IMPLEMENTACIÓN E INTEGRACIÓN DE LA VIRTUALIZACIÓN DE FUNCIONES DE RED, COMPUTACIÓN EN LA NUBE Y LAS REDES DEFINIDAS POR SOFTWARE, PARA LA ADMINISTRACIÓN Y ORQUESTACIÓN DE UNA INFRAESTRUCTURA COMO SERVICIO.

CAROLINA GARZÓN PALACIO
KEVIN ANDRES ATEHORTUA CASTRO

UNIVERSIDAD CATÓLICA DE PEREIRA
FACULTAD DE CIENCIAS BÁSICA E INGENIERÍA
INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES
PEREIRA
2018

IMPLEMENTACIÓN E INTEGRACIÓN DE LA VIRTUALIZACIÓN DE FUNCIONES DE RED, COMPUTACIÓN EN LA NUBE Y LAS REDES DEFINIDAS POR SOFTWARE, PARA LA ADMINISTRACIÓN Y ORQUESTACIÓN DE UNA INFRAESTRUCTURA COMO SERVICIO.

CAROLINA GARZÓN PALACIO
KEVIN ANDRES ATEHORTUA CASTRO

Trabajo de Grado presentado como opción parcial para optar
Al título de Ingeniero de Sistemas y telecomunicaciones

Director
NÉSTOR ALZATE MEJÍA
Magister en Ingeniería de Sistemas y Computación

UNIVERSIDAD CATÓLICA DE PEREIRA
FACULTAD DE CIENCIAS BÁSICA E INGENIERÍA
INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES
PEREIRA
2018

PÁGINA DE ACEPTACIÓN

_____
_____
_____
_____
_____
_____
_____

_____
<NOMBRE COMPLETO>
JURADO

_____
<NOMBRE COMPLETO>
JURADO

_____
<NOMBRE COMPLETO>
JURADO

Pereira, 29 de mayo de 2018

CONTENIDO

LISTA DE ILUSTRACIONES

LISTA DE TABLAS

RESUMEN

Las funciones de red actualmente están ligadas a la capacidad de hardware propietario, lo que ha causado que los proveedores de servicio de internet tengan incrementos en los precios para su despliegue, implicando el uso de más personal capacitado para el manejo del hardware implementado. Por consiguiente se desplegó una infraestructura como servicio (IaaS) en un entorno de laboratorio para implementar al menos dos funciones de red utilizando el paradigma de virtualización de funciones de red, en un ambiente de trabajo con redes definidas por software y computación en la nube, por medio de las plataformas de Open Source MANO (OSM), OpenDaylight (ODL) y OpenStack; teniendo como resultado un despliegue de funciones de red de una manera virtual y quitando la dependencia de hardware propietario. Para la elaboración de este proyecto se utilizó una metodología de paradigma mixto puesto que se inicia realizando una investigación para conocer las herramientas que se van a utilizar, en una segunda fase se realiza la implementación de cada una de ellas, posteriormente se procede a integrar cada una de las plataformas, para finalmente desplegar los servicios de red.

De esta manera se pudo observar que a pesar de que cada una de las plataformas implementadas no necesariamente requieren de las otras para funcionar, al integrarlas se observa como las características de cada una de ellas aportan para tener una infraestructura más robusta, con características que inicialmente no se tendrán al momento de trabajar las funciones de red directamente desde el hardware propietario o serían muy costosas de implementar ya sea en un entorno comercial o educativo.

Palabras clave --- Computación en la nube, redes definidas por software, virtualización de funciones de red

ABSTRACT

Network functions are currently linked to the capacity of proprietary hardware, which has caused Internet service providers to increase the prices for their deployment, implying the use of more trained personnel to manage the hardware implemented. Therefore, an infrastructure as a service (IaaS) was deployed in a laboratory environment to implement at least two network functions using the virtualization paradigm of network functions, in a work environment with software-defined networks and cloud computing, through the Open Source platforms MANO (OSM), OpenDaylight (ODL) and OpenStack; resulting in a deployment of network functions in a virtual way and removing dependency on proprietary hardware. For the elaboration of this project, a mixed paradigm methodology was used since it starts with an investigation to know the tools that will be used, in a second phase the implementation of each of them is carried out, then each of them is integrated one of the platforms, to finally deploy the network services.

Likewise, it was observed that although each of the platforms implemented does not necessarily require the others to function, when integrating them, it is observed how the characteristics of each of them contribute to have a more robust infrastructure, with characteristics that initially do not they will have at the moment of working the network functions directly from the proprietary hardware or they would be very expensive to implement either in a commercial or educational environment.

Descriptors --- Cloud computing, software-defined networking, network functions virtualization

INTRODUCCIÓN

La virtualización de funciones de red (NFV), la computación en la nube, y las redes definidas por software (SDN), son conceptos encargados de manejar diferentes tipos de abstracciones, ya sean de funciones (NFV), computacionales (Computación en la nube) o de red (SDN). Aunque las ventajas de cada una de estos tipos de tecnologías son similares (Agilidad, reducción de costos, automatización, escalabilidad, dinamismo), no dependen unos de otros para ser implementadas, sin embargo al realizar la integración de estas tres tecnologías aportan un valor agregado al despliegue de nuevos servicios de red. Es por esto que este proyecto busca, a través de la plataforma Open Source MANO (OSM), integrar la virtualización de funciones de red, computación en la nube y las redes definidas por software, con el fin de proveer funciones de red a diferentes usuarios, sin necesidad de adquirir hardware propietario y específico para esta función, además de tener un mejor rendimiento, disponibilidad, simplificar la compatibilidad con implementaciones existentes y facilitar la operación, mantenimiento y administración de estas funciones de una forma centralizada.

Es por esto que la investigación de la virtualización tiene gran impacto tanto en el mundo académico como en la industria. En academia, el interés surge por los desafíos de investigación conceptuales, operacionales y procedimentales que estas tecnologías presentan. En la industria, el interés particular es el ahorro en costos de capital (CapEx) y operacionales (OpEx) debido a que la gestión de recursos de manera centralizada, brinda facilidad al momento de identificar los servicios que se han puesto en marcha, teniendo una sola sintaxis de manejo en el sistema, siendo más sencillo de manejar que cuando el hardware se encuentra de manera individual.

1. SITUACIÓN PROBLEMÁTICA

Con el crecimiento de las nuevas tecnologías, los proveedores de servicio tienen la necesidad de acelerar el despliegue de funciones de red, donde es requerida la instalación de hardware propietario, lo que implica costos de adquisición e instalación. Además, los proveedores de servicios no sólo se enfrentan a la dificultad de comprar nuevos equipos, sino que deben tener en cuenta los costos de la planta física como el espacio o el consumo de energía, adicionalmente deben tener presente el mercado competitivo, para contratar personal calificado con capacidades de diseñar, integrar y operar una infraestructura de hardware más compleja. Por estos motivos se aumenta el tiempo de implementación y se limita la innovación en la industria de las telecomunicaciones. Por lo tanto, lo que se pretende es minimizar o eliminar la dependencia de hardware propietario y centralizar las funciones de red, y que dichas funciones estén disponibles desde cualquier sitio gracias a la computación en la nube. Adicionalmente, se busca realizar la implementación integrando las SDN, teniendo como objetivo que estas funciones de red tengan un mayor rendimiento, compatibilidad y mantenimiento de forma más eficiente.

2. OBJETO DE ESTUDIO

Este proyecto pretende implementar e integrar, en un entorno de laboratorio a escala, la virtualización de funciones de red, redes definidas por software y computación en la nube, con la finalidad de comprobar que la integración de estos tres paradigmas cumple con la solución planteada por el Instituto Europeo de Normas de Telecomunicaciones (ETSI) de reducir el tiempo y los costos al momento de desplegar una nueva función de red.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Implementar e integrar la virtualización de funciones de red, computación en la nube y las redes definidas por software, para la administración y orquestación de una infraestructura como servicio.

3.2. OBJETIVOS ESPECÍFICOS

- Examinar y realizar la implementación de la plataforma Open Source MANO (OSM) para el despliegue de funciones de red virtualizadas.
- Proporcionar un ambiente de computación en la nube y redes definidas por software por medio de OpenStack y Opendaylight para preparar los recursos que se orquestaran con NFV.
- Integrar las plataformas implementadas para la administración y orquestación de la infraestructura.
- Virtualizar dos funciones de red para demostrar la orquestación de computación en la nube y las redes definidas por software por medio de la plataforma OSM.

4. JUSTIFICACIÓN

En la actualidad, la virtualización ha sido uno de los avances que ha tenido mayor atención para los investigadores y la industria de telecomunicaciones, debido a que esta abre puertas que generalmente estarían cerradas a la hora de resolver problemas relacionados con funciones, que son muy costosas de generar al sólo contar con equipos físicos. Es por esto que al virtualizar funciones de red los costos de capital (CapEx) y operación (OpEx) se reducen, siendo más ágil la implementación de estas funciones, ya sean: NAT (Network Address Translation), DNS (Domain Network Service), Firewall, quitando así las limitaciones de un equipo físico.

Este proyecto se dirige al ámbito investigativo, documentando el proceso de implementación y virtualización de servicios de red, aportando a futuras investigaciones información que pueda ser tomada para ser aplicada a la industria y en lo académico.

5. DELIMITACIÓN Y/O ALCANCES DEL PROYECTO

En este proyecto se busca analizar la documentación acerca de la orquestación de funciones de red utilizando computación en la nube y redes definidas por software, con el fin de aportar y elaborar documentación acerca del tema abarcado para aportar a investigaciones futuras.

Como objeto de investigación se pretende realizar la implementación de al menos dos funciones de red virtualizadas, utilizando las tecnologías de Open Source Mano, OpenStack y OpenDaylight sobre el sistema operativo Ubuntu 16.04. Proponiendo una infraestructura a escala para orquestar estas funciones de red sobre una topología simple que se definirá a lo largo de la investigación.

6. MARCO TEÓRICO

6.1. ANTECEDENTES

Patel, et al. [1] mencionan que la integración de funciones de red y redes definidas por software en la nube les dan una mejora a los servicios de red y seguridad, describiendo como integrar NFV y SDN en una nube OpenStack para mitigar ataques que pueda sufrir la red mejorando los servicios y sacando ventaja de las SDN.

Mijumbi, et al. [2] presentan una descripción detallada de los servicios SDN / NFV que se ofrecen sobre una plataforma Cloud Computing y transporte de red mediante un Testbed llamado ADRENALINE. Por un lado, proponen una arquitectura genérica para los servicios SDN / NFV desplegados en un multidominio de Redes de transporte y centros de datos distribuidos. Por otro lado, presentan dos casos de uso de posibles servicios NFV: un Virtual Path Computation Element (vPCE) y el despliegue de controladores virtuales SDN (vSDN) sobre el transporte virtualizado de Redes.

Xilouris, et al. [3] describen una arquitectura integrada, diseñada y desarrollada bajo el proyecto T-NOVA, permitiéndoles a los operadores implementar funciones de red virtualizadas según las necesidades propias, además tener los dispositivos de red de forma virtual, proporcionándoles un servicio bajo demanda, eliminando la necesidad de adquirir, instalar y mantener hardware especializado. Así mismo, introducen un “Mercado NFV”, en donde se podrán encontrar servicios y funciones de red, los cuales podrán ser adquiridos e instanciados bajo demanda.

Manzalini and Crespi [4] proponen el término de infraestructura definida por software (SDI) que se integraría a partir de NFV, SDN y Cloud para hacer entornos dinámicos que se controlen como aplicaciones de software, con el fin de ejecutar cualquier función de red, que tiene como objeto proponer: “el modelo del Sistema Operativo Universal (UOS) para SDIs como un Sistema Operativo global y distribuido, que abarca desde terminales de elementos de red, recursos Cloud / IT”.

6.2. MARCO CONCEPTUAL

En esta sección se describirán conceptos relacionados con la virtualización de funciones de red, la computación en la nube y las redes definidas por software, con la finalidad de crear un entorno, para posteriormente ver el proceso de integración de las tres tecnologías.

6.2.1. Virtualización de funciones de red (NFV)

La virtualización de funciones de red (NFV), es un nuevo paradigma para diseñar, implementar y gestionar servicios de red. Permite desacoplar las funciones de red desde los dispositivos de hardware propietario, para que puedan ejecutarse como un software en un servidor estándar [5], como se aprecia en la ilustración 1.
NFV surge a finales del año 2012, por la necesidad de acelerar el despliegue de nuevos servicios de red, debido a que los proveedores de servicios de telecomunicaciones encontraron que los dispositivos basados en hardware limitaban su capacidad para alcanzar este despliegue, puesto que debían adquirir nuevos equipos, generando costos de adquisición, instalación, operación y locación. Debido a esto, formaron un grupo específico de NFV, dentro del Instituto Europeo de Normas de Telecomunicaciones (ETSI) [6], con el objetivo de generar una serie de recomendaciones para implementar NFV de una forma rápida [7] .

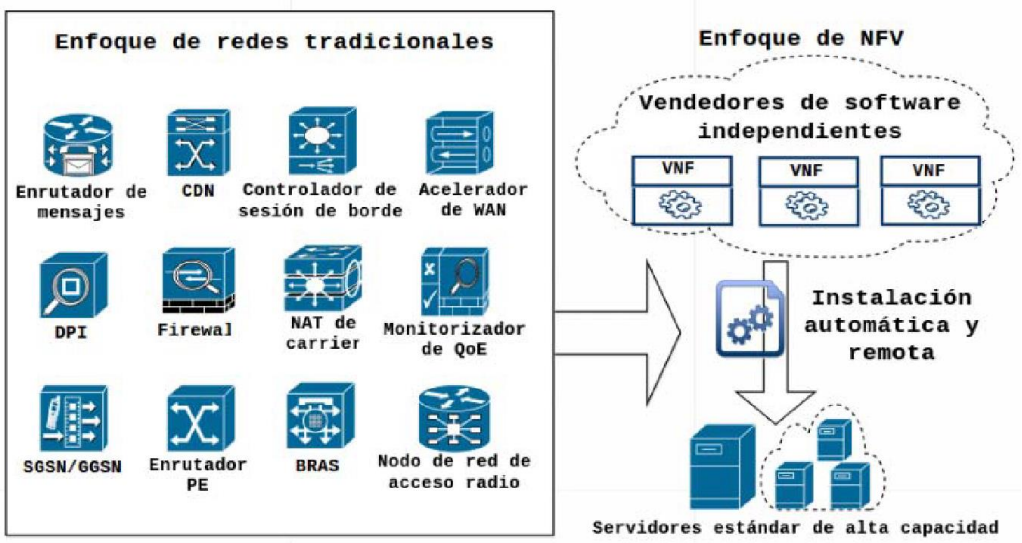

Ilustración 1: Descripción funcionamiento NFV [7]

- **Beneficios de NFV**. NFV se encarga de virtualizar servicios de red a través de software brindándoles a los operadores de red una variedad de beneficios,

puntualizados a continuación:

- ✓ Reduce los costos de capital (CapEx), debido a que disminuye la necesidad de comparar Hardware diseñado específicamente para un servicio.
- ✓ Reduce los costos de operación (OpEx), puesto que reduce los requisitos de una planta física (espacio, energía, refrigeración) y simplifica el despliegue y la gestión de los servicios de red.
- ✓ Acelera el tiempo de salida al mercado, porque minimiza el tiempo necesario para implementar nuevos servicios de red, además reduce los riesgos asociados al momento de desplegar nuevos servicios, porque le ofrece a los proveedores de servicios la capacidad de probar y desarrollar servicios para determinar si realmente satisface las necesidades de los clientes.
- ✓ Aporta agilidad y flexibilidad, debido a que los servicios se pueden ampliar o modificar rápidamente para satisfacer las demandas cambiantes, además las NFV aportan a la innovación dado que, permite que los servicios sean entregados a través de software a cualquier Hardware de un servidor estándar.

- **Arquitectura NFV.** Comúnmente, existen tres componentes principales en una arquitectura NFV de alto nivel (ilustración 2), entre los cuales se pueden encontrar la función de red virtualizada (VNFs), la infraestructura NFV (NFVI) y la orquestación y gestión de NFV (MANO).

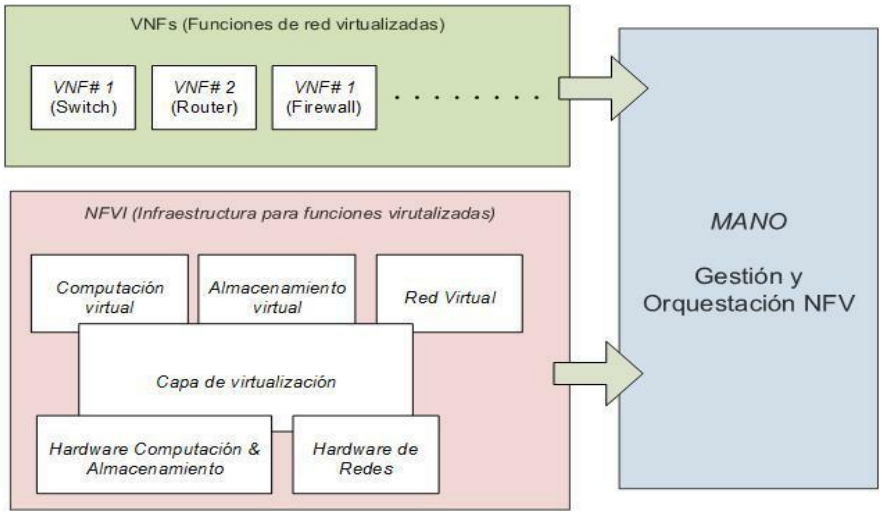

Ilustración 2: Arquitectura NFV [8]

- ✓ **Infraestructura NFV (NFVI).** Son todos los recursos tanto de Hardware como de software que componen un ambiente NFV. Este

componente es el encargado de brindar la conectividad de red entre diferentes ubicaciones, además se basa en elementos de computación estandarizados, ampliamente disponibles y de bajo costo, trabajando así, con servidores virtuales, hipervisores, máquinas virtuales y administradores de infraestructura virtual para habilitar la capa física y virtual de la red.

En esta capa de la arquitectura se encuentra el NFVI (Hardware, sistemas operativos, hypervisor) y el VIM (Administra los recursos del NFVI) tal como se aprecia en la ilustración 3, estos componentes interactúan con la funciones de red virtualizadas (VNFs), es decir, la interfaz de VNFs a NFVI (Vn-Nf) se encarga de formar un canal por el cual se transmite todo el tráfico de la red, mientras que la interfaz que va desde NFVI a VIM (Nf-Vi) forma una ruta de control, la cual se utiliza exclusivamente para la gestión [8]. El VIM se encarga de todos los aspectos operativos como logs, métricas, alertas, cumplimiento de políticas, aseguramiento de servicios, además de que interactúa con la capa de orquestación y el controlador SDN.

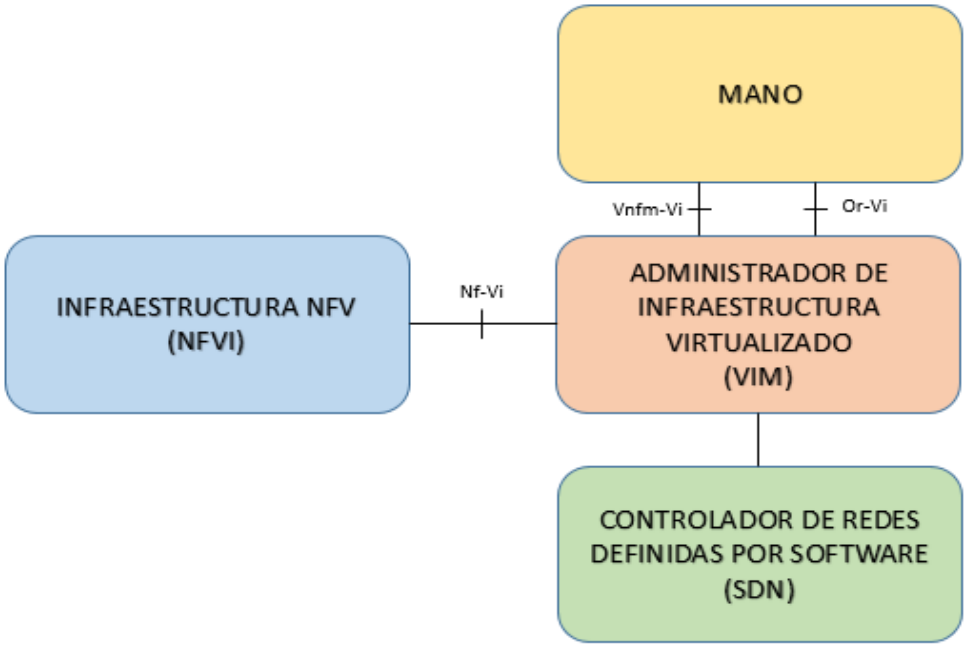

Ilustración 3: VIM con NFV [8]

NFV está compuesta por tres capas: Infraestructura física, capa de virtualización y la infraestructura virtual.

- Infraestructura Física: Se compone del hardware de cómputo, almacenamiento y red.
  Como se dijo anteriormente, el objetivo de NFV es hacer una transición de las funciones de red implementadas en hardware propietario a servidores estándar de la industria. Al existir una gran variedad en este tipo de servidores, supone un problema elegir cual utilizar. Actualmente los servidores que utilizan son servidores rack que funcionan para infraestructura tipo nube, cuya ventaja es que tiene menor necesidad de procesos de entrada y salida. La elección del hardware a utilizar se debe realizar con cuidado puesto que se debe evitar la subutilización de este, teniendo en cuenta que lo importante para implementar NFV es el procesamiento de paquetes y la lógica de aplicación (Procesamiento y red), y es por eso que la ETSI realiza algunas recomendaciones de hardware para un entorno de producción. Estas especificaciones se describen en la tabla 1.

Tabla 1: Especificaciones Recomendadas

Componentes	Especificación
Procesador	Dual-Socket Intel Xeon ES-2600 v4/v3
Memoria RAM	192-256 GB DDR4
Disco Duro	6 x 4 TB 7.5k RPM SATA Drives
Tarjeta de Red	2 x 10 GE
Interfaz administración	Controlador de administración de la placa base (BMC)

- Capa de virtualización: Esta capa se encuentra en la parte superior del hardware y es una plataforma de software que implica un hypervisor [8].
  - Un hypervisor es una herramienta que permite dividir los recursos de la maquina física, para crear máquinas virtuales (VM) emulando todos los periféricos necesarios (ilustración 4). Los hipervisores más utilizados son VMware, vShpere y KVM. VShpere es un hypervisor desarrollado por VMware, este hypervisor se conoce como tipo 1 o bare-metal, es decir, se ejecuta directamente en el hardware [9]. Por otro lado, KVM, es

un hypervisor de código abierto de tipo 2 o hosted, se ejecuta en la parte superior de un sistema operativo Linux [10].

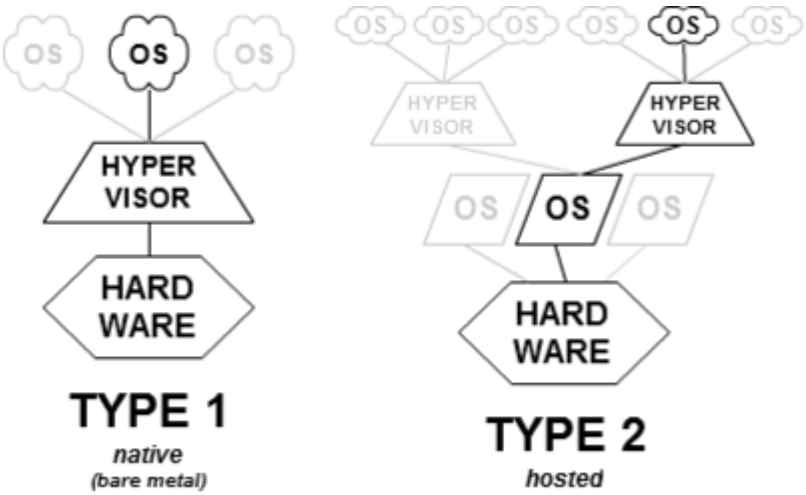

Ilustración 4: Tipos de Hipervisores [11]

- ❖ Infraestructura Virtual: Esta capa está compuesta por las máquinas virtuales, almacenamiento virtual y redes virtuales.
  - Máquinas virtuales (VM): Son creadas, destruidas, migradas y administradas por el hypervisor. En estas máquinas virtuales están alojadas las funciones de red (VNFs)
  - Almacenamiento virtual: El software SAN, NAS o SDS es el encargado de virtualizar el almacenamiento por bloques o archivos, con la finalidad de crear instantáneas, copias de seguridad, replicación, entre otros.
  - Redes Virtuales: Es una función que posee el hypervisor, debido a que permite enviar paquetes desde diferentes máquinas virtuales sin necesidad de acceder a un switch externo, además proporciona servicios de red overlay donde las redes son expuestas, también se encarga de habilitar una pasarela para brindar conectividad hacia internet.

- ✓ **Función de red virtualizada (VNFs):** Son instancias responsables de

manejar las funciones de red específicas que se ejecutan en una o múltiples máquinas virtuales. Las VNFs se pueden conectar o combinarse como bloques de construcción para ofrecer un servicio de comunicación de red a gran escala, puesto que una sola función de red virtualizada no aporta una solución de NFV.

- ✓ **Gestión y Orquestación de NFV (MANO):** Es el encargado de dirigir, gestionar y automatizar el servicio de red de extremo a extremo, el cual es proporcionado por una cadena de VNFs. Generalmente hay un solo orquestador que supervisa la creación de un servicio de red. MANO está compuesta por diferentes elementos, como lo son: Gestión de recursos VIM y NFVI, Administrador de VNF (VNFM), Orquestador NFV (NFVO). Estos elementos se muestran en la Ilustración 5.

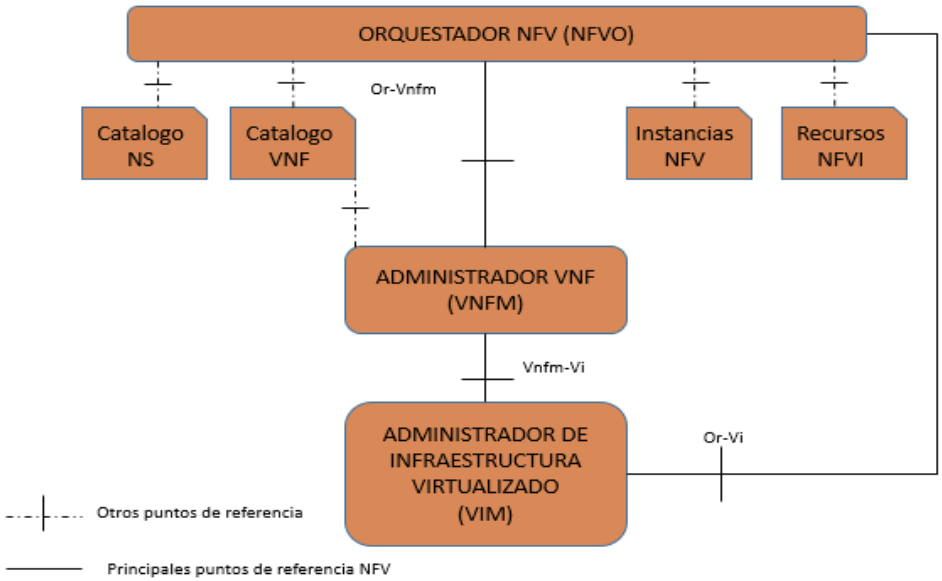

Ilustración 5: Componentes MANO [11]

  - ❖ Gestión de Recursos VIM y NFVI: Como ya se mencionó anteriormente, VIM es el encargado de administrar y controlar los recursos tanto de computación, como de almacenamiento y red del NFVI.
    - ❖ Administrador de VNF (VNFM): Es el encargado de gestionar el ciclo de vida de los VNF, mediante VIM. El VNFM instalada, escala, actualiza, termina las VNFs.
    - ❖ Orquestador NFV (NFVO): Realiza dos funciones principales, incluyendo orquestación de recursos y

orquestación de servicios de red, así como otras funciones [11]. La orquestación de recursos asegura que existan los recursos suficientes de computación, almacenamiento y red, para proporcionar un servicio de red. El NFVO trabaja directamente con el VIM debido a que puede coordinar, autorizar, liberar y contratar los recursos del NFVI.
La NFVO crea servicios de extremo a extremo entre diferentes VNF, con el fin de proporcionar una orquestación de servicios.

Desde que la ETSI dio a conocer el modelo para NFV, algunos proveedores han desarrollados sus propios proyectos de código abierto para la gestión y orquestación de la virtualización de funciones de red (NFV), entre ellas se pueden encontrar la plataforma abierta para proyectos NFV (OPNFV), T-NOVA, CloudNFV, CloudBand, OSM.

6.2.2. Open Source MANO (OSM)

Es un proyecto de código abierto, publicado bajo la licencia de apache 2, creado por telefónica y ahora alojado por la ETSI. OSM incluye un open Source VIM (OpenVIM), permitiendo tener a OSM sin un VIM como OpenStack ya presente [11]. Esta plataforma de código abierto ofrece ser un componente clave para pruebas de laboratorio y de campo, pruebas de interoperabilidad y escalabilidad para funciones de red virtualizadas, entre su facilidad de uso permite una rápida instalación en diferentes sistemas además de mejorar la interoperabilidad con las funciones de red virtualizadas, las VIMs y los controladores de SDN [12]
Como se muestra en la Ilustración 6 se puede observar la arquitectura general y componentes de la plataforma.

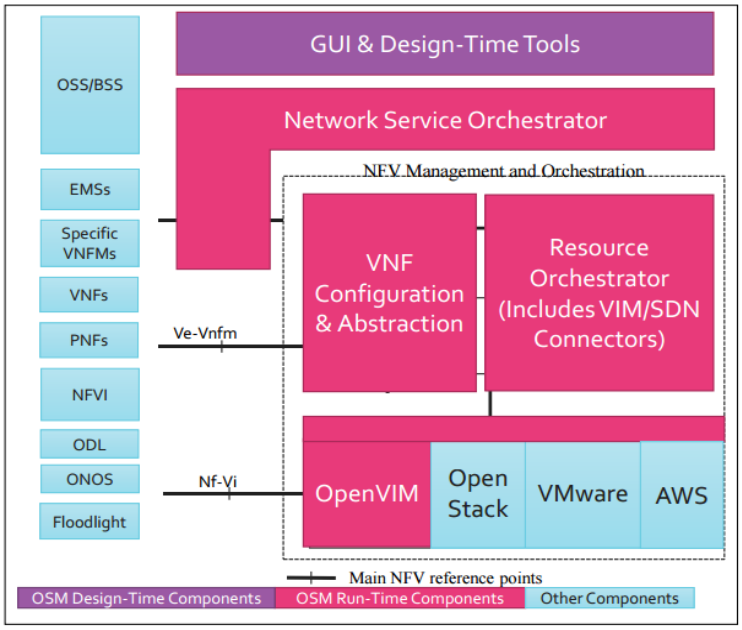

Ilustración 6: Arquitectura y composición de OSM [13]

OSM es una combinación de la orquestación de RIFT.io y Juju, las cuales interactúan como un VNFM.

- ✓ RIFT.io: Es una compañía que se fundó en 2014 y ha aportado a la plataforma OSM una interfaz gráfica de usuario, herramientas de automatización y la organización de los servicios de red
- ✓ Juju: Es una herramienta que se encarga de gestionar la orquestación de servicios de red, abstrayendo el servicio de la máquina o servidor. Permite activar o desactivar escaladamente los servicios a través de un comundo.

- **Objetivos de OSM.** La comunidad de OSM ha definido un amplio objetivo que cubre dos aspectos principales los cuales son el diseño y el tiempo de ejecución para el despliegue de servicios.
  Dentro de los objetivos para el tiempo de ejecución se encuentran:
  - ✓ Un entorno de orquestación de servicios automatizados que permite y simplifica las consideraciones operacionales de las fases del ciclo de vida de los servicios basados en NFV.
  - ✓ Un súper conjunto entre ETSI NFV y MANO donde se incluye aparte de la orquestación de servicios un control explícito de SDNs.
  - ✓ Despliegue de un modelo para la integración de múltiples controladores SDN.
  - ✓ Despliegue de un modelo que integra múltiples VIMS incluyendo una nube basada en estas.

- ✓ Una VIM que ha sido optimizadas por Enhanced Plataform Awareness (EPA) para el despliegue de funciones de red virtualizadas de alto rendimiento.
- ✓ Una VNFM (Virtual Network Function Machine) genérica con soporte para integrar VNFMs específicas.
- ✓ Soporte para integrar funciones de red físicas en un servicio de red automatizado.
- ✓ Clientes GUI, CLI y REST para permitir el acceso a todas las funciones.

En cuanto a los objetivos de diseño se tienen:

- ✓ Una capacidad de operaciones de Crear/Leer/Actualizar/Eliminar (CRUD) en la definición de servicios.
- ✓ Soporte de un entorno de modelado con modelos de datos que van de la mano con ETSI, NFV y MANO
- ✓ Simplificación en la generación de paquetes VNF.
- ✓ Una interfaz gráfica para acelerar el diseño de los servicios de red.

La ETSI define a esta plataforma como “Open Source Management and Orchestration”, la cual es una plataforma para el despliegue de funciones de red virtualizadas, con el objetivo de ser un productor de escala global para este tipo de soluciones.
Esta plataforma en cuanto a su arquitectura lógica se compone 3 componentes principales como se muestra en la Ilustración 7.

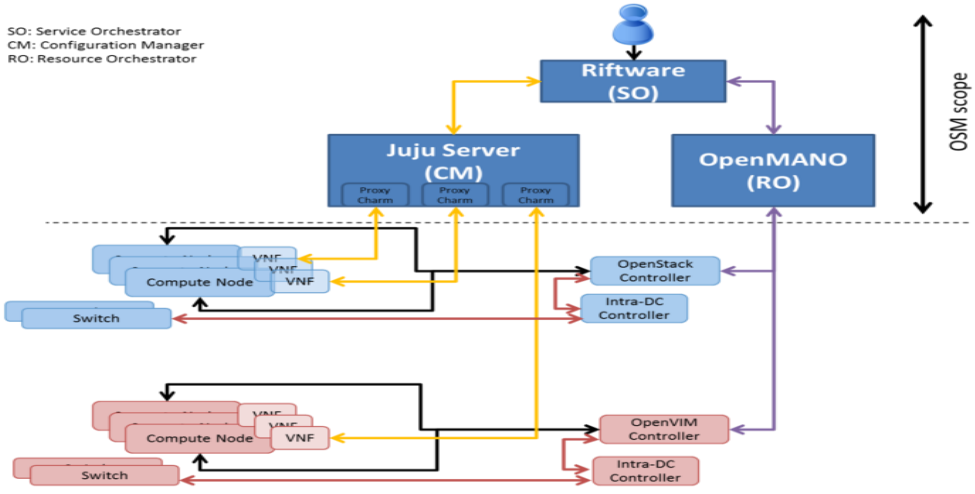

Ilustración 7: Arquitectura Lógica OSM [13]

- ✓ **Service Orchestration (SO):** La orquestación de servicios se basa en la automatización, administración y coordinación de algún servicio determinado. En esta orquestación se usa generalmente un Riftware que es un software que permite hacer virtualización de servicios.
- ✓ **Resource Orchestration (RO):** La orquestación de recursos tiene el mismo principio de SO su diferencia es que en vez de servicios como Firewalls, DPIs, Nats, este haría su administración sobre los recursos de red tales como Servidores y VIMs. Un ejemplo de esto es la plataforma OpenMano.
- ✓ **Configuration Manager (CM, actualizada a VCA):** Los administradores de configuración hace "sencilla" la experiencia debido a que ayudan a instalar paquetes necesarios o software de terceros, Juju server es un ejemplo de esto.

En contexto la plataforma MANO divide su estructura basado en las tecnologías usadas como se describe en la Ilustración 8, allí se puede observar que esta plataforma consta de varios paquetes; VNF package que trae todo lo que respecta a virtualización de funciones, un Data Model Translator que funciona como un controlador y finalmente la plataforma desarrollada por la ETSI que es la que contiene el Resource Orchestration (RO) y el configuration Manager (CM) (JuJu charm y OpenMano).

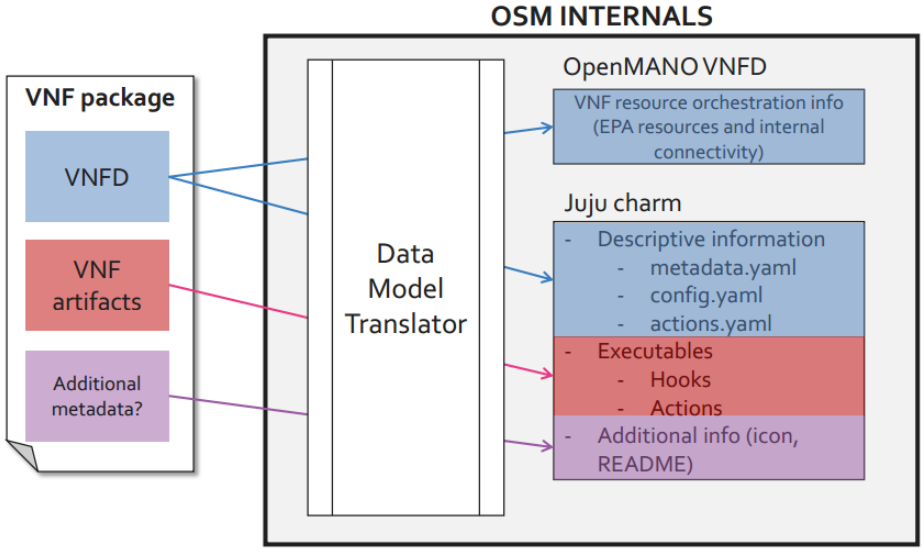

Ilustración 8: Arquitectura de funcionamiento de OpenMano [14]

6.2.3. Computación en la nube

La computación en la nube es un paradigma que se basa en ofrecer servicios de computación a través de una red (Internet), además de proveer de forma remota y escalable los recursos necesarios, tales como servidores, almacenamiento, aplicaciones, servicios [13]. El objetivo de la computación en la nube es permitir a los usuarios finales hacer uso de una infraestructura tecnológica sin la necesidad de tener conocimientos o control sobre esta, siendo una experiencia sencilla y cómoda, lo anterior se puede observar en la ilustración 9.

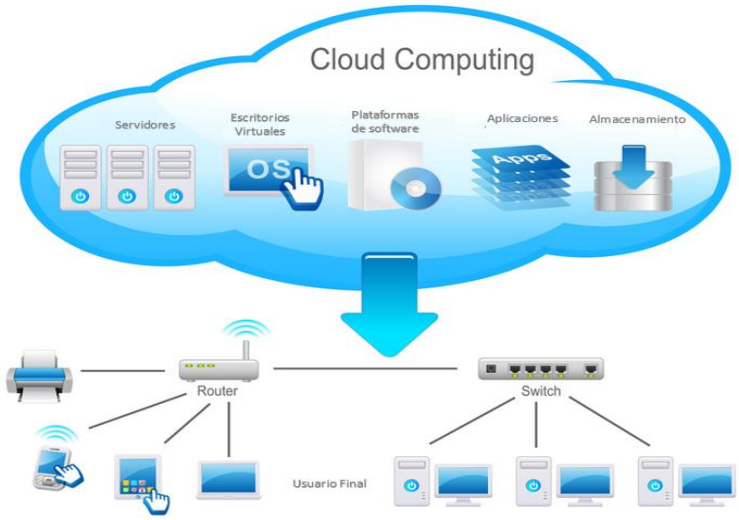

Ilustración 9: Computación en la nube [14]

El modelo de computación en la nube está compuesto por cinco características fundamentales, además de tres modelos de servicio y cuatro modelos de despliegues [14].

- **Características fundamentales.**
  - ✓ **Servicio bajo demanda:** El usuario tiene la capacidad de obtener los recursos (almacenamiento, memoria, servicios de red) de forma automática, sin ayuda del proveedor de servicios.
  - ✓ **Amplio acceso a la red:** Se da debido a que permite que los servicios estén disponibles a través de la red y se puedan acceder a ellos desde cualquier dispositivo.
  - ✓ **Agrupación de recursos:** Los diferentes recursos físicos y virtuales del proveedor son agrupados para que puedan servir a múltiples usuarios. Estos recursos se asignan de forma dinámica según la demanda del usuario, además se genera una sensación de

independencia al no conocer la ubicación exacta de los recursos brindados.

- ✓ **Rápida elasticidad:** Los recursos ofrecidos pueden crecer o disminuir de acuerdo a las necesidades o requerimientos de los usuarios, de una forma rápida y automática.
- ✓ **Medición del servicio:** Los sistemas de computación en la nube, controlan y optimizan automáticamente el uso de los recursos, por medio de la medición apropiada con respecto al servicio ofrecido. La utilización de los recursos pueden ser monitoreados, controlados y reportados.

- **Modelos de servicios.** La computación en la nube al tener una arquitectura basada en la separación del hardware, plataforma y aplicaciones tiene la capacidad de ofrecer diferentes modelos de servicios (ilustración 10).

  - ✓ **Software como Servicio (SaaS):** Se le proporciona al usuario final las aplicaciones solicitadas mediante una infraestructura en la nube. El usuario no controla la infraestructura y puede acceder a los servicios mediante diferentes dispositivos, por medio de una interfaz de cliente ligero (por ejemplo, un navegador web). Algunos ejemplos de SaaS son: Gmail, Google Docs, Dropbox.
  - ✓ **Plataforma como Servicio (PaaS):** Permite a los usuarios crear o desplegar aplicaciones por medio de lenguajes de programación, bibliotecas, servicios y herramientas soportadas por el proveedor. A pesar de que el usuario tiene control sobre las aplicaciones y sobre la configuración del entorno para hospedarlas, no tiene la capacidad de controlar la infraestructura de la nube [15]. En la categoría de PaaS se puede encontrar: Google App Engine, Heroku, Windows Azure.
  - ✓ **Infraestructura como Servicio (IaaS):** El usuario tiene la capacidad de controlar el procesamiento, almacenamiento, redes y otros recursos fundamentales de la computación, además puede ejecutar software como sistemas operativos y aplicaciones. Sin embargo, la capacidad de control sobre la nube es limitado, dado que el usuario no gestiona componentes internos de la nube. Entre los proveedores IaaS se pueden encontrar: Juju, OpenStack, IBM SoftLayer, vCloud.
  - ✓ **Metal como Servicio (MaaS):** Es una mejora de los servicios IaaS, pues permite a los usuarios utilizar los servidores físicos como máquinas virtuales en la nube, sin tener que administrar cada servidor de forma individual.

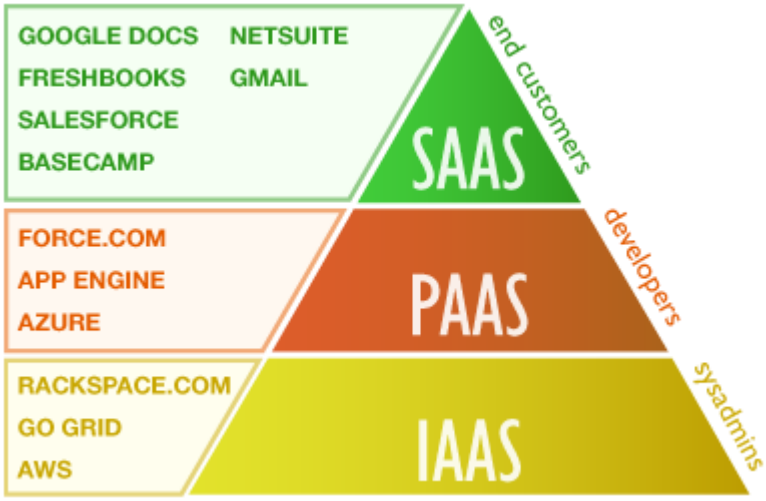

Ilustración 10: Modelos de servicio [16]

- **Modelos de implementación.** Según los requerimientos de los usuarios, es posible implementar diferentes modelos.

  - ✓ Nubes públicas: Es un servicio que permite a una gran cantidad de usuarios utilizarlo de forma simultánea pero independiente, vendiendo el servicio a través de internet. El proveedor de servicio es el encargado de administrar el mantenimiento, seguridad, flexibilidad y escalabilidad para todos los usuarios de esa nube.
  - ✓ Nubes comunitarias: Se desarrollan cuando un grupo de usuarios en específico tienen determinados requisitos o necesidades en cuanto a cuestiones de seguridad o tipo de aplicaciones, es así como diferentes empresas agrupan sus recursos para brindar una solución a un problema compartido.
  - ✓ Nubes privadas: Se da cuando existe una organización con su propia nube de servidores y software para utilizar sin un punto de acceso público. Este tipo de nubes es gestionada por la propia empresa u organización.
  - ✓ Nubes híbridas: En este tipo de nube se puede encontrar dos o más modelos de implementación (Pública, privada o comunitaria), los cuales permanecen como entidades únicas unidas con tecnologías diferentes.

6.2.4. OpenStack

Es una plataforma de código libre, enfocada para trabajar la computación en la nube como una infraestructura como servicio (IaaS), implementando diferentes tipos de nubes (Pública, privadas e hibridas), de una forma simple y escalable [16].
OpenStack fue desarrollado en 2010 inicialmente por RackSpace y la Nasa. RackSpace aportaba al proyecto el código de servicio de cloud files y cloud server, mientras que la Nasa contribuyo con el código de la plataforma Nebula.
La tecnología detrás de OpenStack consiste en proporcionar una serie de servicios interrelacionados (ilustración 11), los cuales aportan la solución de infraestructura en la nube. Cada servicio proporciona una API con la finalidad de que los recursos puedan administrarse por medio de un panel de control a través de una interfaz web [17].

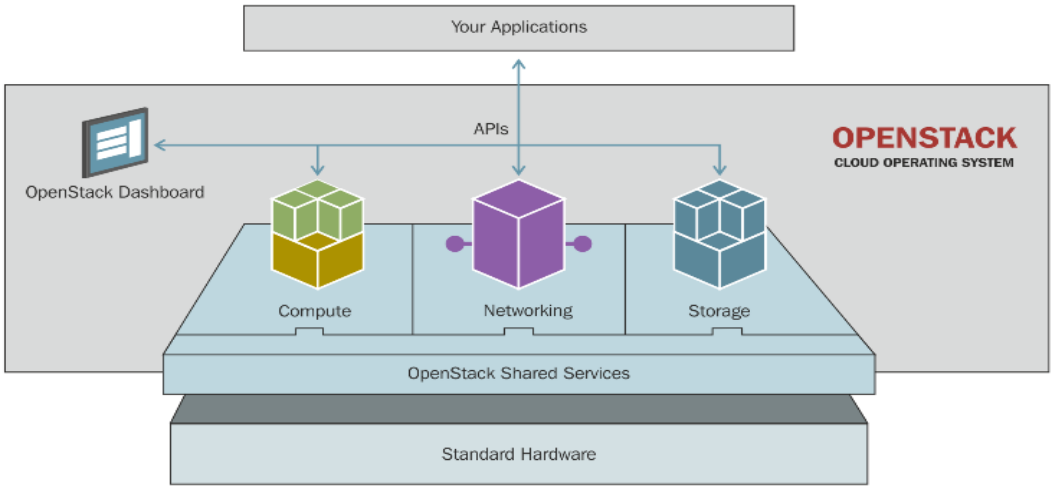

Ilustración 11: Estructura OpenStack [18]

- **Componentes de OpenStack.** La arquitectura modular y configurable de OpenStack permite que la solución sea adaptable a los recursos de hardware disponibles, proporcionándole a las empresas una variedad de servicios a elegir con el objetivo de satisfacer las necesidades en cuanto a la informática, redes y almacenamiento. Esta arquitectura se observa en la ilustración 12.

    - ✓ **Componente de Computo (Compute):** Es el encargado de controlar la nube, configura los recursos asignados a los usuarios y administra la ejecución de las instancias.
        - ❖ **Nova:** Es un controlador que proporciona servidores virtuales bajo demanda interactuando con los hipervisores (KVM, Xen, VMware, Hyper-v). Es el encargado de gestionar los recursos

en la nube, sin este servicio sería imposible implementarse una arquitectura básica [18].

- ❖ **Glance:** Es un sistema para buscar, registrar y recuperar imágenes de máquinas virtuales (VM). Además, permite que estas imágenes se utilicen como plantillas para implementar nuevas instancias de VM.

- ✓ **Componente de Red (Networking):** Permite que exista una conexión entre servidores, redes externas, además de definir nuevas redes con su arquitectura.
  - ❖ **Neutron:** Es el servicio encargado de proporcionar la conectividad de red como un servicio, además de gestionar redes (DHCP, VLAN, IDS) y direcciones IP de una forma simple, ágil y eficiente.
- ✓ **Componente de almacenamiento (Storage):** Aporta un sistema de alta capacidad de almacenamiento, con características de escalabilidad, redundancia y recuperación de información.
  - ❖ **Swift:** Es un sistema de almacenamiento para objetos y archivos escalable y redundante. La información se almacena en varias unidades de discos para garantizar la replicación e integridad de los datos.
  - ❖ **Cinder:** Proporciona un almacenamiento en bloques persistente. Al trabajar con Swift permite realizar copia de seguridad de las VMs, si se integra con Nova proporciona volúmenes para sus instancias, permitiendo la manipulación, tipos e instantáneas de volúmenes [19].
- ✓ **Servicios compartidos (Shared Services):** Permite a los servicios compartidos abarcar los tres componentes básicos, computación, almacenamiento y redes, con el objetivo de facilitar la implementación en la nube.
  - ❖ **Keystone:** Proporciona servicios de identidad para permitir a los usuarios acceder a los servicios que tengan asignados. Es un sistema de autenticación a través de un sistema operativo en la nube.
  - ❖ **Horizon:** Brinda a los administradores y usuarios una interfaz gráfica por medio de una aplicación web modular, con el objetivo de que puedan acceder a los recursos en la nube.
  - ❖ **Ceilometer:** Es el sistema encargado de mantener un conteo verificable del uso del sistema de cada usuario y de los recursos

en la nube, con el fin de proporcionar una facturación individual de cada usuario [20].

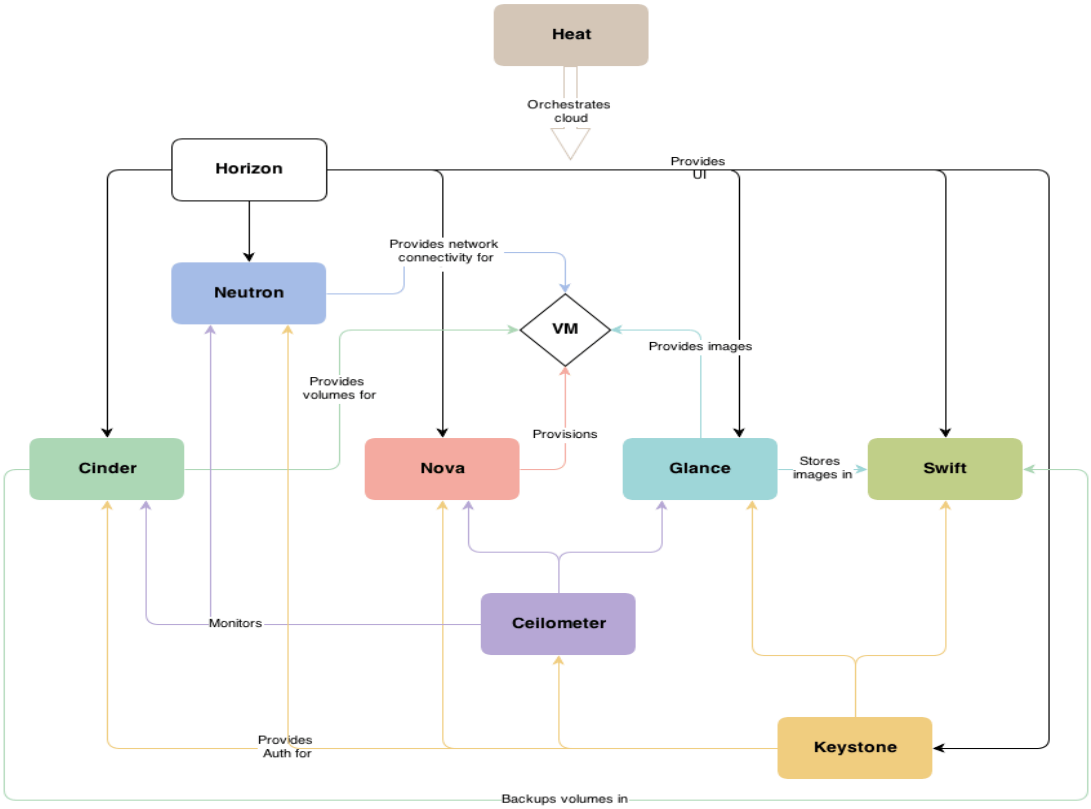

Ilustración 12: Componentes de OpenStack [21]

6.2.5. Redes definidas por software (SDN)

Open Networking Foundation [21] define las SDN como una arquitectura emergente la cual es dinámica, manejable y adaptable, haciéndola ideal para el alto ancho de banda, y para las aplicaciones dinámicas actuales. Esta arquitectura realiza un desacoplamiento entre el plano de control (Software) y el plano de datos (Hardware), convirtiendo la red, en una red programable y con un control centralizado. Es por esto, que al implementar las SDN se brinda mayor agilidad, y se reduce el desarrollo de servicios y costos operativos.

Las SDN permiten programar directamente el control de la red ya que su característica principal es que se desacopla el plano de datos con el plano de control, siendo este último el que controla el comportamiento de una red. En las SDN se le da importancia a 3 planos específicos:

- ✓ Plano de datos.
- ✓ Plano de control.
- ✓ Plano de aplicación.

El desacoplamiento del plano de datos y el de control le permite al administrador ajustar la red de una manera dinámica, es decir que se puede modificar a conveniencia según se requiera ya que dicho desacoplamiento permite programar directamente sobre el control de la red [22].

- **¿Cómo funcionan las SDN? (Arquitectura).** A pesar de que existe una gran variedad de arquitecturas para las SDN, la arquitectura en su forma más básica está compuesta por tres capas, la capa de aplicación, la capa de control y la capa de infraestructura, las cuales se interconectan por medio de protocolos y APIs, permitiendo la programación y el control centralizado de la red, además hace que las actualizaciones, los cambios, las adiciones y las configuraciones sean mucho más sencillas.
  - ✓ Capa de aplicación: Incluye las aplicaciones de red, como VoIP, aplicaciones de seguridad, básicamente, incluye los servicios de red y utilidades.
  - ✓ Plano de control: En esta capa se ubica el controlador SDN, el cual es el elemento más importante, puesto que gestiona la capa de aplicación e infraestructura. Esta capa está lógicamente centralizada y debe monitorizar todos los elementos de la red, para brindar una visión global de está.
  - ✓ Plano de datos: La función de esta capa es mover los paquetes a través de la red, gestionando el tráfico de forma centralizada basándose en tablas de flujo, pudiendo enviar paquetes a un costo mucho menor que en equipos de una red tradicional.

Según un grupo de investigación del Internet Research Task Group [23] enfocados en las SDN, explican que se tienen 3 enfoques en cuanto a la arquitectura de las SDN:

- ✓ OpenFlow SDN: en este enfoque se separa completamente el plano de control y el de datos mediante el protocolo OpenFlow.
- ✓ Plano de control SDN: su principal objetivo es hacer programable los planos de control, si hay más planos de control distribuidos en una red, también los hace programables.

- ✓ Overlay (superposición) SDN: aquí se puede agregar un plano de control o en su defecto uno de datos programables sobre los que ya existen.

Sin embargo las SDN según Lin and Lin [22] permiten combinar los aspectos de estos 3 enfoques siendo OpenFlow SDN y Overlay SDN los enfoques más utilizados y recurrentes.
Como se muestra en la Ilustración 13, el funcionamiento de las SDN se basa globalmente en la separación de los planos de control y de datos mediante el protocolo OpenFlow permitiendo el desacoplamiento de estos 2 planos [23].

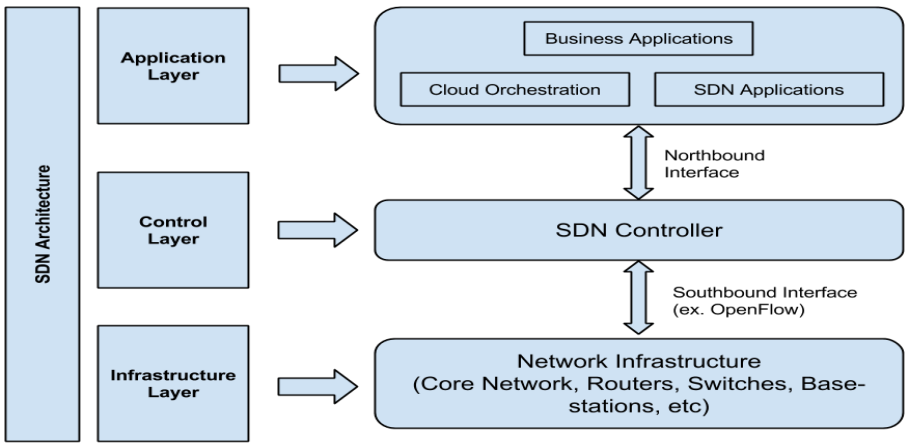

Ilustración 13: Funcionamiento de las SDN [24]

- **Beneficios de SDN**. Además de ofrecer una red programable y con un control centralizado, las SDN ofrecen grandes beneficios, descritos a continuación:
  - ✓ Directamente programable, puesto que las funciones de control están totalmente desacopladas de las funciones de datos, permitiendo que la red pueda ser programada por herramientas de automatización.
  - ✓ Gestión centralizada, la inteligencia de la red (plano de control) está lógicamente centralizada en el software del controlador SDN, manteniendo una vista global de la red.
  - ✓ Reduce los costos de capital (CapEx), puesto que las SDN minimizan la necesidad de adquirir Hardware propietario.
  - ✓ Reduce los costos de operación (OpEx), debido a que con la SDN se pueden desplegar con mayor facilidad y rapidez nuevas aplicaciones, servicios e infraestructura, reduciendo el tiempo de gestión y la

posibilidad de error humano.

- ✓ Ofrece agilidad y flexibilidad, al momento de desplegar nuevas aplicaciones, servicios e infraestructura.
- ✓ Habilita la innovación, dado que SDN permite a las empresas crear nuevos tipos de aplicaciones, servicios y modelos de negocio, que están representados en nuevas fuentes de ingresos y mayor valor de la red.

- **Controlador SDN.** Es un elemento que unifica los planos de control y datos, permitiendo tener un control en la red, es decir, es un tipo de puente programable que unifica estos planos de red. El controlador es un elemento primordial en la arquitectura SDN, debido a que se encarga de tomar decisiones, implementar reglas y ejecutar instrucciones que proceden de las diferentes aplicaciones y que se distribuyen a los dispositivos de la red, asimismo gestiona los diferentes paquetes de red.

  Al momento de seleccionar un controlador se deben tener presente las siguientes características:

  - ✓ Escalabilidad.
  - ✓ Rendimiento.
  - ✓ Programación de red.
  - ✓ Seguridad.
  - ✓ Virtualización de red.
  - ✓ Procesamiento
  - ✓ Monitorización centralizada y virtualizada.
  - ✓ Soporte de plataformas.
  - ✓ Funcionalidad de la red.
  - ✓ Soporte sobre el protocolo OpenFlow

  En la tabla 2 se exponen los tipos de controladores para SDN más conocidos.

  Tabla 2: Lista de Controladores [34]

Nombre	**Lenguaje de desarrollo**	**Versión Openflow**
Beacon	Java	1.0
OpenDaylight	Java	1.0
Ryu	Python	1.0, 1.2, 1.3

Trema	Ruby / C	1.3
Nox	C++	1.0
Pox	Python	1.0
Floodlight	Java	1.0

6.2.6. Controlador OpenDaylight (ODL)

El controlador OpenDaylight es desarrollado en el lenguaje de programación java, por lo tanto es modular y de código abierto (open Source), adaptable a redes de cualquier magnitud y con escalabilidad. La arquitectura de microservicios les brinda a los usuarios la capacidad de controlar aplicaciones, plug-ins y protocolos (ilustración 14).

Actualmente cuenta con varias versiones del controlador que sirven para varios propósitos entre estos propósitos está n:

- ✓ Automatización de servicios.
- ✓ Computación en la nube y NFV.
- ✓ Optimización de recursos de red.
- ✓ Visualización y control de red.

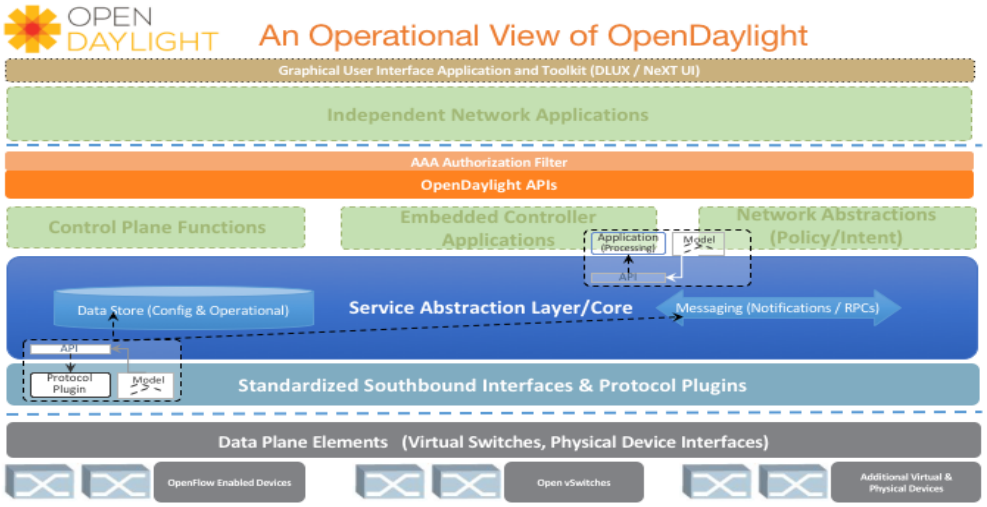

Ilustración 14: Vista operacional de Opendaylight [25]

Esta plataforma contiene soporte multiprotocolo, lo cual permite incluir

protocolos de red en cualquier plataforma de SDN con el fin de resolver cualquier necesidad del usuario que la maneje.
Algunas de las versiones de la plataforma OpenDaylight se exponen en la tabla 3.

Tabla 3: Versiones de Opendaylight [35]

Nombre	**Versión**	**Fecha de lanzamiento**	**Edición**
Hydrogen	1.0	Febrero 4 2014	Proveedor de servicios
Hydrogen	1.0	Febrero 4 2014	Virtualización
Hydrogen	1.0	Febrero 4 2014	Base
Helium	1.0	Septiembre 29 2014	Base
Helium-SR1	1.0	Noviembre 2014	Sin especificar
Helium-SR1.1	1.1	Diciembre 18 2014	Sin especificar
Helium-SR2	1.2	Enero 27 2015	Sin especificar
Helium-SR3	1.3	Marzo 17 2015	Sin especificar
Helium-SR4	1.4	Agosto 11 2015	Sin especificar
Lithium	1.0	Junio 29 2015	Base
Lihium-SR1	1.1	Agosto 18 2015	Sin especificar
Lithium-SR2	1.2	Octubre 8 2015	Sin especificar
Lithium-SR3	1.3	Diciembre 3 2015	Sin especificar
Lithium-SR4	1.4	Marzo 4 2016	Sin especificar
Beryllium	1.0	Febrero 22 2016	Base
Beryllium-SR1	1.1	Marzo 22 2016	Sin especificar
Beryllium-SR2	1.2	Mayo 11 2016	Sin especificar
Beryllium-SR4	1.4	Octubre21 2016	Sin especificar
Boron-SR2	1.2	Diciembre 20 2016	Sin especificar

7. ENFOQUE METODOLÓGICO

Para el proyecto se tomará un enfoque de paradigma mixto ya que se requiere tomar un análisis bibliográfico además de poner en práctica lo que se plantea. Por esto el enfoque que se tomara en esta tesis tendrá 4 fases principales las cuales contendrán los siguientes ítems:

Fase 1
Realizar una recopilación y análisis de bibliografía existente referente a este tema, con el fin de llevar el proyecto por un rumbo fijo para el cumplimiento de los objetivos planteados, además de obtener los resultados esperados. Este análisis de bibliografía también pretende documentar lo que ya se ha hecho sobre este tema para tener una guía sobre lo que se pretende hacer.

Fase 2
Analizar la plataforma OSM (Open Source MANO) haciendo un estudio de esta y sus funciones para su implementación, luego se implementarán las plataformas Opendaylight y OpenStack para crear el entorno de trabajo en donde se propondrá una topología en la cual se aplicara funciones de red virtualizadas (NFV).

Fase 3
Integrar las plataformas mencionadas en la fase 2. Para comprobar su funcionamiento, se orquestarán las plataformas a través de la plataforma Open Source MANO, luego para el cumplimiento de los objetivos se virtualizarán dos funciones de red en la topología planteada.

Fase 4
Con base en lo realizado anteriormente, virtualizar dos funciones de red, con el objetivo de comprobar el correcto funcionamiento y analizar los resultados para determinar si realmente la plataforma Open Source MANO cumple con la solución propuesta por la ETSI sobre el despliegue de funciones de red, teniendo en cuenta que se realizó sobre un laboratorio a escala.

8. PROCEDIMIENTO

Para la integración de la virtualización de funciones de red, computación en la nube y redes definidas por software se lleva a cabo la implementación de las plataformas de OSM, OpenStack y OpenDaylight respectivamente. Con base en la investigación realizada anteriormente, se toma como eje central la plataforma OSM, es decir que tanto OpenStack como OpenDaylight se conectaran a esta plataforma.

Para integrar el VIM de OpenStack con OSM primero hay que asegurarse de que se tenga conectividad entre ambas plataformas, además de configurar una red de administración que será la encargada de proporcionarle las direcciones IP a los VNFs. OSM despliega instancias para los servicios de red asignando interfaces passthrough (se utilizan para el acceso de “invitados”), para que posteriormente OpenStack le asigne una interfaz a dicha instancia. Cuando se integran las SDN (en este caso OpenDaylight) con OSM, este usa las interfaces de red de OpenStack y las asigna a los puertos del switch de OpenFlow. Finalmente, OSM creara las redes del plano de datos comunicándose con el controlador SDN.

Lo anterior se realiza en la máquina de OSM-RO (Orquestador de recursos), la cual utiliza una biblioteca interna que administra la conectividad a través de las SDN, teniendo presente las reglas proactivas de OpenFlow.

8.1. DISEÑO Y CREACIÓN DE LA TOPOLOGÍA DE RED.

Para el desarrollo del proyecto se realiza el diseño de la topología de red (ilustración 15) con el objetivo de tener un ambiente de red más controlado. Para este procedimiento se utilizan dos servidores físicos, donde en uno de ellos se implementará Open Source MANO (OSM) y Opendaylight sobre VirtualBox, y en el otro servidor se implementará OpenStack.

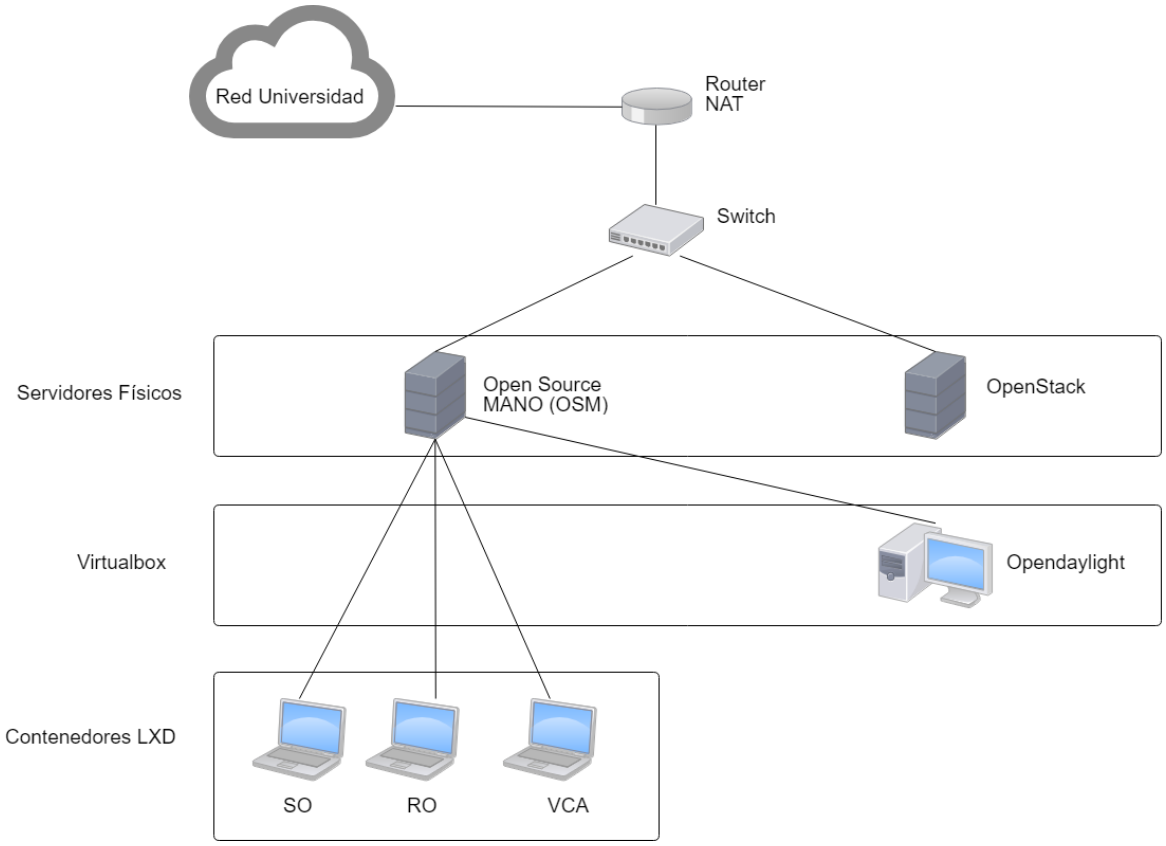

Ilustración 15: Diseño topología

8.2. IMPLEMENTACIÓN OPEN SOURCE MANO (OSM)

En esta fase del procedimiento se busca implementar la plataforma Open Source MANO (OSM) la cual permitirá virtualizar las funciones de red.

Antes de iniciar el proceso de instalación de OSM, se debe tener instalado y configurado el servicio de Linux container hypervisor (LXD), el cual es un hipervisor de contenedores y sobre este se implementará la arquitectura de OSM.

Para empezar lo primero que se hace es actualizar los paquetes y dependencias del sistema operativo.

```
# sudo apt-get update
# sudo apt-get upgrade
```

Se debe instalar el paquete ZFS, con el fin de utilizarlo como backend de almacenamiento LXD y así tener un rendimiento más óptimo, como se aprecia en la ilustración 16.

```
opensource@opensource-VirtualBox:~$ sudo apt-get install zfs
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'zfsutils-linux' instead of 'zfs'
The following additional packages will be installed:
  libnvpair1linux libuutil1linux libzfs2linux libzpool2linux zfs-doc zfs-zed
Suggested packages:
  default-mta | mail-transport-agent samba-common-bin nfs-kernel-server
  zfs-initramfs
The following NEW packages will be installed:
  libnvpair1linux libuutil1linux libzfs2linux libzpool2linux zfs-doc zfs-zed
  zfsutils-linux
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 897 kB of archives.
After this operation, 2.902 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Ilustración 16. Instalación ZFS

Como paso a seguir, se instala LDX y a continuación con el comando "`newgrp`" se cambia de grupo de trabajo.

```
opensource@opensource-VirtualBox:~$ sudo apt install lxd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  liblxc1 lxc-common lxcfs lxd-client uidmap
Suggested packages:
  criu lxd-tools
The following NEW packages will be installed:
  liblxc1 lxc-common lxcfs lxd lxd-client uidmap
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 5.622 kB of archives.
After this operation, 26,8 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
opensource@opensource-VirtualBox:~$ newgrp lxd
```

Ilustración 17. Instalación LXD y asignación del grupo de trabajo

Se inicializa el servicio LXD como se muestra en la ilustración 18, con el objetivo de configurar parámetros como el tipo de almacenamiento backend, el tamaño, nombre del mismo. Por el momento no se configurará el adaptador puente, de igual forma, si se configura no generará ningún problema debido a que más adelante se modificará.

```
opensource@opensource-VirtualBox:~$ sudo lxd init
Name of the storage backend to use (dir or zfs) [default=zfs]:
Create a new ZFS pool (yes/no) [default=yes]?
Name of the new ZFS pool [default=lxd]:
Would you like to use an existing block device (yes/no) [default=no]?
Size in GB of the new loop device (1GB minimum) [default=44]: 15
Would you like LXD to be available over the network (yes/no) [default=no]?
Do you want to configure the LXD bridge (yes/no) [default=yes]?
Warning: Stopping lxd.service, but it can still be activated by:
  lxd.socket
LXD has been successfully configured.
```

Ilustración 18. Inicialización LXD

Por último, verificamos que se haya instalado el servicio correctamente, además de generar un certificado para el cliente (ilustración 19).

```
opensource@opensource-VirtualBox:~$ lxc list
Generating a client certificate. This may take a minute...
If this is your first time using LXD, you should also run: sudo lxd init
To start your first container, try: lxc launch ubuntu:16.04

+------+-------+------+------+------+-----------+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+------+-------+------+------+------+-----------+
```

Ilustración 19. Verificación servicio LXD

Una vez se tenga instalado LXD, se instala Open Source MANO (OSM), para ello se va a descargar el script desde la página oficial.

```
opensource@opensource-VirtualBox:~$ wget https://osm-download.etsi.org/ftp/osm-2
.0-two/install_osm.sh
--2017-10-12 14:21:18--  https://osm-download.etsi.org/ftp/osm-2.0-two/install_o
sm.sh
Resolving osm-download.etsi.org (osm-download.etsi.org)... 195.238.226.47
Connecting to osm-download.etsi.org (osm-download.etsi.org)|195.238.226.47|:443.
.. connected.
HTTP request sent, awaiting response... 200 OK
Length: 14435 (14K) [text/x-sh]
Saving to: ‘install_osm.sh’

install_osm.sh      100%[===================>]  14,10K  69,8KB/s    in 0,2s

2017-10-12 14:21:19 (69,8 KB/s) - ‘install_osm.sh’ saved [14435/14435]
```

Ilustración 20. Descargar paquete de OSM

Como paso a seguir se le cambian los permisos al archivo descargado en el paso anterior.

```
opensource@opensource-VirtualBox:~$ chmod +x install_osm.sh
```

Ilustración 21. Permisos paquete OSM

Por último, como se observa en la ilustración 22, se ejecuta el script para instalar OSM.

```
opensource@opensource-VirtualBox:~$ ./install_osm.sh -b tags/v2.0.2

Creating temporary dir for OSM installation
Checking required packages: git

Cloning devops repo temporarily
./install_osm.sh: line 264: git: command not found
using devops repo tag: tags/v2.0.2
./install_osm.sh: line 273: git: command not found
./install_osm.sh: line 277: /tmp/installosm.uKYnRF/jenkins/common/all_funcs: No
such file or directory

Installing OSM from refspec: tags/v2.0.2

Checking required packages: wget, curl, tar
Checking required packages: lxd
```

Ilustración 22. Instalación OSM

Después de terminada la instalación, se verifican que se hayan creado los contenedores correctamente, como se presenta en la ilustración 23. Se debe tener en cuenta que el contenedor SO (orquestación de servicios) se encarga de la automatización, administración y coordinación de un servicio determinado, también es el encargado de gestionar la interfaz gráfica. El contenedor RO (orquestación de recursos), será el encargado de administrar los recursos de red tales como servidores, VIM, entre otros. Por último, el contenedor VCA, será el encargado de configurar y abstraer las funciones de red.

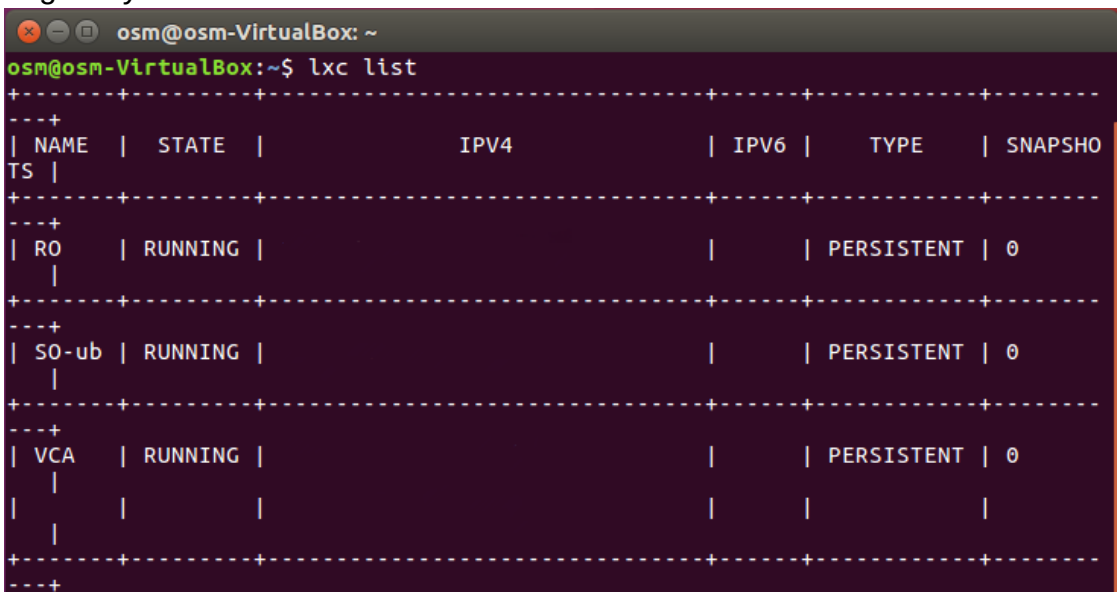

Ilustración 23. Verificación instalación contenedores OSM

Si por algún motivo los contenedores se encuentran apagados, solo deben ejecutar el comando "*lxc start <nombre del contenedor>*" y para entrar a cada uno de los contenedores y verificar su conectividad solo se debe ejecutar "*lxc exec <nombre del contenedor> bash*"

A continuación lo que se debe realizar es la configuración del adaptador puente. Para ello se abre el archivo de configuración "lxd-bridge"

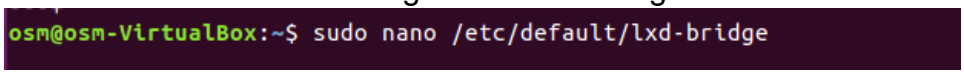

Ilustración 24. Ruta para la configuración del adaptador puente

Lo importante a configurar en este archivo son los siguientes parámetros (ilustración 25): (1) Habilitar el servicio LXD, (2) Nombrar el adaptador puente, (3) Configurar la dirección IP para el adaptador puente, (4) Configurar la máscara de red, (5) Definir la red con su respectiva máscara de red, (6) Establecer el rango de direcciones IP disponibles que serán entregadas por el servicio DHCP, (7) Deshabilitar la opción de NAT en el adaptador.

```
# WARNING: This file is generated by a debconf template!
# It is recommended to update it by using "dpkg-reconfigure -p medium lxd"

# Whether to setup a new bridge or use an existing one
USE_LXD_BRIDGE="true"  (1)

# Bridge name
# This is still used even if USE_LXD_BRIDGE is set to false
# set to an empty value to fully disable
LXD_BRIDGE="lxdbr0" (2)

# Update the "default" LXD profile
UPDATE_PROFILE="true"

# Path to an extra dnsmasq configuration file
LXD_CONFILE=""

# DNS domain for the bridge
LXD_DOMAIN="lxd"

# IPv4
## IPv4 address (e.g. 10.0.8.1)
LXD_IPV4_ADDR="192.168.0.150"  (3)

## IPv4 netmask (e.g. 255.255.255.0)
LXD_IPV4_NETMASK="255.255.255.0" (4)

## IPv4 network (e.g. 10.0.8.0/24)
LXD_IPV4_NETWORK="192.168.0.0/24"  (5)

## IPv4 DHCP range (e.g. 10.0.8.2,10.0.8.254)
LXD_IPV4_DHCP_RANGE="192.168.0.151,192.168.0.200"(6)

## IPv4 DHCP number of hosts (e.g. 250)
LXD_IPV4_DHCP_MAX="49"(6)

## NAT IPv4 traffic
LXD_IPV4_NAT="false" (7)
```

Ilustración 25. Configuración del adaptador puente

Por último, se procede a unificar el adaptador puente con el adaptador de red de la maquina física, como se muestra en la ilustración 26, con el objetivo de que ambos trabajen como si fueran uno. Para esto, se entra a editar las conexiones, se selecciona el adaptador puente creado anteriormente y se edita, teniendo presente que la versión con la cual se está trabajando es Ubuntu 16.04.

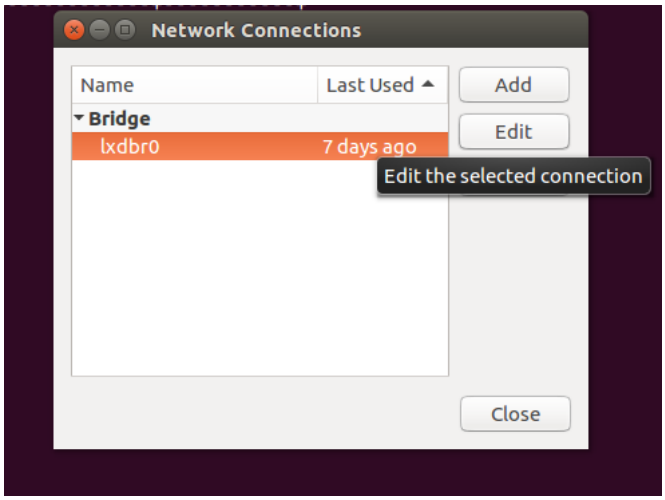

Ilustración 26. Edición adaptador puente

Una vez allí se agregaría el adaptador de la red física, para ello se presiona en “agregar”, como paso a seguir el adaptador “Ethernet” y por último en el dispositivo de red que para este caso es *enp0s3*

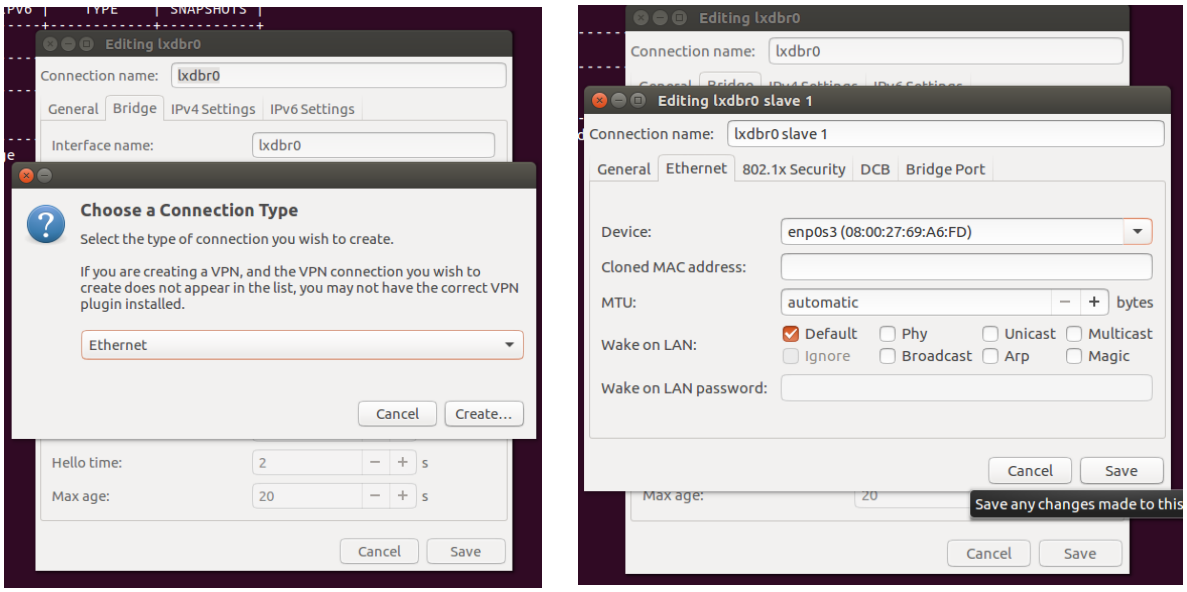

Ilustración 27. Unificar adaptador puente con la interfaz física

Se comprueba que el adaptador Ethernet se haya creado correctamente, además que este unificado al adaptador puente y por último se reinicia el servicio de red.

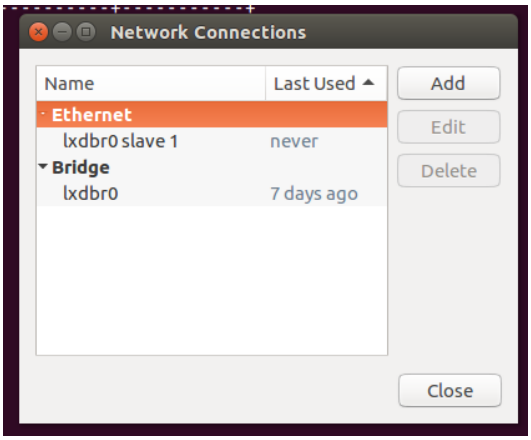

Ilustración 28. Verificar adaptadores de red

```
osm@osm-VirtualBox:~$ sudo /etc/init.d/networking restart
[ ok ] Restarting networking (via systemctl): networking.service.
```

Ilustración 29. Reiniciar servicio de red

Se verifica que los contenedores tomen la dirección IP suministrada por el DHCP del adaptador puente.

```
osm@osm:~$ lxc list
+-------+---------+-------------------------------+------+-----------+-----------+
| NAME  |  STATE  |             IPV4              | IPV6 |   TYPE    | SNAPSHOTS |
+-------+---------+-------------------------------+------+-----------+-----------+
| RO    | RUNNING | 10.84.130.7 (eth0)            |      | PERSISTENT | 0         |
+-------+---------+-------------------------------+------+-----------+-----------+
| SO-ub | RUNNING | 10.84.130.6 (eth0)            |      | PERSISTENT | 0         |
+-------+---------+-------------------------------+------+-----------+-----------+
| VCA   | RUNNING | 10.84.130.8 (eth0)            |      | PERSISTENT | 0         |
|       |         | 10.70.194.10 (lxdbr0)         |      |           |           |
+-------+---------+-------------------------------+------+-----------+-----------+
```

Ilustración 30. Verificación direccionamiento IP

Por último se hacen pruebas observando que exista conectividad entre los diferentes contenedores y con otras máquinas. Una vez comprobado esto, se abre el navegador y se ingresa la IP del contenedor SO-ub, que es el contenedor que suministra el entorno gráfico (ilustración 31). La plataforma OSM utiliza el protocolo HTTPS y el puerto 8443 (https://<IP CONTENEDOR SO-ub>:8443)

El usuario y contraseña por defecto para ingresar a la plataforma es ADMIN

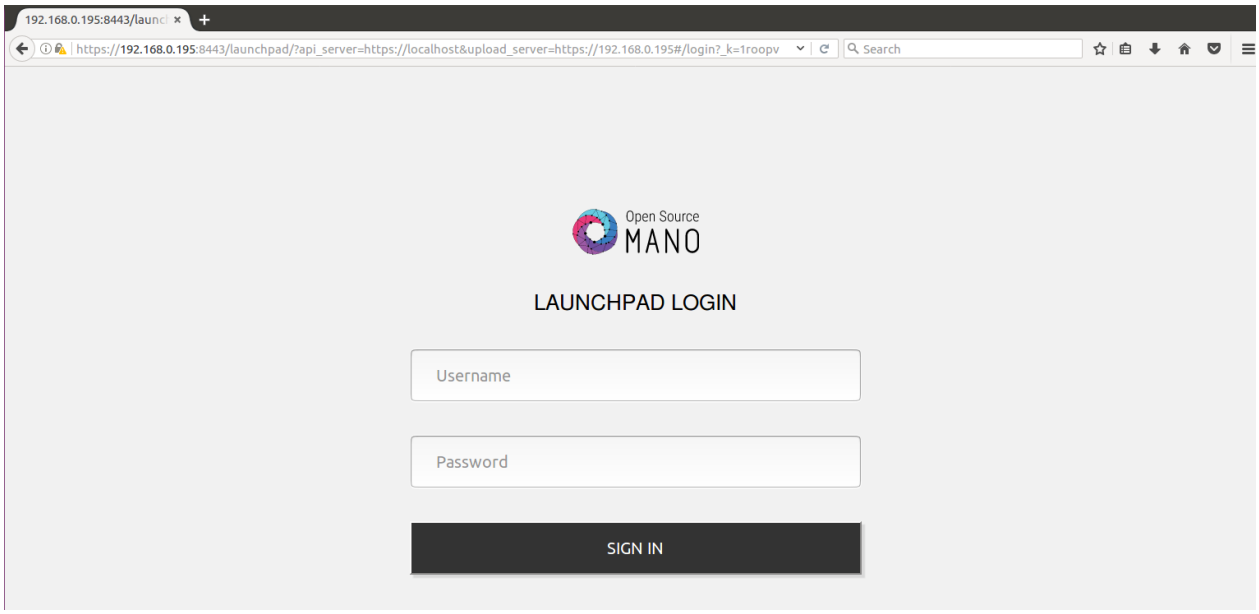

Ilustración 31. Dashboard OSM

8.3. IMPLEMENTACIÓN DE OPENSTACK

Para la instalación de OpenStack se utiliza conjure-up, la cual es una herramienta que abarca varias tecnologías tales como: Juju, metal como servicio (MAAS) y LXD, ofreciendo una solución simplificada. Con el objetivo de proporcionar un enfoque simplificado, conjure-up hace uso de secuencias de comandos, ejecutando scripts en 3 momentos diferentes cuando se hace un despliegue.

Como primer paso debemos actualizar debidamente los repositorios y el sistema, para esto utilizamos las siguientes secuencias de comandos:

```
# sudo apt-get update
# sudo apt-get upgrade
```

Una vez actualizado el sistema, debemos instalar conjure-up en su versión base utilizando snap

```
# sudo apt-get install snap
# sudo snap install conjure-up-classic
```

Se procede a instalar el hipervisor de contenedores LXD, el cual será utilizado por Openstack para desplegar sus nodos

```
# snap install lxd --classic
```

```
# /snap/bin/lxd init --auto
# sudo usermod -a -G lxd
# /snap/bin/lxc network create lxdbr0 ipv4.address=auto
ipv4.nat=true ipv6.address=none
```

Teniendo los pasos anteriores completados con éxito, se procede a instalar OpenStack con conjure-up ejecutando el siguiente comando

```
# conjure-up
```

En la terminal saldrá la ventana principal de instalación (ilustración 32) en donde se seleccionará “Openstack with NovaLXD”. Para seleccionar esta opción se usan las flechas de dirección para mover el cursor hasta la opción deseada.

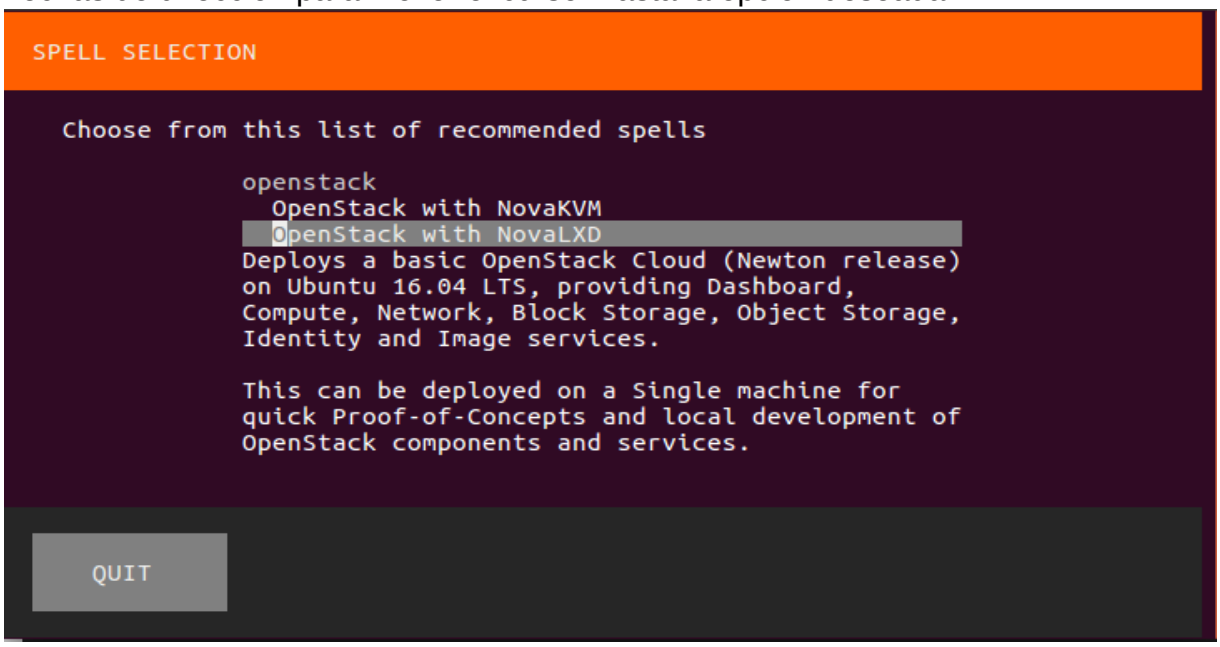

Ilustración 32. Selección de hipervisor

Al seleccionar la opción de instalar OpenStack con LXD, aparecerá la opción de configurar OpenStack en la maquina local

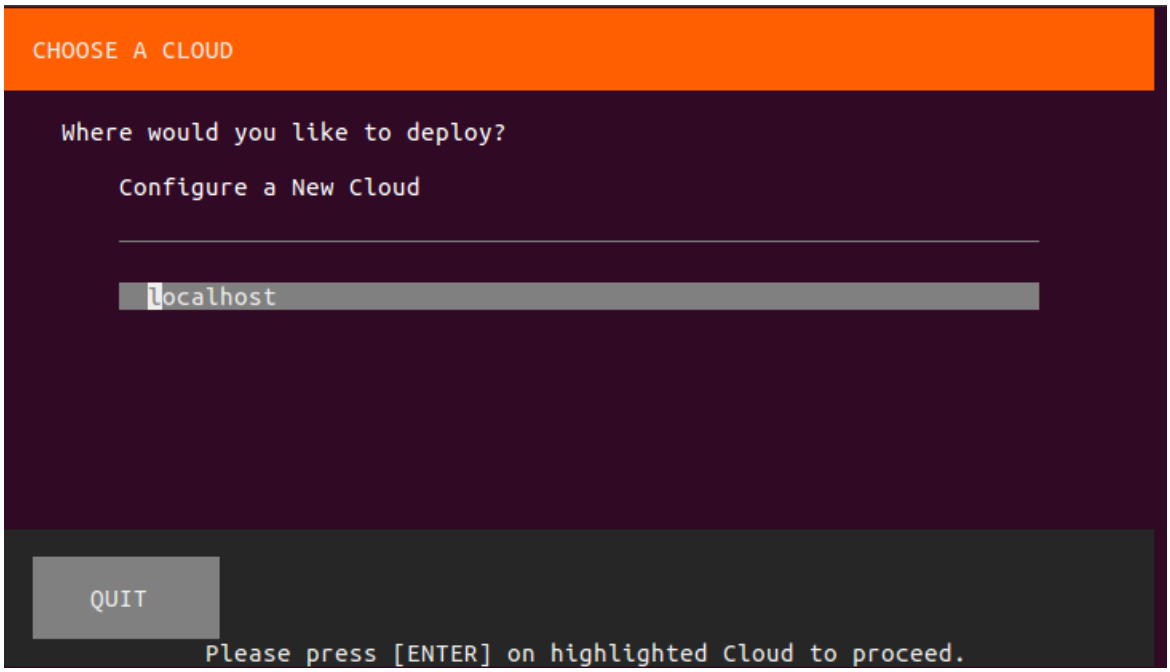

Ilustración 33. Selección sitio de almacenamiento

Como paso a seguir se despliegan todos los elementos de OpenStack

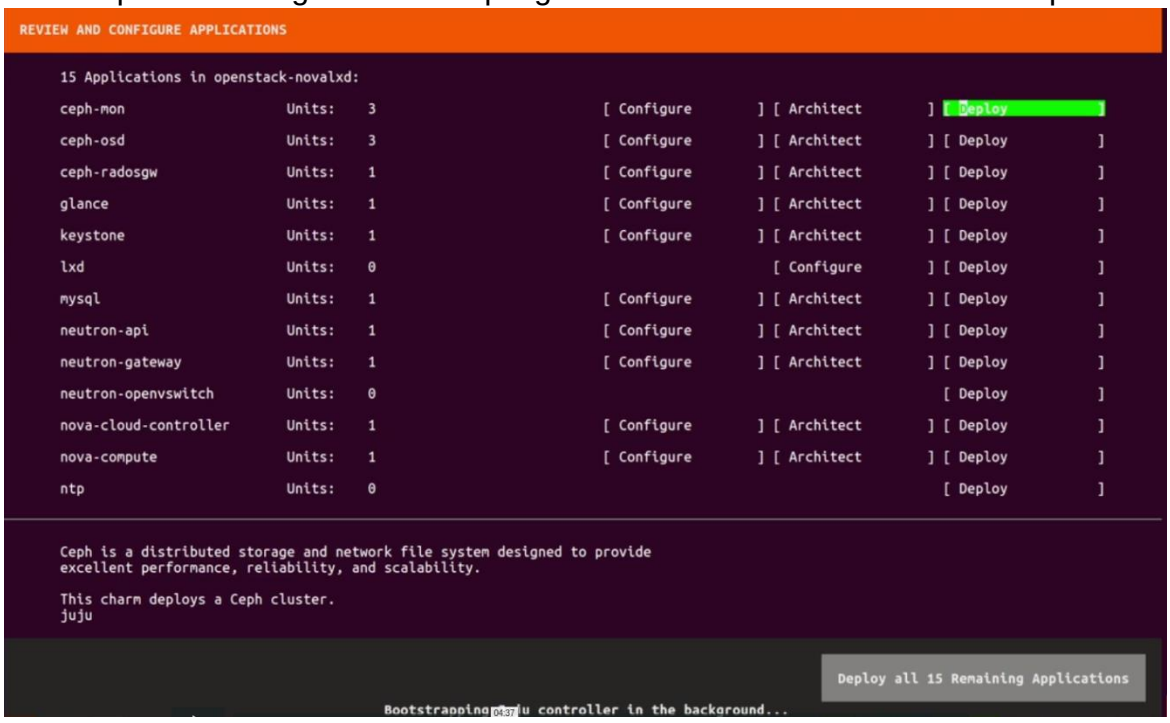

Ilustración 34. Despliegue servicios OpenStack

El script de instalación se ejecutará (ilustración 35), en este punto se debe esperar a que las instancias de OpenStack pasen del estado *Waiting* a *Ready*

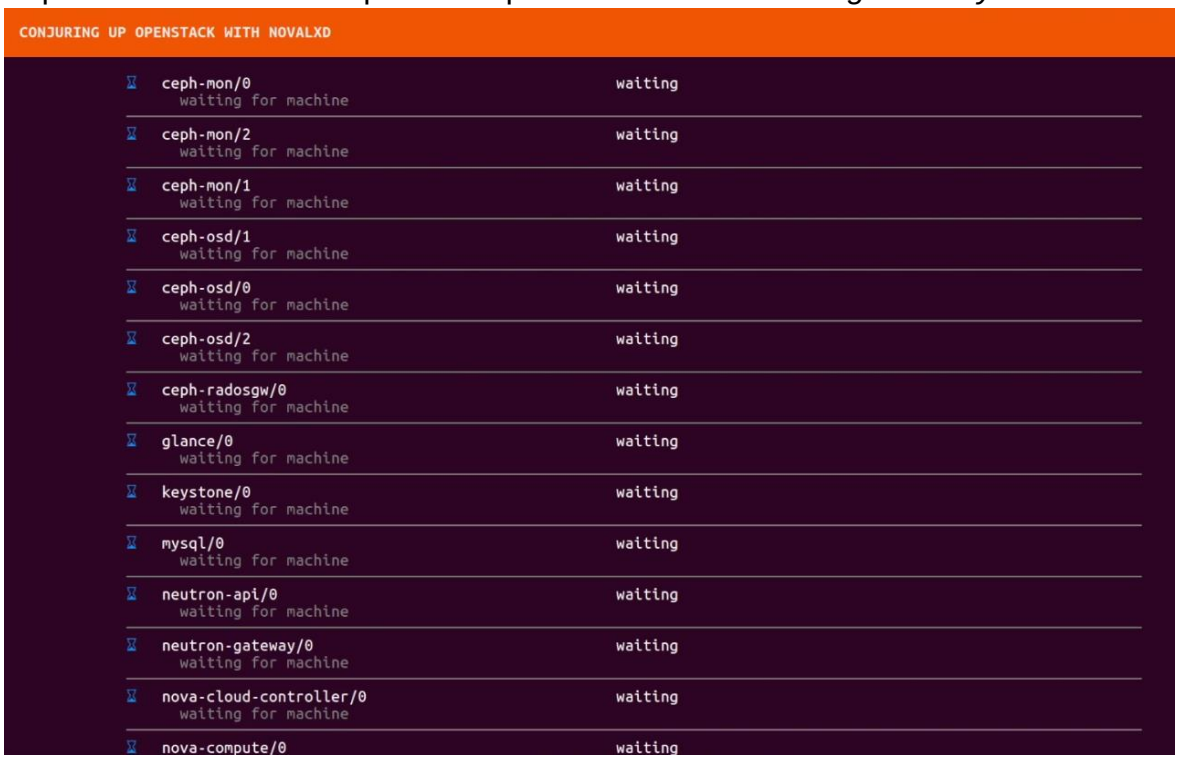

Ilustración 35. Ejecución Script

Una vez terminado el proceso anterior, basta con seleccionar “Run” para inicializar las instancias instaladas.

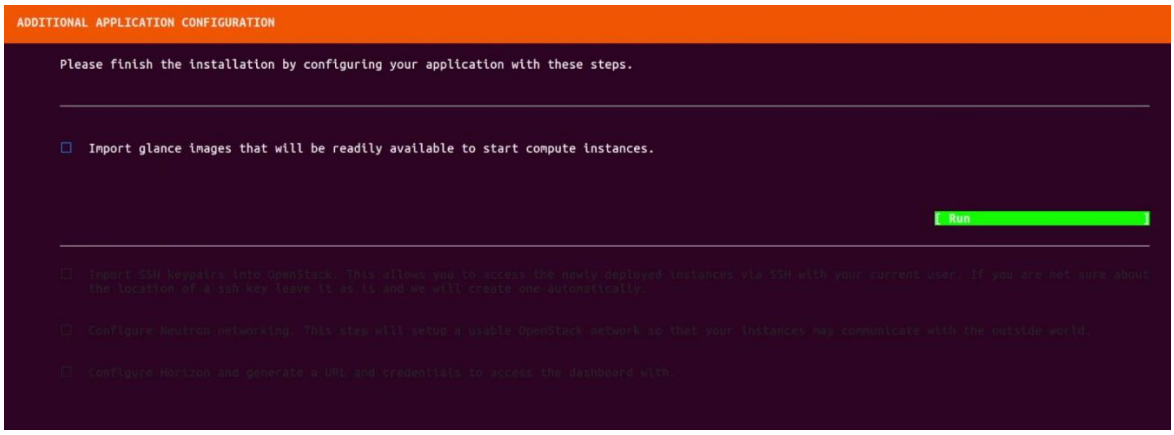

Ilustración 36. Inicialización instancias

Como se observa en la ilustración 37, el paso a seguir es crear la llave SSH. Esta llave SSH se crea para poder acceder de forma remota a cada una de las instancias. Si se desea se puede modificar la ruta donde se almacenara la llave SSH.

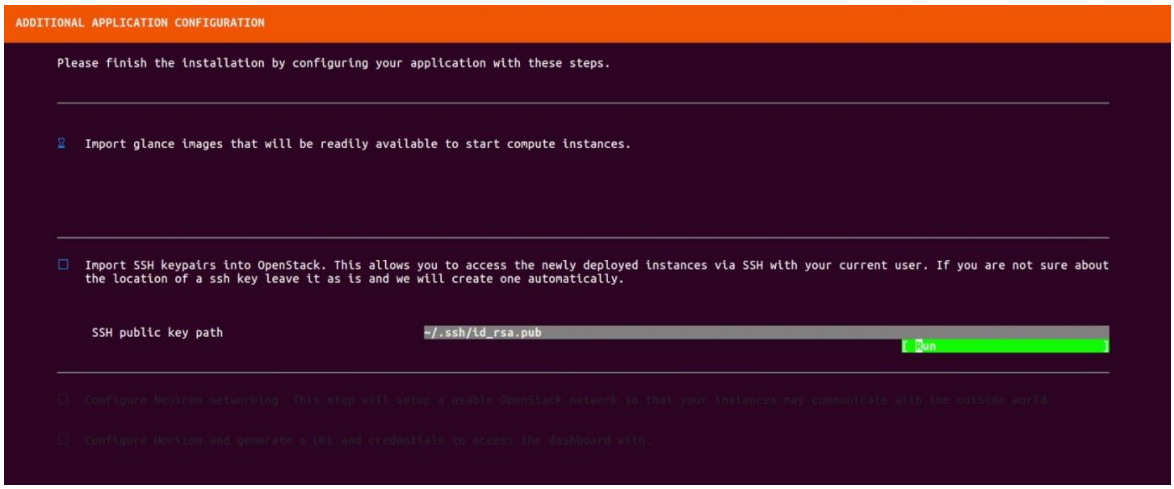

Ilustración 37. Creación llave SSH

La siguiente opción es poner en marcha la red Neutron, la cual configura OpenStack para las instancias instaladas tengan salida a internet.

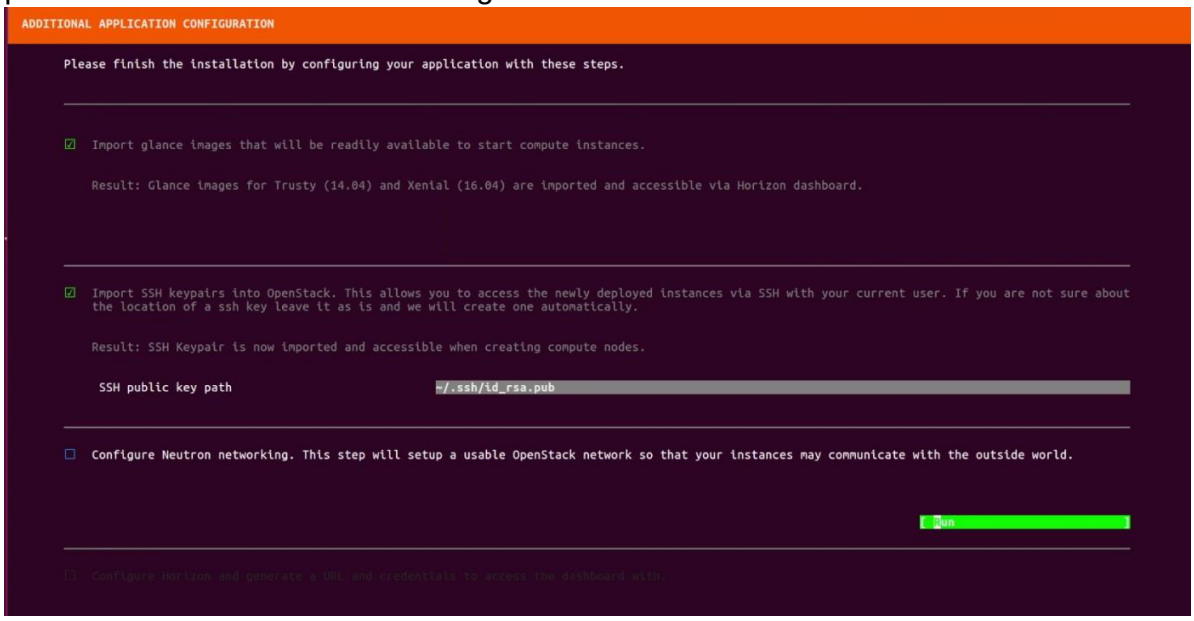

Ilustración 38. Red Neutron

Para verificar que se haya instalado correctamente OpenStack (ilustración 39), primero verificamos con el comando *juju status*, el estado y la dirección IP de cada una de las maquinas instaladas anteriormente.

```
ops@OpenStack:~$ juju status
Model                      Controller               Cloud/Region         Version  SLA
conjure-openstack-novalx-727  conjure-up-localhost-82e  localhost/localhost  2.3.3    unsupported

App                    Version       Status   Scale  Charm                  Store       Rev  OS      Notes
ceph-mon               12.2.1        active       3  ceph-mon               jujucharms  23   ubuntu
ceph-osd               12.2.1        active       3  ceph-osd               jujucharms  257  ubuntu
ceph-radosgw           12.2.1        active       1  ceph-radosgw           jujucharms  256  ubuntu
glance                 15.0.0        active       1  glance                 jujucharms  263  ubuntu
keystone               12.0.0        active       1  keystone               jujucharms  275  ubuntu
lxd                    2.0.11        active       1  lxd                    jujucharms  17   ubuntu
mysql                  5.6.37-26.21  active       1  percona-cluster        jujucharms  259  ubuntu
neutron-api            11.0.2        active       1  neutron-api            jujucharms  258  ubuntu
neutron-gateway        11.0.2        active       1  neutron-gateway        jujucharms  247  ubuntu
neutron-openvswitch    11.0.2        active       1  neutron-openvswitch    jujucharms  249  ubuntu
nova-cloud-controller  16.0.3        active       1  nova-cloud-controller  jujucharms  306  ubuntu
nova-compute           16.0.3        active       1  nova-compute           jujucharms  282  ubuntu
ntp                                  waiting      0  ntp                    jujucharms  24   ubuntu
openstack-dashboard    12.0.1        active       1  openstack-dashboard    jujucharms  257  ubuntu  exposed
rabbitmq-server        3.5.7         active       1  rabbitmq-server        jujucharms  72   ubuntu

Unit                      Workload  Agent  Machine  Public address  Ports              Message
ceph-mon/0                active    idle   4        10.84.130.57                       Unit is ready and clustered
ceph-mon/1*               active    idle   0        10.84.130.88                       Unit is ready and clustered
ceph-mon/2                active    idle   8        10.84.130.125                      Unit is ready and clustered
ceph-osd/0                active    idle   14       10.84.130.113                      Unit is ready (1 OSD)
ceph-osd/1*               active    idle   1        10.84.130.87                       Unit is ready (1 OSD)
ceph-osd/2                active    idle   3        10.84.130.234                      Unit is ready (1 OSD)
ceph-radosgw/0*           active    idle   2        10.84.130.124   80/tcp             Unit is ready
glance/0*                 active    idle   10       10.84.130.188   9292/tcp           Unit is ready
keystone/0*               active    idle   15       10.84.130.189   5000/tcp           Unit is ready
mysql/0*                  active    idle   9        10.84.130.246   3306/tcp           Unit is ready
neutron-api/0*            active    idle   7        10.84.130.185   9696/tcp           Unit is ready
neutron-gateway/0*        active    idle   5        10.84.130.98                       Unit is ready
nova-cloud-controller/0*  active    idle   6        10.84.130.126   8774/tcp,8778/tcp  Unit is ready
nova-compute/0*           active    idle   11       10.84.130.198                      Unit is ready
  lxd/0*                  active    idle            10.84.130.198                      Unit is ready
  neutron-openvswitch/0*  active    idle            10.84.130.198                      Unit is ready
openstack-dashboard/0*    active    idle   13       10.84.130.237   80/tcp,443/tcp     Unit is ready
rabbitmq-server/0*        active    idle   12       10.84.130.242   5672/tcp           Unit is ready
```

Ilustración 39. Verificación instalación OpenStack

Por último, en la ilustración 40, se aprecia el dashboard de OpenStack (dirección IP de la instancia openstack-dashboard), para ingresar a este, se hace uso del usuario por defecto es *admin* y la contraseña *openstack*

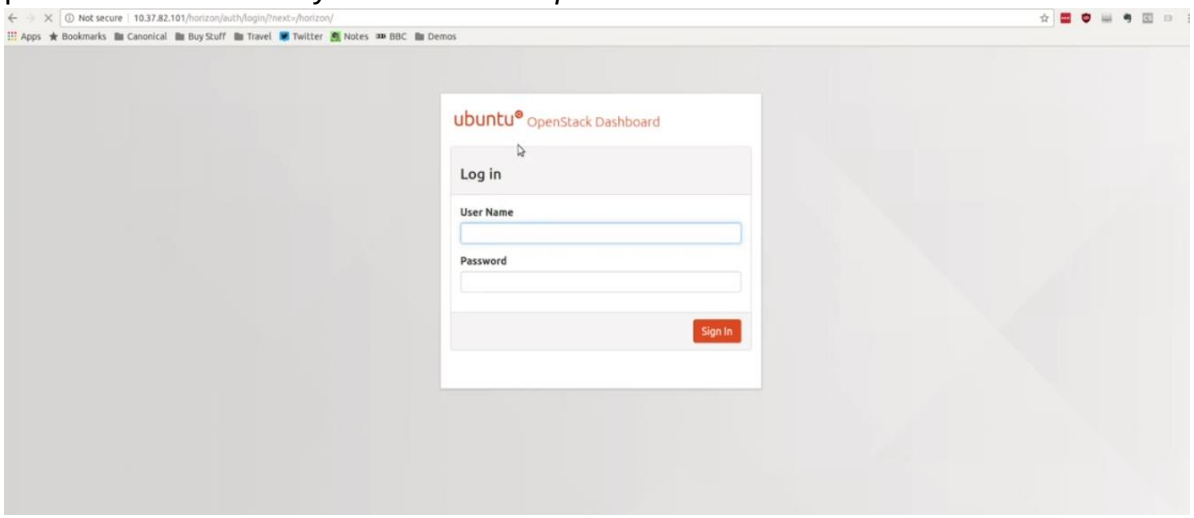

Ilustración 40. Dashboard OpenStack

Para que exista una conectividad con otras máquinas se debe ligar el adaptador puente con el adaptador de red de la maquina física para que ambos trabajen como si fueran uno, tal como se realizó en la sección 8.2

8.4. IMPLEMENTACIÓN DE OPENDAYLIGHT

Como último procedimiento de implementación se implementará Opendaylight, el cual es el controlador de redes definidas por software (SDN).

Como primer paso se accede al siguiente link: https://www.opendaylight.org/technical-community/getting-started-for-developers/downloads-and-documentation

Luego de elegir (según su necesidad) la distribución deseada se procede a dar clic en la sección de “Downloads” donde dice “Pre-build” y se selecciona la extensión “tar” o “zip”.

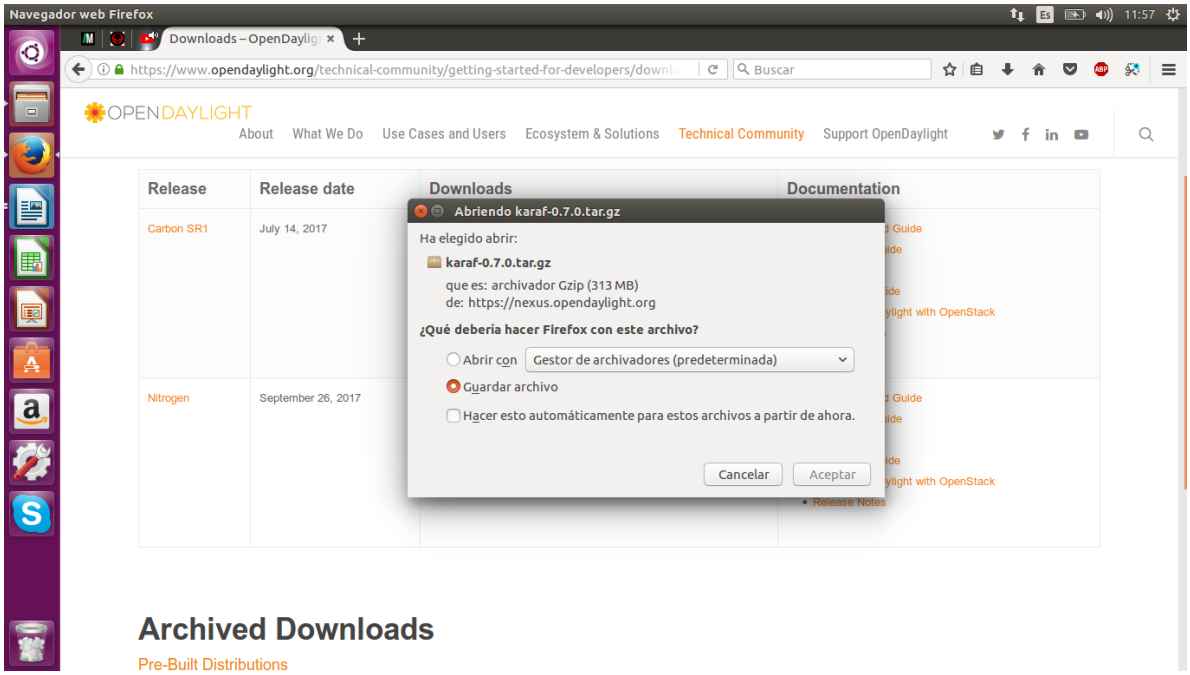

Ilustración 41. Descarga OpenDaylight

Una vez finalizada la descarga, se descomprime el archivo, al terminar el proceso de descompresión se procede a ejecutar el controlador, esto se hace utilizando el siguiente proceso:

- Se accede a la carpeta que descomprimimos

```
# cd karaf-0.7.0/
```

- Se ejecuta el controlador

  ```
  # ./bin/karaf
  ```

Cuando se ejecute el controlador se abrirá una consola de administración (ilustración 42)

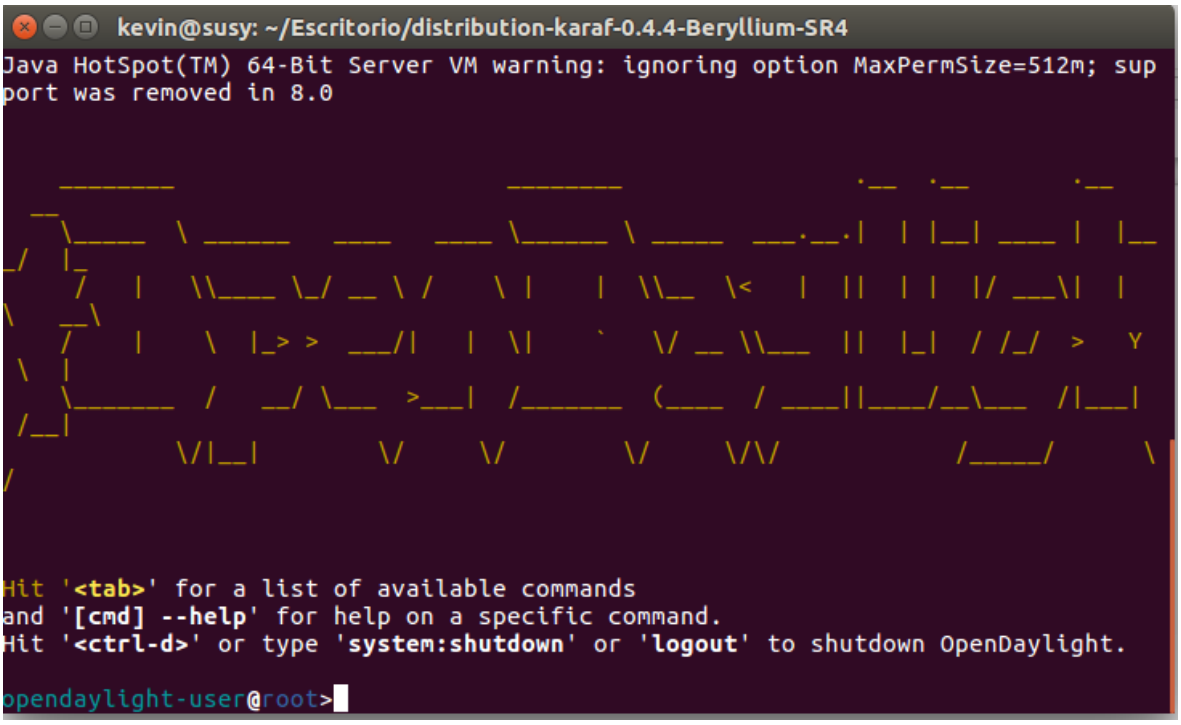

Ilustración 42. Consola de administración ODL

Dentro de la consola del controlador se deben instalar las características extras (Entorno gráfico, enrutamiento, conexiones cliente-servidor, protocolo Openflow, servicio de virtual switch (OVS)) con los comandos de la ilustración 43.

```
opendaylight-user@root>feature:install odl-dlux-all
opendaylight-user@root>feature:install odl-l2switch-all
opendaylight-user@root>feature:install odl-sxp-all
opendaylight-user@root>feature:install odl-openflowjava-all
```

Ilustración 43. Características ODL

Al finalizar este proceso, nos dirigimos al navegador para acceder a la interfaz gráfica del controlador opendaylight (ilustración 44) por medio de la dirección IP donde se encuentra el controlador y por el puerto 8181 (http://DIRECCION_IP:8181/index.html)

Para ingresar al sistema el usuario y la contraseña por defecto es ***admin***

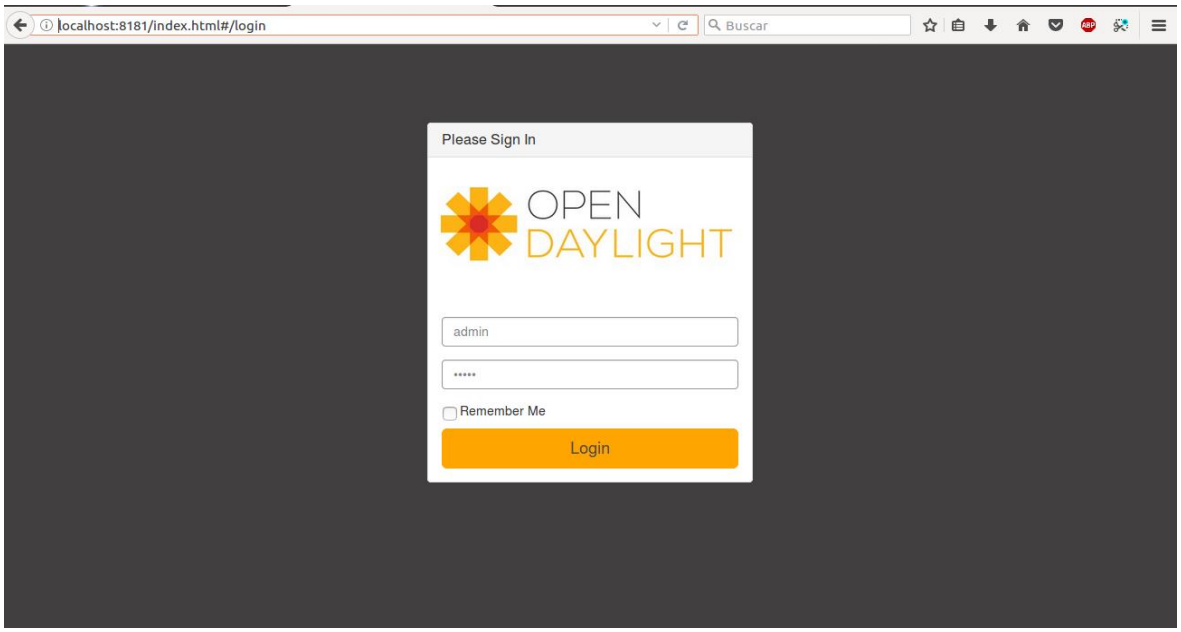

Ilustración 44. Dashboard OpenDaylight

8.5. VERIFICAR LA CONECTIVIDAD ENTRE PLATAFORMAS

En este procedimiento se verificará que exista conectividad entre las diferentes plataformas, con el objetivo de que se pueda realizar la integración entre las mismas.

Lo primero que debemos hacer, es crear rutas estáticas con la intención de que exista conectividad entre todos los elementos de la red, el comando básico para crear rutas estáticas es el siguiente:

```
# sudo route add –net <red destino>/mascara gw <IP acceso>
```

- **OSM**
  Creamos la ruta para acceder a la red interna que está dentro del contenedor VCA y también para acceder a la red de conjure-up que está dentro de OpenStack.

- **OpenStack y OpenDaylight**
  Desde OpenStack y OpenDaylight solo debemos crear la ruta estática para acceder a la red que está dentro de VCA.

Nota: Si se crea alguna red desde OpenStack y no se tiene conectividad a ella, la solución es crear la ruta estática hacia esa red.

8.6. INTEGRACIÓN ENTRE OPEN SOURCE MANO (OSM), OPENSTACK Y OPENDAYLIGHT

Primero se deben ejecutar una serie de pasos con el objetivo de preparar cada una de las plataformas para la integración.

8.6.1. Preparación OpenStack

Se debe crear una red de administración como se observa en la ilustración 45, además asegurarse de que OSM pueda acceder a esta red, para esto nos dirigimos al apartado Red/Redes (desde el administrador gráfico) y creamos una vlan, en estado de administrador, compartida y externa.

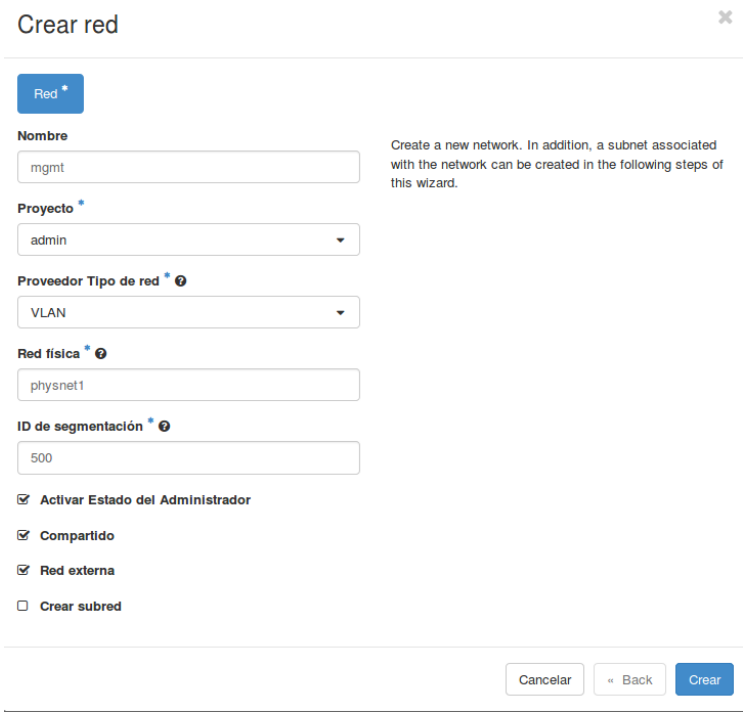

Ilustración 45. Creación red de administración

Se accede a la vlan que se acaba de crear y se agrega una nueva subred definiendo el nombre y la dirección de red; la puerta de enlace toma la primera dirección IP disponible de la subred por defecto.

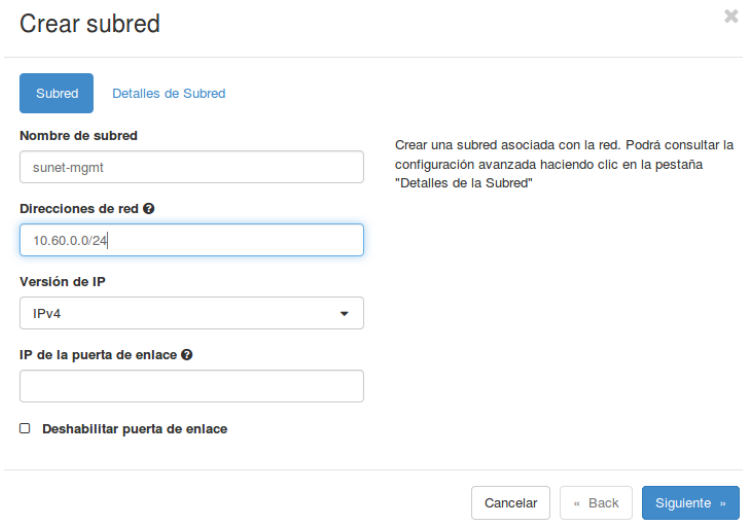

Ilustración 46. Creación de la subred

Por último, se verifica que la red y subred se hayan creado correctamente.

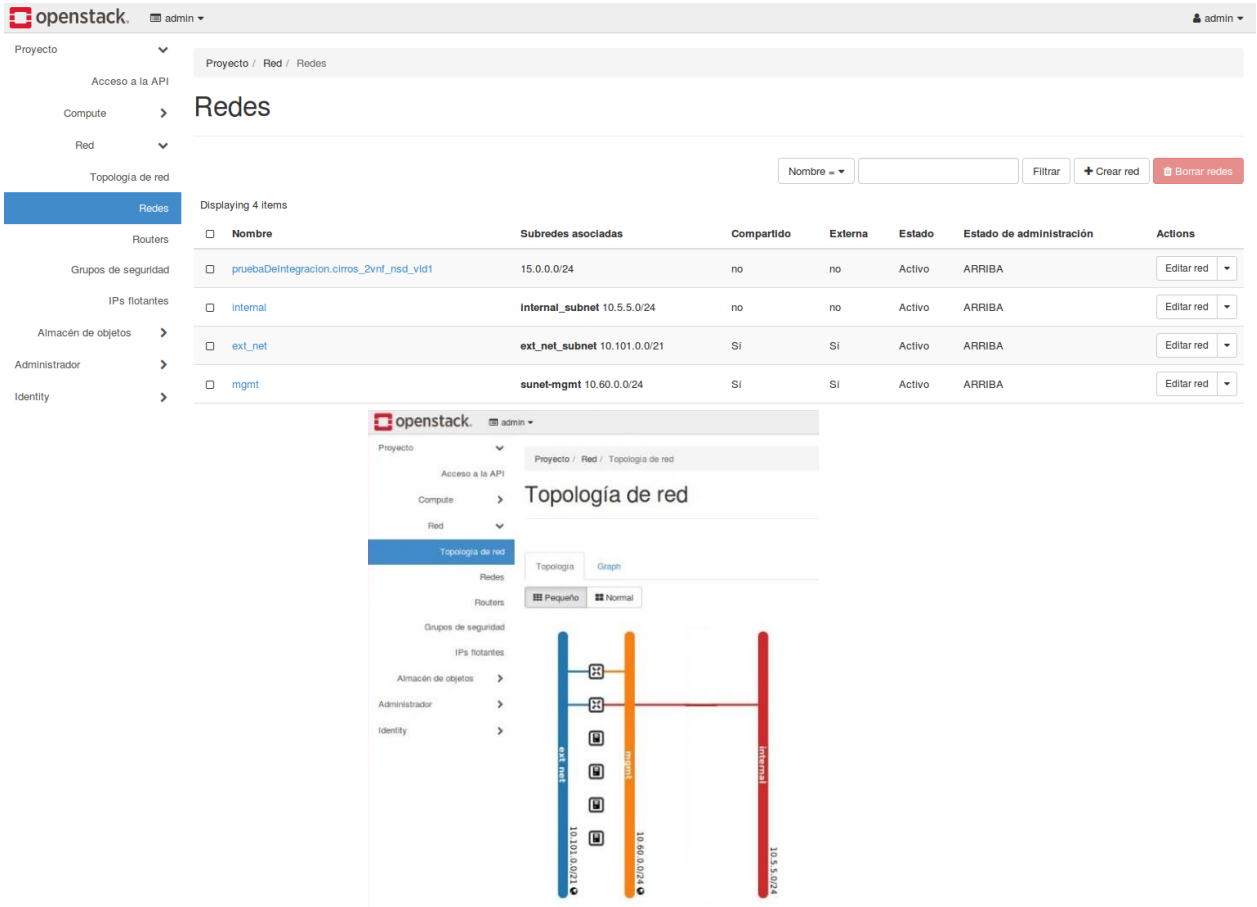

Ilustración 47. Verificación de las redes

Una vez creada la red de administración procedemos a cargar las imágenes de los

VNF, en este caso se subirá la imagen de cirrOS, la cual es una distribución mínima de Linux.

Para cargar la imagen nos dirigimos al apartado Admin/Images y se selecciona la opción “create image”, insertamos un nombre para la imagen, buscamos la imagen en el equipo y seleccionamos el formato, para el caso de CirrOs el formato es QCOW2 (ilustración 48).

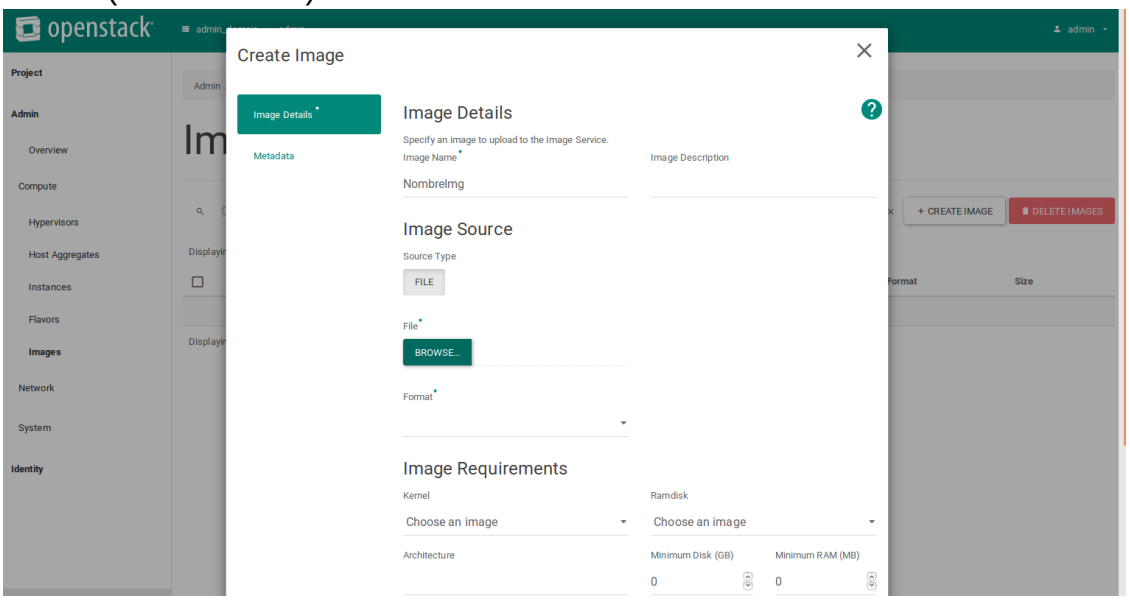

Ilustración 48. Crear imagen

Se verifica que la imagen se haya creado correctamente.

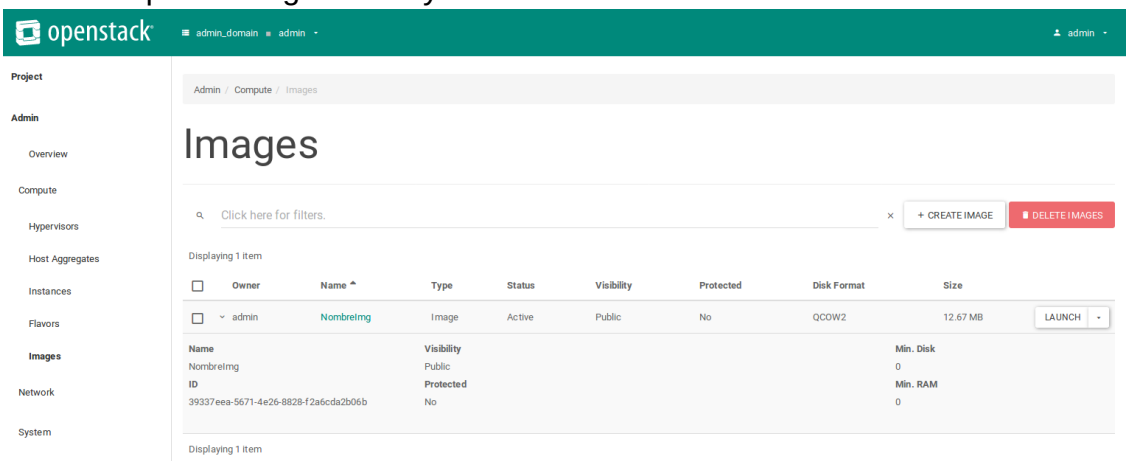

Ilustración 49. Verificación de la imagen

Por último, verificamos que el paquete SSH esté instalado y configurado en la

máquina de OpenStack, de no estarlo, lo instalamos como se muestra a continuación.

```
# sudo apt-get install ssh
```

8.6.2. Preparación OSM para integrar OpenStack

Accedemos al contendor RO que se encuentra dentro de la máquina de OSM

```
# lxc exec RO –bash
```

Creamos y exportamos el usuario que podrá crear instancias en OpenStack

```
# openmano create-tenant Admin
# export OPENMANO_TENANT=Admin
```

Por último, creamos el sitio de OpenStack y lo ligamos al usuario inquilino.

```
# openmano datacenter-create openstack-site
http://IP_ACCESO:5000/v2.0 --type openstack --description
"OpenStack site"
# openmano datacenter-attach openstack-site --user=admin
-- password=openstack --vim-tenant-name=admin
# openmano Datacenter-list
```

8.6.3. Preparación de OSM para integrar OpenDaylight

Accedemos al contenedor RO y exportamos el usuario, podríamos utilizar el mismo usuario creado anteriormente para OpenStack

Como ya se tiene un controlador configurado aplicamos el siguiente comando

```
# openmano sdn-controller-create opendayligh_SDN --
ip=10.84.130.4 --port=8181 --user admin --passwd admin -
-type odl
```

8.6.4. Integración entre OSM y OpenStack

Para integrar OSM y OpenStack ingresamos al administrador de OSM, accedemos en la pestaña *Accounts* y agregamos una nueva cuenta VIM (para el presente caso OpenStack), diligenciamos los datos correspondientes (la URL de autenticación se

encuentra en la sección API de OpenStack), agregamos la cuenta y verificamos que la conexión sea exitosa (ilustración 50).

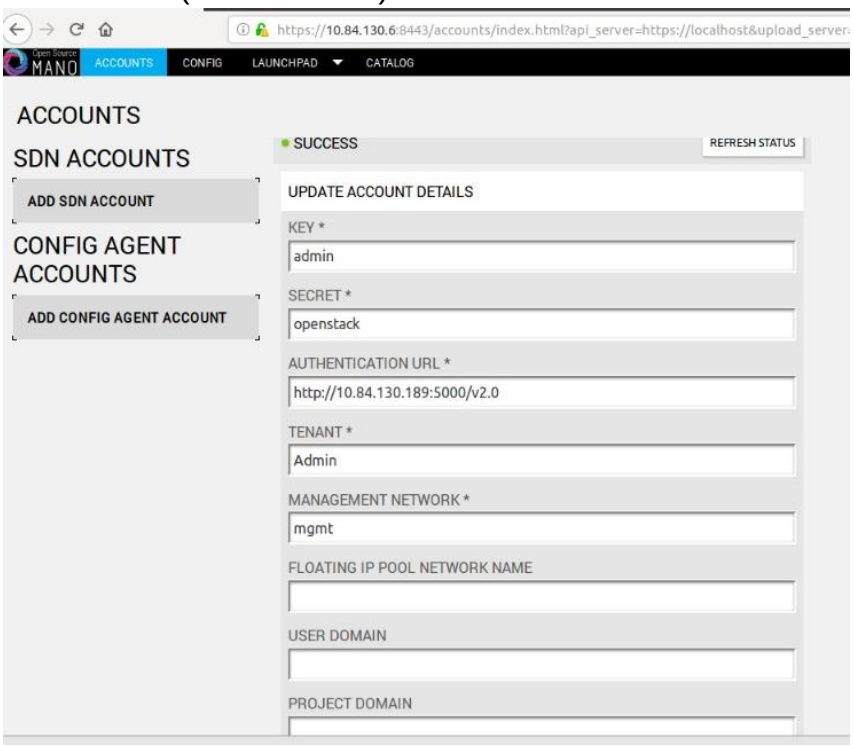

Ilustración 50. Integración de OSM con OpenStack

En la ilustración 51 se observa que desde OpenStack se haya creado una nueva red correspondiente a OSM.

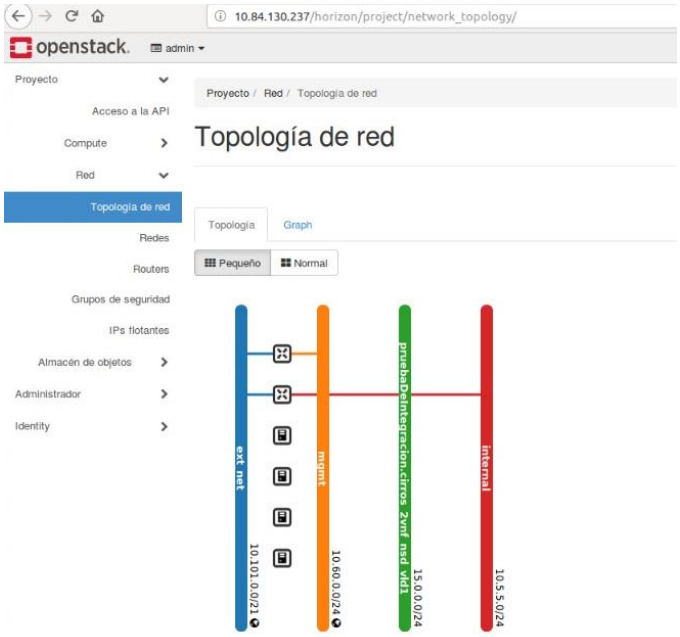

Ilustración 51. Verificación de la integración desde OpenStack

8.6.5. Integración entre OSM y OpenDaylight

Se ingresa a OSM y accedemos a la pestaña *Accounts,* y agregamos el controlador SDN, agregamos un nombre, la dirección URL con el puerto y los datos de acceso (ilustración 52). Una vez realizado esto se verifica que se haya conectado de forma exitosa.

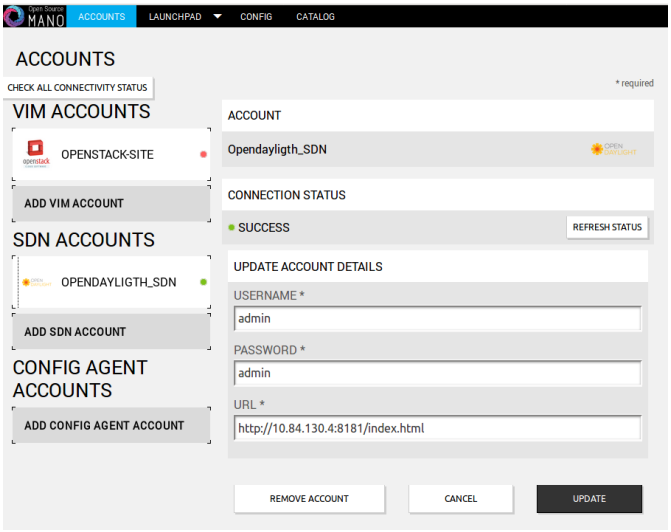

Ilustración 52. Integración de OSM con OpenDaylight

8.6.6. Despliegue de los servicios de red

Para la implementación de los servicios de red virtualizados se seleccionó inicialmente el despliegue de dos VNF simples basado en CirrOS, conectados por medio de un Virtual Link Descriptor (VLD), el VLD es una plantilla donde se describen los requisitos de recursos que se necesitan para crear un enlace. Se selecciona este servicio para comprobar la conectividad entre cada VNF.

Los paquetes de VNF y NS se descargarán directamente de la página de OSM https://osm-download.etsi.org/ftp/osm-3.0-three/examples/cirros_2vnf_ns/

Dentro de OSM accedemos al apartado de “Catalog” y subimos los paquetes correspondientes, estos pasos se observan en las ilustraciones 53 y 54.

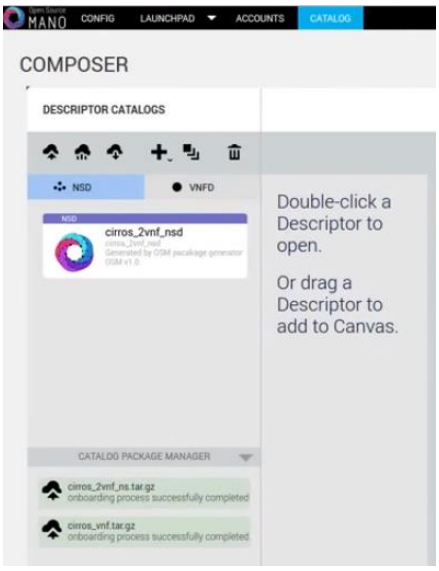

Ilustración 53. Agregar paquete VNFD

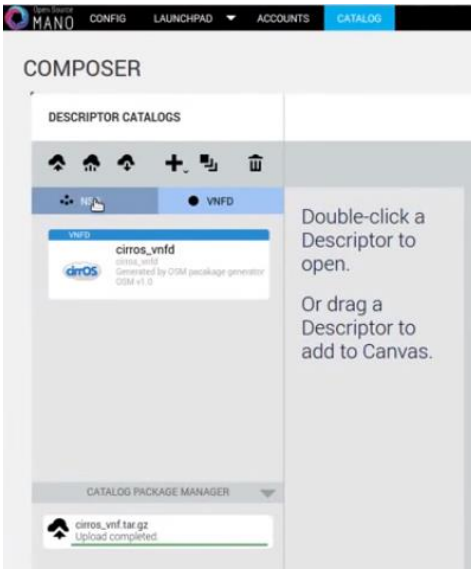

Ilustración 54. Agregar paquete NSD

En la ilustración 55 se evidencia que al hacer doble clic sobre el paquete alojado en NSD se puede observar la topología de los paquetes conectados por medio del VLD.

Ilustración 55. Topología VNF

Y al hacer doble clic sobre el paquete que se encuentra en VNFD se puede ver la imagen de CirrOS que hace posible el despliegue de los servicios.

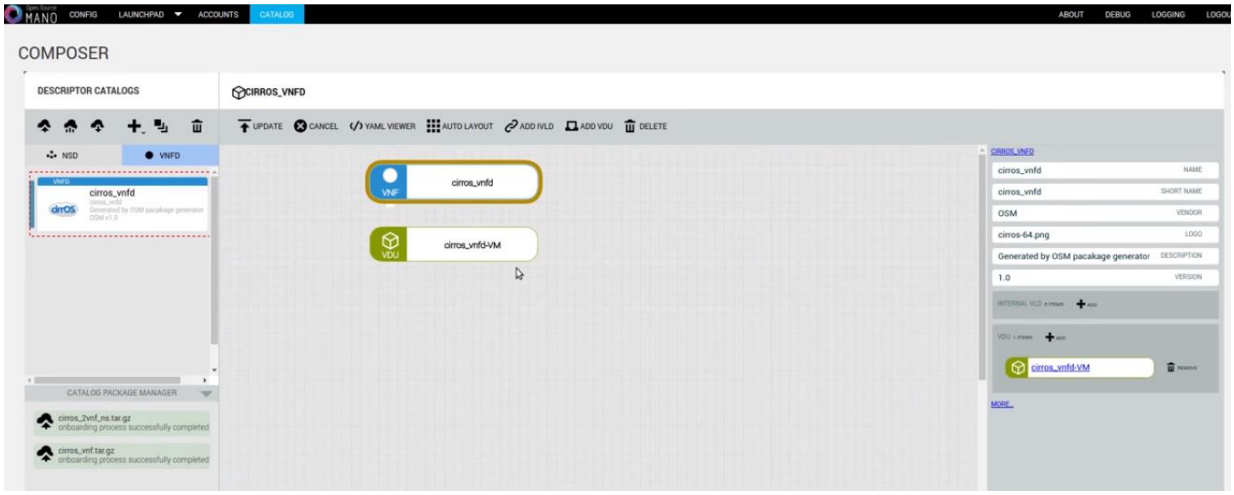

Ilustración 56. Composición paquete VNFD

Instanciamos el paquete VNF dentro del VIM OpenStack, para esto nos dirigimos al launchpad/instantiate, allí se puede observar el paquete, damos doble clic sobre él, seleccionamos el VIM que integramos anteriormente y esperamos a que se instancie el paquete (ilustración 57).

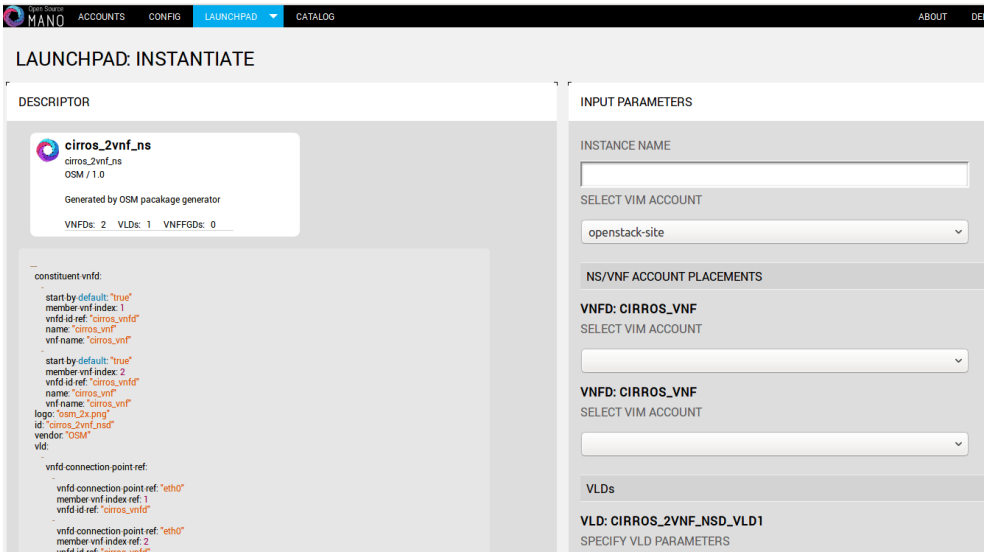

Ilustración 57. Instanciación paquete

Por último, verificamos que el servicio de red se haya instanciado correctamente (ilustración 58).

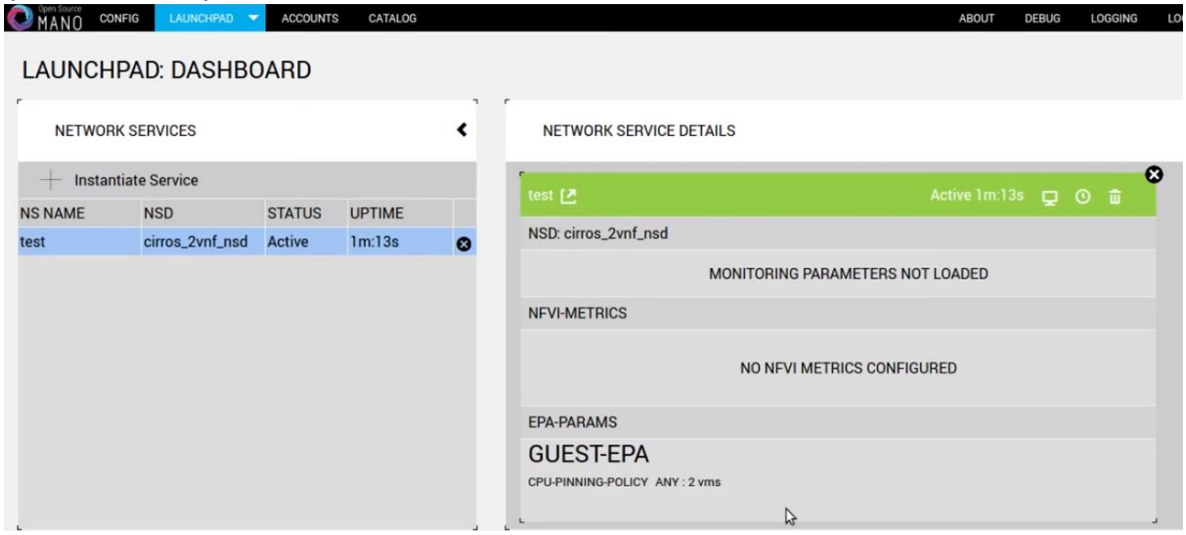

Ilustración 58. Servicio de red instanciado

Al ingresar dentro del servicio de red se puede observar el estado actual del mismo, los paquetes VNF que lo conforma e inclusive crear diferentes conexiones entre paquetes, tal como se muestra en la ilustración 59

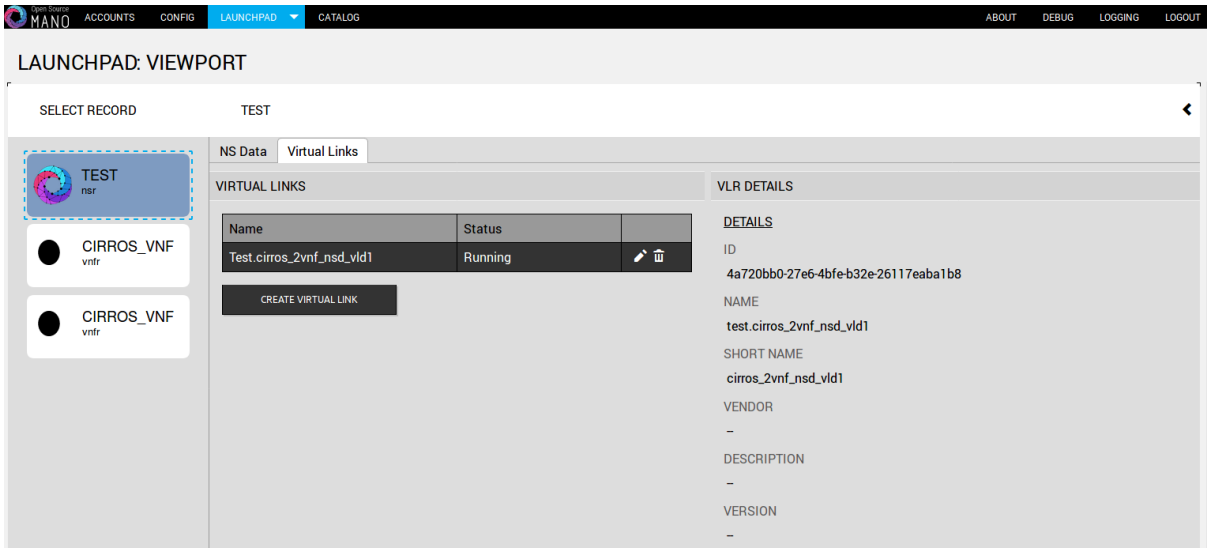

Ilustración 59. Detalles del servicio

Como paso final, ingresamos a OpenStack y verificamos que la instancia se haya creado e iniciado correctamente (ilustración 60), desde donde podemos hacer las configuraciones o cambios a nuestro servicio de red.

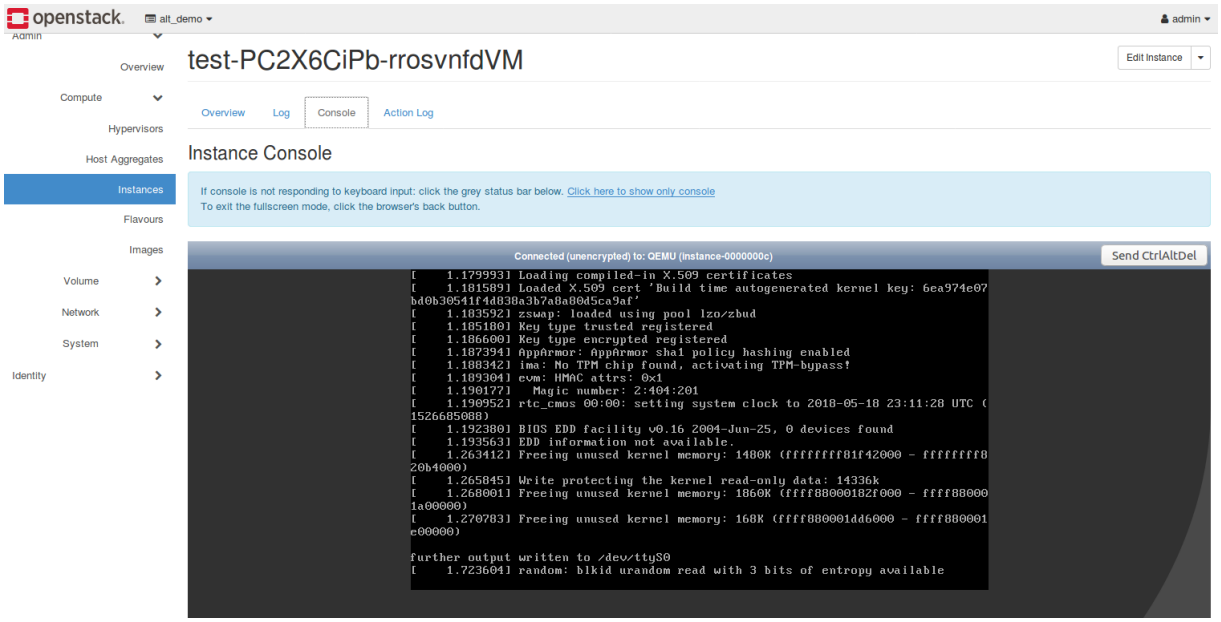

Ilustración 60. Servicio de red instanciado desde OpenStack

8.6.7. Creación de los servicios de red

Es posible la creación de nuestros servicios de red, gracias a la herramienta online creada por la compañía Rift.io, la cual puede ser accedida desde el enlace

https://riftio.com/osm-vnf-package-generator/ y permite desde un entorno gráfico crear los diferentes paquetes para ser instanciados desde OSM.

En la ilustración 61 se puede observar la herramienta OSM VNF Onboarding, en donde se ingresaría el nombre del paquete VNF, el tamaño de la instancia (memoria RAM, almacenamiento, procesador), nombre de la imagen (debe estar creada en OpenStack) y lo más importante el Script de configuración del servicio de red.

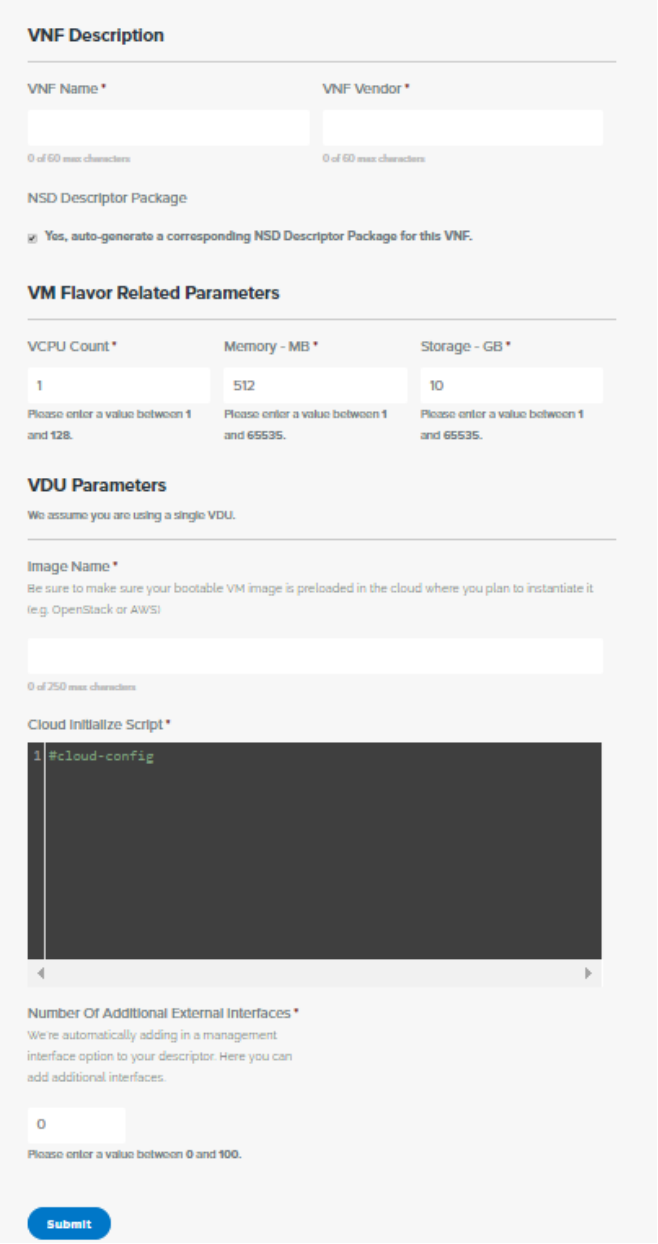

Ilustración 61. Herramienta OSM VNF Onboarding

9. RESULTADOS

En el transcurso del desarrollo del proyecto se alcanzó un nivel de conocimiento sobre los paradigmas de virtualización de funciones de red, redes definidas por software y computación en la nube, asimismo se amplió el conocimiento sobre las plataformas OSM, OpenDaylight y OpenStack, entendiendo sus arquitecturas, servicios, implementación y funcionalidades al momento de integrar cada una de las plataformas al ser aplicadas en un entorno de infraestructura como servicio.

Se implementan e integran las plataformas de OSM y OpenStack, tal como lo se muestra en las ilustraciones 62 y 63 respectivamente. Generando un despliegue de las funciones de red de forma más rápida, ágil, flexible y escalables, lo cual le brindó un valor agregado a la infraestructura de red.

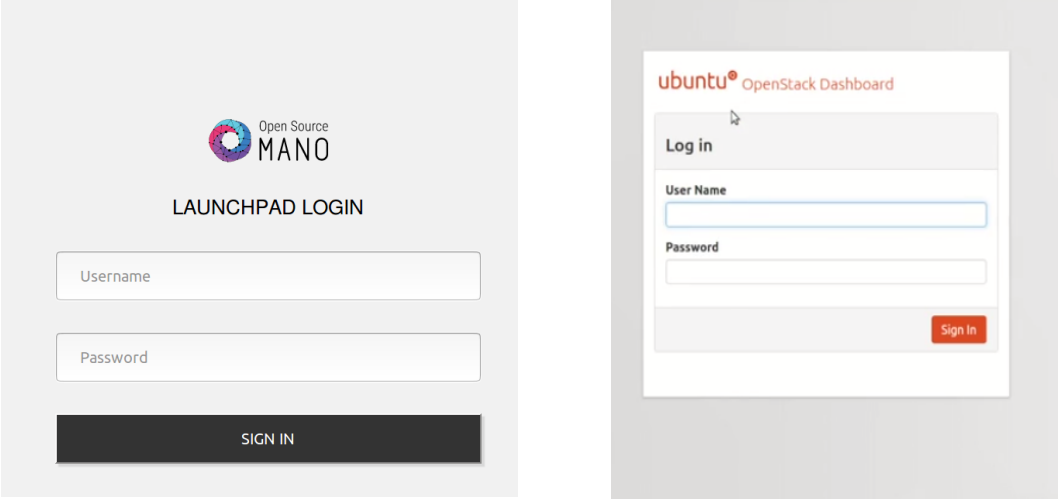

Ilustración 62. Implementación OSM y OpenStack

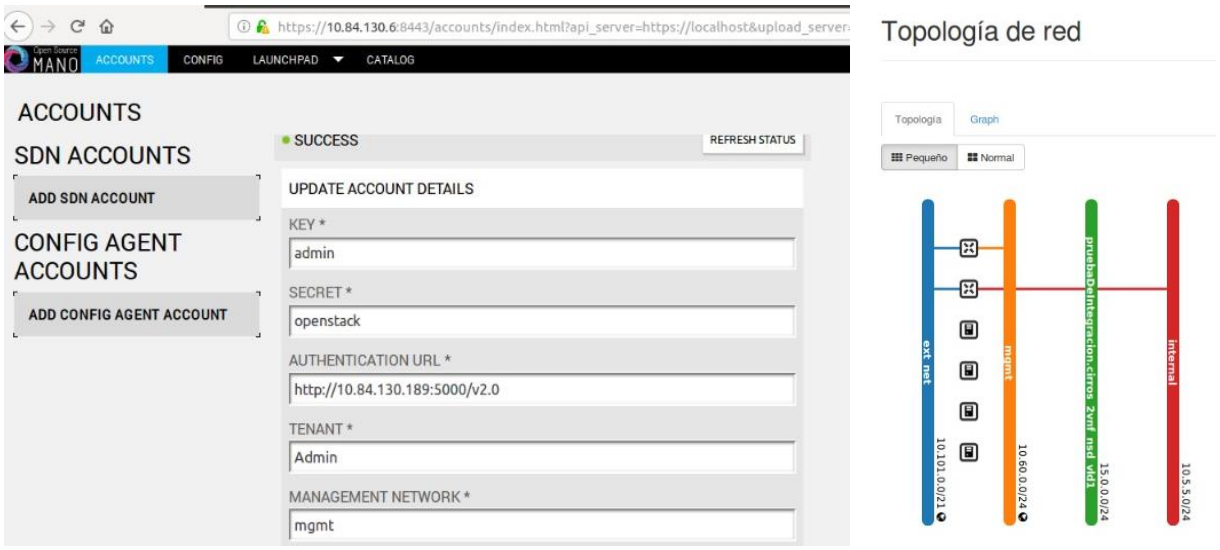

Ilustración 63. Integración de OSM con OpenStack

Se implementó el controlador OpenDaylight (ilustración 64) y posteriormente se integra con OSM (ilustración 65), agregando a la infraestructura la flexibilidad y automatización al momento de desplegar funciones de red, además con políticas de control y seguridad.

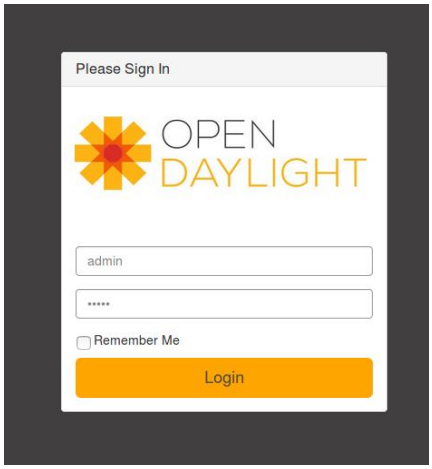

Ilustración 64. Implementación OpenDaylight

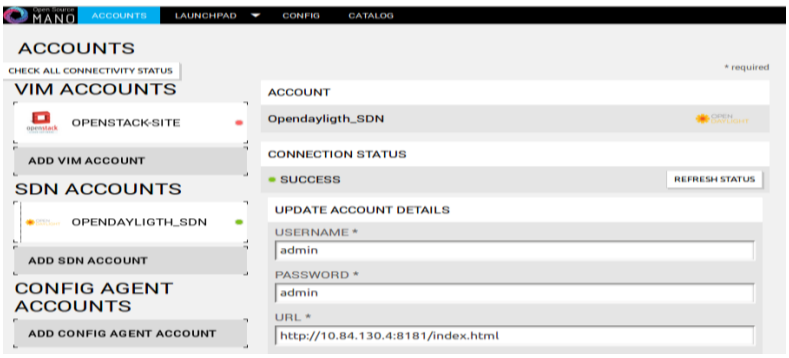

Ilustración 65. OSM con OpenDaylight

Al desplegar las funciones de red en una infraestructura como servicio con características de virtualización se agilizó e independizaron estas funciones del hardware propietario, generando un menor costo tanto de capital como de operación, asimismo, solo se utiliza una sola sintaxis para el despliegue de cada uno de estos.

Se implementaron dos VNF simples basados en CirrOS, conectados por medio de un Descriptor de enlace virtual (VLD, siglas en inglés), permitiendo comprobar la conectividad entre cada VNF. El despliegue de este servicio de red se puede observar en la ilustración 66 y 67, respectivamente.

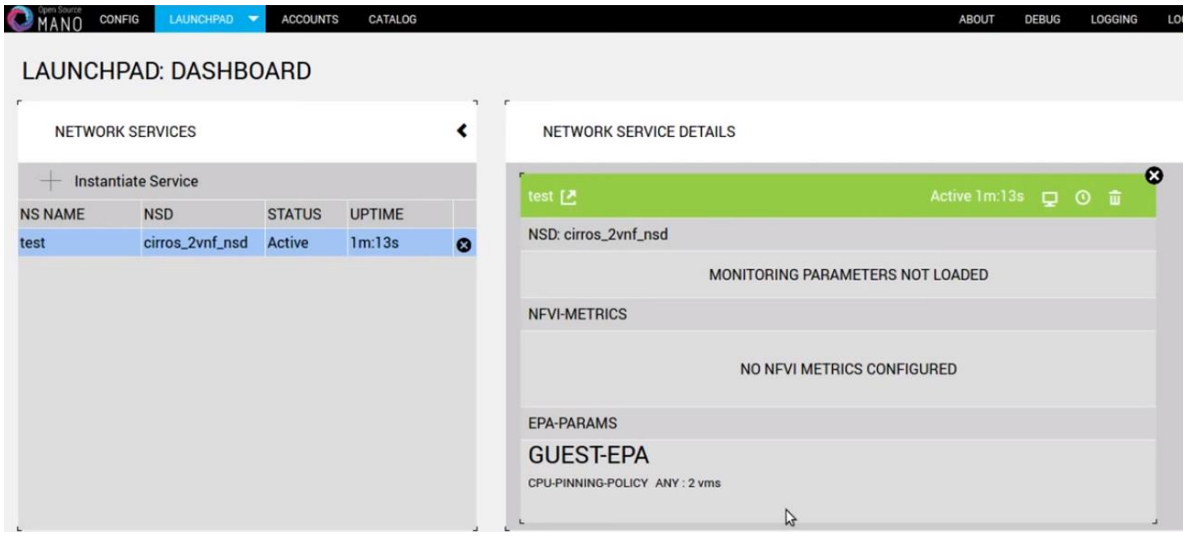

Ilustración 66. Despliegue servicio de red

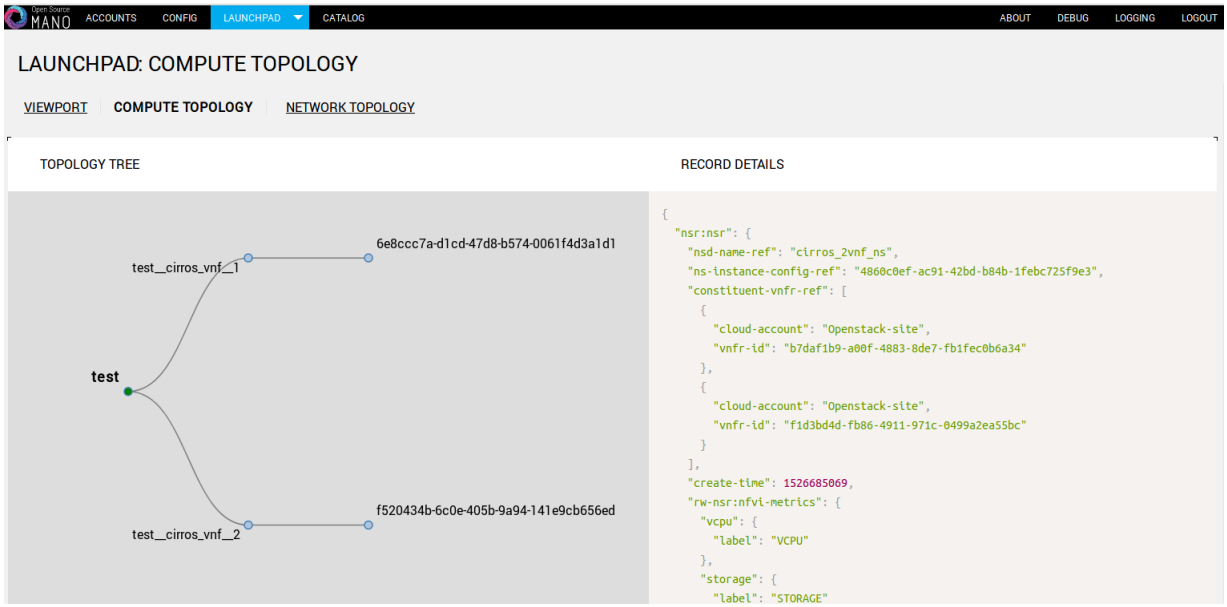

Ilustración 67. Conexión entre VNF

10. CONCLUSIONES

NFV y SDN tienen mucho en común, debido a que ambos están orientados a trabajar con software libre y hardware estándar, aunque se debe tener claro que para implementar NFV no es necesario implementar las SDN.

Es posible la integración entre las plataformas de OSM, OpenDaylight y OpenStack. Pero se debe tener presente que al ser plataformas o herramientas de terceros se presentan bugs, sobre los cuales no teníamos ningún control, por consiguiente mientras se lanzaban un nuevo release resolviendo dicho problema esto nos demoraba en el transcurso del desarrollo del proyecto.

Al implementar una infraestructura como servicio a nivel de laboratorio, se encontró la dificultad de que los computadores a pesar de que contaban con los recursos recomendados por cada plataforma presentaban dificultad en ciertos puntos del proyecto, por ejemplo, al momento de instanciar un servicio de red, puesto que son procesos que consumen gran cantidad de recursos de la máquina. Asimismo, se presentó la dificultad de no contar con una fuente de suministro eléctrico (UPS) dado que cada vez que se reiniciaba la maquina se presentaba gran demora al volver a cargar los servicios.

Al integrar NFV con SDN se aprovecha la automatización y virtualización ofrecidos por ambos, generando una solución de red con un gran valor. Las SDN se pueden implementar como una función de red virtualizada (VNF) beneficiándose de las características de fiabilidad y elasticidad de NFV.

Al integrar NFV con la computación en la nube, permite desplegar de una forma más rápida, ágil y flexible nuevos servicios de red, en vez de implementarlo en hardware especializado. Una vez que se tenía la infraestructura de la nube, se pueden agregar nuevos servicios con software y escalar el centro de datos a medida que aumentan sus clientes.

La realización del proyecto genero aportes valiosos de documentación en cuanto a la implementación e integración de una infraestructura como servicio a nivel de laboratorio, utilizando herramientas de virtualización como lo son OSM, OpenDaylight y OpenStack, y que pueden ser de utilidad para futuros trabajos.

11. RECOMENDACIONES

Debido a los inconvenientes causados al momento de implementar OpenStack en un solo nodo, se puede decir que es más estable la instalación de la plataforma en varios nodos o sistemas, separando controladores tanto de red y computación en distintas áreas, con el objetivo de solucionar problemas de puedan surgir al momento de la instalación, dado que si se presenta algún inconveniente en un nodo solo se tendría que volver a instalar o modificar dicho nodo y no empezar la instalación desde cero de todo el sistema.

Al momento de generar una instancia ya sea desde OSM u OpenStack se debe tener en cuenta el formato y metadatos de la imagen a subir, puesto que un mal uso de las extensiones IMG, Qcow2, RAW, ISO, entre otras, pueden generar conflictos al momento de instanciar, ya sea por compatibilidad o por permisos de acceso.

Se recomienda utilizar una red interna para hacer laboratorios y pruebas generales, dado que al momento de comprobar conectividad y funcionalidad, es muy útil tener control con las direcciones IP que se utilizan, así se evita confusiones y se tiene certeza de que no hay una dirección IP duplicada.

BIBLIOGRAFÍA

[1] P. Patel, V. Tiwari, and M. K. Abhishek, "SDN and NFV integration in openstack cloud to improve network services and security," in *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 2016, pp. 655-660.

[2] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 236-262, 2016.

[3] G. Xilouris, E. Trouva, F. Lobillo, J. M. Soares, J. Carapinha, M. J. McGrath*, et al.*, "T-NOVA: A marketplace for virtualized network functions," in *2014 European Conference on Networks and Communications (EuCNC)*, 2014, pp. 1-5.

[4] A. Manzalini and N. Crespi, "SDN and NFV for Network Cloud Computing: A Universal Operating System for SD Infrastructures," in *2015 IEEE Fourth Symposium on Network Cloud Computing and Applications (NCCA)*, 2015, pp. 1-6.

[5] SDXCentral, in *https://www.sdxcentral.com/nfv/definitions/whats-network-functions-virtualization-nfv/*, ed.

[6] M. Chiosi., D. Clarke, P. Willis, A. Reid., J. Feger, M. Bugenhagen*, et al.*, "Network Functions Virtualisation," *SDN and OpenFlow World Congress,* 2012.

[7] J. d. J. G. Herrera and J. F. B. Vega, "Network Functions Virtualization: A Survey," *IEEE Latin America Transactions,* vol. 14, pp. 983-997, 2016.

[8] SDXCentral., "2017 NFV Report Series Part I Foundations of NFV: NFV Infrastructure and VIM," *MARKET REPORT,* 2017.

[9] VMware. Available: http://www.vmware.com/latam/products/vsphere.html

[10] KVM. Available: https://www.linux-kvm.org/page/Main_Page

[11] SDXCentral.. "2017 NFV Report Series Part 2: Orchestrating NFV - MANO and Service Assurance," *MARKET REPORT,* 2017.

[12] SDXCentral... Available: https://www.sdxcentral.com/nfv/definitions/nfv-cloud/

[13] U. Moghe, P. Lakkadwala, and D. K. Mishra, "Cloud computing: Survey of

different utilization techniques," in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, 2012, pp. 1-4.

[14] P. Mell and T. Grance, "The NIST Definition of Cloud Computing " *National Institute of Standards and Technology,* 2011.

[15] S. Joshi and U. Kumari, "Load balancing in cloud computing: Challenges & issues," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 2016, pp. 120-125.

[16] OpenStack. Available: https://www.openstack.org/

[17] OpenStack. Available: https://docs.openstack.org/ops-guide/preface.html#introduction-to-openstack

[18] M. Bist, M. Wariya, and A. Agarwal, "Comparing delta, open stack and Xen Cloud Platforms: A survey on open source IaaS," in *2013 3rd IEEE International Advance Computing Conference (IACC)*, 2013, pp. 96-100.

[19] T. Rosado and J. Bernardino, "An overview of openstack architecture," presented at the Proceedings of the 18th International Database Engineering & Applications Symposium, Porto, Portugal, 2014.

[20] R. Kumar, N. Gupta, S. Charu, K. Jain, and S. K. Jangir5, "Open Source Solution for Cloud Computing Platform Using OpenStack," *International Journal of Computer Science and Mobile Computing,* vol. 3, pp. 89 – 98, Mayo-2014 2014.

[21] O. N. Foundation. Available: https://www.opennetworking.org/sdn-resources/sdn-definition

[22] L. Lin and P. Lin, "Software-Defined Networking (SDN) for Cloud Applications," *Springer International Publishing Switzerland,* 2014.

[23] D. Meyer, "The Software-Defined-Networking Research Group," *IEEE Computer Society,* 2013.