, 4500 y 1701 todos udp.

Instalación de Openstack

En la instalación de openstack no encontramos con múltiples formas de hacerlo:

-***Manual***: Es la instalación manual de cada componente de openstack. Esta forma de instalación no es recomendable ya que nos llevaría mucho tiempo.

-***Openstack-ansible:*** Esta forma consiste en la instalación de openstack en el que los componentes están albergados en Linux Containers. Hay que tener en cuenta que el componente de nova no puede ir en un linux container ya que no puede virtualizar

-***Kolla-Ansible:*** Es la instalación con contenedores Docker.

-***RDO:*** Esta forma de instalación consiste en utilizar un nodo de despliegue e indicarle que nodo será el de almacenamiento, el controlador y computador y automáticamente este instalará y configurará los paquetes necesarios en cada uno de los nodos.

Estos son unos cuentos de los métodos que se pueden utilizar. En mi caso yo utilizaré el método de instalación de openstack-ansible, por lo que los linux Containers se encontrarán en el nodo controlador batman.
La version de openstack que instalaré será la de pike ya que la de queens todavía esta en release candidate.

Instalación de Openstack utilizando Openstack-Ansible

Antes de la instalación tendremos que preparar cada uno de los nodos, instalando los paquetes necesario y hacer distintas configuraciones en las tarjetas de red.

Preparación del nodo computador y de despliegue (superman)

Como dijimos antes este nodo además de actuar como computador también nos servirá para desplegar openstack.
Primero antes que nada actualizamos todo nuestro sistema

```
root@superman:/home/stack# apt update
root@superman:/home/stack# apt upgrade
```

Una vez hecho esto instalamos los siguiente paquetes necesarios.

```
root@superman:/home/stack# apt-get install aptitude build-essential
git ntp ntpdate openssh-server python-dev sudo bridge-utils
debootstrap ifenslave ifenslave-2.6 lsof lvm2 tcpdump vlan
```

Al igual que en la raspberry tendremos que cargar el módulo de vlan.

```
echo '8021q' >> /etc/modules
```

Ahora pasaremos a configurar las interfaces de red. Como solo tengo una tarjeta de red lo que haré es crear vlans y trataré cada vlan como si fuera una tarjeta de red distinta.
Antes de seguir con la configuración de la red diré los rangos que vamos a utilizar para cada red que utiliza openstack.

-***br-mgmt***: Esta interfaz se utiliza para comunicarse con los distintos componentes de openstack. Tendré dos interfaces como esta una interna para conectarse con los componentes y otra externa para que el nodo pueda salir al exterior. El rango que utilizaremos en el br-

mgmt será uno interno 192.168.51.0/24 y en el br-mgmt:0 nuestro rango externo 192.168.50.0/24.

-***br-vxlan***: Esta interfaz se utilizará para hacer comunicaciones internas en openstack. Utilizaremos el rango 192.168.52.0/24.

-***br-vlan:*** En esta interfaz no tendremos un rango de direcciones ya que esta interfaz solo se utilizará para comunicarse nuestras máquinas virtuales que creemos con neutron(mas tarde si le asignaremos ip).

-***br-storage:*** Esta interfaz la utilizaremos para comunicarnos con el nodo de almacenamiento wonderwoman. Utilizaremos el rango de ip 192.168.53.0/24

Una vez explicado esto seguiremos con la configuración de red de superman.
Las configuración del las diferentes redes es la siguiente:

Creación de las distintas vlan:

```
# The primary network interface
auto enp1s0
iface enp1s0 inet manual

#vlan Para el br-mgmt
auto enp1s0.10
iface enp1s0.10 inet manual
vlan-raw-device enp1s0

#vlan para br-vxlan
auto enp1s0.30
iface enp1s0.30 inet manual
vlan-raw-device enp1s0

#vlan para br-storage
auto enp1s0.20
iface enp1s0.20 inet manual
vlan-raw-device enp1s0
```

Configuración del br-mgmt tanto el externo como el interno:

```
#br-mgmt interno
auto br-mgmt
iface br-mgmt inet static
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp1s0.10
address 192.168.51.12
netmask 255.255.255.0
gateway 192.168.50.1
network 192.168.50.0
dns-nameservers 192.168.50.1
dns-search vargax.com

#br-mgmt externo
auto br-mgmt:0
iface br-mgmt:0 inet static
address 192.168.50.12
netmask 255.255.255.0
```

Configuración de br-vxlan y br-vlan

```
auto br-vxlan
iface br-vxlan inet static
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp1s0.30
address 192.168.52.12
netmask 255.255.255.0

auto br-vlan
iface br-vlan inet manual
bridge_stp off
bridge_waitport 0
bridge_fd 0
pre-up ip link add br-vlan-veth type veth peer name eth12 || true
pre-up ip link set br-vlan-veth up
pre-up ip link set eth12 up
post-down ip link del br-vlan-veth || true
bridge_ports enp1s0 br-vlan-veth
```

Y por último la configuración del br-storage

```
auto br-storage
iface  br-storage  inet
static
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp1s0.20
address 192.168.53.12
netmask 255.255.255.0
```

Reiniciamos el servicio de networking y listo. Ya estaría preparado superman para el despliegue.

Preparación del nodo controlador(batman)

Al igual que con superman primero antes que nada actualizaremos todo nuestro sistema

```
root@batman:/home/stack# apt update
root@batman:/home/stack# apt upgrade
```

Instalaremos los paquetes necesarios

```
root@batman:/home/stack# apt-get install bridge-utils debootstrap
ifenslave ifenslave-2.6 lsof lvm2 ntp ntpdate openssh-server sudo
tcpdump vlan
```

Y al igual que hicimos con superman cargaremos el modulo de vlan

```
root@batman:/home/stack# echo '8021q' >> /etc/
modules
```

Hecho esto pasaremos a configurar la red. Al igual que en el nodo computador solo tengo una tarjeta de red así que seguiré trabajando con vlans.

Creación de las vlan:

```
auto enp1s0
iface enp1s0 inet manual

auto enp1s0.10
iface enp1s0.10 inet manual
vlan-raw-device enp1s0

auto enp1s0.30
iface enp1s0.30 inet manual
vlan-raw-device enp1s0

auto enp1s0.20
iface enp1s0.20 inet manual
vlan-raw-device enp1s0
```

Configuramos el br-mgmt tanto el externo como el interno:

```
auto br-mgmt
iface br-mgmt inet static
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp1s0.10
address 192.168.51.10
netmask 255.255.255.0
gateway 192.168.50.1
network 192.168.50.0
dns-nameservers 192.168.50.1
dns-search vargax.com

auto br-mgmt:0
iface br-mgmt:0 inet static
address 192.168.50.10
netmask 255.255.255.0
```

Configuramos el br-vxlan y br-vlan con la diferencia en el br-vxlan que esta vez no tendrá ip.

```
auto br-vxlan
iface br-vxlan inet manual
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp1s0.30

auto br-vlan
iface br-vlan inet manual
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp1s0
```

Y por último el storage tampoco le pondremos ip ya que no la necesita.

```
auto br-storage
iface br-storage inet manual
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp1s0.20
```

Y listo ya tendríamos configurado el nodo controlador. Ahora pasaremos a configurar el último nodo.

Preparación del nodo de Almacenamiento(wonderwoman)

Primero antes que nada actualizamos el sistema:

```
root@wonderwoman:/home/stack# apt update
root@wonderwoman:/home/stack# apt upgrade
```

E instalaremos los paquetes necesarios al igual que en los otros 2 nodos.

```
root@wonderwoman:/home/stack# apt-get install bridge-utils
debootstrap ifenslave ifenslave-2.6 lsof lvm2 ntp ntpdate
openssh-server sudo tcpdump vlan
```

Cargamos el módulo de vlan

```
root@wonderwoman:/home/stack# echo '8021q' >> /etc/modules
```

Y pasaremos a la configuración de la red.
Creamos las distintas vlan.

```
auto enp4s0
iface enp4s0 inet manual

auto enp4s0.10
iface enp4s0.10 inet manual
vlan-raw-device enp4s0

auto enp4s0.30
iface enp4s0.30 inet manual
vlan-raw-device enp4s0

auto enp4s0.20
iface enp4s0.20 inet manual
vlan-raw-device enp4s0
```

Configuramos la interfaz br-mgmt, externa e interna.

```
auto br-mgmt
iface br-mgmt inet static
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp4s0.10
address 192.168.51.14
netmask 255.255.255.0
gateway 192.168.50.1
network 192.168.50.0
dns-nameservers 192.168.50.1
dns-search vargax.com

auto br-mgmt:0
iface br-mgmt:0 inet static
address 192.168.50.14
netmask 255.255.255.0
```

También configuramos la br-vxlan y br-vlan, al igual que en el nodo controlador br-vxlan no tendrá ip.

```
auto br-vxlan
iface br-vxlan inet manual
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp4s0.30

auto br-vlan
iface br-vlan inet manual
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp4s0
```

Y por último el br-storage, en esta ocasión si que tendrá ip ya que al despliegue tendremos que indicarle la ip del nodo de almacenamiento por la que obtendrá los volúmenes.

```
auto br-storage
iface br-storage inet static
bridge_stp off
bridge_waitport 0
bridge_fd 0
bridge_ports enp4s0.20
address 192.168.53.14
netmask 255.255.255.0
```

Y listo la configuración de red ya estaría.
Como dijimos al describir el esquema de red este nodo tiene conectado un dispositivo de bloques para que actúe como almacenamiento para la creación de volúmenes en cinder. Este dispositivo lo configuraremos en lvm.
Creamos un volumen con el dispositivo dejándoles de metadatos 2GB.

```
root@wonderwoman:/home/stack# pvcreate --metadatasize 2048 /dev/sdb
WARNING: dos signature detected on /dev/sdb at offset 510. Wipe it?
[y/n]: y
  Wiping dos signature on /dev/sdb.
  Physical volume "/dev/sdb" successfully created
```

Y con el volumen creado crearemos un grupo de volúmenes en lvm para cinder.

```
root@wonderwoman:/home/stack# vgcreate cinder-volumes /dev/sdb
  Volume group "cinder-volumes" successfully created
```

De modo que si vemos nuestro grupo de volúmenes nos tiene que aparecer algo parecido a esto:

```
root@wonderwoman:/home/stack# 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               111,00 GiB
  PE Size               4,00 MiB
  Total PE              7121
  Alloc PE / Size       0 / 0
  Free  PE / Size       7121 / 111,00 GiB
  VG UUID               ZpEnVi-qyZ1-UDY0-tbMN-C6mG-9ST4-r26bO7
```

Lo bueno que tiene lvm es que si en un futuro nos gustaría tener mas espacio en nuestro openstack para volúmenes solo tendríamos que añadir un dispositivo al grupo de volúmenes y automáticamente ya tendríamos más.

Pues hecho esto ya tendríamos listo nuestros 3 nodos para comenzar el despliegue e instalar openstack así que ¡Vamos a ello!

Despliegue e instalación de Openstack.

Para el despliegue de openstack tendremos que volver al nodo de despliegue que es el mismo que el computador, superman.

En este nodo lo primero que haremos es clonar la última version estable de pike del repositorio oficial de openstack-ansible.

```
root@superman:/home/stack# git clone -b 16.0.10 https://
github.com/openstack/openstack-ansible.git /opt/openstack-ansible
```

Y ejecutaremos el script de boostrap-ansible que prepara nuestro nodo para el despligue:

```
root@superman:~# /opt/openstack-ansible/scripts/bootstrap-ansible.sh
```

Si todo fue bien nos tendrá que salir al finalizar el script que el sistema fue “bootstrapead” y esta listo par su uso.
Ahora si entráramos en la carpeta /root/.ssh/ veríamos que el script nos ha generado un clave publica y privada.

```
root@superman:~/.ssh# ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts
root@superman:~/.ssh#
```

Por eso nuestra próxima tarea sería la de copiar la clave publica, es decir el id_rsa.pub, en el fichero /root/.ssh/authorized_keys de nuestros otros nodos para conectarnos por clave publica-privada a estos sin la necesidad de utilizar contraseñas.

```
root@batman:~/.ssh# cat authorized_keys
ssh-rsa   AAAAB3NzaC1yc2EAAAADAQABAAABAQDRzWH/
FV9bItkRYoaX5VjRBKRLwdcGYth/LrqlUWPHS47bKhwrK5cLSPd0lD69z6FWFXd/
8M…
```

```
root@wonderwoman:~/.ssh# cat authorized_keys
ssh-rsa   AAAAB3NzaC1yc2EAAAADAQABAAABAQDRzWH/
FV9bItkRYoaX5VjRBKRLwdcGYth/LrqlUWPHS47bKhwrK5cLSPd0lD69z6FWFXd/
8M…
```

Una vez veamos que nos podemos conectar como root sin ningún problema con clave pública-privada pasaremos al siguiente paso.

Copiamos los archivos de configuración de openstack-ansible a /etc

```
root@superman:~# cp -R /opt/openstack-ansible/etc/
openstack_deploy /etc/
```

Y en el directorio al que lo hemos copiado copiaremos el fichero user_config.example y lo copiaremos al que utilizaremos nosotros para la configuración.

```
root@superman:/etc/openstack_deploy# cp /etc/openstack_deploy/
openstack_user_config.yml.example /etc/openstack_deploy/
openstack_user_config.yml
```

Y en este nuevo fichero lo editaremos y empezaremos la configuración para el despliegue.[1]
Primero comenzamos definiendo las cidr_networks:
-container: Es la red que utilizamos desde los nodos en el adaptador br-mgmt
-tunnel: La red que utilizamos en el adaptador br-vxlan
-storage: La red que utilizamos en el adaptador br-storage

```
---
cidr_networks:
  container: 192.168.51.0/24
  tunnel: 192.168.52.0/24
  storage: 192.168.53.0/24
```

Ahora configuraremos el apartado de used_ips, que serán el rango de ips que queramos que estén reservadas para cada una de nuestras redes.

```
used_ips:
  - "192.168.51.20,192.168.51.100"
  - "192.168.52.20,192.168.52.100"
  - "192.168.53.20,192.168.53.100"
```

[1] Tenemos que tener en cuenta que el fichero openstack_user_config.yml es un yaml así que tendremos que tener en cuenta los espacios ya que es muy importante para la sintaxis.

Ahora configuraremos el apartado global_overrides que son configuraciones de cada una de las interfaces que utilizaremos, como es un apartado bastante extenso también lo haremos por parte como hemos hecho hasta ahora
A lo primero de este apartado tendremos que configurar las diferentes ips de nuestro nodo controlador:
-external_lb_vip_address: Es la dirección ip de nuestro nodo controlador de la interfaz br-mgmt:0. Si recordamos dijimos que br-mgmt tendría dos direcciones ips una externa y otra interna, pues en este apartado iría la externa.
-internal_lb_vip_address: En este apartado iría la otra dirección ip de nuestro nodo controlador en la interfaz br-mgmt, es decir la interna.

```
global_overrides:
  external_lb_vip_address: 192.168.50.10
  internal_lb_vip_address: 192.168.51.10
  tunnel_bridge: "br-vxlan"
  management_bridge: "br-mgmt"
```

Ahora en este mismo apartado de global_overrides configuraremos las provider_networks, o en otras palabras las configuraciones de cada una de las interfaces de red que tenemos así como también la red de los contenedores:
-**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
        is_ssh_address: true
```

-**br-vxlan**:

```
- 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
```

-**br-vlan**:

```
- network:
    container_bridge: "br-vlan"
    container_type: "veth"
    container_interface: "eth12"
    host_bind_override: "eth12"
    type: "flat"
    net_name: "flat"
    group_binds:
      - neutron_linuxbridge_agent
- network:
    container_bridge: "br-vlan"
    container_type: "veth"
    container_interface: "eth11"
    type: "vlan"
    range: "1:1"
    net_name: "vlan"
    group_binds:
      - neutron_linuxbridge_agent
```

-**br-storage**:

```
    - 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
```

Una vez configurado el global_overrides pasaremos a indicarle en este mismo fichero donde se econtrarán los distintos componentes de openstack.

```
shared-infra_hosts:
  infra1:
    ip: 192.168.51.10
repo-infra_hosts:
  infra1:
    ip: 192.168.51.10
haproxy_hosts:
  infra1:
    ip: 192.168.51.10
indentity_hosts:
  infra1:
    ip: 192.168.51.10
storage-infra_hosts:
  infra1:
    ip: 192.168.51.10
image_hosts:
  infra1:
    ip: 192.168.51.10
```

```
compute-infra_hosts:
  infra1:
    ip: 192.168.51.10
orchestration_hosts:
  infra1:
    ip: 192.168.51.10
dashboard_hosts:
  infra1:
    ip: 192.168.51.10
network_hosts:
  infra1:
    ip:192.168.51.10
compute_hosts:
  compute1:
    ip: 192.168.51.12
storage_hosts:
  storage1:
    ip: 192.168.51.14
    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: "192.168.53.14"
```

Como vemos casi todos los componentes están en el nodo controlador batman, salvo compute_hosts que dijimos que sería superman y storage_hosts que será wonderwoman.

En storage_hosts en el apartado iscsi_ip_address tendremos que poner la ip de wonderwoman en la interfaz br-storage.

Muy bien una vez configurado este fichero generaremos un fichero que es donde se econtrarán las claves para acceder a nuestros distintos componentes de openstack.
Ejecutamos la siguiente instrucción:

```
root@superman:/etc/openstack_deploy# cd /opt/openstack-ansible/
scripts/
root@superman:/opt/openstack-ansible/scripts# python pw-token-
gen.py --file /etc/openstack_deploy/user_secrets.yml
Creating backup file [ /etc/openstack_deploy/user_secrets.yml.tar
]
Operation Complete, [ /etc/openstack_deploy/user_secrets.yml ] is
ready
```

Si vemos dicho fichero generado veremos que están las distintas claves para los componentes.

Una vez hecho esto ya estaríamos listo para desplegar. Pero para asegurarnos que esta todo correcto podemos ejecutar el siguiente script que nos confirmará que no tenemos ningún error de sintaxis en la configuración:

```
root@superman:/opt/openstack-ansible/playbooks# openstack-ansible setup-
infrastructure.yml --syntax-check
Variable files: "-e @/etc/openstack_deploy/user_secrets.yml -e @/etc/
openstack_deploy/user_variables.yml "
[DEPRECATION WARNING]: docker is kept for backwards compatibility but usage is
discouraged. The module documentation details page may explain more about this
rationale..
This feature will be removed in a future release. Deprecation warnings
can be disabled by setting deprecation_warnings=False in ansible.cfg.

playbook: setup-infrastructure.yml

EXIT NOTICE [Playbook execution success] ***********************************
==============================================================================
```

Nos tendrá que salir algo parecido a lo anterior si todo esta bien.

Una vez visto que todo esta bien comenzaremos con el despliegue. El despliegue consiste en la ejecución de unos determinados playbooks que nos proporciona openstack-ansible en la carpeta /opt/openstack-ansible/playbooks, así que comenzaremos a ejecutar los playbooks:

-Primer playbook(setup-hosts.yml): Este playbook prepara los distintos nodos y crea los contenedores necesarios e instala las características y programas necesarias en estos.

```
root@superman:/opt/openstack-ansible/playbooks# openstack-ansible
setup-hosts.yml
```

Cuando termine de ejecutarse nos tendrá que salir algo parecido a lo siguiente:

```
lxc_container_create : Run container veth wiring script
---------------- 15.83s
lxc_container_create : Read custom facts from previous runs
------------ 14.38s
lxc_container_create : Container service directories
------------------- 14.01s
lxc_container_create     :     Create     hostname
-------------------------------- 13.67s
lxc_container_create : LXC host config for container networks
---------- 13.30s

EXIT   NOTICE   [Playbook   execution   success]
************************************
=============================================================
==============
```

Indicándonos que el playbook se ejecutó perfectamente.

—Errores—
Si en la ejecución de este playbook nos da un error de “LXC cache has been prepared”, como me ha pasado a mi, solo tendremos que irnos al fichero /etc/openstack_deploy/user_variables.yml y añadir una variable para que le de tiempo a preparar la cache.

```
lxc_cache_prep_timeout: 2700
```

-*Segundo Playbook(setup-infrastructure.yml):*Instala los servicios de la infraestructura como son: Memcached, el servidor de repositorio, Galera, RabbitMQ y rsyslog.

```
root@superman:/opt/openstack-ansible/playbooks#  openstack-ansible
setup-infrastructure.yml
```

Al igual que en el anterior libro nos tendrá que salir que se ejecuto perfectamente.

```
pip_install : Install PIP ------------------------------------------
10.71s
pip_install : Install distro packages ------------------------------
10.48s
repo_build : Create venv build options files -----------------------
10.26s
pip_install : Install UCA repository -------------------------------
5.38s
repo_server : Install repo caching server packages -----------------
4.99s
repo_build : Install pip packages ----------------------------------
4.88s

EXIT     NOTICE     [Playbook     execution     success]
************************************
=============================================================================
======
```

—Errores—
Si nos da un error de tipo timeout intentado crear los virtualenvs(build venv). Al igual que antes en el fichero de user_variables.yml tendremos que añadir las siguientes líneas indicándole un timeout mayor.

```
repo_build_venv_timeout:
120
repo build timeout: 2700
```

También podemos probar que se ha ejecutado perfectamente verificando el cluster de galera de la siguiente manera.

```
root@superman:/opt/openstack-ansible/playbooks# ansible galera_container -m
shell \
>   -a "mysql -h localhost -e 'show status like \"%wsrep_cluster_%\";'"
Variable files: "-e @/etc/openstack_deploy/user_secrets.yml -e @/etc/
openstack_deploy/user_variables.yml "
infra1_galera_container-15b3b70b | SUCCESS | rc=0 >>
Variable_name   Value
wsrep_cluster_conf_id   1
wsrep_cluster_size      1
wsrep_cluster_state_uuid        e56db5a8-43b4-11e8-9bf9-b291dc30edc8
wsrep_cluster_status    Primary
```

-*Tercer playbook(setup-openstack.yml):* Instala los componentes principales de openstack como neutron, heat, horizon, nova etc.

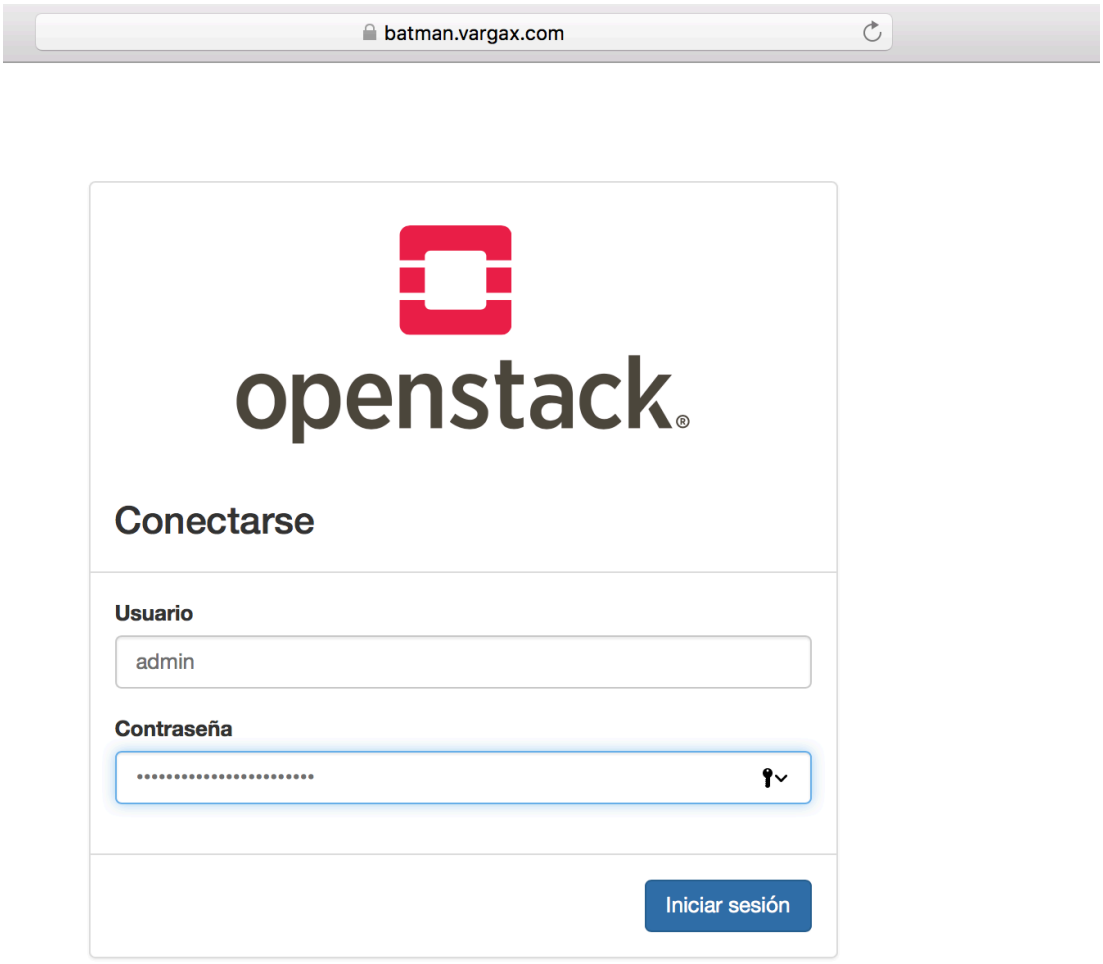
```
root@superman:/opt/openstack-ansible/playbooks# openstack-ansible
setup-openstack.yml
```

Al terminar también nos deberá salir que su ejecución se hizo perfectamente.

```
os_heat : Attempt venv download ---------------------------------------
10.55s
pip_install : Install UCA repository key ------------------------------
10.39s
pip_install : Remove known problem packages ---------------------------
10.28s
pip_install : Install UCA repository key ------------------------------
10.26s
pip_install : Install UCA repository key ------------------------------
10.02s

EXIT     NOTICE     [Playbook     execution     success]
**************************************
```

Y listo, ya podremos entrar en el panel de horizon con el usuario admin y la contraseña de keystone_auth_admin_password que como dijimos antes se encuentra en el fichero /etc/openstack_deploy/user_secrets.yml



Configuraciones Esenciales

Claro una vez instalado tendremos que configurar partes esenciales como puede ser el tema de las redes. Tendremos que conectarnos al contenedor de utilidades y desde ahi crear las redes que necesitemos.
Para saber cual es nuestro contenedor de utilidades desde batman ejecutaremos el siguiente comando para filtrar por dicho nombre.

```
root@batman:~# lxc-ls | grep utility
infra1_utility_container-66cff772
```

Y nos conectaremos a este contenedor y antes de configurar las redes cargaremos las variables de autenticación.

```
root@batman:~# lxc-attach -n infra1_utility_container-66cff772
root@infra1-utility-container-66cff772:~# source /root/openrc
```

Una vez hecho esto crearemos la red externa de tipo flat.

```
root@infra1-utility-container-66cff772:~# neutron net-create shoto --router:external True --provider:network_type flat --provider:physical_network flat
```

Y la subred correspondiente para la red externa, en el que tendremos que asignar el rango de ip flotantes que dé nuestro openstack a las instancias, además de configuraciones de red como el router que ofrecerá internet y el servidor DNS.

```
root@infra1-utility-container-66cff772:~# neutron subnet-create --name shoto --gateway 192.168.54.1 --allocation-pool start=192.168.54.50,end=192.168.54.99 --disable-dhcp --dns-nameserver 192.168.54.1 shoto 192.168.54.0/24
```

[2]

Ahora crearemos un router para conectarlo a la dicha red externa que hemos creado y a su vez podamos en nuestro proyecto conectar nuestra red interna también a este router.

```
root@infra1-utility-container-66cff772:~# neutron router-create toshinori
```

Y mediante el network_id de nuestra subred creada externa y el id de nuestro router los conectaremos.

```
root@infra1-utility-container-66cff772:~# neutron router-gateway-set cce8f9e8-8488-4ee7-834f-9b86d90f30b7 284f8051-b7a2-4eda-b318-3d1777058344
```

[3]

[2] El rango de 192.168.54.0/24 será nuestro rango por la que podremos acceder externamente a las instancias y la ip 192.168.54.1 es la ip de nuestra raspberry pi que tendremos que añadir en la interfaz eth1. Ya que no podremos utilizar el rango 192.168.50.0 ya que están en una vlan diferente y las instancias no podrían estar en esa vlan por eso es mejor optar por la opción de añadir un nuevo rango a nuestro escenario.

[3] La sintaxis es primero id del router y después la id de la subred externa

Y ahora configuraremos también un usuario para administrar nuestra instancias, ya que no es muy recomendable utilizar el usuario admin para esto.
Mi usuario que crearé necesitará un dominio, esto es opcional pero en mi caso por gusto lo pondré. Creamos el dominio.

```
root@infra1-utility-container-66cff772:~# openstack domain create
--description "Dominio vargax.com" vargax
```

Y nos crearemos un proyecto asociado a este dominio.

```
root@infra1-utility-container-66cff772:~# openstack project
create --domain vargax --description "Proyecto del Administrador
Vargax" vargax
```

Crearemos nuestro usuario en nuestro dominio vargax. La contraseña se nos pedirá una vez introducido dicho comando.

```
root@infra1-utility-container-66cff772:~# openstack user create
--domain vargax --password-prompt vargax
```

Y por último le damos el rol de admin a nuestro usuario creado.

```
root@infra1-utility-container-66cff772:~# openstack role add --
project vargax --user vargax admin
```

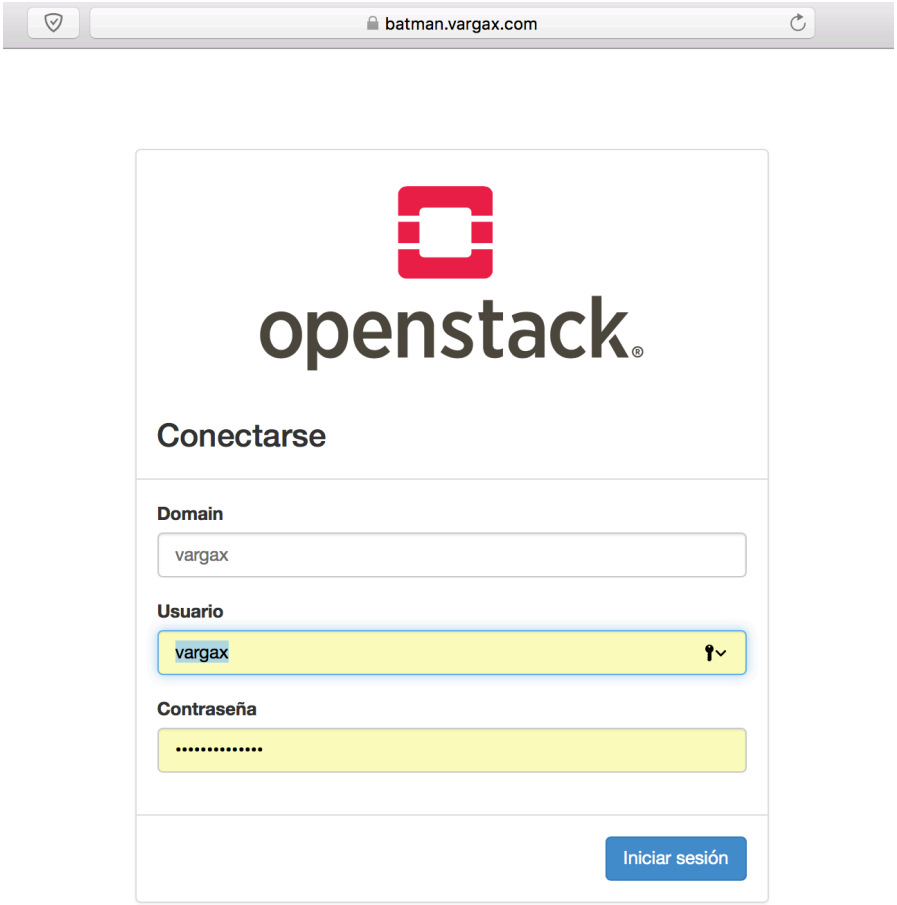
Lo mas probable es que en horizon no nos pida el dominio y no nos podamos autenticar con nuestro usuario. Para solucionar esto tendremos que entrar en el contenedor de horizon y modificar el fichero /etc/horizon/local_settings.py y poner la siguiente línea a True.

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

Y ya por último para tener una imagen al menos en nuestro openstack nos bajaremos la imagen oficial de debian para openstack y la introduciremos en glance.(http://cdimage.debian.org/cdimage/openstack/9.4.3-20180416/)

```
root@superman:~# openstack image create \
>         --container-format bare \
>         --disk-format qcow2 \
>                 --file  debian-9.4.3-20180416-openstack-
amd64.qcow2 \
>         debian-9-amd64
```

Una vez hecho esto ya podremos trabajar sobre nuestro openstack utilizando la cuenta creada de vargax en el dominio vargax.



Instalación y Configuración de Manila

Como dijimos antes wonderwoman también se encargará de tener este componente para tener nuestros volúmenes compartidos entre varias instancias.

A continuación hablaremos de las instalaciones y configuraciones que tendremos que hacer tanto en nuestro nodo controlador y nuestro nodo de almacenamiento para tener a punto el componente de manila.

Configuración en el nodo controlador

Primero desde cualquier nodo nos conectamos como root a nuestra base de datos de galera[4].

```
root@superman:~# mysql -u root -p -h 192.168.51.10
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31023
Server version: 10.1.30-MariaDB-1~xenial mariadb.org binary
distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

MariaDB [(none)]>
```

En la base de datos crearemos una nueva base de datos para manila.

```
MariaDB [(none)]> CREATE DATABASE manila;
Query OK, 1 row affected (0.12 sec)
```

Creamos el usuario en nuestra base de datos para manila y le damos permisos en dicha base de datos creada.

[4] La contraseña de root esta en el fichero /etc/openstack_deploy/user_secret.yml en el nodo de despliegue, en mi caso superman.

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON manila.* TO
'manila'@'localhost' IDENTIFIED BY ‘********’;
Query OK, 0 rows affected (0.23 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON manila.* TO
'manila'@'%' IDENTIFIED BY ‘********’;
```

Ya tendríamos nuestra base de datos preparada para manila. Ahora crearemos un usuario para manila en openstack.
Nos conectamos al contenedor de utilidades. Y ejecutamos la siguiente instrucción para crear el usuario.

```
root@infra1-utility-container-66cff772:~# openstack user create
--domain default --password-prompt manila
User Password:
Repeat User Password:
```

Ponemos la password y ya tendríamos el usuario en openstack.
Y añadimos a este usuario al grupo admin.

```
root@infra1-utility-container-66cff772:~# openstack role add --
project service --user manila admin
```

Crearemos los servicios manila y manilav2 también en openstack.

```
root@infra1-utility-container-66cff772:~# openstack service
create --name manila \
>   --description "OpenStack Shared File Systems" share
```

```
root@infra1-utility-container-66cff772:~# openstack service
create --name manilav2 \
>   --description "OpenStack Shared File Systems" sharev2
```

Creamos los endpoints para la api de manila. La url tendremos que cambiar la ip por la de nuestro nodo controlador en la interfaz br-mgmt

```
root@infra1-utility-container-66cff772:~#  openstack  endpoint
create --region RegionOne \
share public http://192.168.51.10:8786/v1/%\(tenant_id\)s
root@infra1-utility-container-66cff772:~#  openstack  endpoint
create --region RegionOne \
share internal http://192.168.51.10:8786/v1/%\(tenant_id\)s
root@infra1-utility-container-66cff772:~#  openstack  endpoint
create --region RegionOne \
share admin http://192.168.51.10:8786/v1/%\(tenant_id\)s
```

Y también crearíamos los endpoint para la api de manilav2.

```
root@infra1-utility-container-66cff772:~#  openstack  endpoint
create --region RegionOne \
sharev2 public http://192.168.51.10:8786/v2/%\(tenant_id\)s
root@infra1-utility-container-66cff772:~#  openstack  endpoint
create --region RegionOne \
sharev2 internal http://192.168.51.10:8786/v2/%\(tenant_id\)s
root@infra1-utility-container-66cff772:~#  openstack  endpoint
create --region RegionOne \
sharev2 admin http://192.168.51.10:8786/v2/%\(tenant_id\)s
```

Por último nos conectaremos al contendor de rabbit y creamos un usuario en rabbit llamado openstack con una password que nosotros queramos, ya que manila necesitará un usuario para oslo.

```
root@batman:~# lxc-attach -n infra1_rabbit_mq_container-bde5699c
root@infra1-rabbit-mq-container-bde5699c:/etc/rabbitmq#  rabbitmqctl  add_user
openstack ********
```

Y le damos permisos a dicho usuario:

```
root@infra1-rabbit-mq-container-bde5699c:/etc/rabbitmq#
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Listo ya tendríamos el escenario preparado para instalar y configurar manila.

Primero instalaremos y configuraremos el servicio de manila-scheduler y manila-api que estarán en nuestro nodo controlador batman.
Instalamos los siguientes paquetes.

```
root@batman:~# apt-get install manila-api manila-scheduler
python-manilaclient
```

Ahora tendremos que configurar varios parámetros en el fichero de configuración de manila /etc/manila/manila.conf
En la parte de database tendremos que configurar la conexión, introduciendo la contraseña que le pusimos a manila en la base de datos y la ip de nuestro nodo controlador.

```
[database]
connection = mysql+pymysql://manila:********@192.168.51.10/manila
```

En la parte default tendremos que configurar varios parámetros como es el rpc_backend que será rabbit, default_share_type, rootwrap_config que será el fichero /etc/manila/rootwrap.conf, la autenticación que será por keystone y también introducir nuestra ip en una variable.

```
[DEFAULT]
rpc_backend = rabbit
default_share_type = default_share_type
rootwrap_config = /etc/manila/rootwrap.conf
auth_strategy = keystone
my_ip = 192.168.51.10
```

En la parte de keystone_authtoken tendremos que poner las variables necesitadas para que manila pueda conectar con keystone.

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = manila
password = ********
```

La password de manila será la que pusimos al crear dicho usuario.
En oslo_messaging_rabbit tendremos que poner las variables necesarias para rabbit. Tendremos que poner la ip del host de rabbit dicha ip la podemos encontrar en el contenedor de rabbit. Para ver dicha ip podemos ejecutar el siguiente comando:

```
root@batman:~# lxc-ls -f | grep rabbit
infra1_rabbit_mq_container-bde5699c                      RUNNING  1
onboot, openstack 10.0.3.111, 192.168.51.218
```

También tendremos que poner el usuario de openstack que creamos antes y su contraseña. De modo que en dicha parte de oslo tendríamos algo parecido ha esto.

```
[oslo_messaging_rabbit]
rabbit_host = 192.168.51.218
rabbit_port = 5672
rabbit_userid = openstack
rabbit_password = *******
rabbit_virtual_host = /
```

Y por último para terminar la configuración de este fichero configuraremos la parte de oslo_concurrency indicándole el lock_path

```
[oslo_concurrency]
lock_path = /var/lib/manila/tmp
```

Y para terminar la configuración en nuestro nodo controlador ejecutamos el siguiente comando para crear el esquema de tablas en la base de datos de manila.

```
root@batman:/etc/haproxy/conf.d# su -s /bin/sh -c "manila-manage
db sync" manila
```

Se crearan dichas tablas, una vez haya terminado reiniciamos los servicios de manila-scheduler y manila-api y nos pasaremos a la configuración del nodo de almacenamiento.

```
root@batman:/etc/haproxy/conf.d# systemctl restart manila-scheduler
root@batman:/etc/haproxy/conf.d# systemctl restart manila-api
```

Configuración en el nodo de almacenamiento

En el nodo de almacenamiento primero tendremos que instalar unos paquetes necesarios:

```
root@wonderwoman:~# apt-get install manila-share python-pymysql
```

Y tendremos que configurar el fichero de manila.conf al igual que en el nodo controlador de la siguiente manera:
En la sección de **database** configuramos la conexión con el usuario de manila, su contraseña y la ip de nuestro nodo controlador

```
connection = mysql+pymysql://manila:********@192.168.51.10/manila
```

En el apartado de **default** configuramos el rcp que será rabbit, el default_share_type que es el tipo de compartido que lo dejaremos por defecto, habilitamos el fichero de configuración de rootwrap, la autenticación la haremos por keystone y por último pondremos nuestra ip del nodo de almacenamiento en la interfaz br-vlan[5]

```
rpc_backend = rabbit
default_share_type = default_share_type
rootwrap_config = /etc/manila/rootwrap.conf
auth_strategy = keystone
my_ip = 192.168.54.2
```

[5] En esta interfaz es probable que no tengamos ninguna ip asignada. Tendremos que añadir una ip que este en el rango de red de nuestras ips flotantes(192.168.54.0/24), para que nuestras instancias tengan acceso directamente al nodo de manila. Yo he optado por ponerle al nodo de almacenamiento la ip 192.168.54.2

Después configuraremos el apartado de **oslo_messaging_rabbit** y configuraremos el nodo de rabbit que es la ip del contenedor de rabbit al igual que configuramos en batman, el usuario que es openstack y su respectiva contraseña.

```
rabbit_host = 192.168.51.218
rabbit_userid = openstack
rabbit_password = ********
```

A continuación nos iremos al apartado **keystone_authtoken** y configuraremos los parámetros necesarios para keystone

```
memcached_servers = 192.168.51.10:11211
auth_uri = http://192.168.51.10:5000
auth_url = http://192.168.51.10:35357
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = manila
password = ********
```

Y por último configuraremos el apartado de **oslo_concurrency** y configuraremos el lock_path:

```
lock_path = /var/lib/manila/tmp
```

Esta es la configuración básica ahora pasaremos a configurar manila según nuestros criterios por ejemplo en mi caso como dije anteriormente utilizaré un nuevo dispositivo de bloques de 32GB en lvm para que manila tenga un grupo de volúmenes propio al igual que cinder.
El protocolo que utilizaré para compartir los volúmenes sera nfs.[6]

[6] Como dije al principio podríamos configurar manila con otros sistemas como puede ser gluster o ceph entre otros. Yo utilizare la configuración de manila en nfs4

Primero instalamos el paquete nfs-kernel-server para que podamos compartir volúmenes con manila.

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

Y antes de irnos a la configuración de manila crearemos con nuestro dispositivo un nuevo grupo de volúmenes con lvm.
Pero antes tendremos que editar el fichero de configuración de lvm ya que debido a la configuración de cinder es muy probable que lvm se haya configurado de un modo que tengamos que ir añadiendo a una lista los dispositivos que queramos configurar en lvm. Si es así al intentar crear un volumen nuevo nos debería salir el siguiente error.

```
root@wonderwoman:~# pvcreate /dev/sdc
  Device /dev/sdc not found (or ignored by filtering).
```

Vemos que entre paréntesis nos dice “or ignored by filtering” eso quiere decir que el filtro esta activado ya que el dispositivo sdc si existe.
Por lo tanto editamos el fichero /etc/lvm/lvm.conf. Y en la sección devices debe de haber un parámetro llamado filter el cual es una lista de dispositivos, solo tendríamos que añadir nuestro nuevo dispositivo a esta lista de la siguiente forma.

```
filter = [ "a/sdb/","a/loop.*/", "a/sdc/", "r/.*/" ]
```

Vemos que en esa lista ya se encuentra el dispositivo que queremos trabajar con el.
Hecho esto pasaremos a crear primero un volumen en nuestro dispositivo.

```
root@wonderwoman:/dev# pvcreate /dev/sdc
WARNING: iso9660 signature detected on /dev/sdc at offset 32769.
Wipe it? [y/n]: y
  Wiping iso9660 signature on /dev/sdc.
WARNING: dos signature detected on /dev/sdc at offset 510. Wipe
it? [y/n]: y
  Wiping dos signature on /dev/sdc.
  Physical volume "/dev/sdc" successfully created
```

Y crearemos el grupo de volúmenes de manila añadiendo este volumen a dicho grupo.

```
root@wonderwoman:/dev# vgcreate manila-volumes /dev/sdc
   Volume group "manila-volumes" successfully created
```

Ya tendríamos preparado nuestro volumen, ahora volveremos a la configuración de manila para terminar.

Volvemos al fichero /etc/manila/manila.conf y en la sección DEFAULT y habilitamos el backend de compartición con lvm y habilitamos los protocolos para compartir tanto NFS como CIFS

```
[DEFAULT]
enabled_share_backends = lvm
enabled_share_protocols = NFS,CIFS
```

Y al final del fichero añadiremos una nuevo sección para la configuración de los volúmenes compartidos como es el nombre que tendrá, el driver para los volúmenes, el nombre del grupo de volúmenes y la ip por la que será accesible nuestros volúmenes compartidos al igual que antes será por br-vlan.

```
[lvm]
share_backend_name = LVM
share_driver = manila.share.drivers.lvm.LVMShareDriver
driver_handles_share_servers = False
lvm_share_volume_group = manila-volumes
lvm share export ip = 192.168.54.2
```

Reiniciamos manila-share y ya estaría funcionando manila.

```
root@wonderwoman:~# systemctl restart manila-share
```

—Errores—

Uno de los errores mas comunes que nos puede dar manila-share tras reiniciarlo es el siguiente si vemos el fichero de log /var/log/manila/manila-share.log.

```
ERROR oslo_service.service IndexError: list index out of range
```

Y no debemos preocuparnos no es ningún tipo error de sintaxis en el fichero sino que mas bien es un bug de manilla. El error es retornado en la salida del comando “vgs --rows --units g”.
¿Cómo solucionamos dicho bug?
Pues tan simple como ejecutando dicha instrucción

```
root@wonderwoman:~# localectl set-locale LANG=en_US.utf8
```

Aquí le decimos que cambié nuestra localización de la máquina a en_US ya que es muy probable que lo tengamos en es_ES.
Y listo hecho esto si que no nos debería dar ningún error en el log manila.
Para comprobar que tanto el servicio de manila-share como el servicio de manila-scheduler están ejecutandose nos vamos al contenedor de utilidades y ejecutamos la siguiente instrucción de manila.

```
root@infra1-utility-container-66cff772:~# manila service-list
+----+------------------+------------------+------+---------+-------
+--------------------------+
| Id | Binary           | Host             | Zone | Status  | State | Updated_at
|
+----+------------------+------------------+------+---------+-------
+--------------------------+
| 1  | manila-scheduler | batman               | nova | enabled | up    |
2018-06-03T08:08:15.000000 |
| 2  | manila-share       | wonderwoman@lvm | nova | enabled | up    |
2018-06-03T08:08:10.000000 |
+----+------------------+------------------+------+---------+-------
+--------------------------+
```

Vemos que nos tiene que dar como resultado que tanto manila-scheduler como manila-share están un estado habilitado.

Crear y compartir volumenes con Manila

A continuación haré una prueba de la la creación de un volumen compartido con manila y como lo montamos en los equipos clientes.
Nos iremos al contenedor de utilidades y ahí primero crearemos un volumen con manila. La sintaxis para crear un volumen es la siguiente:

```
manila create NFS 1 --name “nombre volumen”
```

Y nos tendrá que salir lo siguiente:

```
root@infra1-utility-container-66cff772:~# manila create NFS 1 --name compartida_prueba_memoria
+-----------------------------------------+--------------------------------------+
| Property                                | Value                                |
+-----------------------------------------+--------------------------------------+
| status                                  | creating                             |
| share_type_name                         | default_share_type                   |
| description                             | None                                 |
| availability_zone                       | None                                 |
| share_network_id                        | None                                 |
| share_server_id                         | None                                 |
| share_group_id                          | None                                 |
| host                                    |                                      |
| revert_to_snapshot_support              | False                                |
| access_rules_status                     | active                               |
| snapshot_id                             | None                                 |
| create_share_from_snapshot_support      | False                                |
| is_public                               | False                                |
| task_state                              | None                                 |
| snapshot_support                        | False                                |
| id                                      | 0429dc8f-bf26-434f-85d7-e345e2536372 |
| size                                    | 1                                    |
| source_share_group_snapshot_member_id   | None                                 |
| user_id                                 | 47f6302f22aa49a4a5edde43d55534e4     |
| name                                    | compartida_prueba_memoria            |
| share_type                              | e0a69747-da9b-4237-acfb-bc8acdbcd6b3 |
| has_replicas                            | False                                |
| replication_type                        | None                                 |
| created_at                              | 2018-06-09T12:17:15.000000           |
| share_proto                             | NFS                                  |
| mount_snapshot_support                  | False                                |
| project_id                              | 55cf199c04cd4620958922fec1f47d88     |
| metadata                                | {}                                   |
+-----------------------------------------+--------------------------------------+
```

Vemos que el estado es creando, para listar nuestros volúmenes de manila y ver su estado ejecutamos la instrucción “manila list”:

```
root@infra1-utility-container-66cff772:~# manila list
+--------------------------------------+---------------------------+------+-------------+---------------+-----------+---------------------+-----------------------------------+-------------------+
| ID                                   | Name                      | Size | Share Proto | Status        | Is Public | Share Type Name     | Host                              | Availability Zone |
+--------------------------------------+---------------------------+------+-------------+---------------+-----------+---------------------+-----------------------------------+-------------------+
| 0429dc8f-bf26-434f-85d7-e345e2536372 | compartida_prueba_memoria | 1    | NFS         | available     | False     | default_share_type  | wonderwoman@lvm#lvm-single-pool | nova              |
| 05311e3d-10be-479a-a82f-e449001c72d3 | prueba4                   | 1    | NFS         | error_deleting | True     | default_share_type  | wonderwoman@lvm#lvm-single-pool | nova              |
| 07de7894-9151-4e7e-9a60-ea3957cf0fa2 | prueba_red                | 1    | NFS         | error         | False     | generic_share_type  |                                   | None              |
| 0db2eeba-6282-4663-8c04-d835f49a390e | prueba10                  | 1    | NFS         | error         | False     | default_share_type  | wonderwoman@lvm#lvm-single-pool | nova              |
| 50f562e1-fe36-4749-b379-6612dbb85458 | prueba_red3               | 1    | NFS         | error         | False     | generic_share_type  |                                   | None              |
| 6e2544ca-92cf-48ef-9e5b-eefa9418fadc | red                       | 1    | NFS         | available     | False     | default_share_type  | wonderwoman@lvm#lvm-single-pool | nova              |
| 8240a5f7-44c6-470d-bbd2-1625959d9cc3 | funcionara                | 1    | NFS         | error_deleting | False    | default_share_type  | wonderwoman@lvm#lvm-single-pool | nova              |
| a415724d-0a52-4785-955a-6ac48ab8d3a0 | prueba_red4               | 1    | NFS         | error_deleting | False    | default_share_type  | wonderwoman@lvm#lvm-single-pool | nova              |
| dae7a25b-ea14-463b-8943-725c586f5686 | funcionara                | 1    | NFS         | error_deleting | False    | default_share_type  | wonderwoman@lvm#lvm-single-pool | nova              |
| ff9414df-083f-4943-9135-cc25149e8a91 | funcionara2               | 1    | NFS         | error_deleting | False    | default_share_type  | wonderwoman@lvm#lvm-single-pool | nova              |
+--------------------------------------+---------------------------+------+-------------+---------------+-----------+---------------------+-----------------------------------+-------------------+
```

Vemos que nuestro volumen creado ya esta disponible.
Ahora para ver las especificaciones de nuestro volumen ejecutamos lo siguiente:

```
manila show “nombre volumen”
```

```
root@infra1-utility-container-66cff772:~# manila show compartida_prueba_memoria
+---------------------------------------+--------------------------------------------------------------------------------------------+
| Property                              | Value                                                                                      |
+---------------------------------------+--------------------------------------------------------------------------------------------+
| status                                | available                                                                                  |
| share_type_name                       | default_share_type                                                                         |
| description                           | None                                                                                       |
| availability_zone                     | nova                                                                                       |
| share_network_id                      | None                                                                                       |
| export_locations                      |                                                                                            |
|                                       | path = 192.168.54.2:/var/lib/manila/mnt/share-5febf1c0-4707-46b5-8d90-6c696f6390ec         |
|                                       | preferred = False                                                                          |
|                                       | is_admin_only = False                                                                      |
|                                       | id = f9accdf1-000f-4d85-8dd1-b01bfd0e5cbe                                                  |
|                                       | share_instance_id = 5febf1c0-4707-46b5-8d90-6c696f6390ec                                   |
| share_server_id                       | None                                                                                       |
| share_group_id                        | None                                                                                       |
| host                                  | wonderwoman@lvm#lvm-single-pool                                                            |
| revert_to_snapshot_support            | False                                                                                      |
| access_rules_status                   | active                                                                                     |
| snapshot_id                           | None                                                                                       |
| create_share_from_snapshot_support    | False                                                                                      |
| is_public                             | False                                                                                      |
| task_state                            | None                                                                                       |
| snapshot_support                      | False                                                                                      |
| id                                    | 0429dc8f-bf26-434f-85d7-e345e2536372                                                       |
| size                                  | 1                                                                                          |
| source_share_group_snapshot_member_id | None                                                                                       |
| user_id                               | 47f6302f22aa49a4a5edde43d55534e4                                                           |
| name                                  | compartida_prueba_memoria                                                                  |
| share_type                            | e0a69747-da9b-4237-acfb-bc8acdbcd6b3                                                       |
| has_replicas                          | False                                                                                      |
| replication_type                      | None                                                                                       |
| created_at                            | 2018-06-09T12:17:15.000000                                                                 |
| share_proto                           | NFS                                                                                        |
| mount_snapshot_support                | False                                                                                      |
| project_id                            | 55cf199c04cd4620958922fec1f47d88                                                           |
| metadata                              | {}                                                                                         |
+---------------------------------------+--------------------------------------------------------------------------------------------+
```

Vemos mucha información que nos proporciona dicha instrucción pero a nosotros lo que mas nos interesa es el path, ya que dicho path es que tendremos que copiar para montar el volumen de manila en nuestras instancias.
Pero todavía no podemos montar nada en ninguna instancia, ya que tendremos que darle permisos a la ip flotante de las instancias que queramos que tenga acceso al volumen creado.
Por ejemplo yo tengo 2 instancias con las siguientes ip flotantes:

- tokoyami(ubuntu-server): 192.168.54.55
- aizawa(debian): 192.168.54.61

A continuación utilizaré manila para darle acceso a tokoyami pero no aizawa.
Para darle acceso a una determinada ip utilizamos la siguiente instrucción de manila.

```
manila access-allow “nombre volumen” ip “ip flotante instancia”/32
```

```
root@infra1-utility-container-66cff772:~# manila access-allow compartida_prueba_memoria ip 192.168.54.55/32
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| access_key   | None                                 |
| share_id     | 0429dc8f-bf26-434f-85d7-e345e2536372 |
| created_at   | 2018-06-09T12:33:26.000000           |
| updated_at   | None                                 |
| access_type  | ip                                   |
| access_to    | 192.168.54.55/32                     |
| access_level | rw                                   |
| state        | queued_to_apply                      |
| id           | fb994e06-593b-41a1-9040-36af685be3dc |
+--------------+--------------------------------------+
```

Y tokoyami ya podría montar el volumen de manila. Para montar el volumen de manila en un equipo linux tendremos que utilizar de la siguiente manera el comando mount.[7]

```
mount -vt nfs “path” “directorio donde montar”
```

El path es la variable que dijimos anteriormente que nos salía al ver las características del volumen de manila.
Una vez ejecutado la instrucción nos debe de dejar montarlo, acceder a dicha carpeta compartida y crear ficheros.

```
root@tokoyami:/home/ubuntu# mount -vt nfs 192.168.54.2:/var/lib/manila/mnt/share-5febf1c0-4707-46b5-8d90-6c696f6390ec /mnt
mount.nfs: timeout set for Sat Jun  9 12:45:41 2018
mount.nfs: trying text-based options 'vers=4,addr=192.168.54.2,clientaddr=192.168.60.6'
root@tokoyami:/home/ubuntu# cd /mnt/
root@tokoyami:/mnt# touch prueba_desde_tokoyami
root@tokoyami:/mnt# ls
lost+found  prueba_desde_tokoyami
root@tokoyami:/mnt#
```

Pero si intentamos montar el volumen desde aizawa no nos dejaría porque no le hemos dado permisos desde manila a dicha instancia.

```
root@aizawa:/home/debian# mount -vt nfs 192.168.54.2:/var/lib/manila/mnt/share-5febf1c0-4707-46b5-8d90-6c696f6390ec /mnt
mount.nfs: timeout set for Sat Jun  9 12:48:50 2018
mount.nfs: trying text-based options 'vers=4.2,addr=192.168.54.2,clientaddr=192.168.60.9'
mount.nfs: mount(2): No such file or directory
mount.nfs: trying text-based options 'addr=192.168.54.2'
mount.nfs: prog 100003, trying vers=3, prot=6
mount.nfs: trying 192.168.54.2 prog 100003 vers 3 prot TCP port 2049
mount.nfs: prog 100005, trying vers=3, prot=17
mount.nfs: trying 192.168.54.2 prog 100005 vers 3 prot UDP port 34547
mount.nfs: mount(2): Permission denied
mount.nfs: access denied by server while mounting 192.168.54.2:/var/lib/manila/mnt/share-5febf1c0-4707-46b5-8d90-6c696f6390ec
```

[7] Para montar volumenes de manila en una instancia de linux tendremos que instalar el paquete nfs-common

Vemos que nos da el error de permiso denegado.
Ahora bien si nos vamos de nuevo al contenedor de utilidades y le damos permisos a la ip de aizawa si que nos dejará montarlo.

```
root@infra1-utility-container-66cff772:~# manila access-allow compartida_prueba_memoria ip 192.168.54.61/32
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| access_key   | None                                 |
| share_id     | 0429dc8f-bf26-434f-85d7-e345e2536372 |
| created_at   | 2018-06-09T12:51:20.000000           |
| updated_at   | None                                 |
| access_type  | ip                                   |
| access_to    | 192.168.54.61/32                     |
| access_level | rw                                   |
| state        | queued_to_apply                      |
| id           | 11d0f3cb-7e95-4db4-a047-6715dcca8901 |
+--------------+--------------------------------------+
```

Ahora si vamos a aizawa nos dejará montar el volumen, acceder a el y crear ficheros.

```
root@aizawa:/home/debian# mount -vt nfs 192.168.54.2:/var/lib/manila/mnt/share-5febf1c0-4707-46b5-8d90-6c696f6390ec /mnt
mount.nfs: timeout set for Sat Jun  9 12:54:56 2018
mount.nfs: trying text-based options 'vers=4.2,addr=192.168.54.2,clientaddr=192.168.60.9'
root@aizawa:/home/debian# cd /mnt/
root@aizawa:/mnt# touch prueba_desde_aizawa
root@aizawa:/mnt# ls
lost+found  prueba_desde_aizawa  prueba_desde_tokoyami
```

Además si nos fijamos podemos ver el fichero que creamos desde tokoyami.
En manila podemos ver las distintas ip que tienen acceso a un volumen con la siguiente instrucción:

```
manila access-list “nombre volumen”
```

```
root@infra1-utility-container-66cff772:~# manila access-list compartida_prueba_memoria
+--------------------------------------+-------------+-----------------+--------------+--------+------------+----------------------------+------------+
| id                                   | access_type | access_to       | access_level | state  | access_key | created_at                 | updated_at |
+--------------------------------------+-------------+-----------------+--------------+--------+------------+----------------------------+------------+
| 11d0f3cb-7e95-4db4-a047-6715dcca8901 | ip          | 192.168.54.61/32 | rw           | active | None       | 2018-06-09T12:51:20.000000 | None       |
| fb994e06-593b-41a1-9040-36af685be3dc | ip          | 192.168.54.55/32 | rw           | active | None       | 2018-06-09T12:33:26.000000 | None       |
+--------------------------------------+-------------+-----------------+--------------+--------+------------+----------------------------+------------+
root@infra1-utility-container-66cff772:~#
```

Conclusión

En conclusión podemos decir que **openstack** es una muy buena herramienta para que distintos usuarios puedan gestionar sus maquinas virtuales de una manera muy sencilla, así como la gestión y creación de redes virtuales dentro del proyecto del usuario, entre otras cosas.

Además el componente de **manila** para openstack nos puede ser muy util para compartir datos entre distintas instancias o hacer un cluster de instancias que atacara a un mismo volumen, entre otras opciones. También nos proporciona una alta seguridad sobre los volúmenes, ya que manila solo ofrecerá volúmenes a las ip flotantes de las instancias que nosotros designemos.

Algo muy interesante sobre manila también es el hecho de ser tan compatibles entre sistemas de ficheros distribuidos actuales como pueden ser de guster y ceph.

Estudiando también manila me he dado cuenta que lo realmente hay detrás de manila es un servidor nfs(en mi caso) y que manila es una forma muy sencilla de interactuar con nuestro servidor NFS.

Por último comentar que hay muchas configuraciones de manila además de la que yo he hecho, como puede ser la configuración de manila con otro tipo de driver en el que se pueden crear redes de manila que realmente están conectadas con las que nosotros le indiquemos de neutron y podemos compartir volumenes por dichas redes virtuales.