

Receta de integración de OpenStack con OpenDayLight

En este documento se realiza una descripción detallada de cómo realizar la instalación y despliegue de la plataforma OpenStack integrada con el controlador SDN OpenDayLight como proveedor de la infraestructura de red. En concreto, se trata de una receta de cómo configurar y desplegar esta infraestructura de integración a partir de un escenario configurado desde la herramienta de virtualización de redes VNX¹. Los pasos a seguir en este documento pueden ser utilizados para cualquier infraestructura OpenStack desplegada previamente donde se requiera la instalación e integración del controlador SDN OpenDayLight, independientemente de si se haya o no desplegado OpenStack con un escenario propio de VNX.

OpenStack puede utilizar OpenDayLight como su proveedor para la gestión de red a través del *plugin North-Bound Modular Layer 2 (ML2)*. OpenDayLight se encargará de utilizar los protocolos *South-Bound* OVSDB y *OpenFlow* para configurar directamente los *bridges* OVS del plano de datos SDN, actuando como nuevo *Mechanism Driver*. De esta forma, Neutron delega en OpenDayLight la gestión de los recursos de red de la infraestructura OpenStack.

Para la configuración del escenario de integración, partimos de un escenario base VNX ya desarrollado con la última versión estable de OpenStack Stein “*VNX Openstack Stein four nodes classic scenario using Open vSwitch*”², cuya infraestructura de despliegue está formada por 4 nodos o *hosts* virtuales: un nodo controlador, un nodo de red y dos nodos de computación; donde se utiliza OVS como mecanismo controlador de red en cada nodo de computación y red (ver la Figura 1).

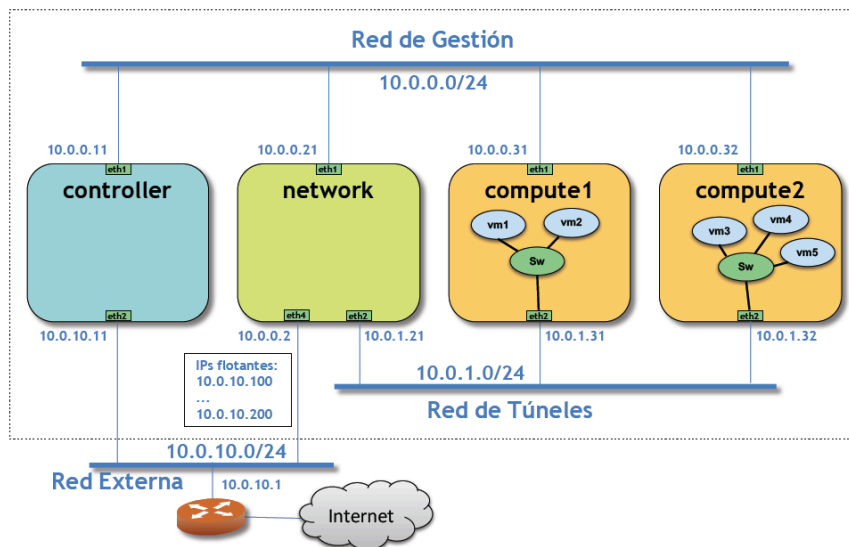


Figura 1: Arquitectura del escenario base OpenStack.

El escenario base de Openstack está formado por cuatro nodos o máquinas virtuales:

- Un nodo controlador, en el que se ejecutan las principales aplicaciones de control y la interfaz de usuario (*Dashboard*) de OpenStack.

¹ http://web.dit.upm.es/vnxwiki/index.php/Main_Page

² <http://web.dit.upm.es/vnxwiki/index.php/Vnx-labo-openstack-4nodes-classic-ovs-stein>

- Dos nodos de computación, que dan soporte a la creación de máquinas virtuales utilizando el hipervisor KVM de Linux.
- Un nodo de red, que se ocupa de proporcionar los servicios de red necesarios (*routers*, NAT, etc.) para que las máquinas virtuales tengan conectividad y acceso al exterior.

Los nodos poseen varias interfaces de red (dos el controlador y los nodos de computación y tres el nodo de red) mediante las cuales se conectan a tres redes distintas:

- Red de gestión (10.0.0.0/24), que es una red interna utilizada para todo el tráfico de control entre los nodos que componen la infraestructura Openstack (por ejemplo, para el tráfico de mensajes y llamadas REST entre todos los módulos *software*).
- Red de túneles (10.0.1.0/24), que es la red que soporta todo el tráfico entre máquinas virtuales y entre máquinas virtuales y el nodo de red. Se denomina “de túneles” ya que el tráfico se intercambia mediante protocolos de *tunneling* como VXLAN o GRE.
- Red Exterior (10.0.10.0/24), que es la red que da acceso al exterior (Internet) y que permite que las máquinas virtuales tengan acceso o sean accesibles desde el exterior a través del nodo de red.

Los nodos del escenario son máquinas virtuales (contenedores LXC) arrancadas en el *host* anfitrión. Asimismo, las redes que interconectan los nodos son *switches* virtuales OVS creados en el *host* anfitrión.

A continuación, se comenta paso a paso, a modo de manual o receta de instalación, cómo integrar el controlador SDN OpenDayLight como *Mechanism Driver* de red de la plataforma OpenStack.

En el enlace del repositorio público GitHub³ se encuentra disponible el escenario VNX desarrollado para la infraestructura de integración OpenStack con OpenDayLight.

Entre la documentación disponible para la configuración de este escenario de integración de OpenStack con OpenDayLight se han seguido las guías oficiales de instalación de la librería “odl-networking” de OpenStack⁴ y del complemento NetVirt de OpenDayLight⁵.

1. Instalación y configuración del escenario base.

1.1. Modificación del escenario “*VNX Openstack Stein four nodes classic scenario using Open vSwitch*”, integrando un nuevo nodo que haga la función de controlador OpenDayLight (modificación en el archivo “openstack_opendaylight_lab.xml”, con la descripción del nodo “opendaylight” con interfaces “MgmtNet”, “ExtNet” y “virbr0” y creación de un nuevo *rootfs* propio “rootfs_lxc_ubuntu64-ostack-opendaylight”). El diseño del nuevo escenario con la inclusión del nodo “opendaylight” se ve reflejado en la Figura 2 de la página siguiente.

³ <https://github.com/daniel-gonzalez-sanchez/openstack-opendaylight-monitoring>

⁴ <https://docs.openstack.org/networking-odl/latest/install/installation.html>

⁵ <https://docs.opendaylight.org/en/stable-boron/submodules/netvirt/docs/openstack-guide/openstack-with-netvirt.html>

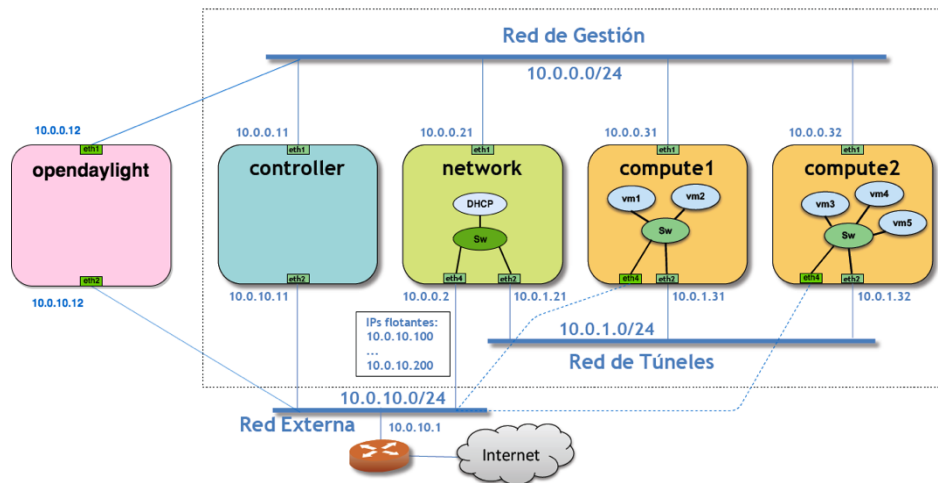


Figura 2: Arquitectura del escenario OpenStack + ODL.

1.2. Iniciar el escenario VNX actualizado y configurarlo:

Arrancar escenario:

```
$ sudo vnx -f openstack_opendaylight_lab.xml -v --create
```

Configurar servicios de OpenStack:

```
$ sudo vnx -f openstack_opendaylight_lab.xml -v -x start-all
```

Cargar las imágenes de VMs en Glance OpenStack:

```
$ sudo vnx -f openstack_opendaylight_lab.xml -v -x load-img
```

Para saber más del escenario base “VNX Openstack Stein four nodes classic scenario using Open vSwitch” y poder descargarlo e instalarlo⁶. Para saber cómo instalar la plataforma de virtualización VNX⁷.

2. Detener y deshabilitar servicios y agentes de Neutron e instalar el *driver* “networking-odl”.

2.1. El siguiente paso es detener los servicios y agentes de Neutron antes de configurar ODL como *Mechanism Driver* del escenario OpenStack. Mientras Neutron administra las instancias de OVS en los nodos de computación, control y red, OpenDayLight y Neutron pueden entrar en conflicto. Para evitar problemas, detener el servidor Neutron en el nodo controlador y deshabilitar los agentes *Open vSwitch* y L3 de Neutron en los nodos de computación y red. OpenDayLight se encargará de proveer y administrar estas funciones por su cuenta como nuevo *Mechanism Driver*.

- En “controller” detener el servidor de Neutron:

```
$ service neutron-server stop
```

⁶ <http://web.dit.upm.es/vnxwiki/index.php/Vnx-labo-openstack-4nodes-classic-ovs-stein>

⁷ <http://web.dit.upm.es/vnxwiki/index.php/Vnx-install>

- En “compute1” y “compute2” detener y deshabilitar los servicios agentes de OVS:

```
$ service neutron-openvswitch-agent stop
$ systemctl disable neutron-openvswitch-agent
```

- En “network” detener y deshabilitar los servicios agentes de OVS y L3:

```
$ service neutron-openvswitch-agent stop
$ systemctl disable neutron-openvswitch-agent
$ service neutron-l3-agent stop
$ systemctl disable neutron-l3-agent
```

- También se detienen momentáneamente los servicios agentes de Neutron para DHCP, metadatos y balanceo de carga como servicio en el nodo de red (“network”):

```
$ service neutron-dhcp-agent stop
$ service neutron-metadata-agent stop
$ service neutron-lbaasv2-agent stop
```

2.2. Para poder configurar ODL como *Mechanism Driver* es necesario instalar la librería de Python “networking-odl” en los nodos que vayan a tener instancias de OVS controladas por el controlador OpenDayLight y en el nodo “controller” que gestiona el servidor de Neutron para que se comuniquen con la API Neutron de OpenDayLight y puedan gestionar la red en conjunto. Por problemas de incompatibilidad de paquetes, es necesario instalar “networking-odl” con la versión del gestor de paquetes de Python pip2 y pip3 en los nodos de computación y red. En el nodo “controller”, habrá que instalar “networking-odl” con pip3. Los pasos son: actualizar repositorios de Ubuntu, instalar “python-pip” y “networking-odl”.

```
$ sudo apt-get update
$ sudo apt-get install -y python-pip
$ pip install networking-odl
$ pip2 install networking-odl
```

3. Configuración del nodo “opendaylight”.

Una vez inicializado el escenario, entrar remotamente en el nuevo nodo “opendaylight” e instalar el controlador ODL en su versión Oxygen-SR4.

- 3.1. Primero instalar JAVA 8 JDK y exportar la variable de entorno “JAVA_HOME”:

```
$ ssh root@opendaylight
$ sudo apt -y install openjdk-8-jdk
$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
```

3.2. Luego proceder a instalar el controlador OpenDayLight, descargando la versión estable Oxygen-SR4 y descomprimiendo el archivo:

```
$ wget https://nexus.opendaylight.org/content/repositories/public/org.opendaylight/integration/karaf/0.8.4/karaf-0.8.4.tar.gz
$ tar -xvfz karaf-0.8.4.tar.gz
$ cd karaf-0.8.4
```

3.3. Configurar el archivo “karaf-0.8.4/etc/custom.properties” con la opción “ovsdb.l3.fwd.enabled = yes” para asegurarse de que el controlador ODL pueda realizar el reenvío y funciones de L3 en lugar del agente L3 de Neutron.

3.4. Desde dentro del directorio principal de la distribución de OpenDayLight, conectarse al cliente Karaf Shell para instalar una serie de complementos (*features* de ODL). Seguidamente, inicializar como servicio el controlador ODL.

Conectarse a Karaf Shell:

```
$ sudo ./bin/karaf
```

Instalar complementos NetVirt para la API Neutron:

```
opendaylight-user@root>feature:install odl-netvirt-api odl-netvirt-openstack
```

Instalar complementos REST:

```
opendaylight-user@root>feature:install odl-mdsal-apidocs odl-restconf-all
```

Instalar complementos para GUI DLUX:

```
opendaylight-user@root>feature:install odl-dlux-core odl-dluxapps-nodes odl-dluxapps-topology odl-dluxapps-yangui
```

Instalar complementos adicionales para gestión de flujos OpenFlow:

```
opendaylight-user@root>feature:install odl-openflowplugin-flow-services odl-openflowplugin-flow-services-rest
```

Instalar complemento adicional para la gestión de *hosts* con OVS:

```
opendaylight-user@root>feature:install odl-neutron-hostconfig-ovs
```

Instalar complementos TSDR para la monitorización de series de datos temporales desde ODL:

```
opendaylight-user@root>feature:install odl-tsdr-core odl-tsdr-hsqldb-all odl-tsdr-netflow-statistics-collector odl-tsdr-syslog-collector odl-tsdr-restconf-collector odl-tsdr-elasticsearch
```

Salir de la sesión con cliente Karaf:

```
opendaylight-user@root>logout
```

Arrancar ODL como servicio

```
$ sudo ./bin/start
```

3.5. Si todo está instalado correctamente, ahora se debería poder iniciar sesión desde el navegador *web* con la interfaz GUI del complemento DLUX de OpenDayLight con la URL “http://opendaylight_MgmtNet_Interface_IP:8181/index.html”, a partir de las credenciales de acceso “*Username/Password: admin/admin*”. Ver la Figura 3.

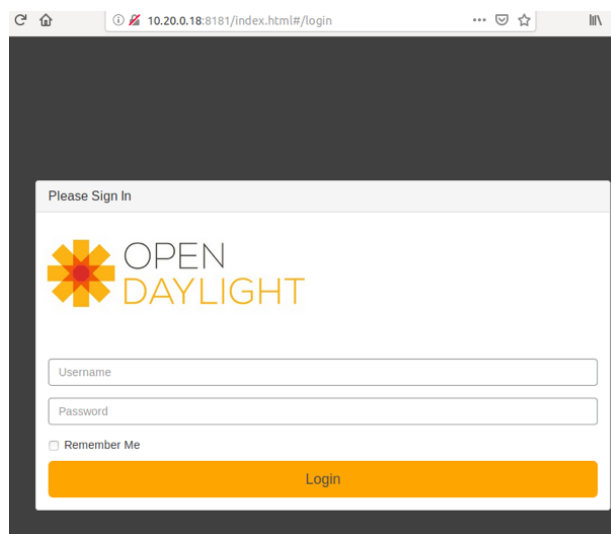


Figura 3: OpenDayLight DLUX GUI.

4. Configuración de los nodos de computación (“compute1” y “compute2”).

Configurar las instancias de *OVSwitches* para ser gestionadas por OpenDayLight. En cada nodo de computación se borrará la configuración de *Open vSwitch* preexistente y se configurará OpenDayLight como orquestador de la red.

4.1. Para ello, en cada uno de estos nodos hay que detener momentáneamente el servicio de *Open vSwitch* y borrar los registros de *logs* y archivos *OVSDb* existentes (OpenDayLight espera administrar completamente los *OVSwitches*). Reiniciar posteriormente el servicio “openvswitch-switch”.

```
$ service openvswitch-switch stop
$ sudo rm -rf /var/log/openvswitch/*
$ sudo rm -rf /etc/openvswitch/conf.db
$ sudo rm -rf /etc/openvswitch/conf.db.~lock~
$ service openvswitch-switch start
```

4.2. En este momento, la configuración de *Open vSwitch* en cada nodo debe estar vacía y tan sólo aparecer el UUID y la versión OVS instalada en cada nodo:

```
$ ovs-vsctl show
9f3b38cb-eeec-4bc7-828b-084b1f66fbfd
    ovs_version: "2.11.0"
```

4.3. Configurar en cada nodo de computación la IP que se utilizará para la conectividad entre hipervisores a través de túneles VXLAN. Esta IP se corresponde con la interfaz física utilizada para la red de túneles (“TunnNet”).

```
# Configurar túnel VXLAN en nodo "compute1":
$ sudo ovs-vsctl set Open_vSwitch . other_config:local_ip=10.0.1.31

# Configurar túnel VXLAN en nodo "compute2":
$ sudo ovs-vsctl set Open_vSwitch . other_config:local_ip=10.0.1.32
```

4.4. Configurar en cada instancia de OVS el mapeo entre la red proveedora OpenStack y la interfaz de salida a la red externa de cada nodo para que las futuras máquinas virtuales alojadas en los hipervisores puedan tener conectividad con la red externa (“ExtNet”).

```
# La interfaz de salida a la red externa (ExtNet) en cada nodo de computación es "eth4".
$ sudo ovs-vsctl set Open_vSwitch other_config:provider_mappings=provider:eth4
```

4.5. Establecer OpenDayLight como administrador de OVS en los nodos. Para ello se abre una conexión TCP con la IP de la interfaz de la red de gestión del nodo “opendaylight” con el puerto TCP 6640 (puerto de OVSDDB). También se abre una conexión local en OVS para que el servicio “nova-compute” pueda “hablar” con *OpenvSwitch*.

```
$ sudo ovs-vsctl set-manager tcp:10.0.0.12:6640 ptcp:6640:127.0.0.1
```

4.6. Configurar los *hosts* de computación para permitir que ODL controle los recursos de nivel L2 gestionados de sus instancias OVS gracias al *driver* “networking-odl” instalado previamente.

```
$ sudo neutron-odl-ovs-hostconfig --noovs_dpdk --ovs_hostconfigs='{"ODL L2":{"allowed_network_types":["local","flat","vlan","vxlan","gre"],"bridge_mappings":{"provider":"eth4"},"supported_vnic_types":[{"vnic_type":"normal","vif_type":"ovs","vif_details":{}}]}}'
```

4.7. Viendo la configuración de cada instancia *Open vSwitch* con el comando “ovs-vsctl show”, se muestra que OVS está conectado al controlador OpenDayLight como gestor a través de OVSDb, y además OpenDayLight crea automáticamente un *bridge* de integración “br-int” que se conecta a través de *OpenFlow* al controlador. Además, crea un puerto en el *bridge* “br-int” conectado al interfaz físico de salida a la red externa. También crea un puerto para la conexión entre los nodos de computación mediante túneles VXLAN. En la Figura 4 se muestra un ejemplo de la salida del comando “ovs-vsctl show” para el nodo “compute1”.

```
[root@compute1:~# ovs-vsctl show
de32965d-f27c-4e87-a4ca-7dae98e67360
  Manager "ptcp:6640:127.0.0.1"
  Manager "tcp:10.0.0.12:6640"
  is_connected: true
Bridge br-int
  Controller "tcp:10.0.0.12:6653"
  is_connected: true
  fail_mode: secure
  Port "tun3674b7560c2"
    Interface "tun3674b7560c2"
      type: vxlan
      options: {key=flow, local_ip="10.0.1.31", remote_ip="10.0.1.32"}
      bfd_status: {diagnostic="No Diagnostic", flap_count="1", forwarding="true", remote_diagnostic="No Diagnostic", remote_state=up, state=up}
  Port "tun53121aef7d0"
    Interface "tun53121aef7d0"
      type: vxlan
      options: {key=flow, local_ip="10.0.1.31", remote_ip="10.0.1.31"}
      bfd_status: {diagnostic="No Diagnostic", flap_count="1", forwarding="true", remote_diagnostic="No Diagnostic", remote_state=up, state=up}
  Port "eth4"
    Interface "eth4"
  Port br-int
    Interface br-int
      type: internal
ovs.version: "2.11.0"
```

Figura 4: Salida del comando "ovs-vsctl show" del nodo “compute1”.

4.8. Asegurarse de que los nodos de computación están conectados a OpenDayLight. Si se vuelve a cargar la GUI DLUX de OpenDayLight, ahora se debería ver que todos los nuevos *bridges Open vSwitch* aparecen en las secciones “Nodes” y “Topology” (ver la Figura 5).

Node Id	Node Name	Node Connectors	Statistics
openflow:202516489569575	None	5	Flows Node Connectors
openflow:234031435379216	None	5	Flows Node Connectors
openflow:246830885410222	None	5	Flows Node Connectors




Figura 5: Paneles Nodes y Topology en GUI DLUX (1).

4.9. Configurar el servicio OpenStack Nova para que OVS sea gestionado internamente desde el nodo de computación local al proveer las máquinas virtuales y crear los puertos que las asocian a la instancia OVS. Para ello, editar el archivo “/etc/nova/nova.conf” y añadir las dos siguientes líneas:

```
[os_vif_ovs]
ovsdb_connection = tcp:127.0.0.1:6640
```


4.10. Justo a continuación, reiniciar el *daemon* del servicio Nova en los nodos de computación:

```
$ service nova-compute restart
```

5. Configuración del nodo “network”.

Configurar la instancia de *OVSwitch* para ser gestionada por OpenDayLight. En el nodo de red borraremos la configuración de *Open vSwitch* preexistente y configuraremos OpenDayLight como orquestador de la red. La configuración será muy similar a la de los nodos de computación, sólo que ahora ODL gestionará, tanto los recursos de L2 como de L3 de OVS como *Mechanism Driver*. Además, hay que configurar el servicio agente DHCP de Neutron para que el nodo de red gestione internamente desde su *bridge* OVS la asignación de direcciones IP a las VMs alojadas en los hipervisores de computación.

5.1. Detener momentáneamente el servicio *Open vSwitch* y borrar los registros de *logs* y archivos OVSDb existentes (OpenDayLight espera administrar completamente los *OVSwitches*). Reiniciar posteriormente el servicio “openvswitch-switch”.

```
$ service openvswitch-switch stop
$ sudo rm -rf /var/log/openvswitch/*
$ sudo rm -rf /etc/openvswitch/conf.db
$ sudo rm -rf /etc/openvswitch/conf.db.~lock~
$ service openvswitch-switch start
```

5.2. En este momento, la configuración de *Open vSwitch* en cada nodo debe estar vacía y tan sólo aparecer el UUID y la versión OVS instalada en cada nodo:

```
$ ovs-vsctl show
9f3b38cb-eeefc-4bc7-828b-084b1f66fbfd
    ovs_version: "2.11.0"
```

5.3. Configurar la IP que se utilizará para la conectividad con los hipervisores a través de túneles VXLAN. Esta IP se corresponde con la interfaz física utilizada para la red de túneles (“TunnNet”).

```
$ sudo ovs-vsctl set Open_vSwitch . other_config:local_ip=10.0.1.21
```

5.4. Configurar en la instancia de OVS el mapeo entre la red proveedora OpenStack y la interfaz de salida a la red externa del nodo para tener conectividad con la red externa (“ExtNet”).

```
# La interfaz de salida a la red externa (ExtNet) en el nodo de red es “eth4”.
$ sudo ovs-vsctl set Open_vSwitch other_config:provider_mappings=provider:eth4
```

5.5. Establecer OpenDayLight como administrador de OVS en el nodo. Para ello se abre una conexión TCP con la IP de gestión del nodo “opendaylight” con el puerto 6640 de OVSDb. También se abre una conexión local en OVS para que el servicio agente “neutron-dhcp-agent” pueda “hablar” con *OpenvSwitch*.

```
$ sudo ovs-vsctl set-manager tcp:10.0.0.12:6640 ptcp:6640:127.0.0.1
```

5.6. Configurar el *host* de red para permitir que ODL controle los recursos de niveles L2 y L3 gestionados de su instancia OVS gracias al *Mechanism Driver* “networking-odl” instalado previamente.

```
$ sudo neutron-odl-ovs-hostconfig --noovs_dpdk --ovs_hostconfigs='{"ODL L2":{"allowed_network_types":["local","flat","vlan","vxlan","gre"],"bridge_mappings": {"provider":"eth4"},"supported_vnic_types": [{"vnic_type":"normal","vif_type":"ovs","vif_details":{}}]}, "ODL L3": {}}'
```

5.7. Viendo la configuración de *Open vSwitch* con el comando “ovs-vsctl show”, se muestra que OVS está conectado al controlador OpenDayLight como gestor a través de OVSDb, y además OpenDayLight crea automáticamente un *bridge* de integración “br-int” que se conecta a través de *OpenFlow* al controlador. Además, crea un puerto en el *bridge* “br-int” conectado al interfaz físico de salida a la red externa. También crea dos puertos para la conexión con los nodos de computación mediante túneles VXLAN. En la Figura 6 se muestra un ejemplo de la salida del comando “ovs-vsctl show” para el nodo “network”.

```
[root@network:~# ovs-vsctl show
b143da43-9d48-4796-9c34-5e9838d99e86
  Manager "ptcp:6640:127.0.0.1"
    is_connected: true
  Manager "tcp:10.0.0.12:6640"
    is_connected: true
  Bridge br-int
    Controller "tcp:10.0.0.12:6653"
      is_connected: true
    fail_mode: secure
    Port "tuna184aff120d"
      Interface "tuna184aff120d"
        type: vxlan
        options: {key=flow, local_ip="10.0.1.21", remote_ip="10.0.1.31"}
        bfd_status: {diagnostic="No Diagnostic", flap_count="1", forwarding="true", remote_diagnostic="No Diagnostic", remote_state=up, state=up}
    Port "tun2384ed9e969"
      Interface "tun2384ed9e969"
        type: vxlan
        options: {key=flow, local_ip="10.0.1.21", remote_ip="10.0.1.21"}
        bfd_status: {diagnostic="No Diagnostic", flap_count="1", forwarding="true", remote_diagnostic="No Diagnostic", remote_state=up, state=up}
    Port "eth4"
      Interface "eth4"
    Port br-int
      Interface br-int
        type: internal
    Port "tuned1fd7b581d"
      Interface "tuned1fd7b581d"
        type: vxlan
        options: {key=flow, local_ip="10.0.1.21", remote_ip="10.0.1.32"}
        bfd_status: {diagnostic="No Diagnostic", flap_count="1", forwarding="true", remote_diagnostic="No Diagnostic", remote_state=up, state=up}
  ovs_version: "2.11.0"
```

Figura 6: Salida del comando "ovs-vsctl show" del nodo “network”.

5.8. Asegurarse de que el nodo de red está conectado a OpenDayLight. Si se vuelve a cargar la GUI DLUX de OpenDayLight, ahora se debería ver que todos los *bridges Open vSwitch* aparecen en las secciones “Nodes” y “Topology” (ver la Figura 7 de la siguiente página).

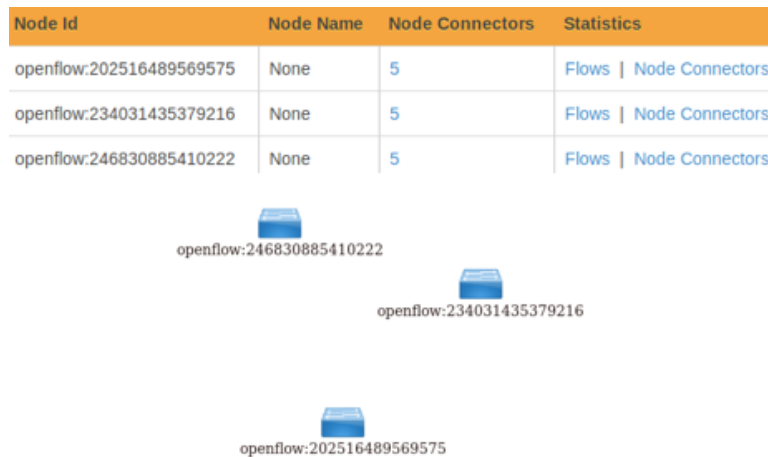


Figura 7: Paneles Nodes y Topology en GUI DLUX (2).

5.9. Configurar el servicio agente DHCP de OpenStack Neutron para que el nodo de red gestione internamente desde su *bridge* OVS la asignación de direcciones IP a las VMs alojadas en los hipervisores de computación. Para ello, editar el archivo “/etc/neutron/dhcp_agent.ini” con las siguientes líneas:

```
interface_driver = openvswitch
force_metadata = true
...
[ovs]
ovsdb_connection = tcp:127.0.0.1:6640
```

5.10. Justo a continuación, reiniciar el *daemon* de los servicios agentes DHCP y metadata de Neutron:

```
$ service neutron-dhcp-agent restart
$ service neutron-metadata-agent restart
```

6. Configuración del nodo controlador OpenStack (“controller”).

Una vez que se hayan configurado las instancias de OVS para conectarse a OpenDayLight, ahora se debe configurar OpenStack Neutron para utilizar OpenDayLight como *backEnd*. Configurar el servidor de Neutron (nodo “controller”) para que ODL actúe de *Mechanism Driver* y pueda gestionar los recursos de red de OpenStack de niveles L2 y L3, usando el *driver* ML2 de OpenDayLight.

6.1. Editar el archivo “/etc/neutron/neutron.conf” para habilitar el *plugin* ML2:

```
[DEFAULT]
core_plugin = neutron.plugins.ml2.plugin.ML2Plugin
```

6.2. Configurar Neutron para usar el complemento de servicio “router-odl_v2” de OpenDayLight para la conectividad y reenvío de L3. Editar mismo archivo “/etc/neutron/neutron.conf”:

```
[DEFAULT]
...
service_plugins = odl-router_v2
...
# Driver to use for scheduling router to a default L3 agent (string value)
router_scheduler_driver = neutron.scheduler.l3_agent_scheduler.LeastRoutersScheduler
# Allow auto scheduling of routers to L3 agent. (boolean value)
router_auto_schedule = true
```

6.3. Configurar el *driver* ML2 de OpenStack. Editar el archivo “/etc/neutron/plugins/ml2/ml2_conf.ini”. Habilitar el *Mechanism Driver* ODL, los *drivers* de tipo de red, los tipos de red de autoservicio (*tenants*) y los *drivers* de extensión de seguridad por puerto y QoS (esto último opcional).

```
[ml2]
type_drivers = local,flat,vlan,vxlan
tenant_network_types = vxlan,flat,vlan
mechanism_drivers = opendaylight_v2
extension_drivers = port_security,qos
```

6.4. Habilitar los grupos de seguridad desde el mismo archivo de configuración “/etc/neutron/plugins/ml2/ml2_conf.ini”:

```
[securitygroup]
enable_security_group = true
# Driver for security groups firewall in the L2 agent (string value)
firewall_driver = neutron.agent.not.a.real.FirewallDriver
```

6.5. Configurar el *Mechanism Driver* ML2 de ODL:

La configuración consiste en crear una nueva sección “[ml2_odl]” en el archivo de configuración “/etc/neutron/plugins/ml2/ml2_conf.ini”, donde se detalle la URL con la que el servicio de Neutron y el *driver* ML2 se comunicarán con ODL mediante la API REST, indicando las credenciales de usuario y *password*. Además, se indica que el servicio DHCP está habilitado y la configuración del *port binding*.

```
[ml2_odl]
enable_dhcp_service = true
port_binding_controller = pseudo-agentdb-binding
```

```
password = admin
username = admin
url = http://10.0.0.12:8181/controller/nb/v2/neutron
```

6.6. Reiniciar la base de datos *mysql* de Neutron con la nueva configuración:

```
$ mysql -u root --password='xxxx' -e "DROP DATABASE neutron;"
$ mysql -u root --password='xxxx' -e "CREATE DATABASE neutron;"
$ mysql -u root --password='xxxx' -e "GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'xxxx';"
$ mysql -u root --password='xxxx' -e "GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'xxxx';"
$ mysql -u root --password='xxxx' -e "flush privileges;"
$ su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
$ neutron-db-manage --subproject networking-odl upgrade head
```

6.7. Reiniciar servicio del servidor de Neutron y los servicios de Nova dependientes:

```
$ service neutron-server restart
$ service nova-api restart
$ service nova-consoleauth restart
$ service nova-scheduler restart
$ service nova-conductor restart
$ service nova-novncproxy restart
```

6.9. Verificar que el servicio de red de Neutron de OpenStack funciona y que se han habilitado todos agentes de red necesarios: DHCP, servicio de metadatos y agentes ODL2 y ODL3 para gestionar los recursos de red de nivel 2 y nivel 3 desde ODL (ver la Figura 8).

```
root@controller:~# openstack network agent list
```

ID	Agent Type	Host	Availability Zone	Alive	State	Binary
13c3db18-fbc3-4c06-9997-625fd7e887db	ODL L2	compute1	None	:-)	UP	neutron-odlagent-portbinding
665b0eb0-5bd8-4d6a-9bcd-be44e094b0c4	ODL L2	network	None	:-)	UP	neutron-odlagent-portbinding
70f2ed23-cab8-44f0-889a-1cc949ee8d53	DHCP agent	network	nova	:-)	UP	neutron-dhcp-agent
72d6d78b-791b-4b86-bfd4-6c477d1cbe0f	ODL L2	compute2	None	:-)	UP	neutron-odlagent-portbinding
93162ecc-0097-4872-a112-51939757d2e3	ODL L3	network	None	:-)	UP	neutron-odlagent-portbinding
a3fdb72c-0169-4278-bf42-e5ea09e4fce6	Metadata agent	network	None	:-)	UP	neutron-metadata-agent

Figura 8: Servicios agentes de OpenStack Neutron.

6.10. Verificar que la interfaz de conexión ML2 de OpenStack con OpenDayLight y la API Neutron de ODL funcionan mediante consulta por la API REST:

```
curl -u admin:admin http://10.0.0.12:8181/controller/nb/v2/neutron/networks
{
  "networks" : [ ]
}
```

7. Configuración del servicio NAT de VNX.

Para configurar NAT en la infraestructura OpenStack desplegada y permitir que las máquinas virtuales que instanciamos tengan salida a Internet, VNX permite hacerlo mediante la ejecución del siguiente comando:

```
$ sudo vnx_config_nat ExtNet <if_externo>
```

Este comando debe ejecutarse desde el *host* anfitrión desde el directorio del escenario VNX desplegado, siendo “ExtNet” la red proveedora externa de OpenStack y “<if_externo>” a interfaz físico del *host* que proporciona la salida a Internet.

8. Ejemplo de creación de escenario virtual OpenStack.

En el siguiente fragmento de código se definen los comandos para la creación de una topología virtual de ejemplo a partir de la CLI de OpenStack, que puede ser probada en el escenario de integración de OpenStack y OpenDayLight desplegado.

```
# Load admin credentials:
source /root/bin/admin-openrc.sh

# Create security group rules to allow ICMP, SSH and WWW access:
openstack security group delete default
openstack security group rule create --proto icmp --dst-port 0 default
openstack security group rule create --proto tcp --dst-port 80 default
openstack security group rule create --proto tcp --dst-port 22 default

# Create internal networks:
openstack network create net0
openstack subnet create --network net0 --gateway 10.1.1.1 --dns-nameserver 8.8.8.8 --sub
net-range 10.1.1.0/24 --allocation-pool start=10.1.1.8,end=10.1.1.100 subnet0
openstack network create net1
openstack subnet create --network net1 --gateway 11.1.1.1 --dns-nameserver 8.8.8.8 --sub
net-range 11.1.1.0/24 --allocation-pool start=11.1.1.8,end=11.1.1.100 subnet1
```

```

# Create external network:

openstack network create --share --external --provider-physical-network provider --provider-network-type flat ExtNet

openstack subnet create --network ExtNet --gateway 10.0.10.1 --dns-nameserver 10.0.10.1
--subnet-range 10.0.10.0/24 --allocation-pool start=10.0.10.100,end=10.0.10.200 ExtSubNet

# Create router:

openstack router create r0
openstack router add subnet r0 subnet0
openstack router add subnet r0 subnet1
openstack router set r0 --external-gateway ExtNet

# Create virtual machines:

mkdir -p /root/keys

openstack keypair create vm1 > /root/keys/vm1
openstack server create --flavor m1.tiny --image cirros-0.3.4-x86_64-vnx vm1 --nic net-id=net0 --key-name vm1

openstack keypair create vm2 > /root/keys/vm2
openstack server create --flavor m1.tiny --image cirros-0.3.4-x86_64-vnx vm2 --nic net-id=net1 --key-name vm2

openstack keypair create vm3 > /root/keys/vm3
openstack server create --flavor m1.smaller --image xenial-server-cloudimg-amd64-vnx vm3 --nic net-id=net0 --key-name vm3

openstack keypair create vm4 > /root/keys/vm4
openstack server create --flavor m1.smaller --image xenial-server-cloudimg-amd64-vnx vm4 --nic net-id=net1 --key-name vm4

# Assign floating IP address to vm1, vm2, vm3 and vm4 openstack servers:

openstack server add floating ip vm1 $( openstack floating ip create ExtNet -c floating_ip_address -f value )

openstack server add floating ip vm2 $( openstack floating ip create ExtNet -c floating_ip_address -f value )

openstack server add floating ip vm3 $( openstack floating ip create ExtNet -c floating_ip_address -f value )

openstack server add floating ip vm4 $( openstack floating ip create ExtNet -c floating_ip_address -f value )

```