UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MÁSTER

INGENIERÍA EN INFORMATICA

Diseño e implementación de una propuesta de Network Slicing

Autor
Freddy Javier Frere Quintero

Directores
Jorge Navarro Ortiz
Juan José Ramos Muñoz

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 14 septiembre de 2018

Diseño e implementación de una propuesta de Network Slicing

Autor
Freddy Javier Frere Quintero

Directores
Jorge Navarro Ortiz
Juan José Ramos Muñoz

Diseño e implementación de una propuesta de Network Slicing

Freddy Javier Frere Quintero

Palabras clave: virtualización de redes, orquestación, MANO, Open Source Mano, OpenStack, NFV, VIM, VNF, Network Slicing

Resumen

El futuro de las redes 5G pasa por adoptar tecnologías que permitan adaptarse de forma ágil a las nuevas aplicaciones y servicios a los que le darán soporte.

Una de las aproximaciones que está adoptando la industria es la de virtualizar los recursos de red. De esta manera, se ahorran costes, al ejecutar sobre servidores de cómputo de propósito general los elementos que hasta hace poco eran hardware, tales como *routers*, *switches*, etc. Esta infraestructura reduce costes tanto de despliegue como de mantenimiento. Además, es fácil crear nuevas máquinas virtuales cuando un servicio esté saturado, o configurar y desplegar rápidamente un nuevo servicio que el operador quiera ofrecer en su infraestructura.

A este respecto, se están proponiendo marcos estándares para virtualizar funciones de red (Network Function Virtualization, NFV), servicios de red (compuestos por varias VNFs), y automatizar las funciones del ciclo de vida de esos servicios (instanciarlos, monitorizar su rendimiento, escalarlos, etc.) mediante el marco de gestión y orquestación de funciones de red virtualizadas (Management and Orchestration, MANO).

Otra de las tecnologías en las que se basan las propuestas actuales para la virtualización de redes es el uso de *Network Slicing*, en las que una infraestructura de red virtualizada se parcela en *slices*, que se ofrecen a distintos tipos de servicio. Cada slice debe gestionar los recursos virtualizados de forma que no interfieran entre ellos, ofreciendo a cada servicio la visión de que la infraestructura subyacente es solo para él.

En este proyecto se pretende diseñar, implementar y desplegar un escenario en el que varios servicios de red comparten una misma infraestructura virtual, gestionados por el marco MANO.

Como resultado, en el presente trabajo se ha estudiado, instalado y configurado un gestor de infraestructura virtual (OpenStack), un gestor y orquestador de servicios virtualizados para soportar Network Slicing (Open Source Mano), y scripts para gestionar la instanciación automática de dos servicios, en caso de caída de los mismos. Concretamente, un servidor de páginas web, y un servidor LoRa (*Long Range*) para Internet de las cosas, ambos ejecutados en una configuración tipo *slice*. Finalmente, se ha evaluado

y documentado el proceso de instalación y configuración, proporcionando así una referencia para futuros trabajos con OpenStack, NVF, y Open Source Mano.

Design and implementation of a Network Slicing proposal

Freddy Javier Frere Quintero

Keywords: network virtualization, orchestration, MANO, Open Source Mano, OpenStack, NVF, VIM, VNF, Network Slicing

Abstract

The future of 5G networks involves adopting technologies that allow them to adapt agilely to the new applications and services they will support.

One of the approaches being adopted by the industry is to virtualize network resources. In this way, costs are saved, by running in compute servers on general purpose, items that until recently were hardware, such as *routers*, *switches*, etc. This infrastructure reduces costs of deployment and maintenance. Also, it is easy to create new virtual machines when a service is saturated, or quickly configure and deploy a new service that the operator wants to offer in his infrastructure.

In this regard, standard frameworks are being proposed to virtualize network functions (Network Function Virtualization, NFV), network services (composed of several VNFs), and automate the life cycle functions of these services (instantiate them, monitor their performance, scale, etc.) through the management and orchestration framework of virtualized network functions (Management and Orchestration, MANO).

Another technology on which the current proposals for network virtualization are based is the use of *Network Slicing*, in which a virtualized network infrastructure is divided in *slices*, which are offered to different types of service. Each slice must manage the virtualized resources in a way that does not interfere with each other, offering each service the vision that the underlying infrastructure is only for it.

This project aims to design, implement and deploy a scenario in which several network services share the same virtual infrastructure, managed by the MANO framework.

As a result, in the present work we have studied, installed and configured a virtual infrastructure manager (OpenStack), a virtualized services manager and orchestrator to support Network Slicing (Open Source Mano), and scripts to manage the automatic instantiation of two services, in case of falling them. Specifically, a server of web pages, and a server LoRa (*Long Range*) for the Internet of things, both executed in a configuration type *slice*. Finally, the installation and configuration process has been evaluated and documented, providing a reference for future work with OpenStack, NVF, and Open Source Mano.

Yo, **Freddy Frere Quintero**, alumno de la titulación TITULACIÓN de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con NIE XXX, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Freddy Javier Frere Quintero

Granada, a 14 días del mes de septiembre del 2018 .

D. **Jorge Navarro Ortiz** , Profesor del Área de Telecomunicaciones del Departamento Teoría de la señal, Telemática y Comunicaciones de la Universidad de Granada.

D. **Juan José Ramos Muñoz** , Profesor del Área de Telecomunicaciones del Departamento Teoría de la señal, Telemática y Comunicaciones de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Diseño e implementación de una propuesta de Network Slicing***, ha sido realizado bajo su supervisión por **Freddy Javier Frere Quintero**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 14 de mes septiembre de 2018.

Los directores:

Jorge Navarro Ortiz **Juan Ramos Muñoz**

Agradecimientos

A Dios por permitirme alcanzar este objetivo, a mi familia por sus consejos, por la motivación constante y por ser ejemplo de perseverancia.

A mi enamorada, por ser mi apoyo y formar parte de esta meta de mi vida.

Y a mis tutores, Juan José Ramos y Jorge Navarro por darme la oportunidad de realizar este proyecto, y haber tenido la paciencia y compromiso para ayudarme en lo que requiriera.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Las redes desde sus inicios han brindado la facilidad de ser el canal por el cual los diferentes servicios puedan proveerse a múltiples usuarios, convirtiéndose en algo indispensable en la actualidad. Esto, ha obligado a distintos entes proveedores de servicios de red a mantener una evolución constante en la arquitectura y en los servicios que ofrecen, resaltando el progreso más notorio en el aumento de la velocidad de trasmisión de datos, lo que ha permitido a los usuarios disponer de múltiples servicios.

Un ejemplo de esto, es la cantidad de usuarios que se han integrado a las redes de comunicación a través de dispositivos móviles, y que continúa creciendo de manera acelerada en los últimos años. En el año 2015 hubo 4.430 millones de usuarios de móviles en el mundo; mientras que, al finalizar el 2017 la cifra ascendió a 5.000 millones, lo que significa que el 66 % de la población mundial ya cuenta con un dispositivo móvil[1][2].

Esto significa, que hay más tráfico de datos que se debe controlar y direccionar, y que genera complicaciones en las arquitecturas de redes, en las técnicas de direccionamiento y en las pasarelas de datos. Además, incrementa los costos de mantenimiento y gestión de los equipos que hoy en día son de propietarios privados, generando a su vez, un incremento en los costos de dichos servicios a los usuarios.

Según el informe de Cisco Visual Networking Index (VNI) sobre Tráfico Global de Datos Móviles 2016-2021 [3] en el 2016 el tráfico global de datos alcanzó los 7 EB (7.000 millones de gigabytes) mensuales; mientras que, para el 2021 han pronosticado que debido al incremento exponencial de usuarios móviles, smartphones y conexiones del Internet of Things (IoT), el tráfico global de datos alcanzará los 49 EB (49.000 millones de gigabytes) mensuales y 587 EB anuales, resultando en un incremento interanual del 47 % entre 2016 y 2021.

Ha sido un problema constante en las redes de comunicaciones, reducir el

tráfico y prever los cuellos de botellas; por lo que, se han venido analizando varias alternativas como medidas para resolver estos obstáculos, pero en el núcleo de las redes no se han realizado cambios fundamentales.

La virtualización de redes y de funciones de red es uno de los pilares en lo que se apoyan las propuestas de la industria para mejorar las redes y así, poder desplegar las próximas redes 5G. Por esto, en la actualidad, organismos de estandarización como la ETSI (European Telecommunications Standards Institute), están definiendo un marco para la gestión y despliegue automáticos de funciones y servicios de red virtualizados: NFV-MANO (Network Functions Virtualization Management and Orchestration).

NFV-MANO, es un framework que se ha planteado como solución para afrontar el futuro de las comunicaciones, ya que nos permite administrar y orquestar todos los recursos de red en la nube. En el corto tiempo que tiene en el medio, se ha ganado una gran aceptación dentro de los centros de investigación, universidades, empresas públicas y privadas, debido a su adaptabilidad en funcionamientos y eficiencia en el coste-beneficio.

Utilizar NFV-MANO como parte del núcleo de las redes, permitirá a los administradores tener un mejor control sobre su tráfico, agilizando sus comunicaciones y aumentando el rendimiento de sus servicios, siendo capaz de soportar una demanda muy alta de usuarios y un enorme flujo de datos. Esto, evitará depender de un núcleo estático, sino de uno configurable, de fácil creación y re-ubicación.

Sin embargo, los operadores y proveedores de infraestructura de red, quienes ejecutarán en sus instalaciones las funciones y servicios virtualizados, necesitan proporcionar un nivel adicional de aislamiento para dichos servicios. Eso implica definir políticas dinámicas de asignación de recursos, de aislamiento de recursos virtuales y físicos, y gestión y orquestación de despliegues de funciones y servicios que convivan en infraestructuras compartidas. A la tecnología que hace posible lo anterior se le llama Network Slicing.

En el presente trabajo, estudiaremos su implementación en las redes de comunicación actuales y futuras. Buscando una solución para el tratamiento del flujo de datos y optimizando los recursos físicos, los cuales son el hardware y el software. Nace la virtualización de la red, con el fin de resolver los problemas de infraestructura tecnológica, aplicando técnicas de virtualización se conseguirá agrupar diversos equipos de varias características dentro de servidores de alto rendimiento con un sistema de almacenamiento alojados en centros de datos.

1.1. Motivación

El servicio de red en la actualidad se halla en cambios constantes, apuntando a una mejor administración y siendo más flexible, eficiente, seguro y escalable, debido a que tiene que adaptarse a los requerimientos modernos de los servicios, generándose una directriz hacia los estudios de la Virtualización de Redes y Redes Programables en estos últimos años.

Estas tecnologías se han ido desarrollando con el objetivo de incrementar el uso de los recursos de red e ir reduciendo los costos de la implementación; como propuestas se han planteado varios conceptos de aplicaciones a diferentes entornos de red, por lo que, existe la necesidad de realizar estudios pertinentes de la utilización en medios específicos, con el fin de analizar su beneficio y seguridad.

El planteamiento sobre la flexibilidad operacional de las redes, específicamente NFV-MANO, provoca una serie de incógnitas referentes a la confiabilidad, rendimiento, seguridad, orquestación, escalabilidad, dónde emplearlo, etc. De igual manera, sobre qué es lo que se puede o no virtualizar y la ubicación en donde se situarían las funciones de red para mejorar el rendimiento de las mismas. Todas estas incógnitas se pueden aplicar a diferentes contextos de casos de uso de la tecnología NFV-MANO.

1.2. Objetivos

El presente trabajo tiene como objetivo principal diseñar, configurar la infraestructura necesaria para desplegar dos servicios de red independientes. Dicho despliegue debe basarse en estándares de gestión y orquestación de redes y funciones de red virtuales, tales como los marcos de trabajo que definen MANO *Management and Orchestration* y VIM *(Virtualized Infrastructure Manager)*. Además, los servicios desplegados deben ser monitorizados para detectar si cae alguno de ellos, y reiniciados automáticamente.

Para ello habrá que satisfacer los siguientes subobjetivos:

- Estudiar, instalar y configurar una implementación de NFV-MANO.
- Estudiar, instalar y configurar una implementación de VIM.
- Ser capaces de instanciar servicios en un VIM.
- Ser capaces de crear *slices* distintas desde un servicio MANO a un servicio VIM.
- Validar la implementación.

1.3. Estructura del Trabajo

Este trabajo incluye un estudio sobre la tecnología NFV-MANO, teniendo como centro principal, el estudio de la administración y orquestación de los recursos de red en la nube. Para ello se ha dividido en ocho capítulos, que son detallados a continuación:

- **Capítulo 1** – *Introducción*: En este capitulo se detalla la situación actual sobre el uso de las redes y como ha ido transformando su tecnología para solventar la alta demanda de servicios generados por los usuarios.
- **Capítulo 2** – *Antecedentes*: En este capítulo se detallan los conceptos previos necesarios para poder entender el desarrollo del proyecto.
- **Capítulo 3** – *Estado del Arte*: En este capítulo se detallan los proyectos relacionados con el TFM a realizar, además, se estudiarán ciertas herramientas que hay en el actual mercado.
- **Capítulo 4** – *Planificación y Presupuesto*: En este capítulo se elabora una planificación de los recursos y presupuestos que implican la realización del proyecto, adicionalmente se incluye un presupuesto total.
- **Capítulo 5** – *Herramientas Usadas*:En este capítulo se justifica la elección de las herramientas utilizadas para el desarrollo del proyecto, a través de un estudio de las mismas.
- **Capítulo 6** – *Diseño e Implementación*:En este capítulo se realiza el esquema finalmente implementado, en el que se describe la instalación de OpenStack, la instalación de Open Source Mano y cómo se interconectan.
- **Capítulo 7** – *Validación*:En este capítulo se valida el éxito del despliegue de los servicios en las diferentes herramientas que se utilizan para el proyecto.
- **Capítulo 8** – *Conclusiones*:Es este capítulo se detalla el cumplimiento de los objetivos planteados.
- **Bibliografía**
- **Apéndices** – *Apéndice A*: Instalación paso a paso de DevStack.

Capítulo 2

Antecedentes

Para poder entender el desarrollo del proyecto es necesario definir previamente algunos conceptos claves relacionados a la virtualización.

2.1. Virtualización

Definimos como virtualización al proceso de crear con la ayuda de un software, un entorno informático simulado o versiones virtuales de ciertos recursos tecnológicos, que pueden ser: versiones de hardware, sistemas operativos, dispositivos de almacenamiento y recursos de red[4].

El objetivo de la virtualización es particionar un servidor físico en varios virtuales, permitiendo que interactúen de manera independiente, mejorando así, las cargas de trabajo y brindando mayor escalabilidad. Entendiéndose como "escalabilidad" a la propiedad de incrementar la capacidad de trabajo de un programa o sistema sin arriesgar el correcto funcionamiento y la calidad del mismo[5]. Además, la virtualización reduce el uso de los recursos físicos, el coste en mantenimiento y el consumo energético.[6]

Dentro de las áreas donde se puede aplicar la virtualización, este se divide en cinco categorías principales:

- **Virtualización de Almacenamiento**: Consisten en combinar varios recursos de almacenamiento en red en un solo dispositivo, creando un centro de almacenamiento que pueda ser administrado por una vista única siendo accesible por varios usuarios.
- **Virtualización de Red**: Consiste en segmentar el ancho de banda de una red en canales independientes, permitiendo que funcionen sobre una infraestructura común, para luego ser asignadas a servidores o dispositivos específicos.

- **Virtualización de Software**: Consiste en ejecutar aplicaciones dentro de un entorno controlado, separándolas del hardware y el sistema operativo.
- **Virtualización de Escritorio**: Consiste en que un servidor centralizado pueda ofrecer y administrar escritorios individualizados.
- **Virtualización de Servidores**: Consiste en el uso de un software que permite ejecutar varios sistemas operativos, como invitados en un único servidor físico. Cada sistema operativo se ejecutaría sobre su propia máquina virtual, usando parte de los recursos informáticos del servidor subyacente.

2.1.1. Virtualización de Redes

La virtualización de red, es la combinación de los recursos de red del hardware con los recursos de red del software, permitiendo que todo se administre de forma centralizada y ofreciéndonos eficacia al momento de experimentar aumentos repentinos e impensados de datos y aplicaciones que requieren de mucha capacidad.

Tiene como objetivo proveer el uso compartido de los recursos de redes de forma eficaz, controlada y segura para los usuarios y los sistemas. El resultado de la virtualización de redes es la red virtual. Las redes virtuales pueden ser externas o internas.

Oracle[7] define las redes virtuales externas e internas de la siguiente manera:

*"Las **redes virtuales externas** constan de varias redes locales que el software administra como una única entidad. Las partes que componen las redes virtuales externas clásicas son el hardware de conmutación y la tecnología de software VLAN. Entre los ejemplos de redes virtuales externas, se incluyen las grandes redes corporativas y los centros de datos.*

*Las **redes virtuales internas** constan de un sistema que usa zonas o máquinas virtuales configuradas en, al menos, una pseudointerfaz de red. Estos contenedores pueden comunicarse entre sí como si estuvieran en la misma red local, por lo que proporcionan una red virtual en un único host. Las partes que componen la red virtual son las tarjetas de interfaz de red virtual, o NIC virtuales (VNIC), y los conmutadores virtuales."*

Las redes continuamente se encuentran acumulando gran cantidad de dispositivos hardware de propietarios. Por ende, si se desea agregar un nuevo servicio de red (es decir, un servicio compartido a través de una infraestructura interconectada), es necesario adquirir el equipo correspondiente, además de disponer de un espacio para ubicarlo; esto genera un costo adicional en

la energía eléctrica, en los conocimientos para integrarlo, administrarlo y mantenerlo.

Hay que tener en cuenta que los dispositivos hardware tienen un ciclo de vida y cada vez es más reducido, debido a que en la actualidad los servicios de red van innovando con mucha rapidez.

Debido a la gran demanda en el uso de la banda ancha en las redes, varias empresas se encuentran constantemente adaptando sus infraestructuras de red a nuevas tecnologías que les brinden adaptabilidad, escalabilidad, rendimiento, reducción de complejidad y sobre todo reducción de costos.

Por las razones mencionadas anteriormente, también se aborda el concepto de virtualización de funciones de red (*Network Function Virtualization*, NFV), ya que esta ofrece una arquitectura de red mucho más flexible, escalable y con estándares abiertos, lo cual permite evitar la dependencia de fabricantes.

2.1.2. Virtualización de Funciones de Red (NFV)

La virtualización de funciones de red fue diseñada con el objetivo de disgregar las funciones de red de dispositivos hardware dedicado, permitiendo que los servicios de red (enrutadores, conmutadores, cortafuegos, balanceadores de carga, sistemas de detección de intrusiones, etc.) sean trasladados a máquinas virtuales, consolidando múltiples funciones en un único servidor físico[8].

Por consiguiente, los administradores de red no tendrán la necesidad de comprar dispositivos de *hardware dedicados* para diseñar una infraestructura de servicios, sino que emplearán funciones de administración y orquestación por medio de un software que gestione los dispositivos virtuales que se ejecutan en una red, brindando mayor flexibilidad en lo referente a la fiabilidad y escalabilidad de recursos.

Por ejemplo, en el caso de que se requiera implementar una nueva función en la red, el proveedor de servicio activa una máquina virtual para realizar dicha función, teniendo opción a desactivarla cuando ya no sea necesaria, evitando que haya un aprovisionamiento excesivo en los centros de datos; lo cual, disminuye los gastos energéticos y operativos. En la siguiente figura(2.1) se muestra un enfoque general de la virtualización de funciones de red:

Estandarización de la virtualización de funciones de red

La ETSI (European Telecommunications Standards Institute), es una organización de estandarización independiente, sin fines de lucro, pertene-

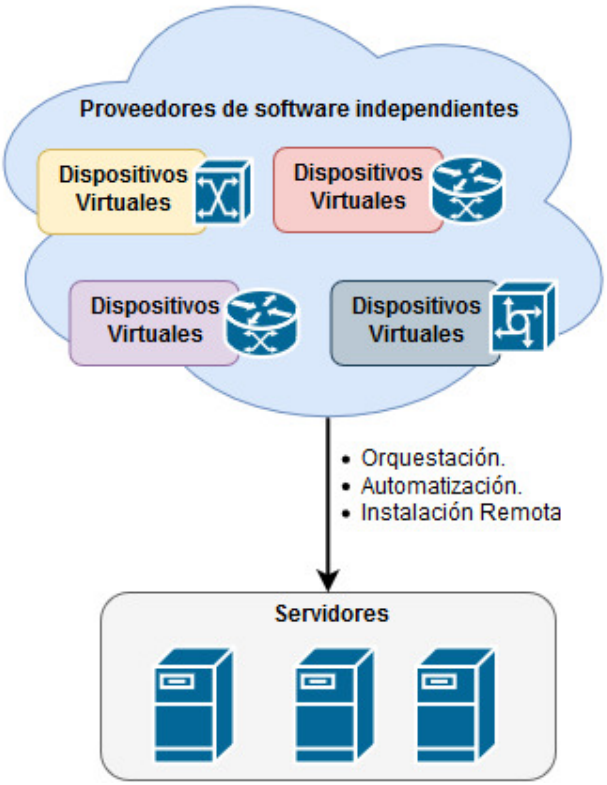

Figura 2.1: Enfoque general de NFV

ciente a la industria de las telecomunicaciones de Europa y con proyección mundial.[9]

Debido a que en la Industria de Telecomunicaciones hay múltiples proveedores de servicios de red, la ETSI crea un grupo llamado ISG (Grupo de especificación de la industria) [10], con el propósito de crear estándares que satisfagan las necesidades de la industria, especificando la arquitectura que debe tener una plataforma NFV, dejando públicos los resultados de las experiencias y pruebas de implementación para los demás grupos de estandarización.

Los estándares especificados por la NFV ISG, nos proporcionan un modelo a seguir para diseñar una arquitectura NFV, definiendo las diferentes funcionalidades y especificando cuáles son los componentes básicos para conseguir una solución interoperable, pluripartidista y sobre todo multiplataforma NFV.

De forma resumida, podríamos dividir el marco en tres bloques principales, los cuales se muestran en la figura(2.2) y se detallan a continuación:

- **Infraestructura NFV (NFVI)**: establece la base general de la arquitectura, contiene el hardware físico para alojar las máquinas virtuales y el software que hace posible la virtualización; permitiendo interoperabilidad entre los componentes virtualizaciones en funciones de red.
- **Funciones de red virtualizadas (VNF)**: Este utiliza las maquinas virtuales que ofrece el bloque NFVI, para construir sobre ellas las funciones virtualizadas de red, agregando el software que se requiere.
- **Administración y Orquestación NFV (NFV MANO)**: Este bloque interactúa con NFVI y con los VNF, aquí se delega toda la gestión de los recursos de la capa de infraestructura; es el encargado de orquestar y administrar el ciclo de vida de los recursos físicos y/o de software específicos de virtualización.

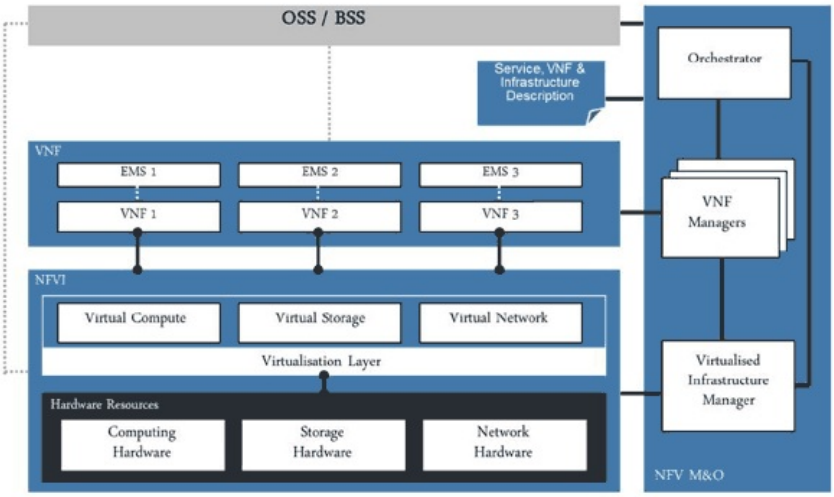

Figura 2.2: Marco de la arquitectura NFV según ETSI.
[11]

2.1.3. NFV Casos de Uso

Según el Grupo de Especificación de la Industria, el marco de NFV es fundamental para la implantación de tecnologías de telecomunicaciones tales como 5G.[12]

Las virtualizaciones de funciones de red se pueden implementar en varios casos de uso, basándose en los requerimientos y necesidades de los usuarios; permitiendo tener el control al procesar los paquetes de datos dentro de una red de comunicación.

En las áreas en las que actualmente resalta la aplicación de NFV es en la virtualización del núcleo de redes móviles e IMS, virtualización estaciones de bases móviles, virtualización de redes de acceso fijo, virtualización de CDNs, etc., permitiendo optimizar las redes, hay que tener en cuenta que la NVF está muy ligado a la computación en la nube, debido a que la evolución de servicios y comunicaciones, se está encaminando cada vez estas soluciones, de tal manera que se va adaptando las soluciones NFV a ciertos servicios que ofrece la nube.

Las aplicaciones que actualmente se consideran como casos de estudios e implementaciones futuras son las siguientes:

- Implementación de elemento de red tales como, conmutadores, enrutadores, dispositivos NAT, dispositivos proxy, servicios de gateway, tunelling (VPN, SSL), etc.
- Implementación de servicios en la nube, tales como IaaS, SaaS y PaaS.
- Implementación de elementos de red para hogares.

2.1.4. Beneficios de NFV

La virtualización se va venido dando durante años, por tal motivo las operadoras de redes buscan adaptarse a los cambios tecnológicos, debido a que las aplicaciones están requiriendo redes con un gran ancho de banda, que brinden flexibilidad y sobre todo velocidad. La NFV, a través de un software y con ayuda de la programabilidad que este brinda, permite un mayor control de las redes y sus funciones.

A continuación, se mencionan varios beneficios que proporciona utilizar NFV, dentro de las redes de comunicaciones.

- **Reducción de Costos**: Es gracias a la centralización de diversos servicios de red virtualidades dentro un conjunto de servidores, así también, al no usar equipos hardware de propietarios, esto aumenta el ciclo de vida de los equipos, se reduce el consumo energético, el gasto de cursos para capacitar a los usuarios para que puedan administrar y darles mantenimientos.
- **Eficiencia operacional**: Permite mayor rapidez en la implementación de nuevos servicios de red, aumentando la escalabilidad, disponibilidad de ejecutar varias aplicaciones en un único equipo de red obteniendo un ecosistema más dinámico.
- **Innovación**: Al no usar equipos hardware propietarios, las operadoras conjunto a empresas desarrolladoras de software, tienen la apertura de desplegar aplicaciones dentro de un entorno con plataformas NFV.

Capítulo 3

Estado del Arte

En este capítulo se analizan algunos de los proyectos relacionados con la gestión y orquestación de NFVs.

En la actualidad hay varios proveedores de la tecnología MANO que trabajan constantemente en sus plataformas con el objetivo de optimizar su gestión y obtener todos los beneficios que brinda NFV. Debido a que esta área innova y evoluciona constantemente, es recomendable analizar cuál es la mejor opción para implementar una tecnología que permita desplegar NFVs, verificando si el proveedor está continuamente mejorando, actualizando y brindando soporte a sus plataformas; de tal manera que se evite adquirir una tecnología MANO que quede obsoleta en poco tiempo.

3.1. Administración y orquestación de virtualización de funciones de red (NFV MANO)

NFV MANO comprende: la administración y orquestación (es decir, secuenciar los servicios y proveer la lógica adicional para acoplarlos) de todas las funciones de red en un centro de datos. Esto incluye los recursos informáticos, de redes, de almacenamiento y las máquinas virtuales. Así, se facilita un diseño e implementación flexible, así como la gestión de los servicios NFV, debido a que están desacoplados de los dispositivos físicos dedicados al estar en máquinas virtuales.[13]

Este marco de trabajo está constituido por tres bloques funcionales, encargados de implementar y conectar funciones y servicios cuando sean necesarios en la red:[14]

- **NFV Orquestador (NFVO)**: Es el encargado de incorporar los nuevos servicios de red (NS) y VNF en un entorno virtual. Gestiona y

autoriza las solicitudes de recursos de la infraestructura de red, las cuales son: el ciclo de vida de NS, la administración de los recursos de un NS y la gestión de las políticas para cada instancia de NS.

- **VNF Manager (VNFM)**: Es el encargado de gestionar el ciclo de vida de las VNF (creación de las instancias, las actualizaciones, las consultas, el escalado y la finalización); además, gestiona la configuración e informes de eventos entre NFVI y EMS (Element Managers System).
- **Gestor de Infraestructura Virtualizada (VIM)**: Controla y administra la infraestructura NFV, compuesta por: recursos de cómputo, de almacenamiento y de red.
- **Repositorios de Datos**: Son bases de datos en las que se alojan diferentes tipos de información sobre NFV Mano, las cuales son:
  - El catálogo de servicio de red, está conformado por un conjunto de plantillas predefinidas, que establecen un modelo para la creación, despliegues y funciones de servicios de red.
  - El catálogo VNF, está conformado por un conjunto de plantillas que definen el despliegue y las características funcionales disponibles de VNFs.
  - El repositorio de recursos NFVI, incluye información sobre los recursos disponibles o asignados de los NFVI.
  - EL repositorio de instancias NFV, contiene información acerca de todas las funciones y servicios durante su existencia,

En la siguiente figura(3.1) se puede visualizar lo que se describió anteriormente acerca de la arquitectura de NFV MANO.

NFV MANO ofrece la facilidad de desplegar funciones de redes virtuales estándares por medio de plantillas, las cuales permiten seleccionar los recursos de infraestructura NFV.

3.1.1. Tecnologías NFV MANO

A continuación mencionaremos algunas de las tecnologías NFV MANO en el mercado:

OpenMano

Este proyecto es de código abierto, liderado por la empresa de Telefónica que actualmente está bajo la estandarización MFV ISG de ETSI. Proporciona un despliegue práctico de la arquitectura de referencia de gestión y

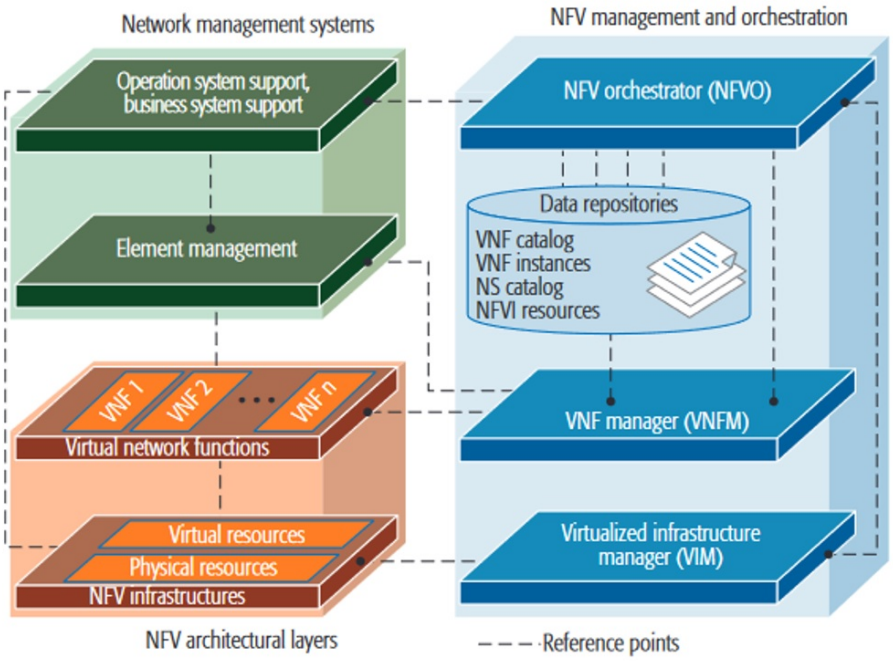

Figura 3.1: Arquitectura NFV MANO.
[14]

orquestación (NFV MANO), es un módulo innovador que permite la creación sencilla y el desarrollo de complejos escenarios de red, ha sido puesto a prueba y validado exitosamente con múltiples VNF's en un entorno de laboratorio. El objetivo de Telefónica con respecto a OpenMano, es impulsar la adopción de esta tecnología por medio del lanzamiento del código abierto, animando y consintiendo a las industrias y desarrolladores de software exploren nuevos medios de NFV.[15]

- **Arquitectura de OpenMano**

  OpenMano está diseñada a través de una arquitectura de tres componentes principales:

  - **Openvim**: Es la implementación de un gestor de infraestructura virtual (VIM-NFV), con soporte para un alto rendimiento, está conectado directamente con los nodos de computación, de almacenamiento en el NFVI y con un controlador llamado "OpenFlow", permitiéndonos crear escenarios de red complejos. Ofrece una interfaz orientada al norte llamada "openvim API", la que ofrece servicios de nube mejorados, en los que incluye la creación, eliminación y gestión de imágenes, sabores, instancias y redes.

- **Openmano**: Es la implementación de un orquestador de funciones virtuales de red (NFVO), nos permite crear escenarios de redes complejas, ofrece una interfaz norte llamada "openmano API", la misma que está basada en "REST" y por medio de la cual se ofrecen los servicios NFVs que incluyen la creación y eliminación de plantillas o instancias VNF.
- **Openmano-gui**: Es la interfaz web de usuario que interacciona con la API de Openmano, proporcionándonos información del funcionamiento de todas las redes.

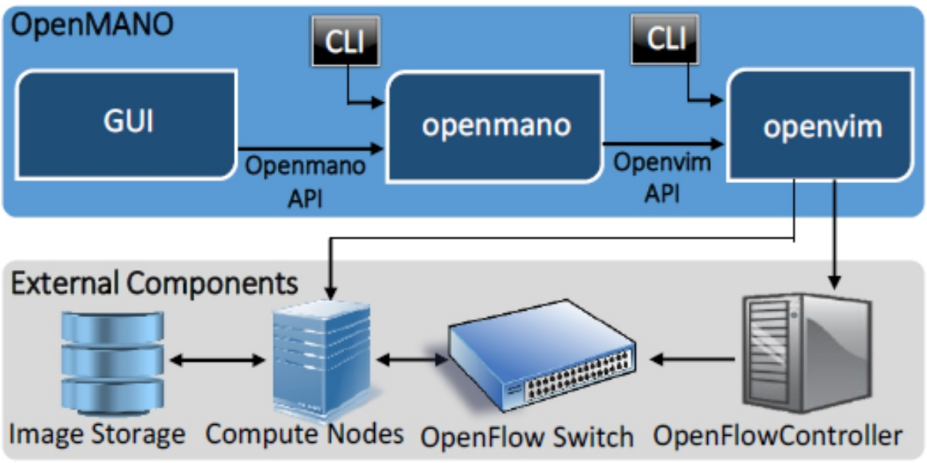

Figura 3.2: Arquitectura de Openmano de Telefónica.
[14]

Open Source Mano

El proyecto de Open Source Mano (OSM) brinda una pila de gestión y orquestación de código abierto (MANO) para producción, basado en los modelos de información ETSI-NFV y cumpliendo sus estándares de redes comerciales; lo cual permite crear un ambiente que facilita proveer NFV, ofreciendo a través de ella soluciones rápidas y rentables a los usuarios. [16]

La comunidad que desarrolla este proyecto está conformada por varios colaboradores y desarrolladores, algunos de ellos son: Intel, TELEFONICA S.A., Canonical, Whitestack, Sandvine, Ericsson LM, CableLabs, Orange, VMware Inc., Sprint Corporation, 6WIND, etc. Al hacerlo en conjunto, incrementa la innovación, eficiencia y el tiempo de la comercialización.[17]

- **Arquitectura de Open Source Mano**

Open Source Mano está diseñada a través de una arquitectura de tres componentes principales:

- **VIM**: Es la implementación de un gestor de infraestructura virtual (NFV) con soporte para un alto rendimiento, está conectado directamente con los nodos de computación y de almacenamiento en la infraestructura NFV, permite crear escenarios de red complejos; además, ofrece una interfaz web, que nos permite obtener servicios de nube mejorados, los mismos que incluyen: la creación, eliminación y gestión de imágenes, instancias y redes.
- **Sistema de Open Source Mano**: Nos permite crear escenarios de redes complejas; ofrece una interfaz norte llamada "Open-Source-Mano API", basada en REST, por la cual se ofrecen los servicios NFVs; es el encargado de comunicarse con el VIM con el fin de poder gestionar y desplegar VNFs; además, este puede comunicarse con los VNFs ya desplegados en un VIM para ejecutar configuraciones requeridas.
- **Open Source Mano UI**: Es la interfaz web de usuario que interacciona con la API de Open Source Mano, proporciona un panel de control que nos permite interactuar con las herramientas que brinda.

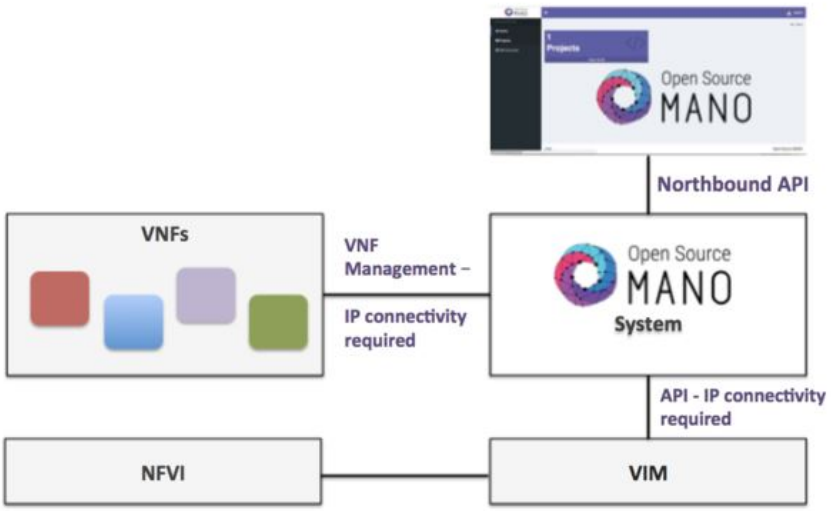

Figura 3.3: Arquitectura de Open Source Mano.
[18]

Open Baton

Es una plataforma de código abierto que está disponible bajo licencia de GitHub y Apache 2.0; proporciona una implementación completa, extensible y personalizable, basada en NFV MANO; tiene como principal objetivo, orquestar servicios de red a través de infraestructura de NFV, permitiendo administrar un ambiente con diversas funciones de redes virtuales (VNF).

Este proyecto es desarrollado por Fraunhofer FOKUS y TU Berlin. Además, cuenta con el apoyo de varios proyectores europeos con fondos públicos, como: NUBOMEDIA, Mobile Cloud Networking, y SoftFIRE. [19]

- **Arquitectura de Open Baton**

  Los componentes que forman parte de Open Baton son los siguientes:

  - **Orquestador de virtualización de funciones de red (NFVO)**: Está encargado de orquestar las virtualizaciones de las funciones de red, siguiendo el diseño e implementación que especifica ETSI MANO.
  - **Administrador de funciones de red virtuales (VNFM) genérico**: Permite gestionar las virtualizaciones de las funciones de red, indicando el ciclo de vida de una VNF.
  - **Infraestructura de virtualización de funciones de red (NFVI)**: Se encarga de permitir y conectar las funciones de red virtuales heterogéneas a la tecnología en la nube.
  - **Servicios de soportes de Operaciones (SSO)**: Se encarga de proporcionar el servicio de auto escalado y gestión de las fallas en función de la información que proviene del sistema de monitoreo del que dispone el componente NFVI.

3.2. Administrador de infraestructura Virtualizada (VIM)

El administrador de infraestructura virtualizada (VIM) es uno de los componentes principales del marco arquitectónico de NFV MANO, siendo este el responsable de controlar y administrar los recursos informativos de almacenamiento de la infraestructura NFV. Los VIMs pueden gestionar el hardware para un entorno multi dominio o pueden optimizarlos para un entorno NFVI específico. [21]

El VIM es el responsable de administrar la infraestructura virtualizada de una solución basada en NFV y es el encargado de realizar las siguientes funciones:

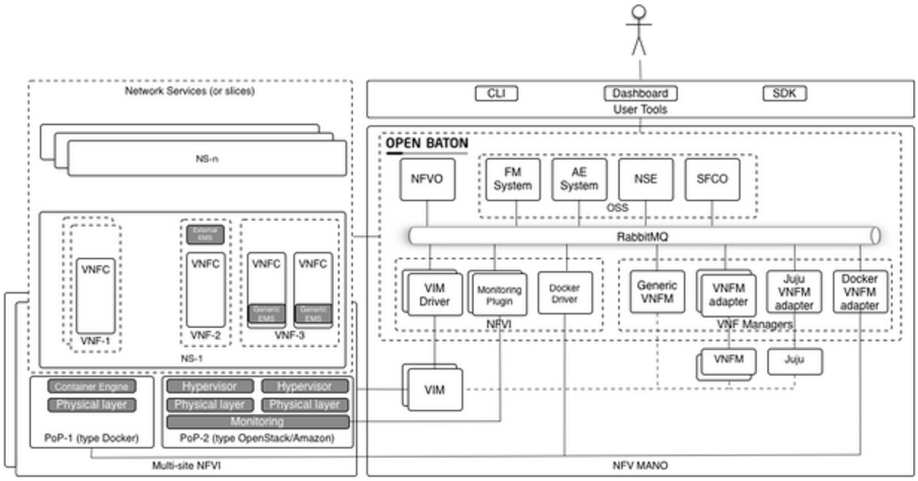

Figura 3.4: Arquitectura de Open Baton.
[20]

- Mantener un registro de las asignaciones realizadas en los recursos físicos a los recursos virtuales; por consiguiente, el VIM podrá orquestar asignaciones y realizar actualizaciones o despliegues.
- Administrar las políticas de grupos de seguridad, garantizando control en el acceso.
- Administrar un repositorio de los recursos hardware NFVI (cómputo, almacenamiento, redes) y de los recursos software (hipervisores), junto con las técnicas dirigidas a optimizar dichos recursos.
- Recopilar información de rendimientos y fallas por medio de notificaciones.
- Administrar un catálogo de imágenes de software.
- Administrar un catálogo de recursos virtualizados.

En el mercado actual hay varias tecnologías VIM, entre las más destacadas tenemos las siguientes:

- **OpenVIM**: Es un VIM ligero, una implementación de referencia de un gestor de infraestructura virtual con soporte de alto rendimiento, interactúa con los nodos de cómputo en la infraestructura NFV, con la ayuda de un controlador de flujo abierto permite crear escenarios de redes y desplegar máquinas virtuales. Ofrece una interfaz orientada al norte llamada “openvim API” donde se ofrecen servicios mejorados

en la nube, incluyendo la creación, eliminación y gestión de imágenes, sabores, instancias y redes. Esta implementación sigue las recomendaciones de ETSI NFV-PER001. [22]

- **Openstack**: Proyecto de computación en la nube que proporciona servicios de infraestructura. Es un software libre y de código abierto distribuido bajo los términos de la licencia Apache 2.0; controla grandes conjuntos de recursos informáticos, de almacenamiento y de red; es un centro de datos administrado a través de un panel o por la API de Openstack; funciona con tecnologías empresariales de código abierto, siendo ideal para infraestructuras heterogéneas. Este proyecto está gestionado por la fundación Openstack. [23]

- **VMware vCloud Director**: Permite un aprovisionamiento y consumo de la nube de forma fácil e intuitiva, incluye servicios integrados tales como protección de datos, recuperación de desastre, administración de sitios múltiples; permite a los proveedores de la nube construir plataformas ágiles, robustas y flexibles, que realmente estén preparadas para la empresa; permite también, construir soluciones de nube privada, pública o híbrida, debido a que VMware ofrece herramientas y servicios necesarios para diseñar, implementar y gestionar un entorno de nube confiable y totalmente sostenible. Sin embargo, para poder acceder a todos estos servicios hay que pagar por su uso, ya que no es un software libre. [24]

- **Amazon Web Services (AWS)**:AWS está compuesta por un conjunto de servicios que forman una plataforma de computación en la nube; brinda una amplia gama de servicios de infraestructura que son ofertados bajo demanda, el valor a pagar dependerá de los recursos solicitados. [25]

Capítulo 4

Planificación y Presupuesto

En este capítulo se abordarán los temas relacionados con la planificación y la elaboración del presupuesto asociado a la implementación y documentación del proyecto.

1. Se realizará una planificación, estimando el tiempo en el que se procederá a desarrollar el proyecto.
2. Se estudiarán los recursos necesarios, tanto materiales como humanos.
3. Se realizará una estimación de costes asociado al desarrollo de todo el proyecto, en función de los recursos utilizados y del tiempo empleado.
4. Se realizará el presupuesto.

4.1. Planificación

En este apartado se detallará de forma resumida una planificación del proceso del desarrollo del proyecto y su documentación, por ende, se definirá una serie de tiempos en los que se lo llevará a cabo. La planificación del desarrollo del proyecto se la presenta de forma gráfica con un diagrama de Gantt, este puede ser visualizado en la siguiente figura(4.1).

Revisión bibliográfica

Es el estudio de los antecedentes y el estado del arte, me permitirá adquirir conocimientos necesarios relacionados para el desarrollo el proyecto. Es necesario tener una base sólida para poder definir las características que tendrá el escenario a desarrollar, así mismo, el software que se utilizará. Se evaluarán ciertas tecnologías similares que servirían para desarrollar y luego se evaluara cuáles son las más indicadas.

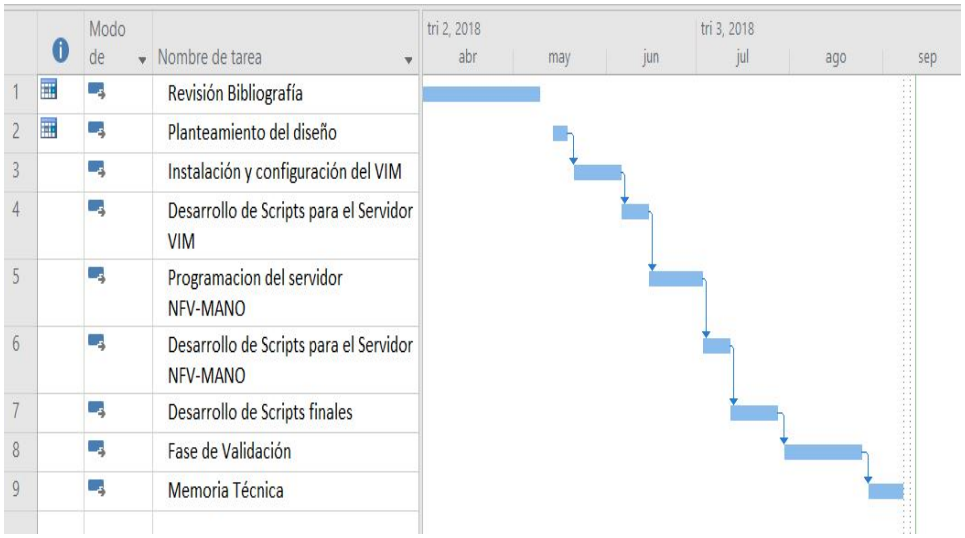

Figura 4.1: Diagrama de Gantt de la planificación temporal del proyecto.

Planteamiento del diseño

Se estudiarán diferentes alternativas para el diseño y el funcionamiento, de los escenarios tentativos. En función de las posibilidades que ofrecen los diferentes software, se elegirá un diseño u otro.

Programación del Servidor VIM

Se realizará la instalación y configuración del sistema operativo del servidor, luego la instalación y configuración de todos los módulos que usa la herramienta VIM y finalmente se verificará que todos los servicios de brinda la herramienta VIM funcionen correctamente.

Desarrollo de Scripts VIM

Se realizará los scripts necesarios para poder realizar el despliegue de las instancias virtuales necesarias para el desarrollo del proyecto.

Programación del Servidor NFV-MANO

Se realizará la instalación y configuración del sistema operativo del servidor, luego la instalación y configuración de todos los módulos que usa la herramienta NFV-MANO y finalmente se verificará que todos los servicios de brinda la herramienta NFV-MANO funcionen correctamente.

Id	Tareas	Tiempo
T1	Revisión bibliografía	70 horas
T2	Planteamiento del diseño	6 horas
T3	Programación del Servidor VIM	65 horas
T4	Desarrollo de Scripts VIM	15 horas
T5	Programación del Servidor NFV-MANO	45 horas
T6	Desarrollo de Scripts NFV-MANO	15 horas
T7	Desarrollo de Scripts finales	10 horas
T8	Fase de Validación	30 horas
T9	Memoria Técnica	140 horas
T10	Trabajo de tutorías	30 horas
	Total de horas	426 horas

Tabla 4.1: Estimación de tiempo de recursos humanos.

Desarrollo de Scripts NFV-MANO

Se realizará los scripts necesarios para poder realizar el despliegue de las NFV desde el NFV-MANO en el VIM.

Desarrollo de Scripts finales

Se realizará los scripts de monitorización que permitan realizar el despliegue automático de diversas instancias en el VIM y el despliegue de NFV desde el NFV-MANO en el VIM.

Fase de Validación

Una vez ya montado los servidores, verificando que haya comunicación entre ellos procedemos a enlazarlos para centralizar todos los despliegues desde el NFV-MANO y realizar las validaciones respectivas de su funcionamiento.

Memoria Técnica

En la memoria técnica se detallará todos los estudios realizados, los procesos de las implementaciones y configuraciones de los diferentes servicios los cuales sirvieron para el desarrollo del proyecto, incluyendo las referencias bibliográficas.

4.2. Recursos

En esta sección se detallaran que recursos intervinieron a lo largo del desarrollo del proyecto.

4.2.1. Humanos

- Jorge Navarro Ortiz y Juan José Ramos Muñoz, profesores de la Universidad de Granada en el Departamento de Teoría de la Señal, Telemática y Comunicaciones, en calidad de tutores del proyecto.
- Freddy Javier Frere Quintero, alumno del Máster Universitario de Ingeniería en Informática de la Universidad de Granada, en calidad de autor del proyecto.

4.2.2. Hardware

- Dos Ordenadores Genéricos, con procesador Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz, memoria RAM de 32 GB y disco duro de 1TB.

4.2.3. Software

- Sistema Operativo CentOs 7 se usará para el servidor VIM.
- Sistema Operativo Ubuntu 16.04.5 LTS se usará para el servidor NFV-MANO.
- OpenStack, distribución RDO versión PIKE.
- Framework Open Source Mano versión cuatro.
- Herramienta de Sublime, se lo usara para crear los scripts.
- Overleaf, herramienta de escritura de LaTeX en línea, se lo usara para redactar la documentación del TFM.
- MobaXterm, herramienta que me permite realizar conexiones remotas.

4.3. Estimación de costes

Humanos

- Horas de trabajo autónomo para un máster en ingeniería en informática se establecerá en 25 euros / hora.
- Horas de tutorías para cada profesor se establecerá en 50 euros /hora.

Hardware y Software

Recursos	Vida media	Coste unitario
Ordenadores	48 meses	900 euros
Sistema Operativo	-	0 euros
Entornos de desarrollo	-	0 euros
Software adicional	-	0 euros

Tabla 4.2: Costes de recursos de hardware y software del proyecto.

4.4. Presupuesto total

El presupuesto total donde se contemplan todos los gastos es el siguiente:

Concepto	Coste
1 desarrollador x 426 horas x 25 euros/hora	10.650 euros
2 tutores x 30 horas x 50 euros/hora	1.250 euros
Hardware 2 x ((ordenador x 850 euros x 8 meses / 48 meses))	283,33
Recursos de Software	0 euros
Total	12.183,33 euros

Tabla 4.3: Presupuesto total del proyecto.

Capítulo 5

Herramientas Usadas

En este capítulo se justifica la elección de las herramientas para el desarrollo de este proyecto y se presenta un análisis, fruto del estudio de las mismas.

Openstack y Open Source Mano (OSM) son las herramientas que hemos seleccionado debido a las características y ventajas que presentan individual y conjuntamente para la consecución de los objetivos planteados. Necesitamos la herramienta de OpenStack ya que nos permite instanciar máquinas virtuales, gestionando los recursos hardware de que se disponen. Y necesitamos Open Source Mano para que orqueste la configuración y el despliegue de los servicios de red y las sus funciones virtuales, sobre la infraestructura gestionada por OpenStack.

Podemos resaltar las siguientes ventajas que comparten estos dos entornos:

- Son software libre, por lo que podemos tener acceso a su código fuente, realizar reformas, añadir elementos.

- Son compatibles mutuamente, por lo que pueden interconectarse para orquestar y administrar NFVs.

- Las comunidades a las que pertenecen, están actualizando constantemente sus herramientas y plataformas, innovando con las últimas tecnologías disponibles.

A continuación, se realiza un resumen de los principales conceptos y arquitecturas de estos dos entornos.

5.1. Open Source Mano

Open Source Mano (OSM) es un proyecto comunitario, de código abierto, alineado a los estándares de la ETSI NFV cumpliendo con los requisitos de las redes comerciales de NFV, este ofrece una pila de administración y orquestación (MANO). OSM ha lanzado varias versiones de pilas, en las cuales ha ido añadiendo mejoras, en la actualidad la última publicada es la versión cuatro.

Es una de las alternativas más importante del mercado para la administración y orquestación de servicios en Infraestructuras NFV heterogéneas.

OSM, brinda las siguientes ventajas dentro del entorno de orquestación y administración:

- **Modularidad**: Debido a que está dividido por capas, permite que los agentes APIs y plug-ins se enlacen entre las distintas capas.
- **Abstracción**: Debido a que se basa en los estándares ETSI-NFV, ofrece una notable diferenciación entre los niveles de la arquitectura.
- **Simplicidad**: Debido que constantemente se encuentra evolucionando, OSM procura proveer una plataforma de fácil administración.
- **Estratificación**: Dispone de una clara delimitación entre las distintas capas y módulos que conforman OSM.

5.1.1. Arquitectura de Open Source Mano

OSM evoluciona con cada versión de *framework* (entorno de trabajo) que se lanza, siento más abierto y de simple integración con nuevos módulos, facilitando una mayor centralización de servicios, está compuesto por: [26] [18]

- **Orquestador de recursos (RO)**: Es el encargado de administrar y coordinar las asignaciones de recursos por medio de múltiples VIM y controladoras SDN distribuidas geográficamente.
- **Network Service to VNF Communication (N2VC)**: Es el Framework responsable de la comunicación entre el SO y la capa de Configuración y Abstracción VNF (VCA).
- **Configuración y Abstracción VNF (VCA)**: Es la responsable de habilitar las configuraciones, acciones y notificaciones hacia y desde los VNF. Cuando se integran el servicio JuJu, este permite crear VNFM permitiendo administrar las interfaces de las VNF.

- **Orquestador de servicios (SO)**: Es el responsable de la orquestación de los servicios, de la gestión del ciclo de vida y la ejecución inicial, conocido como el "Maestro" es el que rige el flujo de trabajo por medio del OSM, es el responsable de soportar los conceptos de usuarios, multi-usuarios, proyectos, y controlar el acceso basados en roles. También es el responsable del servicio de red y la gestión de los descriptores de VNF, validando que estos se rijan al esquema YANG.
- **Monitoreo**: Es una herramienta que nos permite enviar actualizaciones de configuración de monitoreo a una herramienta externa, también es un conducto que dirige los eventos accionables en el orquestador de servicios, estos eventos pueden accionados ejecutando NS/VNF o por las herramientas externas del monitoreo, tiene la capacidad de correlacionar la telemetría relacionada con las VM y los VNF con los servicios de red pertinentes.
- **Modelos de información y API Northbound (IM-NBI)**: Proporciona un panel para controlar el sistema OSM, facilita la interoperabilidad con las aplicaciones ya existentes y de clientes nuevos.
- **Interfaz de Usuario**: Este nos brinda un sistema de visualización, permitiéndonos interactuar con las herramientas de VNF Package Generator y VNF/NS Catalog Composer.
  - **VNF Package Generator**: Nos permite crear paquetes correctamente estructurados para luego ser incorporados en el OSM.
  - **VNF/NS Catalog Composer**: Nos permite modelar y crear descriptores que representan con precisión la esencia de la entidad para las implementaciones.

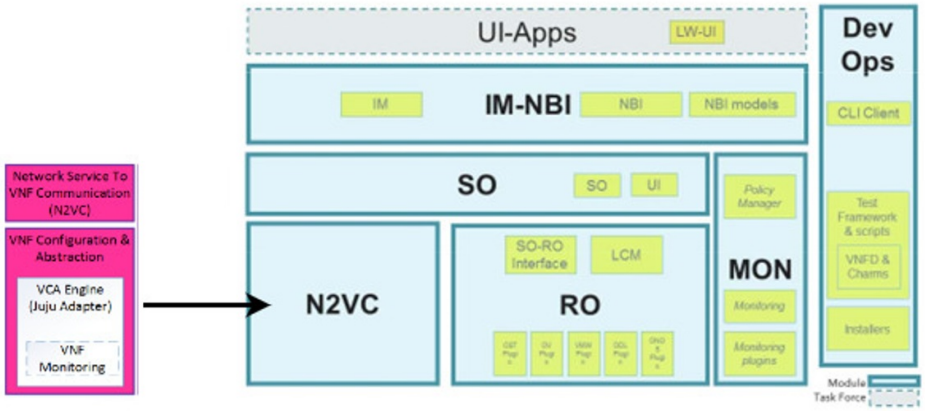

Figura 5.1: Diagrama de la arquitectura lógica de Open Source Mano.

5.2. Openstack

OpenStack es una plataforma de Computación en la Nube, de software libre y código abierto. Está compuesta por varios servicios relacionados entre sí que forman la infraestructura, puede ser instalado de forma separada o conjunta, según la distribución que se elija, ofrece una Interfaz de Programación de Aplicaciones (API) la cual nos facilita su integración con la plataforma y nos permite gestionar y controlar los recursos de computación, almacenamiento y red que forman parte de la misma a través de una interfaz web, con el fin de proveer una solución flexible para la nubes publicas y nubes privadas.

5.2.1. Arquitectura de Openstack

Procederemos a describir brevemente los servicios que conforman la arquitectura lógica de OpenStack en la siguiente Tabla(5.1):

Tabla 5.1: Servicios que conforman OpenStack.

Servicio	Proyecto	Descripción
Identidad	Keystone	Proporciona los servicios de: autentificación, permitiendo confirmar la identidad de un usuario; y, autorización, para determinar el derecho de accesos que tiene el usuario en proyectos de OpenStack.
Interfaz de Gestión Web	Horizon	Proporciona una interfaz de usuario basada en web, permitiendo que los administradores y clientes puedan interactuar con los servicios de Openstack.
Imagen	Glance	Proporciona servicios de almacenamiento y gestionamiento, tales como: descubrir, registrar y recuperar imágenes de los discos de las máquinas virtuales.
Computación	Nova	Proporciona el servicio de gestiones relacionadas con las máquinas virtuales, que incluye en su creación, la selección del nodo de computación y su eliminación.

Continúa en la siguiente página...

Servicio	Proyecto	Descripción
Red	Neutron	Proporciona conectividad de red como servicio a otros proyectos de OpenStack, como por ejemplo, el servicio de nova, que utiliza la API de Neutron para solicitar la conexión de las máquinas virtuales a un segmento de red específico. Permite a los usuarios crear redes virtuales privadas, redes, subredes, enrutadores, cortafuegos y balanceadores de carga. Tiene una arquitectura modular que le permite integrarse con múltiples tecnologías de proveedores de red externos.
Almacenamiento en bloque	Cinder	Proporciona y administra volúmenes de almacenamiento en las máquinas virtuales en ejecución. Brinda un interfaz de programación de aplicación (API) de autoservicio, la cual le permite al usuario solicitar y consumir dichos recursos de almacenamiento en bloques, los mismos que se adaptan a las máquinas virtuales.
Almacenamiento de Objetos	Swift	Proporciona un servicio de almacenamiento redundante y escalable. Consiste en escribir objetos y ficheros en múltiples dispositivos(discos duros), garantizando que los datos estén replicados a lo largo del conjunto de servidores.
Base de Datos	Trove	Proporciona bases de datos escalables y fiables de aprovisionamiento. Las bases de datos pueden ser relacionales y no relacionales
Telemetría	Ceilometer	Proporciona la facilidad de recopilar, normalizar y transformar de manera eficiente los datos generados por los diferentes servicios de OpenStack. Los datos que recoge están destinados a ser utilizados para crear diferentes vistas y ayudar a resolver varios casos de uso de telemetría, como por ejemplo: facturación, escalabilidad y reportes estadísticos.

Continúa en la siguiente página...

Servicio	Proyecto	Descripción
Orquestación	Heat	Permite modelar, configurar y automatizar los recursos de infraestructura de una aplicación en la nube basada en plantillas, estas son en forma de archivos de texto. Heat brinda una API ReSt nativa de Openstack y una API de consulta compatible, que es CloudFormation.

En la siguiente figura (5.2) se podrá ver como se relacionan los diferentes servicios que conforman Openstack.

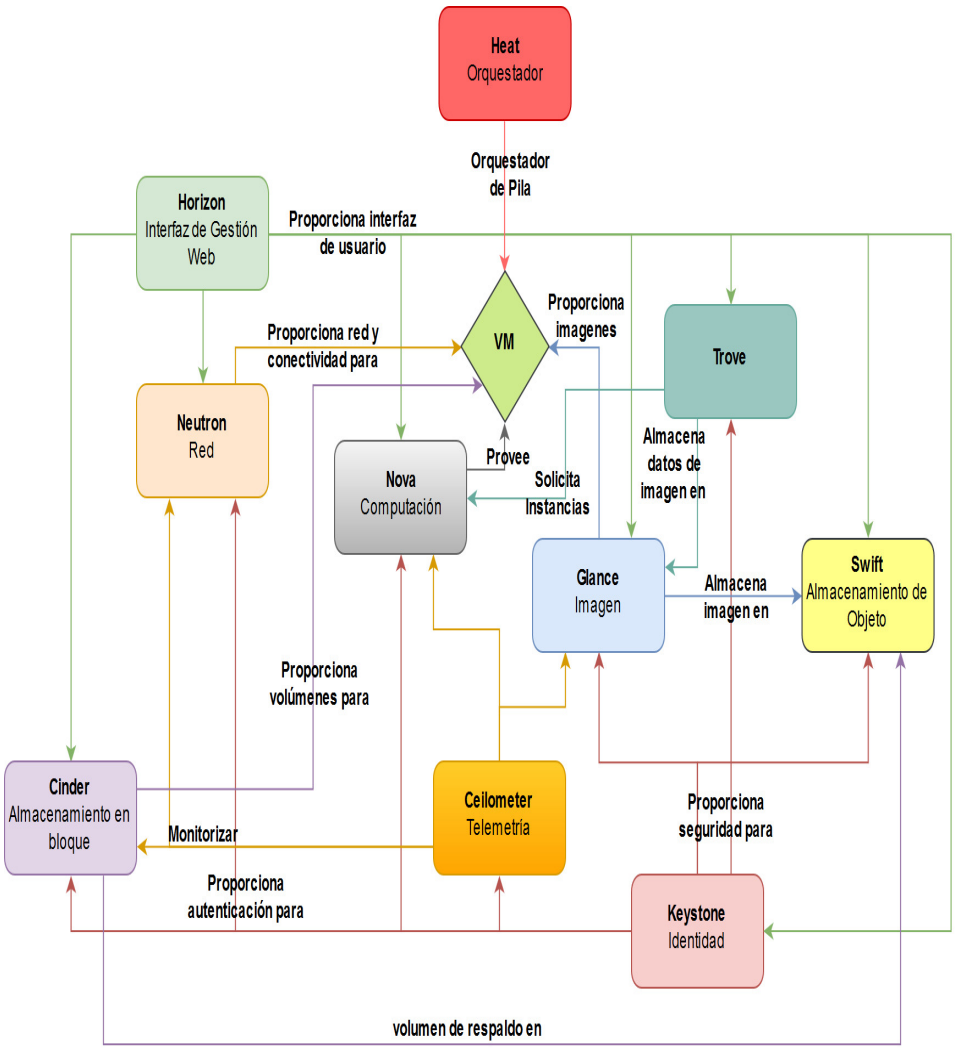

Figura 5.2: Diagrama de la arquitectura lógica de Openstack.

Los diferentes servicios que conforman Openstack, utilizan el servicio de autenticación llamado keystone. Además, se puede destacar que el servicio de nova es quien se encarga de provisionar las maquinas virtuales, el servicio de glance proporciona la imagen del sistema operativo con las diferentes aplicaciones que se ejecutaran en la instancia, el servicio de neutron provee la conectividad de las diferentes instancias a un segmento de red que el usuario desee, el servicio de cinder proporciona un volumen para el almacenamiento de los datos, el servicio de swift se encarga de almacenar en forma de objetos los volúmenes de almacenamiento en bloques, el servicio de ceilometer nos ayuda midiendo y contabilizando el uso de los recursos en la nube.

Los servicios interactúan entre si a través de API's y mantienen su información en una base de datos compartida.

Los usuarios de Openstack pueden acceder a cada uno de los servicios por medio de una interfaz de gestión web, que brinda el servicio de horizon, pero también pueden acceder a cada uno de los servicios a través de sus API's, ya sea por algún complemento del navegador o por línea de comandos.

Internamente los servicios de OpenStack están conformados por varios procesos y como mínimo cada servicio tiene un proceso API, que escucha las solicitudes API, realiza un procesamiento inicial y luego las pasa al resto de los procesos internos del servicio.

OpenStack se muestra como una solución poco compleja de implementar, escalable y muchas funcionalidades. En la actualidad, es empleada por numerosos proveedores de servicios de nube tanto públicas, híbridas y privadas, con una tasa de aceptación que ha ido de manera ascendente estos últimos años.

5.3. Descripción detallada de cada uno de los módulos que conforman Openstack

5.3.1. Servicio de identidad

Servicio de Identidad o también llamado keystone, es el encargado de proveer autentificación y autorización a los usuarios y servicios que componen OpenStack.

- **Autentificación**, es el acto de confirmar la identidad de un usuario especifico.
- **Autorización**, es la función de determinar los derechos de acceso para ese usuario especifico.

Todos los usuarios y servicio necesitan el servicio de identidad para poder

autentificarse y poder crear las solicitudes de los usuarios. Tales como crear usuarios, roles, proyectos y dominios.

Los usuarios de OpenStack pueden autentificarse a través de la interfaz de gestión web (horizon), por línea de comando o directamente hablando con la API, una vez que el usuario es autentificado, el servicio de identidad no participa en la determinación de los derechos de acceso de ese usuario en concreto.

Derechos de acceso incluye tales cosas como si un usuario puede crear un enrutador virtual en neutron o imágenes en glance que sean de acceso público, esta autorización es manejada por un archivo llamado "policy.json" y reside en cada directorio de cada servicio instalado.

Arquitectura lógica del servicio de Identidad.

El servicio de identidad es el sistema de autentificación común en toda la nube por ende recibe más solicitudes que los otros servicios de OpenStack, keystone utiliza Apache2 como interfaz de API con el puerto 5000, es la puerta de entrada principal a la misma.

En la siguiente figura (5.3 se puede observar como es la arquitectura lógica del servicio de identidad.

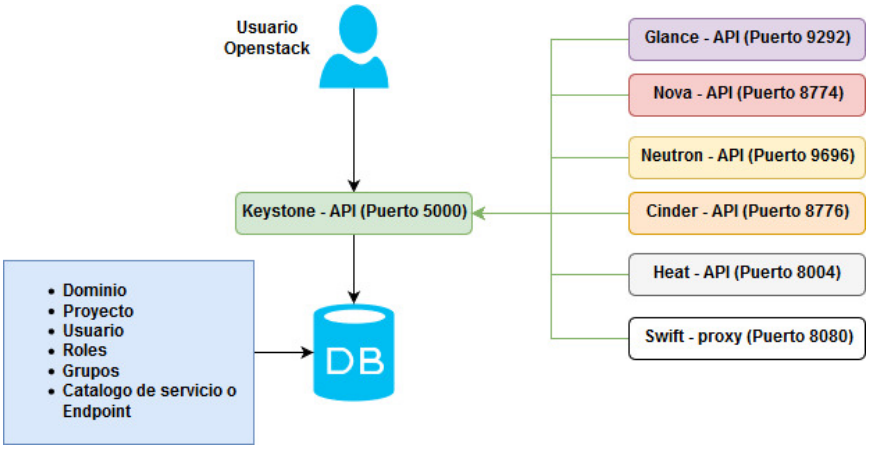

Figura 5.3: Diagrama de la arquitectura lógica de keystone.

El usuario interactúa la keystone API, la cual le permite al usuario autentificarse frente a sus proyectos, mediante él envió de un mensaje "token"de autorización la cual es validado entre los servicios de OpenStack. Esto nos sirve para corroborar si se tiene permiso para utilizar los servicios, como por ejemplo el almacenamiento o el cómputo.

El sistema de administración de base de datos relacionado al entorno de

OpenStack es generalmente "mariadb" y contiene todos los datos para todos los diversos recursos de keystone los cuales son los siguiente:

- **Detalles de Dominios**: Son contenedores lógicos de altos niveles utilizados para agrupar a los usuarios, grupos y los proyectos.
- **Detalles de Proyectos**: Representa un área donde se crean los recursos.
- **Detalles de Usuarios**: Son creados para personas individuales en una organización.
- **Detalles de Roles**: Son etiquetas que se han aplicado a un usuario o a un grupo.
- **Detalles de Grupos**: Proporciona una forma fácil de aplicar un rol existente a múltiples usuarios.
- **Catálogo de servicio o endpoint**, es un mapa almacenado en keyston que contiene una lista de servicio y enlaces finales, se lo asemeja a una guía que proporciona enlaces finales para todos los servicios instalados en el entorno de Openstack.

5.3.2. Servicio de Gestión Web

El servicio de gestión web, también conocido como horizon o dashboard, es la implementación acorde de OpenStack que proporciona una interfaz de usuario basada en web para los servicios de este, tales como nova, Swift, Keystone, Neutron, Heat, etc. Horizon se ejecuta a partir de una instalación de Apache2 utilizando la puerta de enlace del servicio web de Python y Django.

La interfaz de usuario basada en web de OpenStack permite a los usuarios acceder y administrar los diferentes servicios los cuales sirven para provisionar, automatizar los recursos basados en la nube. También nos da la facilidad de incorporar servicios adicionales que nos permiten realizar monitorizaciones, facturaciones y herramientas de servicios adiciónales.

EL servicio de gestión de web ofrece las siguientes funcionalidades:

- Como usuario administrador, te proporciona un panel de control que ofrece una vista general la infraestructura y del estado de la nube. Permitiéndote crear usuarios, proyectos, grupos, asignar usuarios a proyectos, configurar los permisos de los usuarios clientes en diferentes proyectos, etc.

- Permite a los usuarios administradores y clientes controlar los recursos que poseen, tales como el almacenamiento, recursos de red y nivel de computo.

- A los usuarios clientes le proporción un panel de control de autoservicio para provisionar sus propios recursos dentro de los parámetros establecidos por el administrador.

5.3.3. Servicio de Imagen

EL servicio de imagen o también llamado glance, permite a los usuarios registrar, almacenar y gestionar imágenes, nos ofrece las distribuciones de almacenamiento de registro de todas las imágenes en nuestro entorno. Dichas imágenes son solicitadas por el servicio de computación (nova) para poder crear instancias.

Cabe recalcar que una imagen es un fichero que contiene la imagen de un disco duro con un sistema operativo preinstalado y sirve como plantilla para generar el contenido del disco duro de una instancia. Por ende, cuando se crea una instancia, esta inicia con el sistema operativo que detecta el su disco duro que es una imagen de glance.

Las imágenes en la nube, que aparecen en glance son generalmente instantáneas de los contenidos de un disco, estas imágenes han sido configuradas previamente por una persona o un conjunto de códigos que ha pasado por el procedimiento de instalación inicial y ha instalado programas específicos y archivos de configuración para garantizar que sean compatible con la nube, glance puede almacenar estas imágenes en un almacén de datos, admite una variedad de almacenes de datos amplios e incluidos el sistema de archivo local NFS o simplemente un contenedor OpenStack en swift. Las imágenes de disco almacena todo el contenido del sistema operativo y vienen en una variedad de formato de archivos tales como:

- **RAW**: Este formato es simple y es soportado de forma nativa por los hipervisores KVM y XEN, es un formato de imagen de disco sin comprimir.

- **QCOW2 (QEMU copy-on-write)**: Este formato de archivos para archivos de imágenes de disco utilizado por “QEMU” que es un monitor de maquinas virtuales alojadas. Lleva el nombre por su estrategia de optimización de almacenamiento en el disco que retrasa la asignación de almacenamiento hasta que realmente se necesita, las imágenes que usan este formato tienen un tamaño pequeño y permiten crear imágenes instantáneas.

- **VMDK (Disco de máquina virtual)**: Inicialmente fue diseñado para utilizarlo con el hipervisor de VMware, pero ahora es un formato mucho mas abierto y compatible con otros hipervisores como por ejemplo VirtualBox.
- **VHD y VHDX (Disco duro virtual)**: Se usa con productos relacionados con Microsoft como por ejemplos Hyper-v y Azure.
- **VDI (Imagen de disco virtual)**: Este tipo de formato es muy usado por VirtualBox
- **ISO**: Es uno de los formatos mas comunes, es un archivo de disco óptico y todos los contenidos de los datos están escritos en este fichero incluido el sistema de archivos y el sistema operativo.

Arquitectura lógica del servicio de Imagen

EL servicio de imagen está compuesto por dos demonios principales, el glance–API que utiliza el puerto 9292 y el glance–registro.

En la siguiente figura (5.4 se puede observar como es la arquitectura lógica del servicio de imagen.

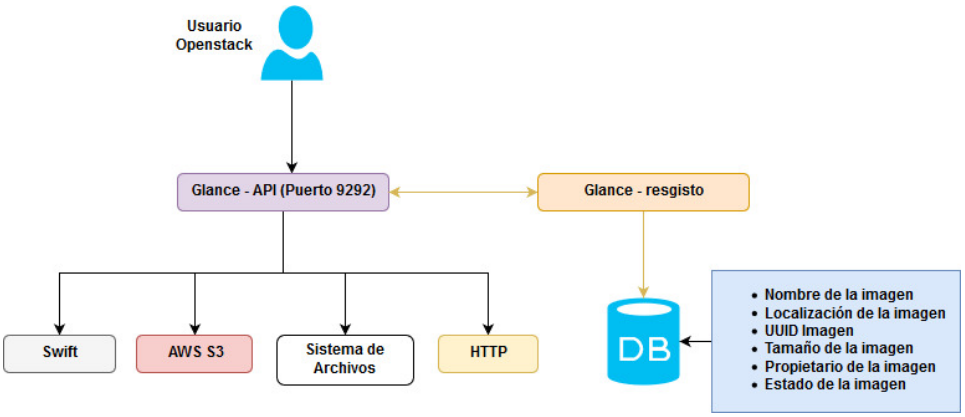

Figura 5.4: Diagrama de la arquitectura lógica de glance.

El usuario interactúa con la glance-API para almacenar y recuperar las imágenes de disco, en cambio el glance–registro es el responsable de almacenar los metadatos asociados a esa imagen en la base de datos relacional, por ejemplo: el nombre, la ubicación, el propietario y el tamaño de la imagen, así como el formato del disco.

Glance puede almacenar imagen en una variedad de respaldos de almacenamiento de datos incluyendo swift, Amazon s3, sistemas locales de archivos

en el que reside el propio demonio de glance-API o incluso en un servidor web de acceso público.

5.3.4. Servicio de Computación

El servicio de computo también conocido como nova, es el núcleo de la construcción en la nube en OpenStack, ha sido diseñado para crear, gestionar, eliminar y automatizar un conjunto de recursos de computo para trabajar con una variedad de tecnologías existentes, este es el componente más importante del IaaS (infraestructura como servicio).

Nova soporta diferentes hipervisores tales como: KVM/QEMU, Xen, Xen-server, Hyper-V, VMware ESXi, también es compatible con la capacidad de aprovechar las tecnologías de contenedores como Docker y LXC.

Arquitectura lógica del servicio de Computo

En la siguiente figura (5.5 se puede observar como es la arquitectura lógica del servicio de computo.

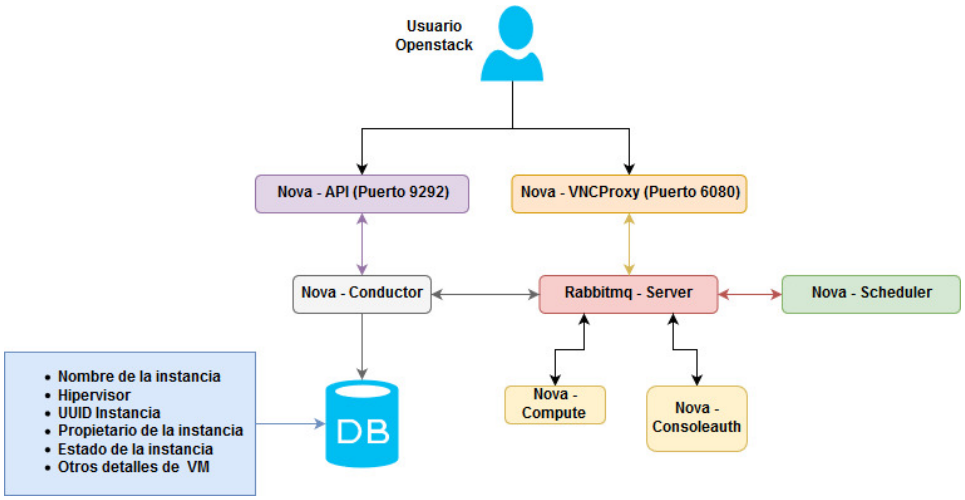

Figura 5.5: Diagrama de la arquitectura lógica de nova.

Nova esta compuesta por seis demonios principales los cuales son los siguientes:

- **Nova-API**: Este corre en el puerto 8774, es la entrada principal y cualquier servicio que desee interactuar con Nova debe interactuar con nova – API para poder crear, listar, borrar y gestionar cualquiera de las instancias que tengamos en nuestro entorno.

- **Nova–scheduler**: Permite evaluar y filtrar todos los hosts de cómputos disponibles en el centro de dato para determinar el mejor nodo de computo para poder arrancar una instancia, el comportamiento de este se puede personalizar a gusto en función de las características específicas, como por ejemplo la arquitectura de la CPU de los servidores o la ubicación especifica de tu centro de datos.

- **Nova–conductor**: Es conocido como un agente de bolsa (bróker) de la base de datos, este se conecta directamente a la base de datos relacional del entrono OpenStack. Principalmente debido a un tema de seguridad y escalabilidad ya que el nodo de computo hipervisor es el componente menos confiable de un entorno virtualizado "multi - tenencia" por lo que toda la comunicación de la base de datos debe de pasar por nova - conductor.

- **Nova-novncproxy**: Proporciona acceso a la consola de las instancias a través de un cliente VNC o navegador web.

- **Nova-consoleauth**: Recibe solicitudes de nova – novncproxy para utilizar el token de un usuario y mapear el host privado y el peurto del servidor VNC de una instancia.

- **Nova-compute**: Gestiona las máquinas virtuales en los hipervisores.

Para poder iniciar una instancia con nova el administrador debe proporcionar una imagen de glance deseado y un "sabor" (es decir, un perfil previamente aprovisionado de procesador, memoria y disco local deseado para la máquina virtual) de nova.

Arquitectura lógica del orquestador del servicio de Computo.

En la siguiente figura (5.6 se puede observar como es la arquitectura lógica del orquestador del servicio de computo.

Nova-Api nos permite la comunicación de los hipervisores con nuestro entorno, las colas de mensajes (rabbit queues) son las responsables de la comunicación entre los diferentes componentes ya sean estos nodos de computo, de red, o del volumen a la hora de provisionar los recursos. Esto quiere decir que cuando un usuario administrador inicia una instancia nova aprovecha los recursos de procesador, memoria y disco disponible en los nodos de computo.

Cabe recalcar que nova no es un hipervisor, sino el gestor de recursos de hipervisores de los cuales ya los hemos mencionado en el apartado de arriba.

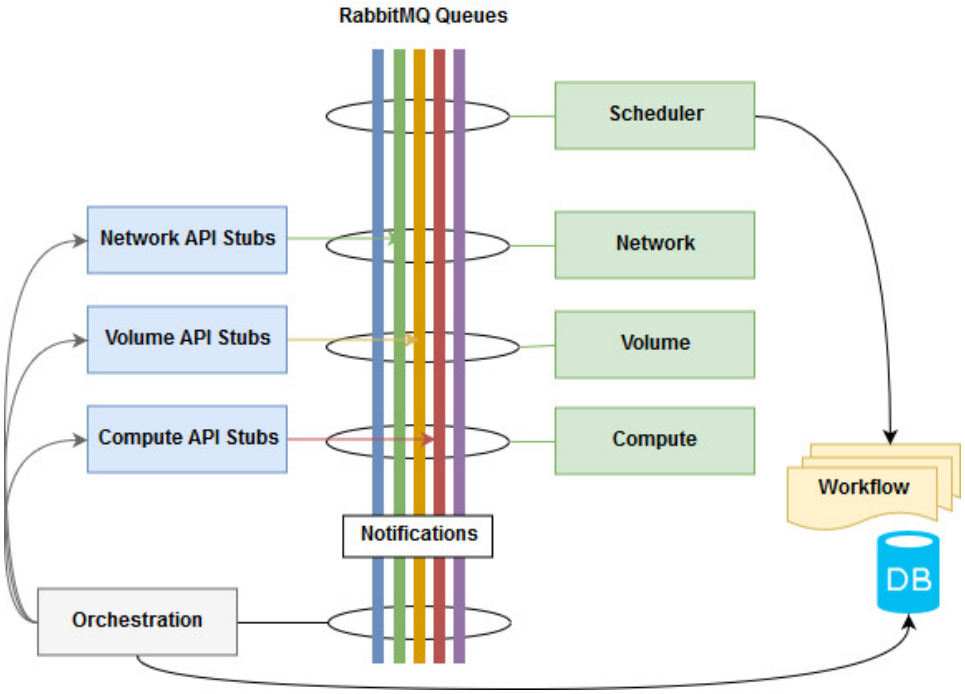

Figura 5.6: Diagrama de la arquitectura lógica de nova.

Tipos de Hipervisores

Nova es compatible con una gama de hipervisores, a continuación se detallará alguno de ellos:

- **QEMU**: Es un hipervisor de código abierto que proporciona una emulación completa del sistema, puede emular uno o varios procesadores sin asistencia del CPU el cual tiene una tendencia al ser un poco lento.

- **XEN**: Es un hipervisor de código abierto e implementa una técnica llamada para-virtualizacion (PV), este no requiere de procesadores con extensiones de virtualización, pero en cambio depende de controladores que hay que instalar dentro de las máquinas virtuales o de las instancias, Xen ha sido utilizado por muchas nubes publicas como por ejemplos: “Rackspace” y “AWS”.

- **QEMU-KVM**: Utiliza QEMU para virtualizar los periféricos de sistema ”host”pero también aprovecha los procesadores con extensiones de virtualización de hardware, este es mucho más rápido debido a su capacidad de virtualizar instancias con velocidad nativas.

5.3.5. Servicio de Red

EL servicio de red también conocido como neutron, es el componente SDN "Redes Definidas por Software", es el que permite administrar las redes virtuales las subredes, las direcciones IP, los enrutadores, las reglas de cortafuego, etc.

Permite a los usuarios crear los recursos virtuales necesarias, para garantizar que sus instancias obtengas direcciones IP internas conocidas como IP fijas, sino que también nos permite asignar direcciones IP externas, conocidas como direcciones IP flotantes, estas permiten a las aplicaciones que residen en instancias de OpenStack que sean accesibles externamente.

Neutron permite a los usuarios ver sus propias redes, reglas de cortafuego, enrutadores y balanceadores de carga, a través del tablero de horizon, por la interfaz de línea de comando o por la API.

Conceptos del servicio de Red

A continuación, se procederá a definir varios conceptos del servicio de red utilizando la siguiente figura (5.7 como ejemplo.

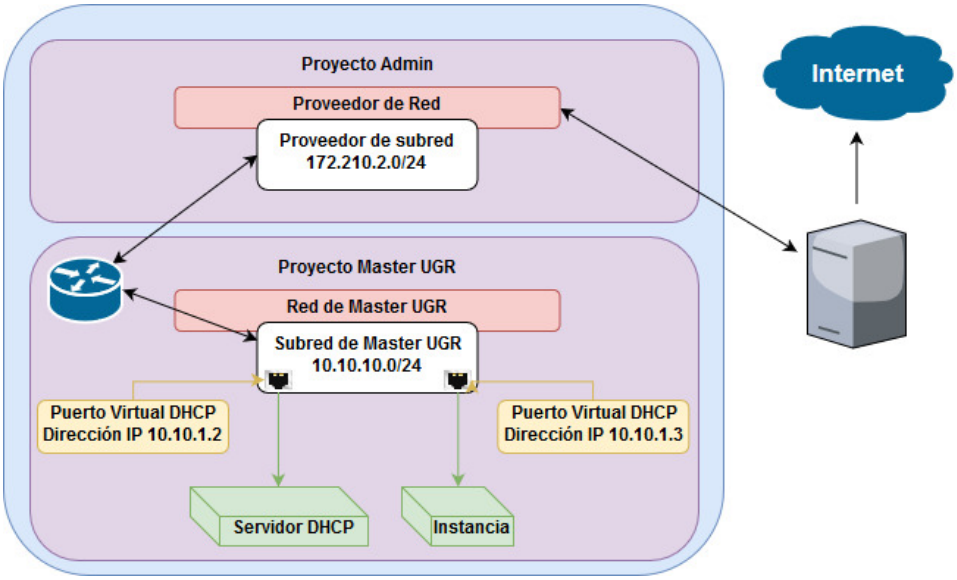

Figura 5.7: Conceptos del servicio de neutron.

- **Red**: Es una analogía de una red física de datos en OpenStack, hay dos tipos de redes en neutron:
  - **Redes privadas**: Son redes creadas por los propios usuarios que

desean iniciar instancias en sus propios dominios de redes de capa dos, estas redes siempre son propiedad de dicho proyecto al que el usuario tiene acceso durante la creación de esa Red.

- **Redes de proveedores**: Son redes creadas por el administrador de la Nube o un usuario con rol de nivel administrador, estas se utilizan para proporcionar acceso a la red de recursos fuera del entorno de OpenStack, estos pueden ser Internet o incluso un servidor de base de datos que existan en una VLAN o XLAN especifica en el algún centro de datos.

- **Subred**: Es un bloque de direcciones IPv4 o IPv6 asociado en una red especifica en OpenStack, permite la asignación de direcciones IP a instancia de máquinas virtuales u otros recursos de red. Esta debe estar asociada a una red para iniciar una instancia en ella. La subred proporciona un rango de red.

- **Puerto**: Se asemeja a una tarjeta de interfaz de red virtual, representa puntos de entrada y salida para entrada del tráfico de datos, este tiene asociado una dirección MAC y un UUID asociado. Un puerto se crea automáticamente cuando se inicia una instancia en una red o se los puede reservar si el usuario prefiere una dirección especifica.

- **Grupos de seguridad**: Controla el tráfico de entrada y salida de un puerto, Por defecto OpenStack permite el trafico de salida en todas las instancias sin embargo en trafico de entrada si está limitado excepto para aquellas instancias que estén en el mismo grupo de seguridad.

- **Enrutador**: Son dispositivos de red generados por el demonio de neutron L3 - Agent, este permite conectar el trafico de diferentes redes.

Como podemos observar en la figura (5.7 tenemos una red privada que es la red de “Master UGR” y se ha creado dentro de esa red una subred “10.10.10.0/24” a la cual se le ha conectado un servidor DHCP y una instancia, un enrutador que nos permite conectarnos con diferentes redes privadas o a una red de proveedor, este tráfico se denomina de “Este” a “Oeste” y nos permite acceder a redes externas como puede ser tráfico Internet y este tipo de tráfico se denomina “Norte” a “Sur”.

Arquitectura lógica del servicio de Red

El servicio de red está compuesto por cinco demonios principales como podemos observar en el siguiente figura (5.8.

- **Neutron-server**: Corre por el puerto 9696, es la API y la puerta de enlace principal a neutron, cualquier servicio o usuario debe interac-

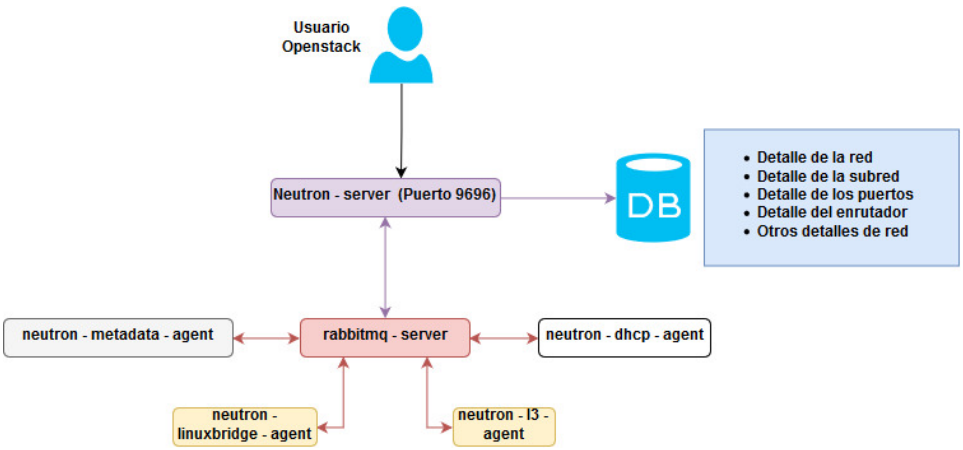

Figura 5.8: Diagrama de la arquitectura lógica de neutron.

tuar con él, para crear, enumerar, eliminar administrar redes y otro recurso de red.

- **Neutron-DHCP-agent**: Proporciona los servicios de DHCP que es la asignación dinámica de una dirección IP a cada dispositivo de red para que se puedan comunicar entre sí, realmente se usa un programa llamado "dnsmasq", es un software de servidor de DHCP ligero que está corriendo dentro de un espacio de nombre de red (namespace), son una característica del Kernel de Linux, divide el uso de red permitiendo la segregación de los recursos de red.
- **Neutron-L3-agent**: Es el encargado de crear los enrutadores, dentro de Neutron son "namespace" con tablas de enrutamiento únicas y reglas de IP tables.
- **Neutron-metadata–agent**: Proporciona servicios de metadatos a todas las instancias, es la dirección IP no enrutable 192.254.168.254 a la que un usuario puede conectarse para obtener información especifica con respecto a dicha instancia.
- **Neutron-linuxbridge-agent**: Se ejecuta en cada nodo de cálculo, es el responsable de la creación y gestión de todas las funciones relacionadas en la red en el nodo de computo, incluida en las fases de las redes virtuales para las instancias recién creadas, así como la creación y conexión de los puertos virtuales y las reglas de enrutamiento.

5.3.6. Servicio de Almacenamiento en bloque

El servicio de almacenamiento en bloque también conocido como cinder, es el cual le permite a los usuarios gestionar la creación y eliminación los

volúmenes persistentes para que sean montados en las instancias creadas por el servicio de nova.

Conceptos del servicio de almacenamiento en bloque

Hay que tener claro los tres conceptos en el servicio de almacenamiento en bloque más importantes:

- **Volumen**: Es un dispositivo de bloque sin formato que se puede conectar a una instancia de maquina virtual en nova, luego este volumen puede ser utilizado cono un disco duro tradicional como el sistema operativo de la instancia.
- **Snapshot (instantánea)**: Es una copia en un punto en el tiempo en concreto del contenido de un volumen de solo lectura, se puede crear a partir de un volumen que este en uso dentro de una instancia.
- **Backup**: Es un archivo comprimido de los contenidos de dicho volumen, esta almacenado en un contenedor de almacenamiento de objetos como por ejemplo Swift u otro proveedor externo como por ejemplo s3 de AWS.

Arquitectura lógica del servicio de almacenamiento en bloque

El servicio de almacenamiento en bloque está compuesto por cuatro demonios principales los cuales los vamos a describir en la siguiente figura (5.9.

- **Cinder–API**: Es el API y la puerta de enlace principal al servicio de volúmenes por bloque y utiliza el puerto 8776,Swift los usuarios deben interactuar con este demonio para poder crear, enumerar, eliminar y administrar los volúmenes de bloques, instantáneas (snapshot) y copias de seguridad de estos volúmenes.
- **Cinder–scheduler**: Evalúa y filtra todos los nodos de almacenamiento disponible en el entorno de OpenStack para determinar el mejor y poder crear el volumen en bloque, el comportamiento de este demonio se puede modificar en función de características específicas como, por ejemplo, la capacidad de los discos o la latencia de los mismos.
- **Cinder–volumen**: Reside en los nodos de almacenamiento es el responsable de la creación y eliminación de los volúmenes de bróker.
- **Cinder–Backup**: Ayuda a realizar copias de seguridad en volúmenes de bloque en OpenStack swift u otros objetos de almacenamiento de terceros que podamos tener en el entorno de OpenStack.

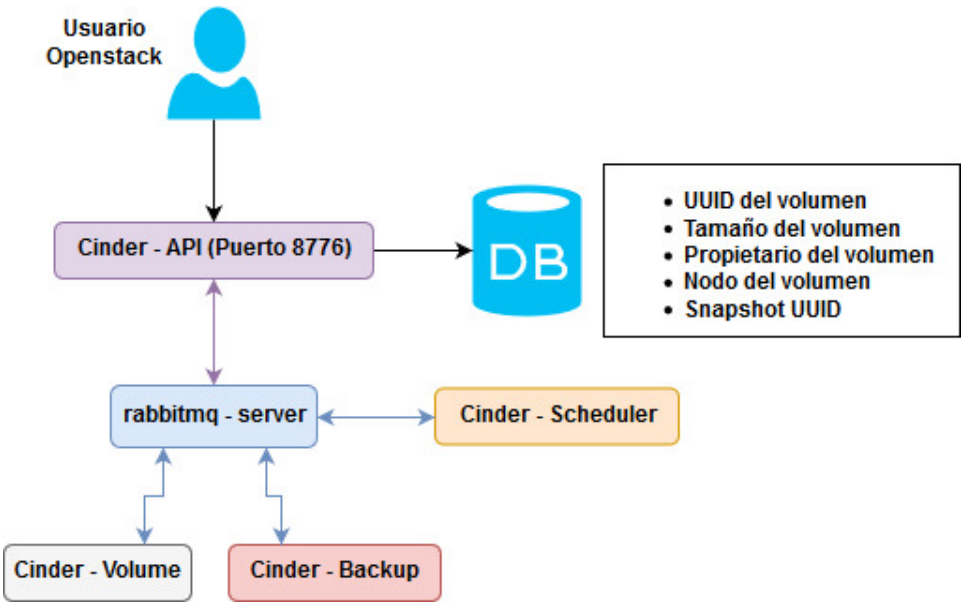

Figura 5.9: Diagrama de la arquitectura lógica de cinder.

Tecnologías del servicio de almacenamiento en bloque

Las tecnologías más importantes en las cuales el servicio de almacenamiento por bloque se basa son las siguientes:

- **Administrador de volúmenes lógicos (LVM)**: Es un conjunto de herramienta que le permite a los usuarios agregar y reemplazar discos duros sin inactividad o interrupción del servicio. Esto se hace mediante la creación de volúmenes lógicos únicos de discos duros completos lo que permite el cambio de tamaño dinámico.
- **Internet SCSI**: Es un estándar de almacenamiento basado en IP para proporcionar acceso a nivel de bloque a dispositivos de almacenamiento a través de una red IP.

En la siguiente figura (5.10) se demostrará cómo funciona el servicio de cinder cuando un usuario crea un volumen en bloque.

1. El usuario envía una solicitud de API a **cinder-API** a través del panel de horizon o CLI.
2. **Cinder-API** recibe la solicitud, la base de datos de cinder se actualiza con los detalles del volumen y lo coloca en el bus de mensajes.
3. Cinder Scheduler determina el mejor nodo para provisionar el volumen solicitado.

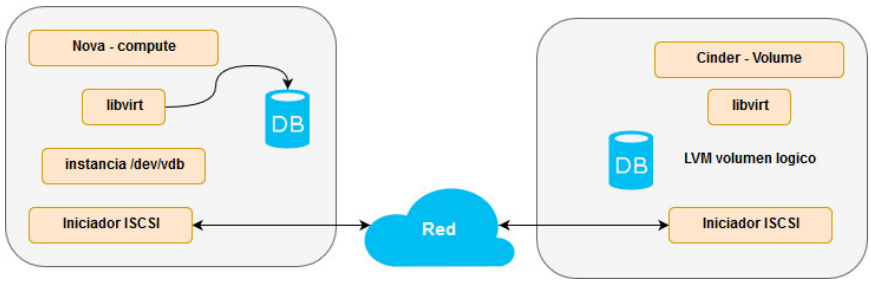

Figura 5.10: Creación de un volumen en cinder.

4. Se selecciona el volumen, **cinder–volumen** es responsable de generar los comandos LVM para crear el volumen lógico a partir de un grupo de volúmenes predefinidos.

5. El usuario envía una solicitud al **nova-API** para adjuntar el volumen de la instancia a su elección, este es conocido como el proceso de conexión iSCSI entre los dos nodos.

6. El usuario puede formatear el dispositivo de bloque y escribir datos dentro del sistema operativo de la instancia.

7. Todos los datos escritos en cinder se almacenan en el base de cinder.

El usuario puede desconectar dicho volumen de la instancia cuando el quiera y volver a conectarlo a otra instancia reestableciendo así la conexión iSCSI a un nodo de computo diferente. Hay que tener en cuenta que cinder utiliza LVM e iSCSI para presentar los dispositivos de bloques sin formatos a las instancias de nova y es a través de una red.

5.3.7. Servicio de Almacenamiento de objetos

El servicio de almacenamiento basado en objetos también conocido como swift, proporciona a los usuarios almacenamiento redundante, escalable, de código abierto en la nube accesible a través de una interfaz API. Nos brinda servicios adicionales tales como alojamientos de sitios web estáticos, control de versiones de objetos, listas de control de acceso (ACL) y caducidad de acceso.

Hay que destacar que el servicio de swift se ha basado en el servicio de almacenamiento de objeto de Amazon llamado S3.

Swift escribe los objetos en varias unidades de disco distribuida por los servidores, este servicio es responsable de garantizar la replicación de datos y la integridad en todo el clúster, en caso de que falle un servidor o un

disco duro, este replica su contenido desde otros nodos activos a nuevas ubicaciones en los servidores.

Los servidores de almacenamiento escalan horizontalmente agregando simplemente nuevos servidores. Debido a que OpenStack utiliza un software para asegurar la replicación y distribución de datos a través de diferentes dispositivos se pueden usar discos duros económicos y servidores de bajo costo.

Arquitectura lógica del servicio de almacenamiento de objetos

El servicio de almacenamiento de objetos está compuesto por catorce demonios los cuales los vamos a describir en la siguiente figura (5.11).

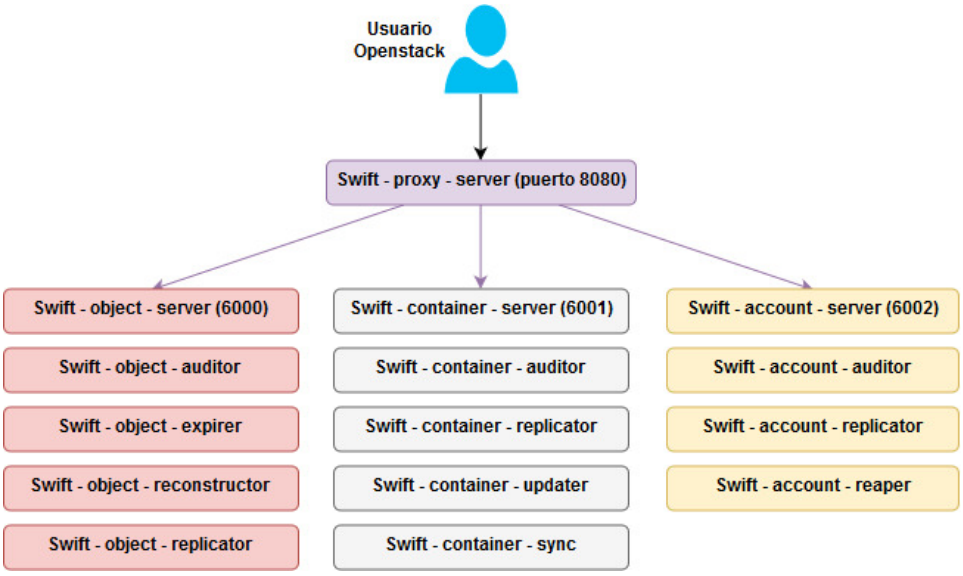

Figura 5.11: Diagrama de la arquitectura lógica de swift.

- **Swift– proxy-server**: Es el API y la puerta principal para swift, es el responsable de vincular el resto de la arquitectura swift y utiliza el puerto 8080, los usuarios deben interactuar con este demonio para crear contenedores, cargar objetos en estos contenedores, establecer ACL's y habilitar características especiales como el control de versiones de objetos o la habilitación de sitios web estáticos.
- **Swift–object–server**: Es el responsable de recuperar y eliminar objetos almacenados en las unidades locales en el clúster del swift, los objetos se almacenan como archivos binarios en el sistema de archivos, con los metadatos almacenados en los atributos extendidos del archivo,

conocidos como “xattrs”. Cada objeto se almacena una ruta derivada del has del nombre del objeto y la marca de tiempo de la operación

- **Swift–object-auditor**: Encargado de rastrear todo el sistema de objeto local comprobando la integridad de los objetos, si encuentra un archivo corrupto, el archivo se ponen en cuarentena y en su momento se reemplazará con otra replica.
- **Swift–object–expirer**: Ofrece la eliminación programada de los objetos.
- **Swift–object–reconstructor**: Permite reconstruir los objetos que han sido borrados.
- **Swift-object–replicator**: Mantiene el sistema en un estado coherente frente a las conexiones de error temporal, como por ejemplo intrusiones en la red o falla en la unidad de disco. Los procesos de replicación comparan los datos locales con cada copia remota para garantizar que todos contengan la última versión, este usa una “lista hash” para comprobar sus secciones de cada partición.
- **Swift-container–server**: Maneja la lista de objetos dentro de un contenedor particular, los listados se almacena como archivos de base de datos como SQLite.
- **Swift–container–auditor**: Rastrea el sistema del servidor local verificando la integridad de la base de dato SQLlite, si se encuentra aún la corrupción el archivo se pone en cuarentena y la replicación reemplazará el archivo por otra.
- **Swift–container–replicator**: Mantiene el sistema en un estado coherente frente a las conexiones de error temporal que hay con los archivos bases de datos SQLite del contenedor.
- **Swift–container–updater**: Esta encargado de actualizar la información del contenedor en la base de datos de la cuenta.
- **Swift–container–sync**: Es el responsable de permitir que el contenido de un contenedor se pueda migrar o copiar a otro contenedor a través de una sincronización en segundo plano.
- **Swift–account–server**: Administrar la lista de contenedores dentro de una cuenta, los listados se almacenas como archivos de base de datos SQLite y se replican en el clúster.
- **Swift–account–auditor**: Rastrea el sistema del servidor local verificando la integridad los archivos de la base de datos SQLite.

- **Swift–account–replicator**: Mantiene el sistema en un estado coherente frente a las conexiones de error temporal que hay con los archivos de la cuenta de la base de datos SQLite.
- **Swift–account–reaper**: Es el encargado de eliminar las cuentas que un administrador de OpenStack a solicitado para su eliminación.

5.3.8. Servicio de Telemetría

El servicio de telemetría también conocido como ceilometer, es el encargado de la recopilación centralizada de las métricas y los datos de monitorización de la nube de OpenStack. Nos permite obtener la información sobre los recursos de un proyecto, para ser utilizados para resolver diferentes casos de medición. Permite a los usuarios configurar los tipos de datos a recopilar accediendo a su API.

Arquitectura lógica del servicio de telemetría

El servicio de telemetría está compuesto por tres demonios los cuales los vamos a describir en la siguiente figura (5.12).

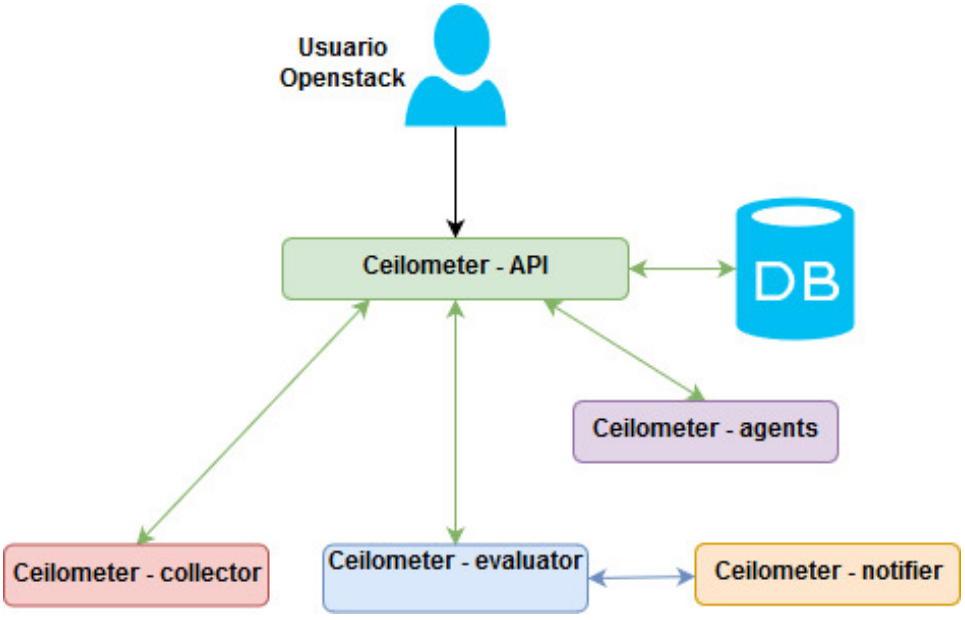

Figura 5.12: Diagrama de la arquitectura lógica de ceilometer.

- **Ceilometer–API**: Es la puerta de enlace principal para el servicio de ceilometer, proporciona acceso a la base de datos.

- **Ceilometer-agent**: Vigila los nodos y permite obtener una estadística del consumo de recursos.
- **Ceilometer–collector**: Permite monitorizar las colas de mensajes tales como notificaciones y métricas del demonio "ceilometer–agent".
- **Ceilometer–notifier**: Permite a los usuarios establecer alarmas basadas en las evaluaciones referentes a un proyecto.
- **Ceilometer–evaluator**: Permite evaluar las condiciones de los nodos para ponerlos en cola antes de pasar al demonio de "ceilometer–notifier".

5.3.9. Servicio de Orquestación

El servicio de orquestación, también conocido como *heat*, tiene como objetivo ayudar a los usuarios a modelar, configurar y automatizar la creación y administración de los recursos de OpenStack.

Para que los usuarios puedan desplegar instancias, usan *pilas* (es decir, plantillas que están en formato JSON, que describen tareas en términos de recursos, parámetros, restricciones y dependencias).

Arquitectura lógica del servicio de orquestación

El servicio de orquestación está compuesto por tres demonios los cuales los vamos a describir en la siguiente figura (5.13).

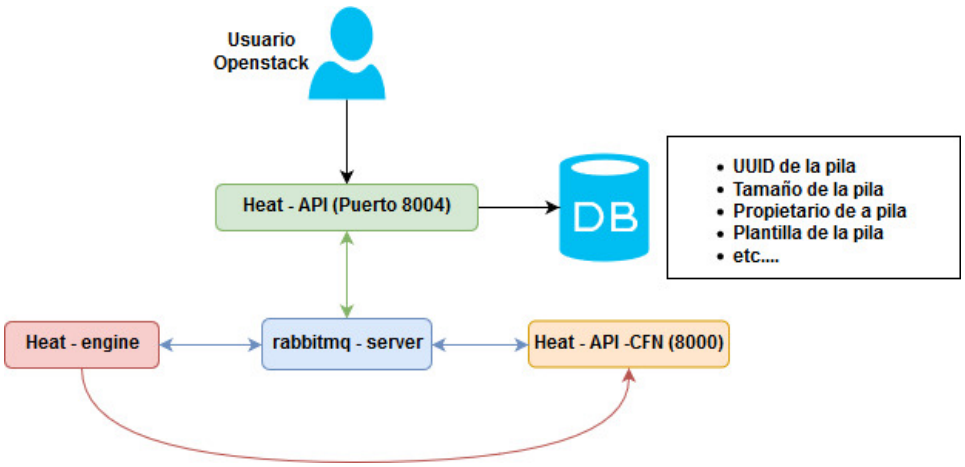

Figura 5.13: Diagrama de la arquitectura lógica de heat.

- **Heat–API**: Es la API y la puerta de enlace principal a heat, este está corriendo por el puerto 8004, los usuarios deben interactuar con ella crear, enumerar, administrar y eliminar las pilas.
- **Heat-API–CFN**: Proporciona una API de consulta, permite a los servicios y usuarios utilizar funcionalidades de CloudFormation, tales como condiciones de espera y opciones de autoescalado.
- **Heat–engine**: Es el responsable de iniciar las pilas y administrador todos los recursos específicos en la plantilla, comunicándose con el demonio "heat–API–CFN".

Capítulo 6

Diseño e Implementación

6.1. Diseño

El diseño del desarrollo del proyecto comprende la utilización de una plataforma de orquestación Open Source Mano (OSM), que se encargará de administrar y orquestar varios escenarios sobre la infraestructura de Openstack (encargada de emular un entorno de computación en la nube).

Por medio de OSM se despliegan dos servicios sobre la plataforma de Openstack: el servicio HTTP y el Servidor Lora.

- **EL servicio de HTTP**, es un programa informático que ejecuta una aplicación, la cual permite realizar una conexión con un cliente. [27]
- **Los servidores de Lora**, son los receptores de la información enviada por motas IoT, que utilizan la tecnología LoraWAN. [28]

Hay que tener en cuenta que OSM nos permitirá automatizar el despliegue y la escalabilidad, incluyendo la instancia automática cuando alguno de los nodos deja de funcionar, por ejemplo: en caso de alguna situación crítica como un incendio, un terremoto, etc.

En la figura 6.1 se muestra la arquitectura por capas y la relación que tienen los componentes. Podemos observar que se encuentra dividida en dos secciones; a la derecha tenemos OSM, la cual está encargada de gestionar y orquestar los servicios mencionados anterior (HTTP y Lora) y que se ejecuten en su propia "slice" (es decir, sobre los recursos reservados y gestionados por el orquestador); y, a la izquierda se muestra la plataforma Openstack, que provee los recursos físicos para implementar las máquinas virtuales, haciendo referencia a NFVI.

En la figura 6.2 se muestra el diseño de la arquitectura del proyecto. Consiste en interconectar la plataforma de Openstack con la de OSM de manera

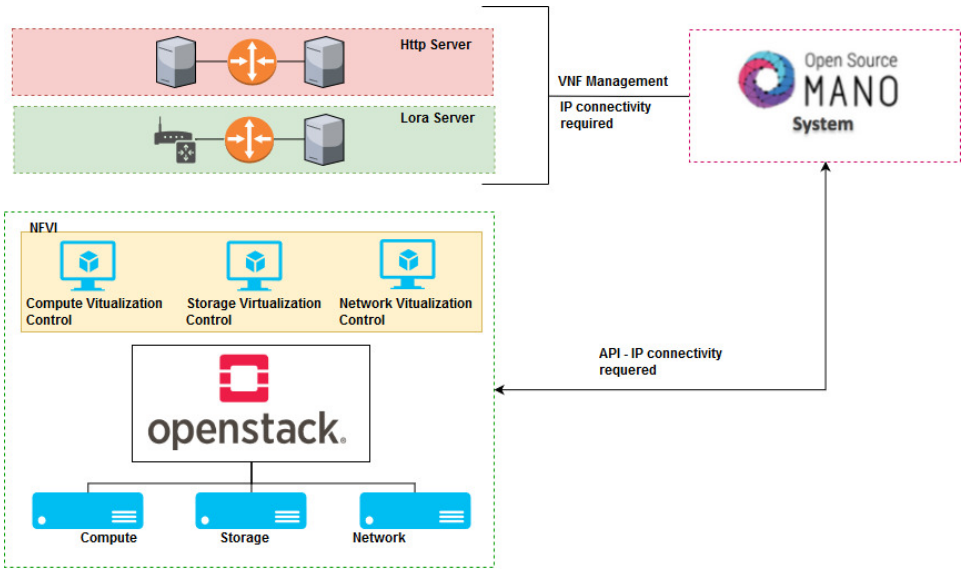

Figura 6.1: Diseño de la arquitectura lógica del proyecto.

que puedan comunicarse dinámicamente, permitiendo desplegar NFVs de forma automática y autoescalable.

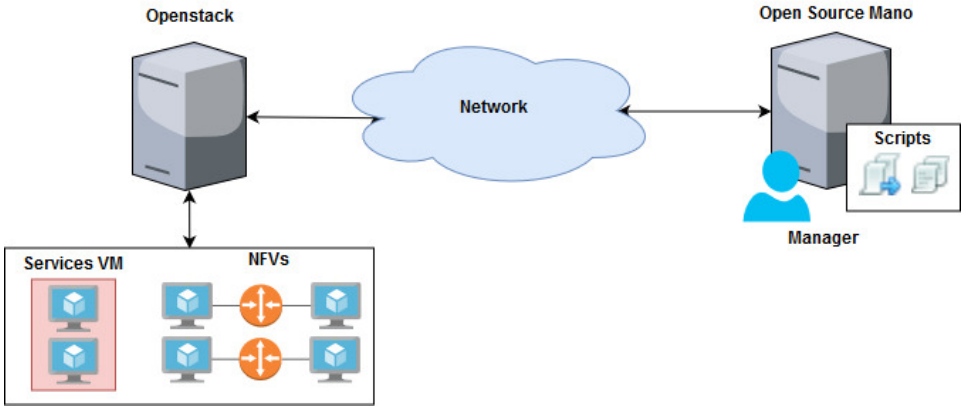

Figura 6.2: Diseño de la arquitectura del proyecto.

6.2. Implementación

En esta sección, se detallan los procedimientos realizados para la instalación de las herramientas Openstack y Open Source MANO, la interconexión de estos, y finalmente, la creación de los scripts correspondientes para el desarrollo del proyecto.

6.2.1. Despliegue de Openstack

A continuación, se va a describir cómo se ha realizado el despliegue de Openstack en un ordenador de altas prestaciones. En este despliegue, Openstack incluye todos los servicios mencionados en la tabla 5.1. El ordenador tiene las siguientes características:

- Sistema Operativo CentOs 7
- Procesador Core i7
- Memoria RAM de 32 Gb
- Disco duro de 1 Tb

La versión de Openstack que se utiliza en este proyecto es "PIKE", debido a que se intentó trabajar con las últimas versiones disponibles; pero, se presentaron varios inconvenientes al momento de agregar o configurar un servicio en específico, mostrándonos errores al desplegar una instancia o al crear un servicio de red e incluso al momento de agregar imágenes o sabores; a pesar de que todos los servicios estaban instalados correctamente.

Instalación de Openstack

Para la instalación de Openstack emplearemos la utilidad **PackStack all-in-one**, es una utilidad desarrollada en el lenguaje de programación Python, creada por desarrolladores de la comunidad de RDO Project[29], ya que es una solución que nos permite realizar el despliegue de Openstack en un único o en múltiples ordenadores con todas sus funcionalidades. Además, es más recomendada que la solución DevStack para proyectos educativos, debido a que es más estable al momento de agregar o configurar módulos.

Se comprobó la inestabilidad de la solución DevStack, ya que luego de instalarla, funcionó correctamente; sin embargo, al momento de agregar algún módulo, otro dejaba de funcionar, además, cuando se intentó acceder a instancias desplegadas, no se logró tener acceso a ellas.

Procedemos a configurar el servicio de red del ordenador para poder tener acceso externo a las instancias de OpenStack, para esto, es necesario desactivar varios servicios de red utilizando los siguientes comandos:

 # sudo systemctl disable firewalld

 # sudo systemctl stop firewalld

 # sudo systemctl disable NetworkManager

 # sudo systemctl stop NetworkManager

 # sudo systemctl enable network

 # sudo systemctl start network

Deshabilitamos el módulo de seguridad *SELinux*, ya que este proporciona un sistema de control obligatorio de acceso (políticas de seguridad). Para ello, procedemos a ir a la ruta ***/etc/sysconfig/*** y editamos el archivo **seLinux**:

 # nano /etc/sysconfig/seLinux

SELinux = ***disabled***

Después, se procede a actualizar el sistema:

 # sudo yum update -y

Luego, se añade el repositorio de la versión a utilizar, en este caso utilizamos **PIKE** de OpenStack con el siguiente comando:

 # sudo yum install -y centos-release-openstack-pike

Actualizamos el sistema con el siguiente comando:

 # sudo yum update -y

Se procede a instalar el **Packstack Installer** con el siguiente comando:

 # sudo yum install -y openstack-packstack

Y finalmente, ejecutamos la utilidad de PackStack para proceder a instalar el entorno de OpenStack con todos los módulos mencionados en la tabla(5.1) utilizando el siguiente comando:

$\#\ sudo\ packstack\ --allinone\ --provision-demo=n\ --os-heat-install=y\ --os-neutron-ovs-bridge-mappings=extnet:br-ex\ --os-neutron-ovs-bridge-interfaces=br-ex:enp0s31f6\ --os-neutron-ml2-type-drivers=vxlan,flat$

Una vez terminada la instalación, la utilidad *PackStack* creará dos archivos que se encuentran en el directorio */root*, los cuales son:

- ***keystonerc_admin***: Contiene los datos establecidos como variables de entorno de usuarios para OpenStack y el enlace de acceso al dasboard.
- ***packstack-answers-20180227-112339.txt*** : Contiene la configuración establecida en la instalación y que posteriormente se podrá usar para modificaciones de algún componente de OpenStack.

Además, la utilidad **PackStack** genera como salida en la línea de comandos un enlace que nos permitirá acceder al dashboard de OpenStack, por ejemplo: *http://10.5.1.2/dashboard/*.

Para visualizar el archivo *keystonerc_admin* y obtener el usuario y la contraseña escribimos el siguiente comando:

 # cat keystonerc_admin

Hay que tener en cuenta que para poder ejecutar algún comando relacionado con los componentes de OpenStack, tenemos que cargar como variable de entorno las variables establecidas en el archivo *keystonerc_admin*, y esto se lo hace ejecutando el siguiente comando:

 # source keystonerc_admin

A continuación, en la figura 6.3 se muestra la interfaz de usuario de Openstack como prueba de la instalación exitosa.

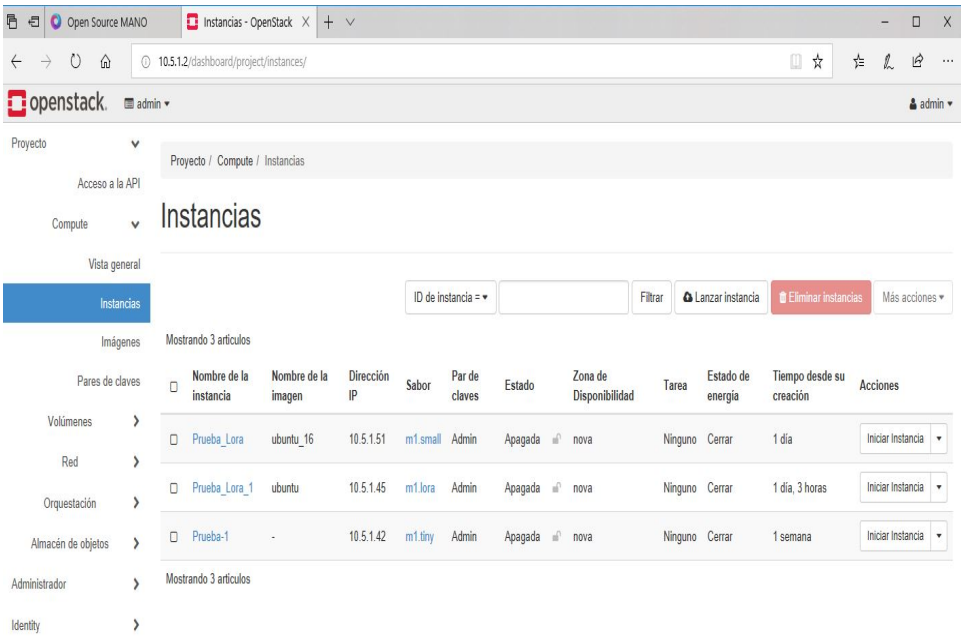

Figura 6.3: Interfaz de usuario de Openstack.

Configuramos la red externa, que proveerá el acceso a Internet a las instancias en Openstack con los siguientes comandos:

 # neutron net − create public − −provider : network_type flat − −provider : physical_network extnet − −shared − −router : external = True

Luego, creamos una subnet que será asociada a la red externa:

 # neutron subnet-create public –name public_subnet–allocation-pool –start=10.5.1.10,end=10.5.1.40 –disable-dhcp–gateway 10.5.1.1 10.5.1.0/8

También, creamos una red privada:

 # neutron net-create private_prueba

 # neutron subnet-create private_prueba –name private_subnet_prueba –dns-nameserver 8.8.8.8 –gateway 192.168.1.1 192.168.1.0/24

Finalmente, creamos el enrutador de Openstack y le asignamos las redes creadas anteriormente:

 # neutron router-create Router-Virtual

 # neutron router-interface-add Router-Virtua private_subnet_prueba

 # neutron router-gateway-set Router-Virtua public

Obteniendo una arquitectura como se muestra en la figura: 6.4.

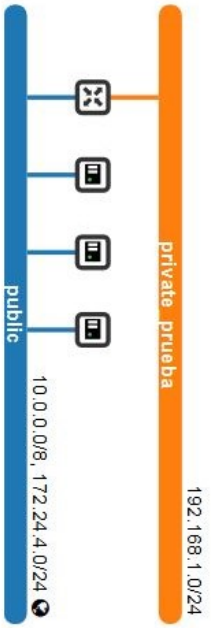

Figura 6.4: Topología de red en Openstack.

También se procedió a realizar la instalación de la distribución DevStack de Openstack, el procedimiento seguido para dicha instalación se encuentra en el apéndice A.

6.2.2. Despliegue de Open Source Mano

A continuación, se va a describir cómo se ha realizado el despliegue de OSM en un ordenador de altas prestaciones. En este despliegue, OSM está compuesto por un *framework* que ofrece la comunidad Open Source Mano. El ordenador tiene las siguientes características:

- Sistema Operativo Ubuntu 16.04.5 LTS
- Procesador Core i7
- Memoria RAM de 32 Gb
- Disco duro de 1 Tb

La versión que se utiliza en este proyecto es la cuarta, debido a que ofrece una instalación OSM con contenedores nativos de la nube, presenta un bus "kafka" que permite las comunicaciones asíncronas y ofrece características que permiten gestionar las fallas y el rendimiento.

Instalación de Open Source Mano

Una vez listo el ordenador procedemos a descargar la pila versión 4:

 # wget https://osm-download.etsi.org/ftp/osm-4.0four/install_osm.sh

Modificamos los permisos de la pila descargada:

 # chmod +x install_osm.sh

Procedemos con la instalación:

 # ./install_osm.sh

La comunidad de OSM, nos indica que es recomendable, guardar el registro completo del proceso de instalación para que a futuro se puedan solucionar posibles problemas generados al momento de la instalación.

*# ./install_osm.sh*2 > &1|*teeosm_install_log.txt*

Cuando se procede a instalar se mostrarán varios mensajes de diálogo relacionado con diversas configuraciones:

- *Desea instalar y configurar LXD, juju, docker CE, inicializar los contenedores y pre requisitos*:**SI**
- *Desea configurar el brigde LXD*: **SI**
- *Desea configurar la subred IPv4*:**SI**
- *Las siguientes preguntas, dejarlas con las respuestas predeterminadas*
- *Desea configurar la subred IPv6*:**NO**

Luego de la instalación, es necesario exportar y definir las siguientes variables de entorno:

 # export OSM_HOSTNAME=10.5.1.3

 # export OSM_SOL005=True

Verificamos que todos los contenedores estén ejecutándose correctamente como se muestra en la figura 6.5:

 # docker stack ps osm | grep −i running

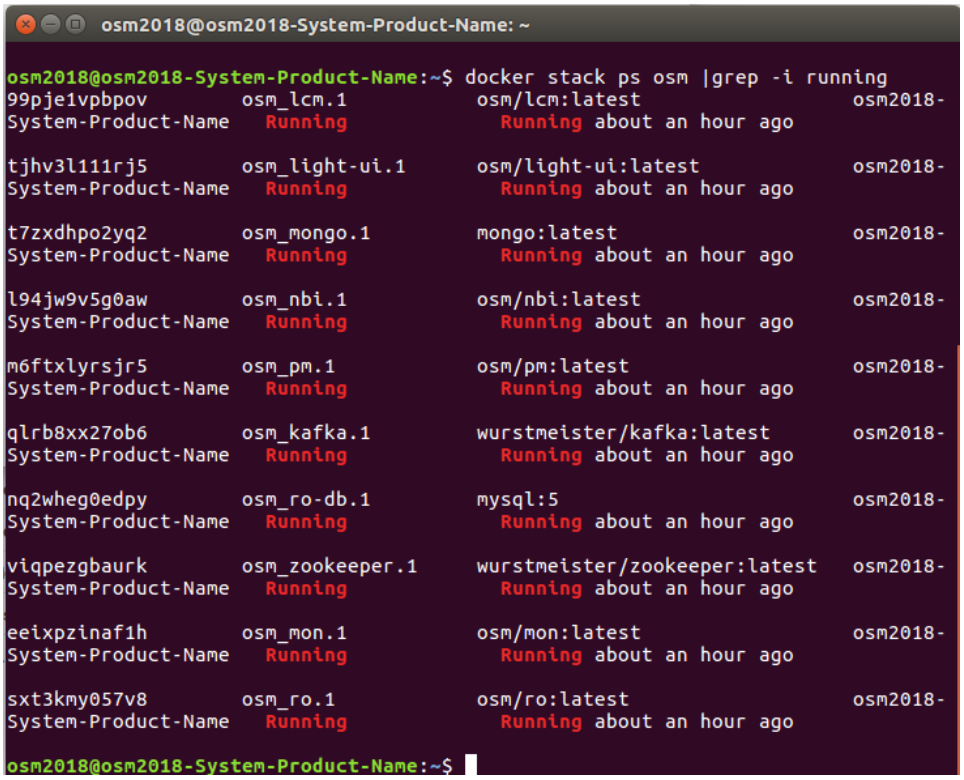

Figura 6.5: Contenedores que integran OSM.

Después de que finalice la instalación, se podrá acceder a la interfaz de usuario con el siguiente URL:

 # http://10.5.1.3/projects/

En la figura 6.6 se muestra la interfaz de usuario de OSM como prueba de la instalación exitosa.

6.2.3. Acoplamiento de Openstack en Open Source Mano

Se puede acoplar Openstack al OSM de dos formas:

1. A través del cliente OSM, con el siguiente comando:

   # osm vim-create –name openstack-site –user admin –password wmnt2018

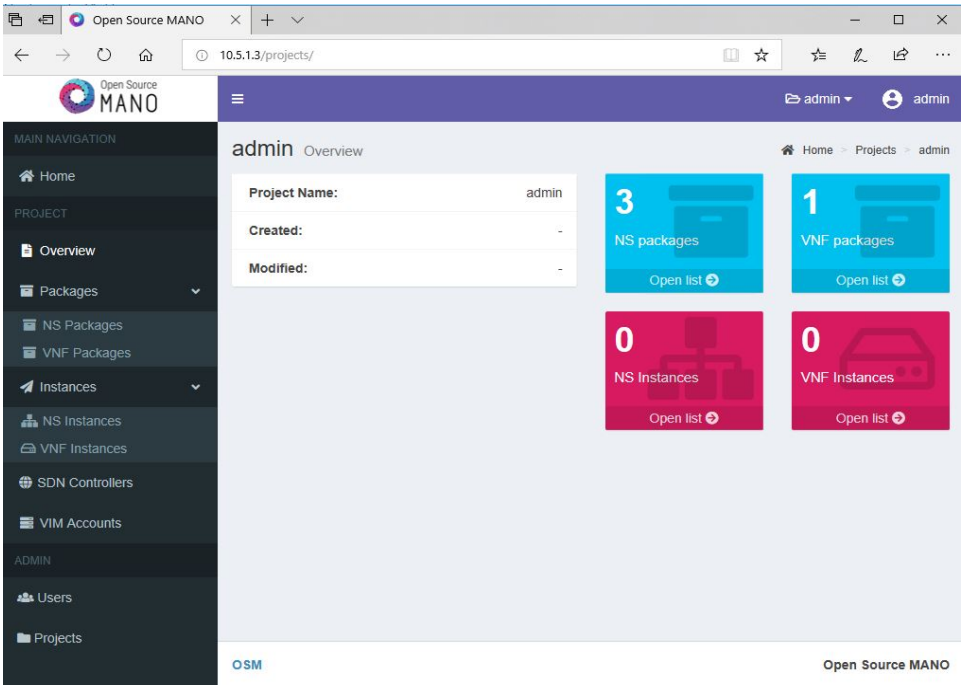

Figura 6.6: Interfaz de usuario de OSM.

–auth_url http://10.5.1.2:5000/v3 –tenant admin –account_type openstack –config='{security_groups:wimunet, keypair: Admin}'

En este comando se indica: el usuario administrador de Openstack, la contraseña, el enlace de Openstack y el nombre del proyecto donde se va a trabajar; además, el grupo de seguridades, y la llave de acceso.

2. A través de la interfaz gráfica de OSM, accediendo a la pestaña *VIM Accounts*, hacemos clic en el botón *New VIM* y completamos los parámetros.

   En la figura 6.7 se puede observar los parámetros que son solicitados al agregar un VIM.

6.2.4. Desarrollo de Scripts

Se procedió a desarrollar diferentes scripts, que mencionaremos a continuación:

- Se desarrolló un script que permite instanciar máquinas virtuales en Openstack, incluyéndole diferentes servicios.

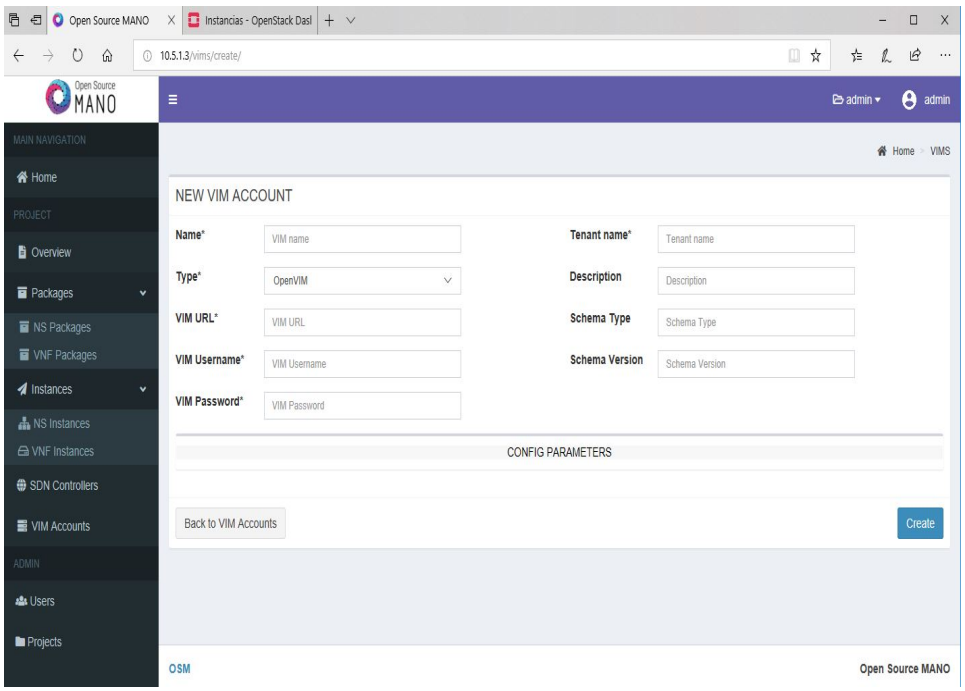

Figura 6.7: Interfaz de usuario de OSM para agregar un VIM.

Listing 6.1: Script para instanciar HTTP.

```
#!/bin/bash
echo "Creando la VM"

source keystonerc_admin

echo $(openstack server create --flavor m1.http  --image cirros
    ↪ --nic net-id=public --security-group wimunet --key-name
    ↪ Admin Servidor_HTTP --user-data http.sh)
echo "Creacion Finalizada"

DIA=‘date +"%d/%m/%Y"‘
HORA=‘date +"%H:%M"‘

echo "Se creo la VM el $DIA a las $HORA!"
```

1. Muestra un mensaje indicando que se va a crear la máquina virtual.
2. Ingresa a Openstack.
3. Ordena que se instancie una máquina virtual indicándole las características requeridas y agregándole los parámetros de *http.sh* que contiene la configuración de dicho servicio.
4. Muestra un mensaje indicando que se instanció correctamente la maquina con el día y la hora.

- Se desarrolló un script que permite instanciar máquinas virtuales, utilizando la herramienta heat que proporciona Openstack.

Listing 6.2: Script para instanciar HTTP con Heat.

```
#!/bin/bash
echo "Creando la VM"

source keystonerc_admin

echo $(openstack stack create -t instancia-http.yaml prueba-heat)

echo "Creacion Finalizada"

DIA=‘date +"%d/%m/%Y"‘
HORA=‘date +"%H:%M"‘
echo "Se creo la VM el $DIA a las $HORA!"
```

1. Muestra un mensaje indicando que se va a crear la máquina virtual.
2. Ingresa a Openstack.
3. Ordena que se instancie una máquina virtual usando la pila *instancia-http.yaml*
4. Muestra un mensaje indicando que se instanció correctamente la maquina con el día y la hora.

- Se crearon los paquetes VNF y NS, los que contienen scripts que permiten desplegar instancias VNF y NS desde OSM en Openstack.
- Se desarrolló un script que me permite monitorizar si las instancias VNF y NS creadas están levantadas, en caso de que no lo estén, las vuelve a desplegar en Openstack.

A continuación, con ayuda de la figura 6.8 se explicará en que consiste el script de monitorización desarrollado en *Pyhton*, mismo que contiene a todos los scripts mencionados anteriormente.

- Se Ingresa al bucle la cual me permitirá realizar la funcion cada cierto tiempo.
- Se llama a la ejecución de un script llamado *dameip.sh* con el fin de saber cual es la IP del servicio especifico que esta ejecutado en Openstack y a su vez es almacenada en un fichero de texto llamado *IP.txt*.
- Se procede a leer el fichero de texto y le asignamos dicho contenido a la variables *hostname*.

```
import time
import os
import subprocess
import shlex

while(1):
    subprocess.call(shlex.split('./dameip.sh'))                    1
    hostname = open("ip.txt").readline().rstrip()                  2
    response = os.system("ping -c 1 " + hostname)                  3
    if response == 0:                                              4
        print "Hay conexion"
    else:
        print "No hay conexion"
        subprocess.call(shlex.split('./instancia.sh'))             5
    time.sleep(30)
```

Figura 6.8: Script de monitoreo de instancias en Openstack.

- Se procede a verificar el estado del servidor mediando el uso del *ping*.
- Dependiendo del resultado se muestra un mensaje o se procede a ejecutar el script llamado *instancia.sh* la cual despliega las instancias VNF y NS en Openstack.

El script de *monitorización.py* se lo puede ejecutar desde el ordenador de Openstack o desde el de OSM.

Capítulo 7

Verificación del funcionamiento del sistema

En este capítulo se valida la configuración y funcionamiento del despliegue de los servicios. Se comprueba la configuración de las diferentes herramientas que se utilizan para el proyecto, y la correcta ejecución de los *scripts* desarrollados.

Cada sección explica la ejecución de cada prueba para verificar el funcionamiento, mostrando el resultado.

7.1. Descripción del entorno experimental

En la figura 7.1 se detalla el entorno experimental del proyecto, en donde se indica la dirección IP y el nombre que tiene cada servidor, para poder referirnos a cada uno de ellos en las explicaciones de la siguiente sección.

7.2. Monitorización e instanciación de servicios desde el servidor de Openstack

Aunque el objetivo final es ejecutar dos slices, una con un servicio de red, gestionado por MANO, en este escenario se comprueba que desde OpenStack pueden levantarse las funciones virtualizadas utilizando solo OpenStack.

Procedemos a instanciar el servicio de Lora desde el cliente Openstack. Para aquello se ejecuta el script llamado *instancia_lora.sh* con el siguiente comando:

- *source instancia_lora.sh*

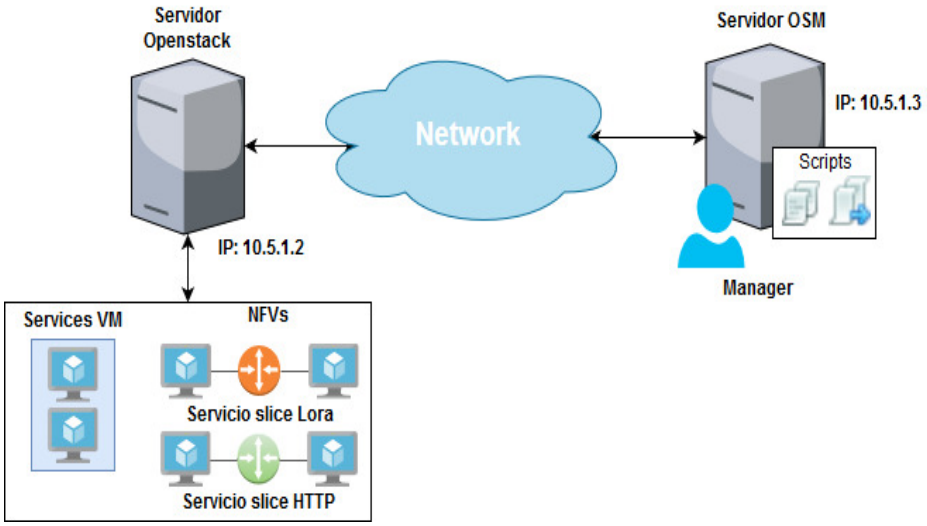

Figura 7.1: Arquitectura del proyecto.

Cabe mencionar que este script es similar al que se utilizó en el apartado anterior para instanciar HTTP 6.1, con la única diferencia de que en este se agregan los parámetros de Lora.

Procedemos a ejecutar el script de monitorización llamado *monitoring_lora.py* para el servicio que se instanció anteriormente con el siguiente comando:

- *python monitoring_lora.py*

Cabe mencionar que este script es similar al que se utilizó en el apartado anterior de monitorización 6.2.

En la figura 7.5 se puede observar que el servicio se encuentra levantado.

Procedemos a apagar el servicio forzosamente ejecutando el script llamado *apagado.sh*:

- *source apagado.sh*

Listing 7.1: Script para apagar instancias.

```
1 #!/bin/bash
2 echo "Apagando Servidor"
3
4 source keystonerc_admin
5
6 echo $(nova stop Servidor_Lora)
7
8 DIA=‘date +"%d/%m/%Y"‘
```

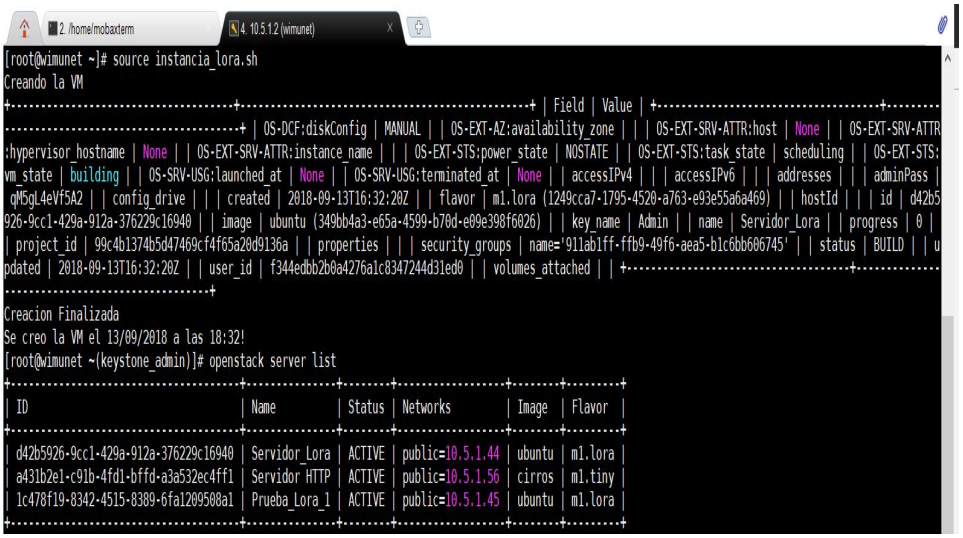

Figura 7.2: Instancia del Servidor Lora.

```
9  HORA=`date +"%H:%M"`
10 echo "Se Apago la VM el $DIA a las $HORA!"
```

1. Muestra un mensaje indicando que se a apagar la instancia.
2. Ingresa a Openstack.
3. Ordena que se apague instancia requerida.
4. Muestra un mensaje indicando que se apago correctamente la maquina virtual con el día y la hora.

En la figura 7.6 se puede observar que el servicio se apagó correctamente.

En la figura 7.7 se puede observar que el script de *monitoring_lora.py* presento un mensaje diciendo que no tiene acceso al servicio y que inmediatamente procede a instanciar nuevamente el servicio.

7.3. Monitorización e instanciación de servicios desde el Servidor de OSM sobre el Servidor de Openstack

Procedemos a monitorizar e instanciar el servicio de HTTP desde OSM, ejecutando el script de *monitoring_HTTP.py*:

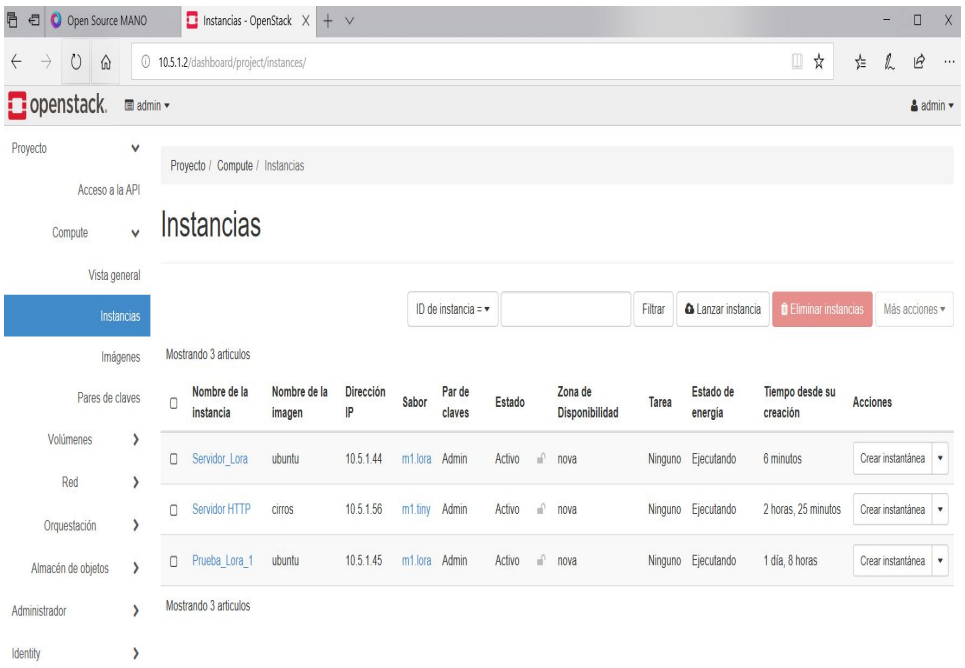

Figura 7.3: Instancia del Servidor Lora en Interfaz gráfica de Openstack.

- *python monitoring_HTTP.py*

Cabe mencionar que este script es similar al que se utilizó en el apartado anterior de monitorización 6.2.

En la figura7.9 se puede observar que el servicio HTTP se encuentra levantado.

Procedemos a apagar el servicio forzosamente ejecutando el script llamado *apagado.sh*:

- *source apagado.sh*

Cabe mencionar que este script es similar al que se utilizó para apagar el servicio de Lora,7.1.

En la figura7.8 se puede observar que el servicio se apagó correctamente.

En la figura7.9 se puede observar que el script de *monitoring_http.py* presento un mensaje diciendo que no tiene acceso al servicio y que inmediatamente procede a instanciar nuevamente los servicios de HTTP.

Cabe mencionar que este script es similar al que se utilizó en el apartado anterior de monitorización 6.2, con la diferencia de que ahora llama al script

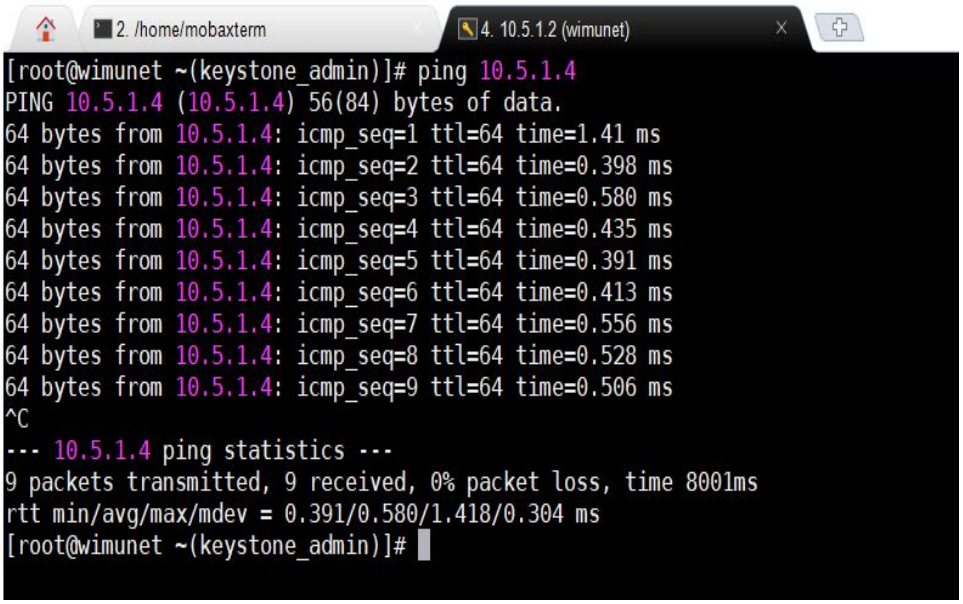

Figura 7.4: Verificación de ping al Servidor Lora.

instanciar_slicing_http.sh cuando no tiene acceso al servicio.

Listing 7.2: Script para instanciar el servicio HTTP slice.

```
#!/bin/bash

echo "Los servicios instanciados desde OSM a Openstack son los
    ↪ siguientes:"

echo $(osm --hostname 10.5.1.3 vnfd-list)
echo $(osm --hostname 10.5.1.3 ns-list)

echo "Se proceden a instanciar los servicios de HTTP"

echo $(osm --hostname 10.5.1.3 ns-create --nsd_name cirros_2vnf_ns --
    ↪ ns_name servicio_http --vim_account wimunet)

DIA=‘date +"%d/%m/%Y"‘
HORA=‘date +"%H:%M"‘
echo "Se instanciaron los servicios de HTTP el $DIA a las $HORA!"

echo "Los servicios instanciados desde OSM a Openstack son los
    ↪ siguientes:"
echo $(osm --hostname 10.5.1.3 vnfd-list)
echo $(osm --hostname 10.5.1.3 ns-list)
```

1. Muestra los servicios instanciados en el OSM.

2. Procede a instanciar el servicio de HTTP en Openstack desde OSM.

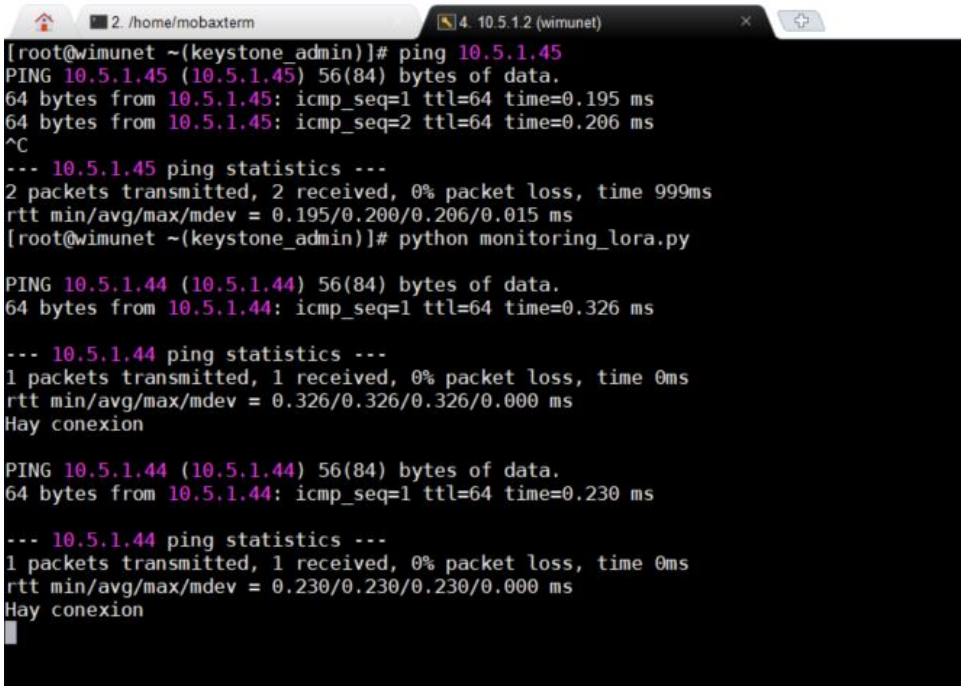

Figura 7.5: Ejecución del script que monitoriza el Servidor Lora.

3. Muestra un mensaje indicando que se instanció correctamente el servicio HTTP slice con el día y la hora.

4. Muestra los servicios instanciados en el OSM.

En la figura 7.10 podemos observar que se han creado servicios slices de Lora y de HTTP.

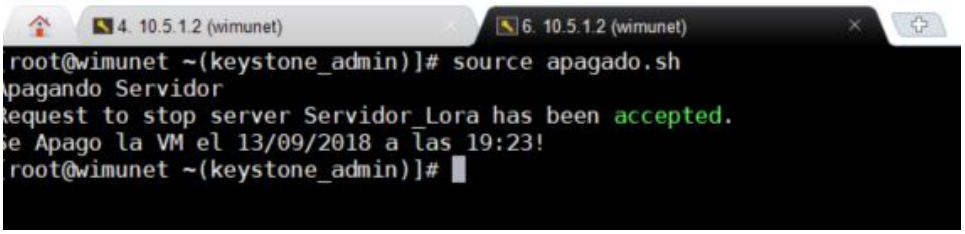

Figura 7.6: Apagado forzoso del Servidor Lora.

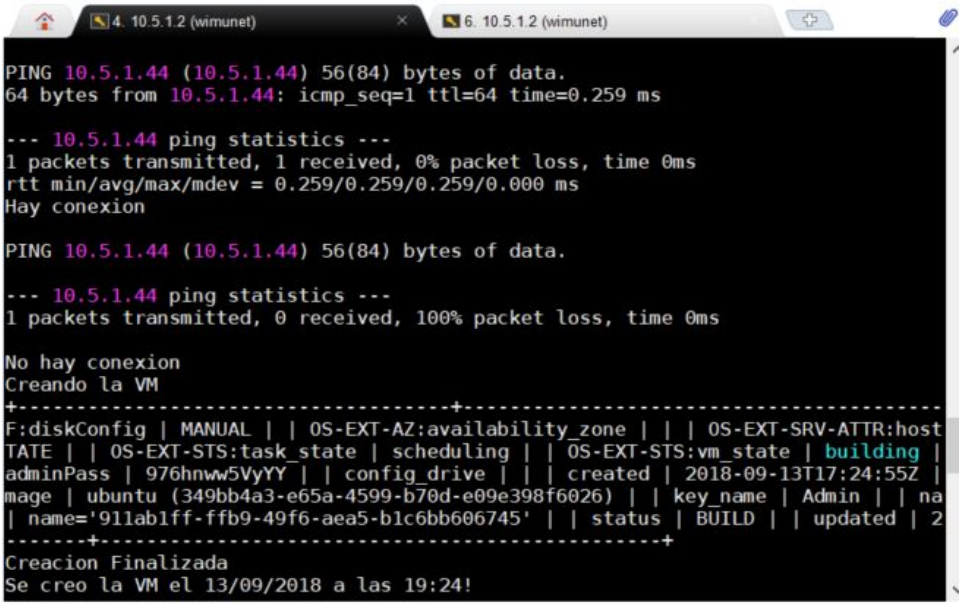

Figura 7.7: Apagado forzoso del Servidor LoRa.

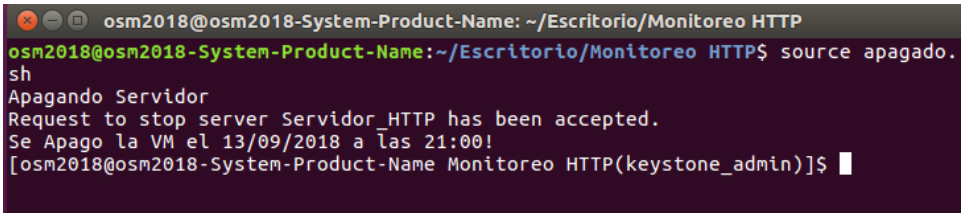

Figura 7.8: Apagado forzoso de los servicios HTTP.

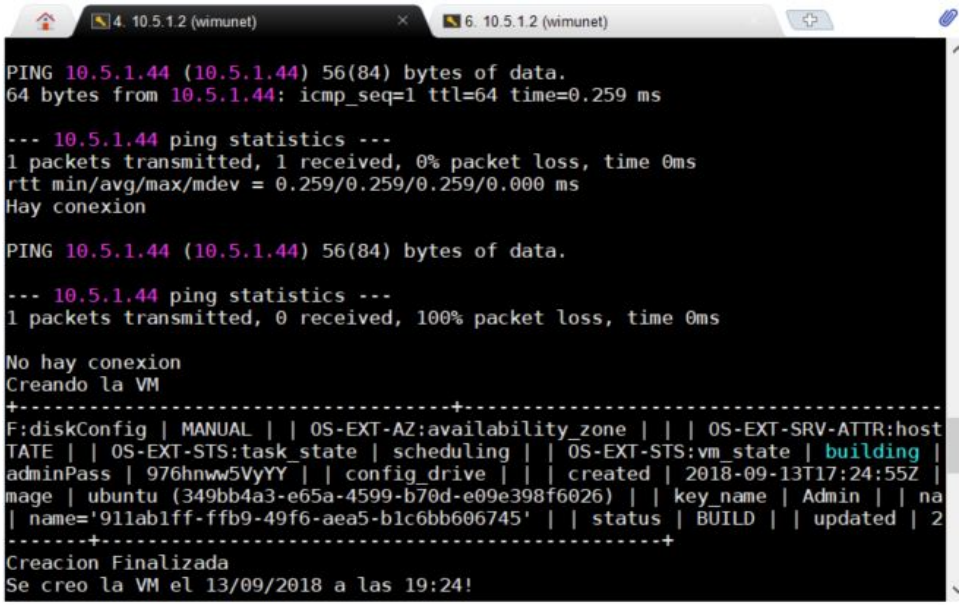

Figura 7.9: Instancias de los servicios HTTP desde OSM.

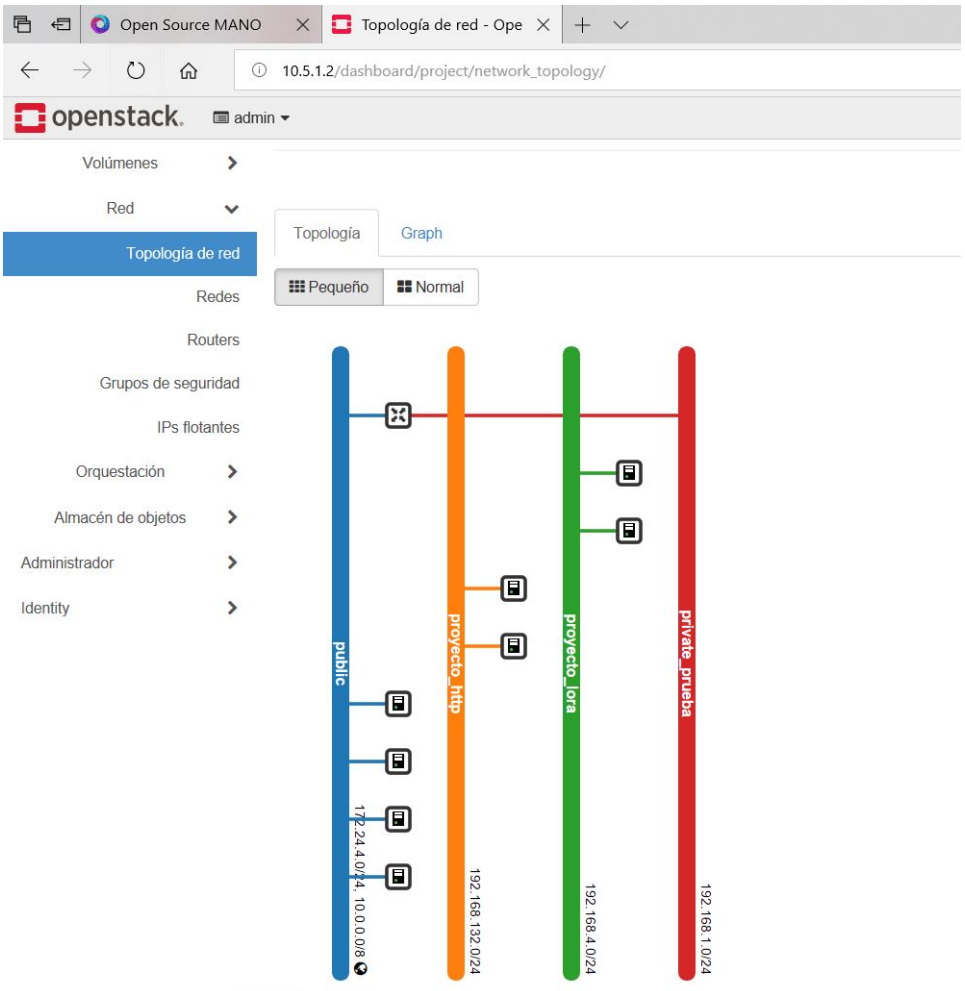

Figura 7.10: Instancias de los servicios HTTP desde OSM.

Capítulo 8

Conclusiones

En este capítulo se desglosan las conclusiones obtenidas tras la realización de este trabajo. Dichas conclusiones corresponden a cada uno de los objetivos planteados inicialmente:

- Para satisfacer el objetivo general, se ha diseñado e implementado un escenario con dos servicios de red diferentes. Estos servicios se han ejecutado sobre dos *slices* de red, como ejemplo de caso de uso de las tecnologías de virtualización de redes y gestión y orquestación de funciones de red virtualizadas.

- Para ello, se hizo una revisión bibliográfica acerca del marco estandarizado para orquestación de servicios de red MANO (*Management and Orchestation*), para poder plasmar en este documento la información más relevante. Así mismo, con esa información se realizó la instalación y configuración de este servicio.

- Se llevó a cabo una revisión bibliográfica acerca de la gestión de infraestructura virtual (*Virtual Infrastructure Management*), para poder plasmar en este documento la información más relevante. Así mismo, con esa información pudimos realizar la instalación y configuración de este servicio.

- Se mostró cómo se crearon las instancias de los servicios virtualizados de Lora y HTTP en Openstack.

- Se crearon *slices* de red distintas desde un servicio OSM (*OpenSource MANO*) con Openstack como VIM.

- Se verificó la implementación del diseño realizado, probando el correcto funcionamiento de la configuración realizada.

- Por último, se ha diseñado un orquestador propio, que monitoriza el estado de los servicios desplegados, y que es capaz de volver a instanciar dichos servicios en caso de caída. Este monitor funciona haciendo uso de las herramientas de MANO y de OpenStack de manera automática. Esta solución permite que los servicios desplegados puedan levantarse automáticamente ante situaciones críticas de caída de la infraestructura virtualizada.

Como contribución adicional, el presente documento aspira a ser una guía para las personas que necesiten instalar y configurar servicios virtuales con tecnologías de funciones virtuales de red, OpenStack para ofrecer infraestructuras virtualizadas, y Open Source MANO para orquestarlos.

Bibliografía

[1] Fernando Rivero. Informe ditrendia 2016: Mobile en españa y en el mundo, 2016.

[2] Teléfonos móviles y smartphones: usuarios mundiales 2014-2019 | estadística. URL: https://es.statista.com/estadisticas/723622/usuarios-de-telefonos-moviles-y-smartphones-en-el-mundo/.

[3] Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[4] Amit Singh. An introduction to virtualization. *kernelthread. com, January*, 2004.

[5] Escalabilidad. URL: https://definicionyque.es/escalabilidad/.

[6] Microsoft Azure. Qué es virtualización - definición — microsoft azure. URL: https://azure.microsoft.com/es-es/overview/what-is-virtualization/.

[7] La virtualización de redes y las redes virtuales - administración de oracle solaris: interfaces y virtualización de redes. URL: https://docs.oracle.com/cd/E26921_01/html/E25833/gfkbw.html.

[8] Juliver de Jesus Gil Herrera and Juan Felipe Botero Vega. Network functions virtualization: A survey. *IEEE Latin America Transactions*, 14(2):983–997, 2016.

[9] ETSI - welcome to the world of standards! URL: https://www.etsi.org/.

[10] Sabine Dahmen-Lhuissier. Industry specification groups (ISGs). URL: https://www.etsi.org/about/how-we-work/how-we-organize-our-work/industry-specification-groups-isgs.

[11] Ultan Mulligan. Network functions virtualisation. URL: https://www.etsi.org/technologies-clusters/technologies/nfv.

[12] Network Functions Virtualisation. Use cases. *ETSI GS NFV*, 1:V1, 2013.

[13] Management and orchestration (MANO). URL: https://iricent.com/solutions/mano/.

[14] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Steven Latré, Marinos Charalambides, and Diego Lopez. Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105, 2016.

[15] OpenMANO. URL: http://www.tid.es/long-term-innovation/network-innovation/telefonica-nfv-reference-lab/openmano.

[16] Sabine Dahmen-Lhuissier. Open source MANO. URL: https://www.etsi.org/technologies-clusters/technologies/nfv/open-source-mano.

[17] List of OSM members. URL: https://portal.etsi.org/TBSiteMap/OSM/ListofOSMMembers.aspx.

[18] OSM release FOUR - OSM public wiki. URL: https://osm.etsi.org/wikipub/index.php/OSM_Release_FOUR.

[19] OpenBaton documentation. URL: http://openbaton.github.io/documentation/.

[20] Open baton: an open source reference implementation of the ETSI network function virtualization MANO specification. URL: http://openbaton.github.io/.

[21] What is virtualized infrastructure manager (VIM)? definition. URL: https://www.sdxcentral.com/nfv/definitions/virtualized-infrastructure-manager-vim-definition/.

[22] OpenVIM | open source projects from telefonica NFV lab. URL: https://www.sdxcentral.com/projects/openvim/.

[23] Open source software for creating private and public clouds. URL: https://www.openstack.org/.

[24] vCloud director - multi tenant cloud provisioning and management platform. URL: https://www.vmware.com/products/vcloud-director.html.

[25] ¿qué es AWS? – amazon web services. URL: https://aws.amazon.com/es/what-is-aws/.

[26] OSM release THREE - OSM public wiki. URL: https://osm.etsi.org/wikipub/index.php/OSM_Release_THREE.

[27] Tim O'Reilly. Qué es web 2.0. patrones del diseño y modelos del negocio para la siguiente generación del software. *Boletín de la Sociedad de la Información: Tecnología e Innovación*, pages 177–201, 2006.

[28] LoRa server, open-source LoRaWAN network-server. URL: https://www.loraserver.io/.

[29] Inc. Red Hat. Packstack: Create a proof of concept cloud. URL: https://www.rdoproject.org/install/packstack/.

[30] Openstack.org. OpenStack Docs: DevStack. URL: https://docs.openstack.org/devstack/latest/.

[31] Openstack.org. Devstack - openstack. URL: https://wiki.openstack.org/wiki/DevStack.

Apéndice A

Instalación de DevStack

A.1. Guía de instalación de Openstack con DevStack

DevStack nos brinda un conjunto de líneas de códigos llamados también scripts, los que hacen posible la instalación de OpenStack de forma automática, se basa en un repositorio "git máster" que es actualizado constantemente por varios desarrollares ya que se usa de manera interactiva y es una base para realizar pruebas futuras en OpenStack.[30]

Hay que recalcar que DevStack no fue diseñado para un entorno empresarial, es porque después de un reinicio, puede presentar algún tipo de falla al iniciarlo. [31]

Para realizar la instalación de OpenStack con DevStack, montaremos en el ordenador una la distribución de Linux llamado Ubuntu 16.04, ya que según las investigaciones es el más probado, y el más simple.

Procedemos a crear un usuario para poder instalar DevStack, el cual tendrá los privilegios de administrador y lo hacemos con los siguientes comandos.

- *useradd -s /bin/bash -d /opt/stack -m stack*
- *echo "stack ALL=(ALL) NOPASSWD: ALL»¿/etc/sudoers*

Iniciamos sesión con el nuevo usuario creado e instalamos el software de *Git.*

- *sudo apt-get -y install git*

Descargamos la última versión de DevStack a través del siguiente comando:

- *git clone https://git.openstack.org/openstack-dev/devstack*

Ingresamos al directorio llamado *devstack*

- *cd devstack*

Creamos un archivo llamado *local.conf* la cual tendrá las siguientes configuraciones que son necesarias para que DevStack nos permita instalar OpenStack:

FLOATING_RANGE = 192.168.1.1/24
FIXED_RANGE = 10.10.10.0/24
FIXED_NETWORK_SIZE = 256
FLAT_INTERFACE = enp3s0:1
ADMIN_PASSWORD = pa$$w0rd
DATABASE_PASSWORD = pa$$w0rd
RABBIT_PASSWORD = pa$$w0rd
SERVICE_PASSWORD = pa$$w0rd

- *FLOATING_RANGE*, es el rango de direcciones IP Flotantes.
- *FIXED_RANGE*, es el rango de direcciones IP Internas para asignar a las instancias.
- *FIXED_NETWORK_SIZE = 256*
- *FLAT_INTERFACE*, es la interfaz ethernet que conecta la instancia a su red local.
- *ADMIN_PASSWORD*, es la contraseña para usuario administrador.
- *DATABASE_PASSWORD*, es la contraseña para administrar la base de datos.
- *RABBIT_PASSWORD*, es la Contraseña para administrar la colas de mensajes (RabbitMQ).
- *SERVICE_PASSWORD*, es la contraseña que utilizan ciertos servicios de OpenStack para autenticarse con Keystone.

Iniciamos el proceso de instalación de Openstack ejecutando el script *stack.sh*, este se encarga de leer los parámetros de configuración del archivo *local.config* con el siguiente comando:

- *./stack.sh*