INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Desarrollo de equipos de red virtuales para la Red Institucional del Instituto Politécnico Nacional

TESIS
QUE PARA OBTENER EL GRADO DE
Maestro en Ciencias en Ingeniería de Telecomunicaciones

PRESENTA:

Ing. Salvador Ruiz León

Directores de Tesis
Dr. Salvador Álvarez Ballesteros
M.C. Miguel Ángel Bennetts Fernández

Ciudad de México Mayo, 2019

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 12:00 horas del día 23 del mes de Mayo del 2019 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de E.S.I.M.E. para examinar la tesis titulada:

"DESARROLLO DE EQUIPOS DE RED VIRTUALES PARA LA RED INSTITUCIONAL DEL INSTITUTO POLITÉCNICO NACIONAL"

Presentada por el alumno:

RUIZ	LEÓN	SALVADOR
Apellido paterno	Apellido materno	Nombre(s)

Con registro: B 1 7 1 1 7 4

aspirante de:

MAESTRO EN CIENCIAS EN INGENIERÍA DE TELECOMUNICACIONES

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

DR. SALVADOR ÁLVAREZ BALLESTEROS

M. EN C. MIGUEL ÁNGEL BENNETTS FERNÁNDEZ

Presidente

DR. VLADIMIR KAZAKOV

Segundo Vocal

M. EN C. MIGUEL ÁNGEL BENNETTS FERNÁNDEZ

Tercer Vocal

DR. HÉCTOR OVIEDO GALDEANO

Secretario

DR. RAÚL CASTILLO PÉREZ

PRESIDENTE DEL COLEGIO

DR. MIGUEL TOLEDO VELÁZQUEZ

I. P. N. SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, el día **23** del mes de **Mayo** del año **2019**, el que suscribe **Salvador Ruiz León**, alumno del Programa de **Maestría en Ciencias en Ingeniería de Telecomunicaciones**, con número de registro **B171174**, adscrito a la **Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Zacatenco**, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del **Dr. Salvador Álvarez Ballesteros** y cede los derechos del trabajo titulado **"Desarrollo de equipos de red virtuales para la Red Institucional del Instituto Politécnico Nacional"**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección **sruizl1700@alumno.ipn.mx**, **sruizl1700@ipn.mx** o **salvarezorama@gmail.com.** Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Salvador Ruiz León

Nombre y firma

RESUMEN

Este documento presenta una propuesta de implementación de Virtualización de Funciones de Red (NFV) en la red de datos del Instituto Politécnico Nacional, tomando como referencia el resultado del análisis realizado en la red de datos de la ESIME Zacatenco, mismo que tiene como finalidad evaluar ciertos parámetros como el throughput, latencia, tasa de pérdidas y el jitter (variabilidad en el retardo) en un entorno virtualizado.

Para poder plantear la implementación de una tecnología nueva, es necesario conocer aspectos avanzados del funcionamiento de las redes de datos (IPv4/IPv6), conocimientos en *Cloud Computing*[1], conocimientos avanzados de sistemas operativos, conocimientos avanzados de programación, conocimientos avanzados en técnicas y tecnologías de virtualización, así como de los diferentes protocolos que operan para la intercomunicación de los equipos que forman parte de la red, y la necesidad de virtualizar determinadas funciones de red (NFV[2]) mediante componentes de *Cloud Computing*.

El paradigma del *Cloud Computing* tiene como objetivo la dotación de flexibilidad, elasticidad y alta disponibilidad a los servicios que se proporcionan a través de la nube. OpenStack[3], un software de código abierto, hace de la nube una infraestructura para el despliegue de servicios de red, en este caso, de virtualización de funciones de red. Ésta investigación se centra en el diagnóstico, el análisis de la problemática de la red, el desarrollo de una metodología para plantear una propuesta de solución para la problemática encontrada en la red institucional del IPN, así como del análisis de las posibles conFiguraciones, distribuciones y el software para una mejor integración y darle un mayor tiempo de respuesta a la red Institucional IPN.

Se ha realizado una amplia investigación bibliográfica sobre tecnologías de *Cloud Computing*, obteniendo como resultado múltiples posibilidades y aplicaciones, centrado en el lanzamiento de instancias, el cual es el pilar fundamental de esta tecnología. Gracias a eso se podrá acceder a hosts de forma remota y el aprovisionamiento dinámico de los

[1] Cómputo en la nube, por sus siglas en inglés
[2] Virtualización de funciones de red, por sus siglas en inglés
[3] Sistema operativo de código abierto para Cómputo en la nube

mismos. Se implementa Neutron[4], el cual nos permite hacer uso de su API[5] a través de plugins y agentes capaces de conectar/desconectar puertos, provisionar redes y subredes, etcétera; todo ello vinculado al proveedor que se vaya a usar en nuestra nube, como por ejemplo switches físicos o virtuales de Cisco, productos de NEC OpenFlow[6], Open vSwitch[7], Linux Bridging[8], Ryu Network Operating System o VMware NSX[9].

Mediante pruebas con equipos virtuales en escenarios similares a los de la red de datos ESIMEZ[10], se han aplicado técnicas de virtualización bare-metal[11], que permiten aprovechar mejor el hardware de los servidores estándares x86[12] actualmente en uso por la Institución.

Se presentan los resultados obtenidos en las pruebas y las conclusiones que determinan, el cumplimiento o incumplimiento de los objetivos que se plantearon en este proyecto, además de las recomendaciones pertinentes para comenzar con la virtualización de funciones de red en la red de datos de la ESIMEZ.

[4] Proveedor de NaaS integrado en OpenStack
[5] Interfaz de programación de aplicaciones, por sus siglas en inglés
[6] Solución SDN de NEC basada en el estándar OpenFlow
[7] Software open-source diseñado para ser utilizado como switch virtual en entornos virtualizados
[8] Es la forma más común de dar acceso a la red mediante puentes (bridges) a las máquinas virtuales bajo Linux
[9] Solución SDN de VmWare
[10] Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Zacatenco, perteneciente al Instituto Politécnico Nacional
[11] Virtualización que tiene acceso directo sobre los recursos de hardware
[12] Arquitectura de CPU que soporta 4092 MB de RAM como máximo

ABSTRACT

This thesis presents a proposal for the implementation of Network Functions Virtualization in the data network of the National Polytechnic Institute, using as a model the results of analysis of the data network of ESIME Zacatenco with the purpose of evaluating certain parameters such as throughput, latency, rate of losses and jitter (variability in the delay between each data packets) in a virtualized environment.

In order to propose an implementation of an innovative technology, it is necessary to have advanced skills about operation of IPv4/IPv6 and IP Next-generation networks, advanced knowlegde in Cloud Computing, Data Centers, Operating Systems, advanced programming skills, knowledge about techniques and technologies of virtualization, respectively. As well as of the different protocols that operate between each Networking devices and the need to virtualize certain network functions (NFV) through the Cloud Computing technology.

This research focuses on the diagnosis, the analysis of the issues of the network, development of a methodology to propose a solution for the issues found in data network of National Polytechnic Institute (IPN) as well as the analysis of posible conFigurations and software for giving a faster response time to the IPN data network users.

Through multiple tests with virtual machines on similar scenarios to those of the data network of ESIME-Zacatenco, bare-metal virtualization techniques have been applied, which make it posible to take better advantage of the hardware of the standard x86_64 physical servers currently running on the data network of ESIME-Zacatenco.

This document presents the results obtained in the tests and the conclusions which determine, the fulfillment or non-compliance of the objectives that were proposed for this Project, in addition to the pertinent recommendations to begin with the network function virtualization in the ESIME Zacatenco data network.

AGRADECIMIENTOS

Aprovecho este espacio para expresar mis agradecimientos para todos los que fueron parte activa y fundamental durante esta etapa de mi vida académica.

Al Dr. Salvador Álvarez Ballesteros, quien siempre estuvo al tanto de mi proyecto a lo largo de todo el proceso, estuvo a disposición para solucionar mis dudas y guiarme en todo este proceso, compartiendo sus experiencias profesionales, personales y académicas.

Al M.C. Miguel Ángel Bennetts Fernández, quien contribuyó enormemente en mi formación profesional en áreas específicas de las redes de telecomunicaciones, se mantuvo al tanto del desarrollo de este proyecto y siempre estuvo dispuesto a solucionar cualquier tipo de duda.

Un agradecimiento muy especial a mi familia. A mis padres que me han apoyado de forma incondicional. A mi padre David Ruiz Zúñiga que ha sido un ejemplo de disciplina férrea a seguir. A mi madre Leonila León Pérez que con su comprensión siempre me ha motivado para seguir avanzando con paso firme. A mis hermanos, Jairo y David, que me demuestran su apoyo en todo momento.

Agradezco al Departamento de Conectividad y Virtualización de la Dirección de Cómputo y Comunicaciones del Instituto Politécnico Nacional que me permitieron realizar la Estancia Industrial y me apoyaron para poder realizar una parte de este trabajo. En especial a la M.C. Griselda Jazmín Ramos Olivares quien me asesoró durante la estancia.

Al Instituto Politécnico Nacional y a la Sección de Estudios de Posgrado e Investigación de la Escuela Superior de Ingeniería Mecánica y Eléctrica, donde se me brindaron todas las herramientas y la ayuda posibles para formarme académicamente. Especialmente a los coordinadores de la Maestría en Telecomunicaciones; el M.C. Miguel Sánchez Meraz y el Dr. Marco Antonio Acevedo Mosqueda, así como al personal administrativo, quienes siempre estuvieron ahí para solucionar todas mis dudas.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CONTENIDO

INTRODUCCIÓN

La Red Institucional de Cómputo y Telecomunicaciones (en adelante, Red Politécnica) se ha caracterizado por estar a la vanguardia en tecnología de redes de computadoras, esto con el fin de abarcar todos los escenarios posibles y que siempre se cubran las necesidades de todos sus usuarios. Sin embargo, hoy en día no se dedica mucho tiempo a las redes y tampoco mucho presupuesto, al mismo tiempo que las demandas de servicios y/o aplicaciones de red de los usuarios se van incrementando.

El Instituto Politécnico Nacional se ha mantenido actualizado en cuanto a hardware de red, pero no se ha realizado un proceso riguroso de planeación y escalabilidad de la Red Politécnica. La tarea de mantener la Red Politécnica ha recaído en la Dirección de Cómputo y Comunicaciones (DCyC).

Actualmente, la Red Politécnica brinda diferentes servicios y aplicaciones de red como Internet, portal web, correo electrónico institucional, servidores de nombre de dominio, cursos en línea, telefonía IP, videoconferencias, el Sistema de Administración Escolar (SAES), entre otros.

La Red Politécnica cuenta con servidores capaces de soportar servicios de virtualización en varias de sus áreas, por lo que es posible agregar nuevos equipos de red virtualizados (VNF) sin tener que invertir en infraestructura física, ya que, según la DCyC, hasta el momento no se ha visto mucha iniciativa por parte de las escuelas para la mejora de sus servicios de red, todo lo anterior, aunado a restricciones en el presupuesto.

En este trabajo, el caso de estudio es la ESIME Zacatenco, donde hasta finales de diciembre 2018 no existía un diagrama de la topología física y lógica de la red de telecomunicaciones. En esta red se encontraron los siguientes problemas:

- Aumento en las demandas de servicios y/o aplicaciones de red.
- Crecimiento desmesurado y sin control.
- Fallas derivadas de problemas de diseño de la red en su última actualización (2016).
- Incertidumbre acerca de la ubicación de determinados nodos de red.

Los problemas mencionados anteriormente se deben a cambios constantes del personal administrativo de la Unidad de Informática (UDI), misma que se encarga de gestionar la red de datos de la ESIMEZ, y aunque tiene un encargado cuyo objetivo es brindar un servicio de calidad, el hecho de no darle continuidad a los procesos, la sustitución del personal y la falta de un modelo adecuado de gestión de red resultan perjudiciales para la Red Politécnica.

En el presente trabajo se realizará un análisis general de las capas núcleo y distribución de la Red Politécnica, para finalmente hacer un diagnóstico minucioso de la red de datos de la ESIMEZ para determinar si cuenta con los servidores físicos capaces de soportar virtualización de equipos de red, las características mínimas requeridas de los mismos, así como para decidir qué tipo de tecnologías de código abierto serán viables para cada escenario que se plantee. Finalmente, será posible proponer una metodología para la implementación de equipos de red virtualizados dentro de la Red Politécnica con la información recabada durante el diagnóstico.

PLANTEAMIENTO DEL PROBLEMA

La expansión constante de las redes de área local (LAN), así como de las redes de área extensa (WAN) y el crecimiento que se viene dando en el desarrollo de las redes de telecomunicaciones ha motivado la búsqueda de nuevas alternativas y paradigmas para adaptarse a la rápida evolución de las redes de computadoras.

El Instituto Politécnico Nacional se ha destacado por ser una de las Instituciones de Educación Superior y de Posgrado que se encuentran a la vanguardia en tecnología de redes (tanto tecnología basada en pares de cobre, como en fibra óptica e incluso tecnologías inalámbricas). Sin embargo, dado el crecimiento sin control de la red Institucional del Instituto Politécnico Nacional así como la aparición de tecnologías emergentes, gradualmente la red de datos del Instituto Politécnico Nacional se ha visto reducida en sus capacidades. Debido a múltiples factores, por mencionar algunos factores como: falta de equipo de Networking, falta de servidores DHCP, equipos que ya están obsoletos y que nunca fueron explotados (se mantuvieron en pre-producción), e incluso temas burocráticos como la falta de presupuesto, todo lo anteriormente mencionado resulta en una red congestionada, limitada, con escasa escalabilidad, agilidad y flexibilidad.

Dado que hoy en día es cada vez mayor la cantidad de información que hay que recibir, procesar y enviar de manera rápida y confiable, dentro de la red de datos del Instituto Politécnico Nacional se hace necesario realizar una propuesta para mejorar la conectividad de red, y en general, la experiencia del usuario final. Para ello una de las soluciones que comienza a tener auge dentro del entorno de investigación y desarrollo, es la Virtualización de Funciones de Red (NFV), que a su vez ofrece Funciones de Red Virtualizadas (VNF), tales como: enrutadores virtuales, balanceadores de carga virtuales, servidores web virtuales, switches virtuales, firewalls virtuales, etcéteraétera. Todo lo anterior está basado en tecnología de *Cloud Computing* de código abierto (Licencia OpenSource) para ofrecer servicios como: IaaS (Infraestructura como Servicio), SaaS (Software como Servicio), NaaS (Red como Servicio), entre otros; que en su conjunto conforman el nuevo paradigma en Ingeniería de Redes llamado SDN (Redes Definidas por Software).

El propósito de esta propuesta es brindar una alternativa de conectividad para estudiantes, personal docente y administrativo de una de las Unidades del Instituto Politécnico Nacional y que la metodología que se obtenga sea de utilidad para replicarla en las distintas Unidades del Instituto Politécnico Nacional. En este caso, la propuesta alcanza únicamente a la Escuela Superior de Ingeniería Mecánica y Eléctrica; Unidad Zacatenco (ESIMEZ), de manera que puedan disponer de una mejor experiencia de usuario (Ux) dentro de la red LAN/WAN de ESIMEZ. Una consideración importante es que la propuesta del uso de tecnologías emergentes/experimentales como la aquí planteada (NFV) no pretende sustituir a las redes tradicionales, sino complementarlas, y a su vez, disminuir la carga de trabajo del administrador de redes, al tener un solo panel de administración para todos sus equipos virtuales (virtual appliances). En este sentido, se busca proporcionar flexibilidad, elasticidad, escalabilidad y agilidad a la red de datos de ESIMEZ, mismas que actualmente no están disponibles mediante la red cableada (par de cobre o fibra óptica) o inalámbrica (2.4 y 5 GHz, respectivamente) ya existente en ESIMEZ.

En nuestro país, en este año 2019, encontramos universidades privadas que ya cuentan con tecnología de *Cloud Computing* y ofreciendo servicios, principalmente SaaS, y el Instituto Politécnico Nacional, en su calidad de icono de la educación superior mexicana, no puede y no debe quedarse atrás en el diseño e implementación de nuevas tecnologías basadas en virtualización bare-metal. Para ello es necesario motivar a los estudiantes a que se involucren en el desarrollo de proyectos relacionados con ellas, en este caso específicamente, con el paradigma de SDN que está en constante desarrollo, y de forma más puntual con OpenStack, que cuenta con repositorios Git[13] para clonar el código fuente y diseñar aplicaciones y/o servicios propietarios del Instituto Politécnico Nacional.

Por lo anterior, se propone este trabajo de tesis, con el fin de mejorar los servicios de red, y aumentar la capacidad de Networking que ofrece el Instituto Politécnico Nacional.

[13] Es un software de control de versiones, su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

JUSTIFICACIÓN

El Instituto Politécnico Nacional, desde sus inicios se ha destacado por mantener actualizada su infraestructura de red. Sin embargo, actualmente no es tan fácil agregar nuevos dispositivos de red especializados, debido a que no hay direcciones IP qué asignarles y al no poder obtener nuevas direcciones IP se limita el crecimiento de la red. Además, la red de datos de la ESIMEZ presenta un congestionamiento cuando el número de usuarios conectados a la misma aumenta. Por lo tanto, para que la red de datos de la ESIMEZ pueda mejorar su desempeño, además de brindar servicios y/o aplicaciones de red de alta disponibilidad, se plantea el uso de recursos ya existentes, como servidores x86 estándar para virtualizar funciones de red específicas (switching, routing, firewall) en determinados nodos de red donde existe congestionamiento.

La virtualización de funciones de red (Network Functions Virtualization, NFV por sus siglas en inglés), es una iniciativa para virtualizar funciones de red que se llevaron a cabo con anterioridad por hardware propietario y dedicado. El concepto está siendo desarrollado por el Grupo de Especificaciones de la Industria ETSI (SIG) para Virtualización de las Funciones de Red y fue presentada por primera vez por un grupo de proveedores de servicios de red en el Congreso Mundial de SDN y OpenFlow realizado en octubre de 2012.

El objetivo de NFV es disminuir la cantidad de hardware propietario que se necesita para poner en marcha y operar servicios de red – las funciones de red que anteriormente eran hechas por ruteadores, firewalls, balanceadores de carga y otros equipos – que ahora estarían alojados en máquinas virtuales.

Si la propuesta de virtualización de funciones de red (NFV) se desarrolla e implementa con éxito para cada caso específico de la ESIMEZ, se revolucionará la forma en que se construyen, gestionan y utilizan las redes para crear servicios.

ALCANCES

El presente trabajo es de tipo exploratorio y a su vez, de tipo descriptivo. Se dice que es exploratorio porque el estudio de la Red Institucional del Instituto Politécnico Nacional en su conjunto ha sido pobremente abordado, únicamente se han limitado al estudio de unidades específicas.

Por otra parte, se dice que es de tipo descriptivo, porque dentro de los objetivos particulares se plantea la necesidad de establecer la topología física de la red completa que constituye la llamada Red Institucional del Instituto Politécnico Nacional.

Lo anterior, ayudará a dar una idea de cómo se encuentra constituida la Red Institucional del Instituto Politécnico Nacional para finalmente estudiar el caso específico de la Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Zacatenco (ESIMEZ). Es decir, se irá de lo general a lo específico. No se puede ir en sentido contrario.

Como se ha mencionado, este trabajo no pretende implementar NFV en la red del Instituto Politécnico Nacional que se encuentra a lo largo de todo el territorio nacional.

En resumen, se estudiará la Red Institucional del Instituto Politécnico Nacional, con el objetivo de determinar su topología física y direccionamiento lógico. Partiendo de ello, se inicia el proyecto que únicamente abarca la red de datos de la Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Zacatenco.

OBJETIVOS

General

Proponer la virtualización de funciones de red (NFV) en la red de datos de la ESIME Zacatenco

Específicos

- Analizar y definir la topología física actual la Red Institucional del Instituto Politécnico Nacional
- Analizar y definir la topología lógica actual de la Red Institucional del Instituto Politécnico Nacional
- Conocer las principales características del Cloud Computing
- Conocer las principales capas: capa de infraestructura (IaaS) y plataforma (PaaS)
- Conocer diversos usos de tecnologías de Cloud Computing
- Conocer algunos proveedores de Cloud Computing de IaaS o PaaS
- Conocer el proyecto de software libre OpenStack para proporcionar IaaS
- Conocer las principales características de OpenStack
- Definir una metodología para la implementación de la iniciativa NFV en la capa de distribución de la red de la ESIME Zacatenco (ESIMEZ)
- Determinar el tipo de tecnología, así como requerimientos mínimos de hardware en servidores físicos para implementar NFV en ESIMEZ
- Proponer escenarios de prueba de NFV
- Evaluar de rendimiento de los escenarios de prueba
- Proponer la adquisición de servidores físicos para mejorar las capacidades de respuesta de la red ante el aumento en la demanda de servicios y/o aplicaciones de red.

CAPÍTULO I: MARCO CONTEXTUAL

Cloud Computing

El término Cloud Computing se refiere a una nueva concepción tecnológica y modelo de negocio en el que se prestan servicios de almacenamiento, acceso y uso de recursos informáticos radicados en la red. Toda la gestión del hardware es realizada por el proveedor de Cloud Computing asegurando cierta disponibilidad y potencia bajo unos niveles de servicio determinado (SLAs[14]).

De esta forma el cliente o usuario puede desentenderse de los recursos físicos y enfocarse en la actividad central del negocio (desarrollo web, red privada, etcéteraétera). Por otra parte, un usuario final no necesita un dispositivo de altas prestaciones puesto que las tareas de mayor desempeño pueden ser trasladadas a la computación en la nube.

Según el organismo NIST (National Institute of Standards and Technology) del departamento de comercio de los Estados Unidos, para que un servicio pueda considerarse Cloud debe cumplir cinco características básicas:

- Servicio bajo demanda: Un usuario debe poder provisionarse automáticamente de recursos Cloud, como almacenamiento o capacidad de cómputo, sin requerir la interacción humana del proveedor de servicio.
- Acceso amplio desde la red: Todos los servicios deben ser accesibles a través de estándares y plataformas comunes (por ejemplo, un servicio web para computadoras, dispositivos móviles, etcéteraétera)
- Puesta en común de recursos: Los recursos Cloud deben ser puestos en común para servir a varios consumidores a través de un modelo multi-tenancy[15]. De esta manera, los recursos físicos y virtuales se asignarán o reasignarán dinámicamente para satisfacer la demanda de consumo.

[14] Es un contrato que describe el nivel de servicio que un cliente espera de su proveedor

[15] Es un principio de arquitectura de software en la cual una sola instancia de la aplicación se ejecuta en el servidor, pero sirviendo a múltiples clientes

- Rápida elasticidad: la capacidad de los recursos contratados debe ser elástica. Es decir, los recursos asignados deben crecer o decrecer bajo demanda de forma automática o con la mayor celeridad posible.
- Capacidad de medición del servicio: los sistemas Cloud deben ser controlados y optimizados a través de métricas que permitan un nivel de abstracción apropiado al tipo de servicio. Los recursos deberán ser monitorizados y bien presentados para ofrecer transparencia al cliente final.

Tipos de sistemas Cloud

Una implementación de Cloud Computing puede clasificarse siguiendo dos criterios explicados a continuación:

- Según el modelo de despliegue se tienen cuatro clasificaciones:
    - Nube pública: En las nubes públicas los servicios ofrecidos se encuentran en servidores externos al usuario o empresa que desea hacer uso de ellos. Normalmente el proveedor Cloud cobra según el tiempo y capacidad de los recursos. Tiene como ventaja un aumento de recursos sin la necesidad de instalar y mantener las máquinas en local. De esta manera se reduce la inversión inicial y permite gran escalabilidad a empresas que necesiten el uso de estos recursos digitales. Como inconvenientes se encuentra la dependencia de terceras personas, donde habitualmente se produce el conocido "lock-in[16]" por parte del proveedor. Para evitar dicha cautividad es recomendable elegir plataformas Open Source que faciliten la migración de nuestro software o servicio.
    - Nube privada: en la nube privada los servicios se encuentran dentro de las instalaciones del usuario o empresa que hará uso de ellos. Normalmente las nubes privadas se implementan para la obtención de infraestructura (IaaS) como máquinas virtuales, contenedores, almacenamiento, redes, etcétera. En este caso será más sencillo integrar los servicios Cloud con sistemas

[16] Es una situación en la que un cliente que utiliza determinado servicio no puede realizar la transición al mismo servicio, pero de otro proveedor

propietarios y la seguridad y carga de trabajo correrá a cargo de la empresa que despliegue dicha nube. El principal inconveniente se encuentra en la inversión inicial del hardware y de operarios capacitados para mantener la infraestructura física y lógica de la instalación.

- Nube híbrida: la nube híbrida combina las características de los dos casos anteriores. Una empresa puede mantener el control de sus aplicaciones principales y aprovechar la flexibilidad de una nube pública para una tarea o momento determinado. Las nubes híbridas ofrecen la promesa de la escalabilidad provisionada externamente y a demanda, permitiendo cubrir momentos o tareas críticas del modelo de negocio (por ejemplo, el SAES-IPN con alta demanda en periodos de inscripción). Este tipo de nube comienza a imponerse sobre las anteriores debido a su versatilidad.
- Nube comunitaria: son nubes implementadas para el uso exclusivo de una comunidad, normalmente organizaciones que comparten asuntos (por ejemplo, la red de policía, red de hospitales, etcéteraétera)

- Según el modelo de servicio tenemos ocho clasificaciones:
  - Software as a Service (SaaS): El software como Servicio permite al cliente final usar una aplicación que corre sobre una infraestructura cloud. La aplicación debe ser fácilmente accesible ya sea a través de la web o a través de un cliente de escritorio. El usuario no puede controlar la infraestructura subyacente como los sistemas operativos, redes o almacenamiento. Ejemplos de SaaS son OwnCloud, WordPress, Gmail, Dropbox y Salesforce.
  - Platform as a Service (Paas): La Plataforma como Servicio permite a un desarrollador conFigurar su propio sistema de programación a través de lenguajes de programación, librerías, APIs, servicios y otras herramientas. El desarrollador no puede controlar la infraestructura subyacente como los sistemas operativos, redes o almacenamiento. Ejemplos de PaaS son OpenShift, CloudFoundry, IBM Bluemix y Google App Engine.

- Infrastructure as a Service (IaaS): La Infraestructura como Servicio permite a un operador de red gestionar el poder de cómputo, almacenamiento, conexionado de redes y otros recursos fundamentales para ofrecer un servicio a un cliente final. El operador no gestiona la infraestructura subyacente del Cloud, sino que gestiona la infraestructura virtual superpuesta (máquinas virtuales, contenedores, redes virtuales, etcéteraétera). Por ejemplo, mientras el operador puede ejecutar sus aplicaciones y sistemas operativos, el proveedor se encargará de la replicación y los backups del sistema en general. Ejemplos de IaaS son OpenStack, CloudStack, IBM SoftLayer y AWS.
- Metal as a Service (MaaS): Desde 2012, Canonical introdujo el concepto MaaS, el cual pretende ser un gran apoyo a la hora de querer trabajar con servidores físicos (metal) y de igual manera contar con los beneficios de cualquier servicio en la nube. Como el propio Canonical lo menciona: "MaaS trae el lenguaje de la nube a los servidores físicos". Esto hace que sea fácil conFigurar el hardware en el que se requiere desplegar servicios que necesitan escalar tanto de forma vertical como horizontal de forma dinámica.
- DataBase as a Service (DBaaS): Los DBaaS ofrecen acceso tanto a bases de datos clásicas, es decir, con el lenguaje SQL, como a bases de datos *sin esquemas* o NoSQL como Redis, CouchDB o MongoDB. También hay modelos *freemium*[17] tales como Amazon DynamoDB o ClearDB, que provee servicio MySQL.
- Desktop as a Service (DaaS): DaaS ofrece una infraestructura virtual (VDI) alojada en un proveedor de servicios en la nube y generalmente se basa en un modelo de costo por suscripción mensual. Utiliza una arquitectura multi-tenancy, lo que significa que una sola instancia de una aplicación se sirve a múltiples usuarios, conocidos como "tenants".

[17] Modelo de negocio que ofrece servicios básicos gratuitos, mientras que cobra cierta cantidad monetaria por servicios más avanzados

El proveedor de servicios es responsable de administrar la nube y la infraestructura subyacente, y el nivel de servicio se puede variar según las necesidades de la compañía. El resultado de esta infraestructura es que los usuarios pueden acceder a sus datos y aplicaciones desde casi cualquier dispositivo, en cualquier lugar.

- Hardware as a Service (HaaS): HaaS tradicionalmente significa que una empresa usa hardware de un proveedor externo. El hardware "alquilado" pueden ser servidores, computadoras, escáneres, fotocopiadoras, etcétera. El equipo se instala en la empresa, pero sigue siendo propiedad del proveedor. Éste último también suele ser el responsable del mantenimiento y la posible sustitución del hardware cuando esté desactualizado. Es lo contrario a la IaaS (Infraestructura como Servicio). Aporta flexibilidad, garantía y elasticidad para proponer soluciones cuando y donde se necesite.

Para este proyecto se ha elegido la plataforma de Cloud Computing OpenStack por su formato Open Source y el gran apoyo recibido por el sector privado.

Virtualización

Es la técnica empleada sobre las características físicas de algunos recursos de computación, permitiendo así solucionar problemas de rendimiento sin necesidad de afectar el trabajo del usuario final. Esto implica hacer que un recurso físico, como un servidor o un dispositivo de almacenamiento, aparezca como si fueran varios o un único recurso lógico a la vez.

Por ejemplo, la virtualización de un sistema operativo es el uso de una aplicación de software que permitirá que un mismo sistema operativo manipule varias imágenes de sistemas operativos a la vez.

Virtualización es un término que se refiere a la abstracción de los recursos de una computadora llamada Hypervisor o VMM (Monitor de máquina virtual) donde se crea una de la abstracción entre el hardware de la máquina física (host) y el sistema operativo de la máquina virtual.

El VMM maneja los recursos de las máquinas físicas (designadas por el computador central) de una manera que el usuario pueda crear varias máquinas virtuales presentando a cada una de ellas una interfaz del hardware que sea compatible con el sistema operativo elegido.

Esta capa de software (VMM) maneja, gestiona y actúa con los cuatro recursos principales de una computadora (CPU, Memoria, Red, Almacenamiento, etcéteraétera) para que así pueda repartir dinámicamente dichos recursos entre todas las máquinas virtuales en el computador central.

Típicamente muchas máquinas virtuales son simuladas en un computador central para que el sistema operativo "huésped" funcione, la simulación debe ser lo suficientemente robusta (dependiendo del tipo de virtualización).

La virtualización es una tecnología que permite que en el sistema operativo (SO) anfitrión que se encuentra en el computador se pueda implementar y hacer funcionar otro SO, utilizando una herramienta de virtualización como pueden ser VMware, QEMU, KVM, etcéteraétera. De una manera "totalmente" independiente, utilizando como recursos compartidos el hardware del computador.

En la Figura 1 se muestra de forma gráfica lo descrito en los párrafos anteriores.

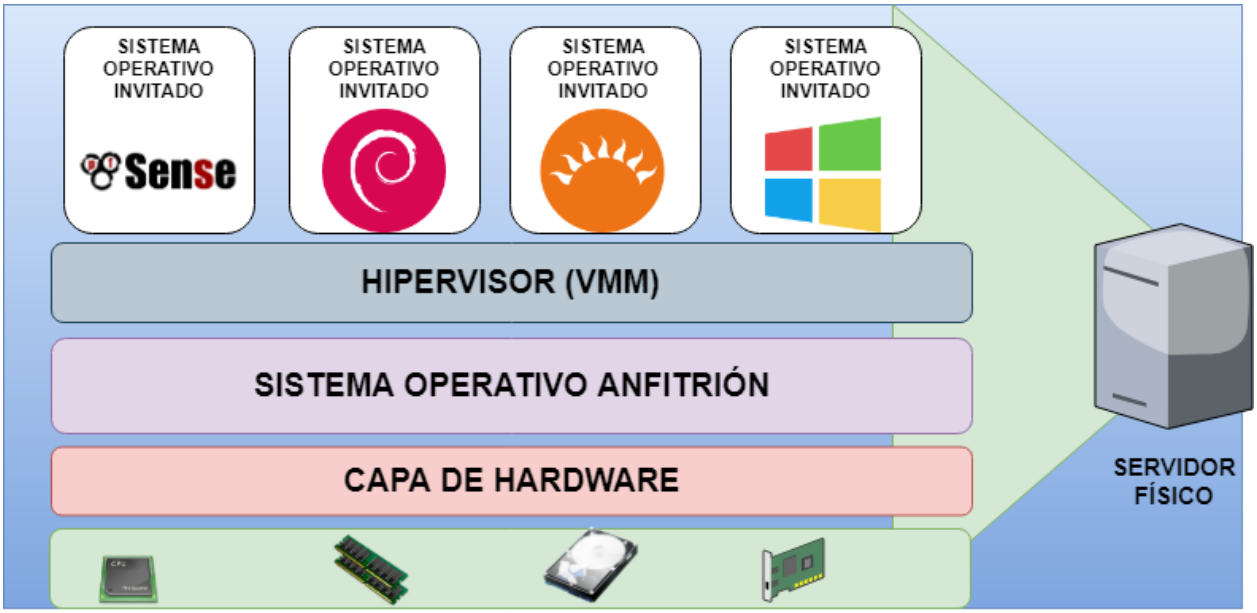


Figura 1. Estructura de una arquitectura virtualizada (elaboración propia)

Se puede decir además que la virtualización permite coordinar en un computador físico el uso de los recursos para que varios sistemas operativos puedan funcionar al mismo tiempo de forma independiente y compartiendo los mismos recursos.

Para lograr esto, son necesarios los siguientes componentes:

- Un servidor estándar x86_64
- De acuerdo al tipo de virtualización, la segunda capa es un sistema operativo o un hipervisor que va instalado como anfitrión o sistema principal. Ésta es la parte de la capa que coordina los recursos del sistema como memoria, CPU, archivos, impresora, tarjeta de red.
- Uno o más sistemas operativos que son los huéspedes (vRouter, vSwitch, Ryu, etcétera)

En la Figura se observa que en la parte inferior se encuentra el hardware o la parte física del servidor (CPU, Memoria EEC[18], Tarjetas de red y Disco duro o SSD). Sobre esa capa física va una capa con el sistema operativo, que es el que coordina el acceso a las partes físicas del servidor, esta capa se denomina hipervisor.

Dentro del hipervisor que se encuentre en un sistema operativo anfitrión van los otros sistemas operativos huéspedes, donde se crea una capa virtual que permitirá a los huéspedes utilizar los recursos físicos que tiene el servidor.

Tipos de virtualización

La virtualización tiene múltiples usos y de acuerdo a éstos se puede determinar el tipo de virtualización que sea la más apropiada.

Las más comunes son la virtualización de servidores, virtualización de clientes y virtualización de almacenamiento de datos o conocido también como virtualización de discos duros. En el presente trabajo se explora un nuevo campo: la virtualización de funciones de red o NFV.

[18] Código de corrección de errores, por sus siglas en inglés

Virtualización de clientes

Son aquellos hosts de escritorio, portátiles o terminales que se conectan a uno o varios servidores para hacer el trabajo que necesitan, es lo que conoce como arquitectura cliente-servidor.

Se recomienda la virtualización ya que, si una determinada organización posee 50 computadores en diferentes departamentos y al departamento encargado de su administración le toca actualizar un programa o simplemente revisar y borrar archivos no deseados o programas que son instalados sin autorización al no tener restricciones en el sistema, la opción sería ir de computadora en computadora y hacer las correcciones correspondientes.

En la Figura 2, se muestra este tipo de virtualización.

Figura 2. Estructura de la virtualización de clientes (elaboración propia)

Pero si en la organización antes mencionada usan virtualización de clientes, lo único que se debe hacer es acceder al servidor o los servidores donde están instalados los clientes y hacer las actualizaciones.

Es decir, no hay necesidad de acudir físicamente a la ubicación de cada uno de los clientes, sino realizar el proceso de actualización o corrección desde un solo punto, ahorrando tiempo y recursos.

Emulación de hardware

Esta técnica se suele llamar virtualización completa de hardware, y puede ser implementado utilizando un “tipo 1” o “tipo 2” de hipervisor. Linux (Tipo 1 hipervisor se ejecuta directamente sobre el hardware, el tipo 2 se ejecuta en otros sistemas operativos, como Windows o Linux).

En la Figura 3, se muestra el diagrama esquemático de este tipo de virtualización.

Cada máquina virtual es particularmente útil en el desarrollo de sistemas operativos, cuando el nuevo código de experimentación se puede ejecutar al mismo tiempo con el código más antiguo, más estable, etcétera, de forma independiente en una máquina virtual.

Esta forma de virtualizar es donde el software de virtualización genera o crea una capa de software que representa el software.

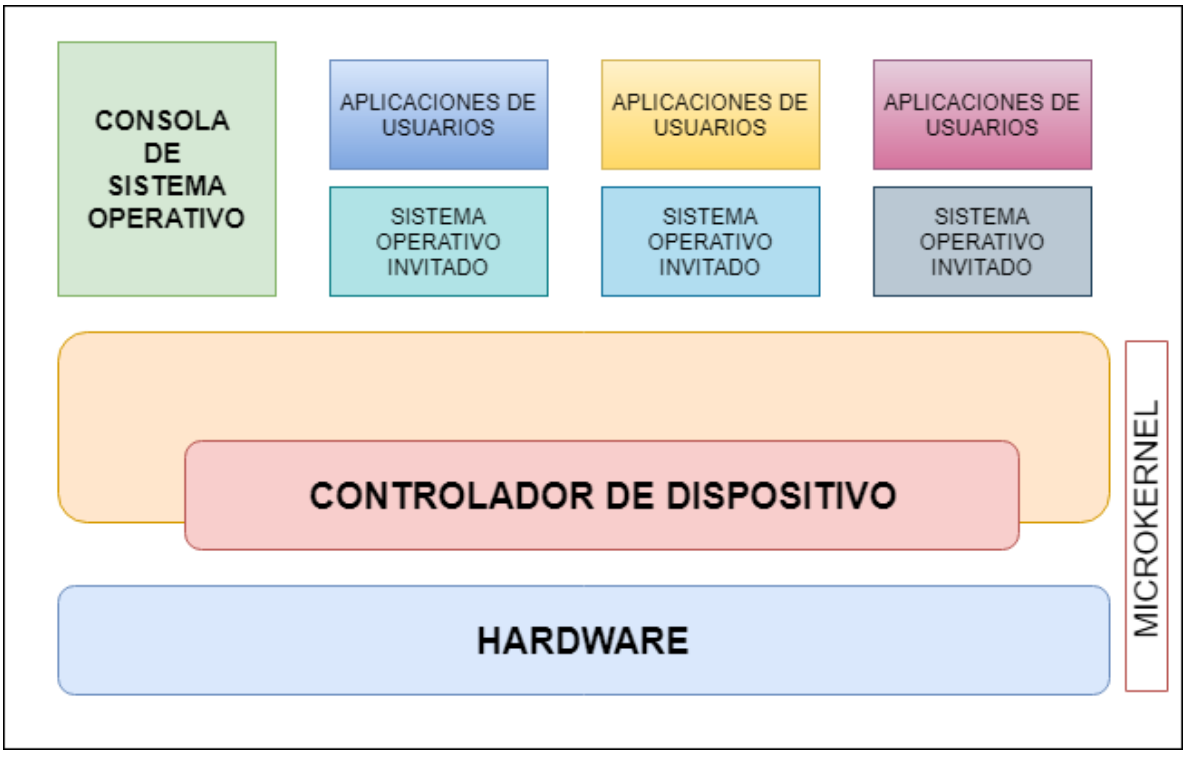


Figura 3. Estructura de la emulación de hardware (elaboración propia)

Para realizar esta virtualización, lo que se hace es tener un sistema operativo instalado en el cliente, luego se instala el software de emulación de hardware que una vez instalado y configurado queda listo para instalar otro sistema operativo invitado. Esto se hace a través del software de virtualización sin instalarse directamente en el computador anfitrión, quien configura el contenedor o lo que conocemos como máquina virtual.

Después de esto la instalación del nuevo sistema operativo invitado se hace igual como si se estuviese haciendo en un computador nuevo.

Virtualización de servidores

Es el tipo de virtualización más usada, y es por las ventajas que genera el virtualizar un servidor: ahorro de energía, de espacio, facilidad para administrar menos servidores físicos, entre otras.

La virtualización de servidores es, como su nombre lo indica, la virtualización de un servidor, y servidores son aquellas computadoras principales a las que los clientes u otras computadoras se conectan para obtener archivos, compartir impresoras o en general manejar todos los recursos de la red. En la Figura 4 se muestra cómo es una virtualización de servidores.

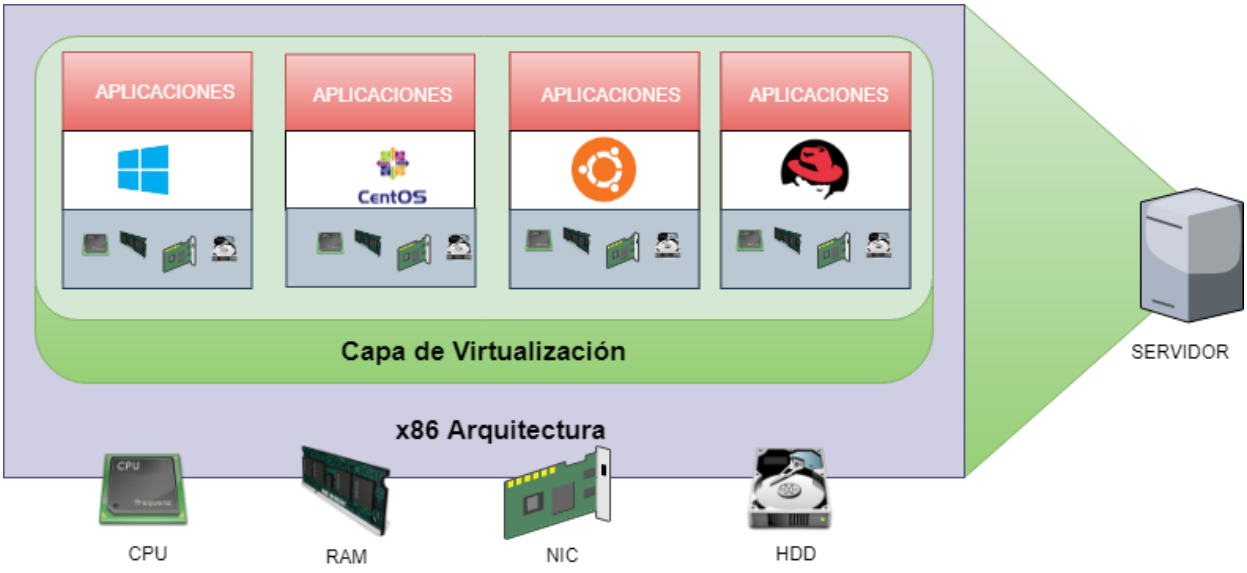


Figura 4. Estructura de la virtualización de servidores (elaboración propia)

Paravirtualización

Esta forma de virtualizar, es en realidad una forma de compartir los recursos por tiempos cortos o a quien los necesite, asignando procesador, memoria o tarjeta de red al anfitrión que lo solicita e intercalando el uso de estos entre los anfitriones. Este sistema utiliza memoria compartida que puede ser usada por dos programas diferentes, de esta forma envía y recibe información de los invitados para el hipervisor, alcanzando así buenos niveles de rendimiento.

Otra de las ventajas es la poca carga que le da al procesador al no tener que poseer una capa completa de virtualización que se encarga de administrar los recursos y virtualizarlos, es decir, no son necesarios procesadores con tecnología Intel-VT o AMD-V. Además, los invitados no tienen que limitarse a los accesorios de hardware que sean soportados por el hipervisor, pues el invitado actúa directamente con la parte física haciendo posible el manejo de todos los accesorios que opera el sistema operativo anfitrión en el invitado.

La desventaja es que para poder hacer esto, el hipervisor necesita modificar los sistemas operativos que se instalan como invitados, es decir, toma el código del sistema operativo y le agrega algunas líneas, siendo así los sistemas operativos como Linux, BSD o cualquier sistema operativo de código libre que puedan ser usados.

En la Figura 5 se muestra el diagrama a bloques de la paravirtualización.

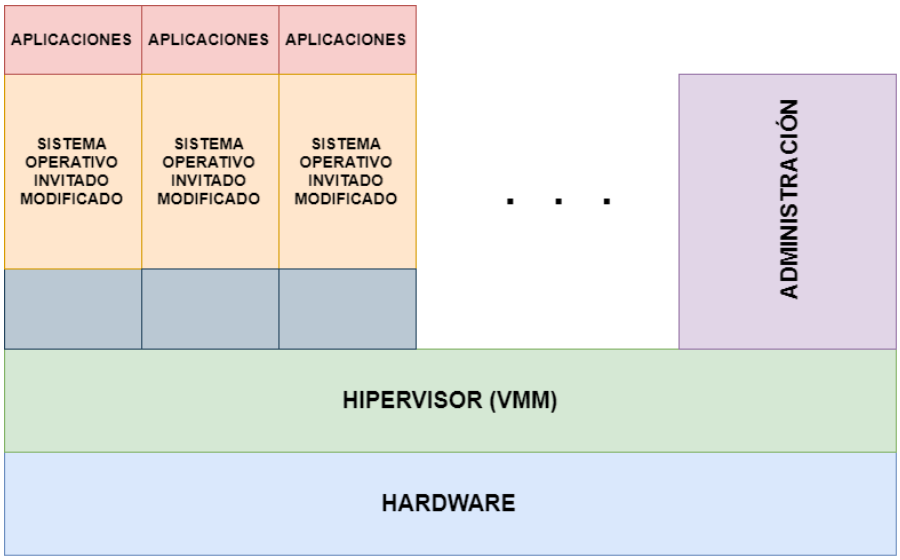


Figura 5. Estructura de la paravirtualización (elaboración propia)

Virtualización a nivel de sistema operativo

Éste es otro tipo de virtualización. En ésta no se utiliza hardware virtualizado, sino que se ejecuta una única instancia del sistema operativo (kernel). Los distintos procesos pertenecientes a cada servidor virtual se ejecutan aislados del resto.

En la Figura 6 se muestra la estructura de las capas de que conforman un sistema de virtualización a nivel de sistema operativo.

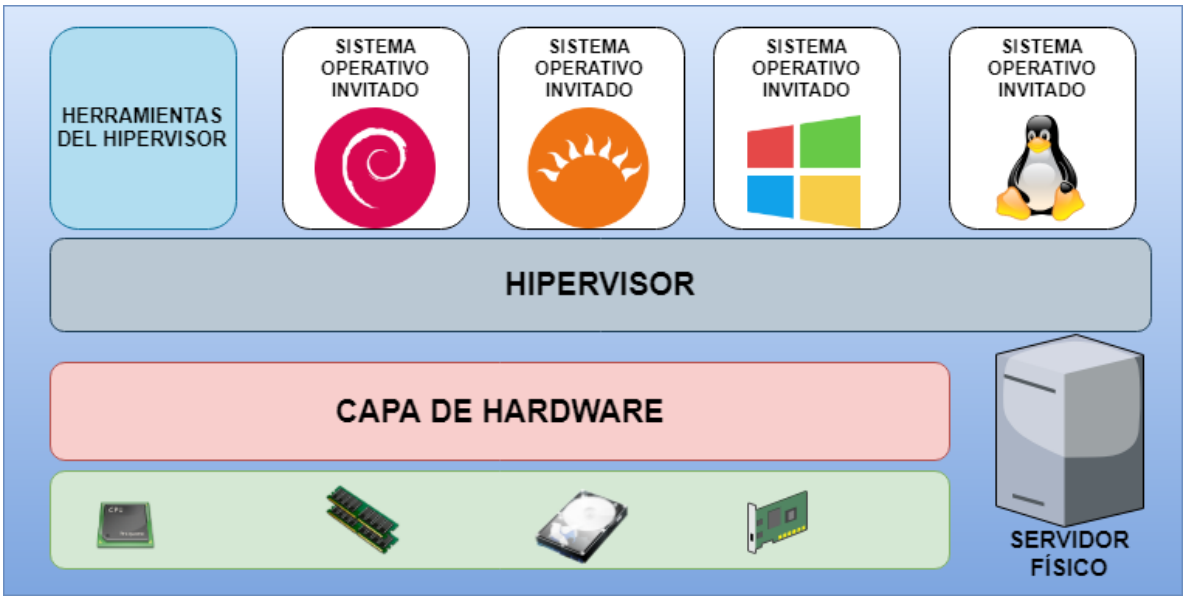


Figura 6. Estructura de la virtualización a nivel de sistema operativo (elaboración propia)

La ventaja de este enfoque es la separación de los procesos de usuario prácticamente sin pérdida en el rendimiento, pero al compartir todos los servidores virtuales con el mismo kernel no se puede obtener el resto de ventajas de la virtualización.

Virtualización de aplicaciones

La virtualización de aplicaciones también conocida como “portabilidad de aplicaciones” o la virtualización de servicios de aplicaciones es la facilidad de ejecutar el software desde un servidor remoto en lugar de en el ordenador del usuario. La virtualización de aplicaciones tiene la capacidad de implementar software sin modificar el equipo o realizar cambios en el sistema operativo local.

En la Figura 7 se muestra la estructura de la virtualización de aplicaciones.

Las ventajas de la virtualización de aplicaciones incluyen:

- Ahorro en hardware
- Los ahorros de costos de software y las licencias del sistema operativo
- Posibilidad de ejecutar varias versiones de un programa de aplicación al mismo tiempo en un solo equipo
- Facilidad de gestión de aplicaciones, la mejora y la migración
- Capacidad para aprovechar los recursos sin repercusiones negativas sobre los usuarios
- Flexibilidad en la adquisición de recursos de hardware (servidores estándares x86)
- Mejora la fiabilidad del sistema y la escalabilidad.

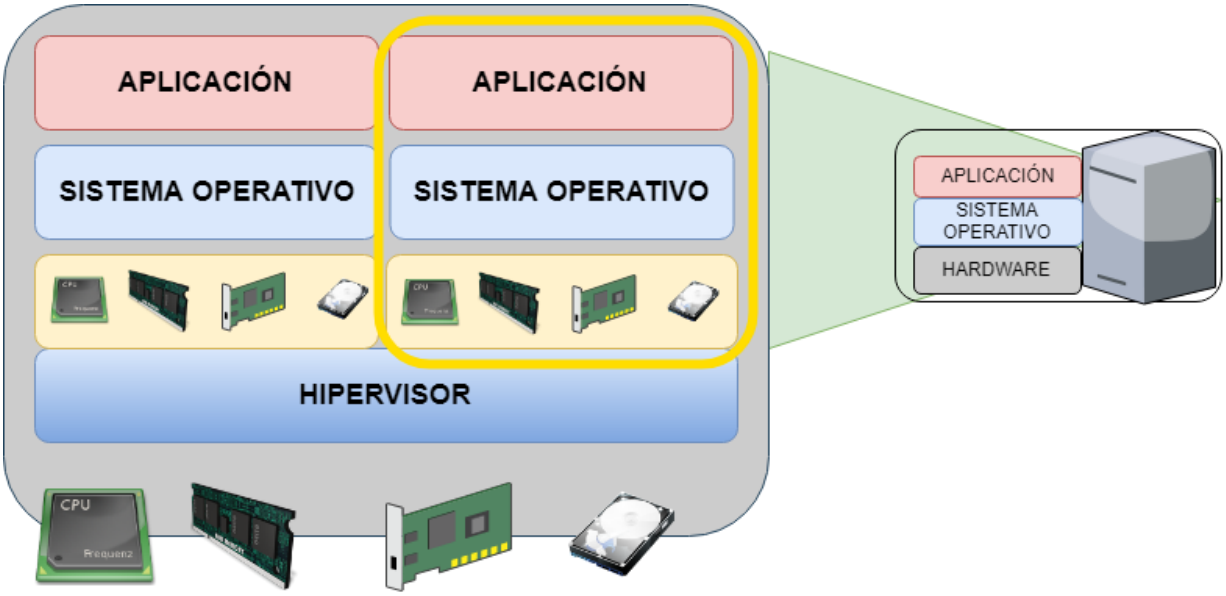


Figura 7. Estructura de la virtualización de aplicaciones (elaboración propia)

Virtualización de red

Este tipo de virtualización tiene por objeto mejorar la productividad, eficiencia y satisfacción en el trabajo del administrador mediante la realización de muchas de estas tareas de forma automática, disimulando así la verdadera complejidad de la red.

Archivos, imágenes, programas y carpetas pueden ser centralizadas en un sitio físico, medios de almacenamiento tales como discos duros (actualmente desplazados por SSDs). El espacio de almacenamiento se puede compartir o repartir entre los servidores.

Además, tiene por objeto optimizar la velocidad de la red, la fiabilidad, flexibilidad, escalabilidad y seguridad. Es un método de combinación de recursos disponibles en una red mediante el fraccionamiento de la disposición de ancho de banda en el canal, cada uno de ellos independiente de los demás, y cada uno de los cuales se puede asignar (o reasignar) a un servidor en particular o un dispositivo en tiempo real. Cada canal es independiente garantizado. Cada abonado tiene acceso compartido a todos los recursos en la red desde un servidor.

Es la segmentación o partición lógica de una red física, para usar los recursos de la red. La virtualización de red es lograda instalando software y servicios para gestionar el almacenamiento compartido, los ciclos de computación y las aplicaciones. La virtualización de red que se muestra en la Figura 8, trata a todos los servicios y servicios en la red como un único grupo de recursos que pueden ser accedidos sin considerar sus componentes físicos.

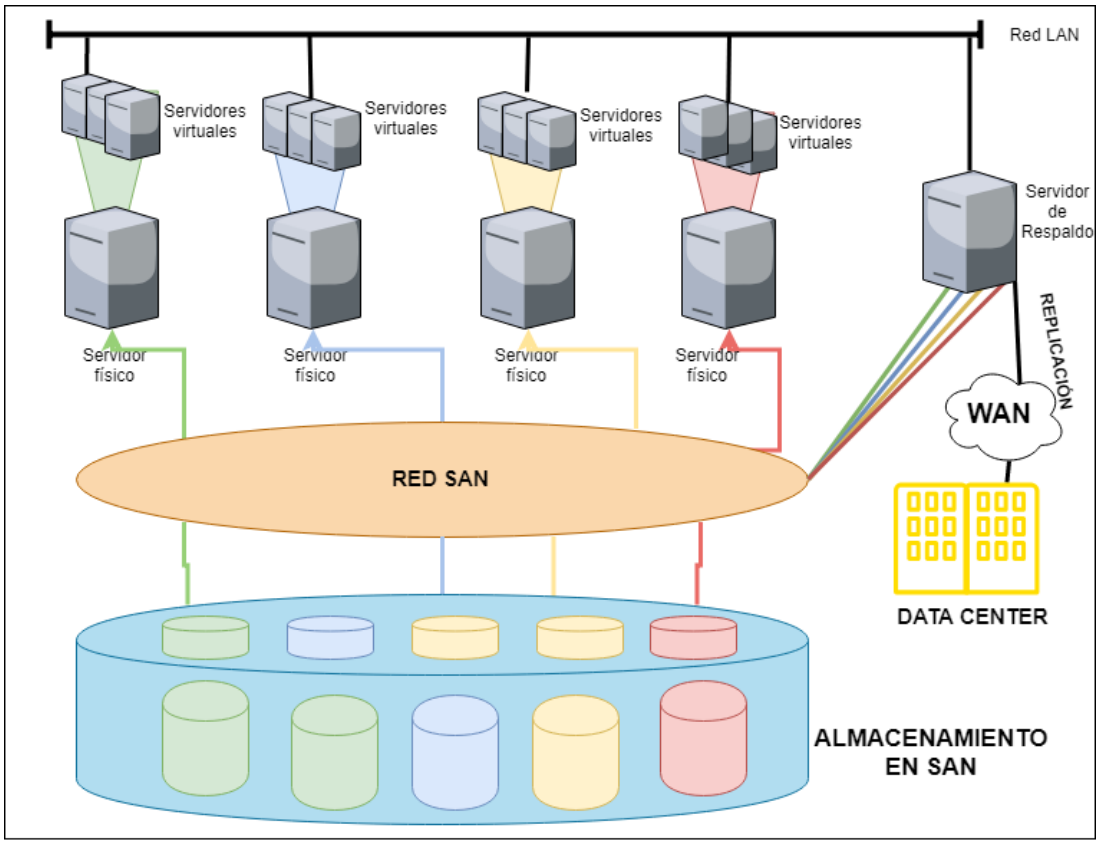


Figura 8. Estructura de la virtualización de red

Hipervisor

También llamado monitor de máquina virtual (VMM). Es el núcleo central de algunas de las tecnologías de virtualización de hardware más populares y eficaces. Son aplicaciones que presentan a los sistemas operativos virtualizados (sistemas invitados) una plataforma operativa virtual (hardware virtual), a la vez que ocultan a dicho sistema operativo virtualizado las características físicas reales del equipo sobre el que operan. También son los encargados de monitorizar la ejecución de los sistemas operativos invitados.

Con el uso de hipervisores es posible conseguir que múltiples sistemas operativos compitan por el acceso simultáneo a los recursos hardware de una máquina virtual de manera eficaz y sin conflictos.

Existen tres tipos de hipervisores en el mercado:

1. Hipervisores de tipo 1 (también llamados nativos, un-hosted o bare-metal): en ellos el hipervisor se ejecuta directamente sobre el hardware físico; el hipervisor se carga antes que ninguno de los sistemas operativos invitados, y todos los accesos directos a hardware son controlados por él.

   Aunque ésta es la aproximación clásica y más antigua de la virtualización por hardware, actualmente las soluciones más potentes de la mayoría de fabricantes usan este enfoque. Es el caso de Microsoft Hyper-V, Citrix XEN Server y VMware ESX-Server.
2. Hipervisores de tipo 2 (también llamados hosted): en ellos el hipervisor se ejecuta en el contexto de un sistema operativo completo, que se carga antes que el hipervisor. Las máquinas virtuales se ejecutan en un tercer nivel, por encima del hipervisor. Son típicos de escenarios de virtualización orientada a la ejecución multiplataforma de software, como en el caso de CLR de .NET o de las máquinas virtuales de Java.
3. Hipervisores híbridos: en este modelo tanto el sistema operativo anfitrión como el hipervisor interactúan directamente con el hardware físico. Las máquinas virtuales se ejecutan en un tercer nivel con respecto al hardware, por encima del hipervisor,

pero también interactúan directamente con el sistema operativo anfitrión. Es la aproximación usada en Parallels, VirtualBox, VMware Server.

En algunas clasificaciones es posible que se incluyan los hipervisores de tipo híbrido dentro de los de tipo 2.

Condicionantes de hardware para la virtualización

Al momento de virtualizar, se debe tener claro que necesitamos que la memoria RAM del equipo desde donde vamos a virtualizar, sea capaz de contener al sistema anfitrión (para VMM tipo 2) y a las MMVV[19]. También se debe disponer de suficiente espacio en disco (HDD o SSD) y además que el procesador (CPU) se haya construido pensando en la virtualización.

Hoy en día, todos los procesadores traen instrucciones para ayudar a la virtualización. Esta característica puede que tenga que activarse en el BIOS/UEFI de la computadora. concretamente este paso es necesario para realizar virtualización de tipo 1 o de tipo 2 de MMVV con SSOO[20] de 64 bits sobre arquitecturas x86. En los microprocesadores de Intel se denomina Intel VT-x y en los de AMD su nombre es AMD-V.

Intel VT-x

La tecnología Intel® de virtualización (Intel® VT) está formada por componentes tecnológicos que soportan la virtualización de plataformas basadas en procesadores Intel®, lo que permite la ejecución de múltiples sistemas operativos y aplicaciones en particiones independientes.

Existen dos tipos de virtualización:

- Tecnología de virtualización Intel® (Intel VT): Mejora la flexibilidad y robustez fundamentales de las soluciones de virtualización basadas en software tradicionales al acelerar las funciones clave de la plataforma virtualizada, incluyendo:
    - Agilizar la transferencia del control de la plataforma entre los sistemas operativos huésped (OSs) y el administrador de máquinas virtuales (VMM)/hipervisor.

[19] Acrónimo para hacer referencia a “Máquinas virtuales”

[20] Acrónimo para hacer referencia a “Sistemas Operativos”

    - Habilitación de VMM para asignar dispositivos de e/s exclusivos a OSs de cliente
    - Optimización de la red para la virtualización con aceleración basada en adaptadores.
- Tecnología Intel® de virtualización (Intel® VT) para e/s dirigida (Intel® VT-d): Proporciona un rendimiento, seguridad y flexibilidad adicionales al proporcionar al VMM las siguientes funcionalidades:
    - Asignación de dispositivos de e/s
    - Reasignación de DMA (Acceso directo a memoria, por sus siglas en inglés)
    - Reasignación de interrupciones
    - Fiabilidad.

AMD-V®

La tecnología Virtualization de AMD® es un conjunto de características exclusivas en el chip que permite a los clientes que utilizan AMD PRO, ejecutar múltiples sistemas operativos y aplicaciones en un solo equipo. La tecnología AMD-V® ayuda a ejecutar de forma eficiente aplicaciones fundamentales de la línea de negocios.

AMD-V™ admite las siguientes funcionalidades:

- Virtualización del lado del cliente
- Infraestructura de escritorio virtual (VDI)
- PC blade, streaming del sistema operativo y servicios de terminal.

CentOS

CentOs (Community ENTerprise Operating System) Linux proporciona una plataforma informática, es decir, un sistema operativo (SO), de código libre y abierto a cualquier persona que desee utilizarlo. Es una distribución mantenida por la comunidad y derivada de los paquetes fuentes liberados al público por Red Hat para “Red Hat Enterprise Linux” (RHEL). De esa manera, CentOS Linux está enfocado en ser operacionalmente compatible con RHEL. El Proyecto CentOS, principalmente, cambia paquetes para eliminar las marcas comerciales y trabajos artísticos de Red Hat.

La redistribución de CentOS Linux es totalmente libre y no hay que pagar para poder usarla, además desde la versión de CentOS 5, cada versión de CentOS es mantenida por 10 años, por medio de actualizaciones de seguridad. La duración de los intervalos de mantenimiento ha variado a lo largo del tiempo con relación a los paquetes fuente liberados.

Una nueva versión de CentOS es liberada aproximadamente cada dos años y cada versión de CentOS es periódicamente actualizada, normalmente cada seis meses para incorporar nuevo hardware. Esto da como resultado un entorno Linux seguro, con un bajo mantenimiento, confiable, predecible, reproducible y fácil de instalar y utilizar.

Por ejemplo, la última versión de CentOS, la versión 7, recibirá actualizaciones de seguridad hasta el 30 de Junio de 2024.

CentOS es creado o desarrollado por un pequeño conjunto de desarrolladores que ha ido creciendo con el tiempo, a la vez que los desarrolladores centrales están apoyados por un gran número de usuarios que trabajan activamente, entre ellos, los usuarios de empresa, los administradores de red, administradores del sistema, gerentes, principales contribuyentes de Linux de todas partes del mundo.

Tabla 1. Comparativa de características de servidores CentOS y Ubuntu.

Características	CentOS	Ubuntu
Package Command	RPM /YUM	APT-GET
Variantes	Basadas en Red Hat Enterprise Linux (RHEL)	Basado en Debian
Porcentaje de uso por sitios web a nivel mundial	20.4%	34.8%
Ciclo de lanzamiento	Periodos largos	Periodos cortos
Seguridad	Seguro fuera de la plataforma (OOTB)	Menos seguro (fuera de la plataforma) comparado con CentOS

Requisitos de instalación

Para tener éxito en la instalación de CentOS es necesaria una serie de requisitos mínimos del sistema. En principio se tiene que diferenciar entre dos tipos de entorno de instalación:

- Con entorno de escritorio, se refiere a que ya existe un conjunto principal de programas que ofrece al usuario del aparato una sencilla y cómoda interacción con su equipo. En este caso se necesita:
    - Memoria RAM: 2 GB, como mínimo
    - Espacio en el Disco Duro o SSD (20 GB, como mínimo)
    - Procesador: Intel x86_64 (64 bits) o AMD 64 (AMD-V).
- Sin entorno de escritorio, es decir, cuando no existe el conjunto de software, los requisitos para la instalación son los siguientes:
    - Memoria RAM: 64 MB, como mínimo
    - Espacio en Disco Duro o SSD: 1024 MB, como mínimo
    - Procesador: Intel x86_64 o AMD 64.

Características

Funciones de red

1. Dispone de Tramas gigantes, para poder aumentar el tamaño de los marcos de red y como consecuencia se aumenta el rendimiento de la red.
2. Funciona para etiquetado VLAN, es decir, se pueden crear redes lógicas independientes dentro de una misma red física.
3. Permite realizar “Migración en vivo”, para mover una máquina virtual desde un host a otro host.
4. Incluye la “Segmentación de TCP y la suma de comprobación”, para dividir los datos recibidos de la capa de segmentos, adjuntando un encabezado a cada segmento donde se indican los puertos de envío y recepción, de esta manera se garantiza que los datos se han transferido sin errores.

Almacenamiento

1. Dispone del “cambio de tamaño de VHDX” para poder cambiar el tamaño fijo de un archivo al adjuntarlo a una máquina virtual.
2. Dispone de “Canales de Fibra Virtual” para que las máquinas virtuales puedan reconocer un dispositivo de canal de fibra y montarlo de forma natural.
3. Se puede realizar una “copia de seguridad de máquina en vivo” cuando hay identificadores de archivos abiertos durante una operación de copia de seguridad.

Memoria

1. ConFiguración de “gap MMIO”, permite conFigurar la ubicación de la brecha de E/S asignada de memoria (MMIO). Este espacio se utiliza para dividir la memoria física disponible entre el sistema operativo y la infraestructura software del dispositivo.
2. Agregar “memoria dinámica activa”, con esto se puede aumentar la cantidad de memoria que está disponible para una máquina virtual en vivo.
3. Dinámica: el aumento de memoria, permite acumular y desactivar la cantidad de memoria asignada a una máquina virtual en vivo dinámicamente.

Varios

1. Se puede habilitar “la infraestructura de clave/valor par (KVP)” en máquinas virtuales para realizar la lectura y escritura de las operaciones de datos personalizados.
2. “No es enmascarable”, se utiliza para saber cuándo un sistema operativo ha dejado de responder debido a errores de la aplicación.
3. “Compatibilidad de núcleo PAE”, permitiendo a un kernel de 32 bits acceder a un espacio de dirección física mayor que 4 GB.
4. “Copia de archivos de host a invitado”, permite copiar en las máquinas virtuales los archivos copiados desde el equipo host físico sin utilizar el adaptador de red.

Estado del arte de entornos NFV

Debido a la importancia del proceso de evaluación de NFV existen varias organizaciones trabajando en su estandarización. Las dos fundamentales que han desarrollado un relevante trabajo son: el ETSI NFV ISG y el Grupo de Trabajo de Ingeniería de Internet (IETF). Se muestran sus logotipos en la Figura 9.

Figura 9. Principales organizaciones de estandarización para NFV

También existen otras entidades que realizan actividades muy importantes como son el Centro Europeo Avanzado de Pruebas de Red (EANTC), la Nueva Agencia IP (NIA) y la Plataforma Abierta para NFV (OPNFV), además de soluciones industriales como Spirent e Ixia.

Organizaciones de estandarización

ETSI NFV ISG

El ETSI NFV completó la fase 1 de su trabajo a finales de 2014 con la publicación de once especificaciones. Uno de los documentos más relevantes fue el referido a las metodologías de desempeño de NFV para evaluar las VNFs.

El objetivo es unificar las pruebas y referencias de varias VNFs heterogéneas bajo una metodología común. En adición a lo anterior, durante la fase 2 realizada durante el periodo 2015-2016 se dio continuación a trabajo de la fase 1.

También se definieron cuatro tipos de cargas de trabajo para facilitar el análisis de desempeño: cargas de trabajo del plano de datos, plano de control, procesamiento de señal y almacenamiento.

En esta fase se publican dos importantes documentos que actualmente se encuentran en desarrollo, uno que aborda las pruebas de evaluación pre-despliegue de los bloques

funcionales de la NFV en un ambiente de laboratorio. El otro, cubre el análisis de metodologías de evaluación de interoperabilidad NFV.

IETF

Dentro del IETF existe un grupo llamado Grupo de Trabajo de Métricas de Referencias (BMWG), el cual propone las métricas de desempeño necesarias a ser medidas en un ambiente de laboratorio que se acercarán a las observadas en un entorno implementado.

El BMWG se encuentra examinando el desempeño y estado de la red a través de varias métricas, que pueden ser usadas para validar diversas aplicaciones, redes y servicios. Las principales métricas que han sido identificadas son: Throughput, tiempos de respuesta de las aplicaciones, número de flujos actuales soportados, latencia unidireccional de paquetes, entre otras.

El grupo también ha propuesto metodologías para medir el funcionamiento de varios dispositivos interconectados, aunque estas pruebas se basen en entornos físicos. Las principales metodologías pueden también aplicarse a entornos virtualizados para el análisis del desempeño de VNFs.

Estas métricas y metodologías se recogen en una serie de publicaciones de la IETF llamadas Petición de Comentarios (RFC), estas son: RFC 1944 *Benchmarking Methodology for Network Interconnect Devices*, RFC 2889 *Benchmarking Methodology for LAN Switching Devices* y RFC 3511 *Benchmarking Methology for Firewall Performance*.

EANTC

Es el laboratorio independiente de pruebas públicamente activo en evaluaciones de NFV. Tiene vasta experiencia en evaluaciones de aplicaciones de casos de uso de NFV y creación de reportes técnicos independientes con verdadero contenido validado. Es miembro activo del ETSI NFV ISG. Define estándares de pruebas para desempeño e interoperabilidad. Contribuye con los proyectos de pruebas de OPNFV. Entre las actividades más sobresalientes que realiza están las pruebas de conceptos (PoC) a SP[21]; verificación de las

[21] Proveedores de Servicios, por sus siglas en inglés

funciones, desempeño e interoperabilidad NFV y pruebas de integración para despliegues tempranos.

NIA

Es una iniciativa independiente que provee información, educación, análisis, servicios comunitarios y pruebas para soportar y acelerar el desarrollo de redes IP avanzadas virtualizadas de economía global. Como parte de los servicios ofrecidos por esta organización, están las pruebas de interoperabilidad entre elementos NFV. El objetivo de todas estas pruebas es proveer a la industria de las telecomunicaciones de información de soluciones NFV interoperables, para de acelerar la adopción de NFV. Actualmente, se encuentra involucrada en evaluaciones de los siguientes bloques funcionales de la NFV:

- Evaluación de interoperabilidad MANO
- Evaluación de interoperabilidad NFV-VNF
- Evaluación de interoperabilidad VNFM.

NIA y sus miembros continuarán en la definición de evaluaciones de interoperabilidad para realizar evaluaciones de las cadenas de funciones de servicio en las redes de acceso tradicionales y Redes Definidas por Software (SDN). Esto permite monitorizar la red, así como manejar el tráfico, la alta disponibilidad y auto recuperación ante fallas.

OPNFV

Es de suma importancia el rol que desempeña OPNFV en el paradigma de esta tecnología. OPNFV es un proyecto de código abierto creado y presentado por la Fundación Linux. Está compuesto por varios SP y vendedores.

Su objetivo es establecer una plataforma integrada de referencia de código abierto para impulsar la evolución de NFV y asegurar la consistencia, funcionalidad e interoperabilidad entre múltiples componentes de código abierto. El proyecto ya ha publicado dos versiones de software, la primera en junio de 2015 llamada *Arno*, la cual provee una estructura inicial de los componentes de la NFVI y del VIM de la arquitectura del ETSI.

Esta se centra en el desarrollo y por tanto, puede ser usada para explorar los despliegues de NFV, desarrollar aplicaciones NFV, evaluar el desempeño NFV y pruebas de casos de uso.

En particular, *Arno* tiene capacidades de integración, despliegue y evaluación de componentes de otros proyectos como Ceph, KVM, OpenDaylight, OpenStack y OpenvSwitch.

En adición, usuarios finales y desarrolladores pueden desplegar sus propias VNFs en Arno para probar sus funcionalidades y desempeño en varios escenarios de tráfico y casos de uso.

La otra versión software es *Brahmaputra* la cual incluye casi el doble de proyectos que tenía *Arno* y trae mejoras como: gestión de capa 3 de Redes Privadas Virtuales (VPN), soporte inicial para IPv6, mejor detección de fallos y recuperación y habilidades mejoradas para las pruebas de infraestructura.

NFV

Principios básicos

A medida que la infraestructura de TI, incluidas las funciones de almacenamiento y computación, continúa su inevitable marcha hacia la virtualización, las redes constituyen la próxima área lógica donde los operadores de red pueden virtualizar para seguir el ritmo de los avances tecnológicos. Debido a que las aplicaciones se sobrecargan a las redes con mayor ancho de banda, flexibilidad y velocidad, la noción de la construcción excesiva de redes a fin de contar con la capacidad necesaria para las cargas máximas de tráfico resulta difícil de sostener y afrontar.

Ya no es aceptable comprar hardware específico para una aplicación, diseñarlo y configurarlo para la aplicación en cuestión y esperar que funcionará por 10 años o más. Lo que se necesita es mayor agilidad y control de la red y de sus funciones centrales. Esta virtualización implica programabilidad a través del control del software. Conforme a las recomendaciones del Instituto Europeo de Normas de Telecomunicaciones (ETSI) y el Grupo de Estándares de la Industria (ISG) para NFV, la NFV emergió como el medio de virtualización de las funciones de red, idealmente a través de SDN.

Actualmente, muchas funciones de red se implementan como dispositivos específicos y personalizados. Estos dispositivos cuentan con hardware, firmware y chipsets personalizados que ayudan a acelerar el rendimiento. Con NFV y SDN, la misma funcionalidad se implementa cada vez más en software que en hardware.

Gracias a la Ley de Moore, que se centra en el concepto de disminución del retorno, las funciones de red que antes solo eran posibles a través de hardware y software altamente personalizados ahora se pueden implementar totalmente en el software. Este hecho básico modifica drásticamente el panorama de las redes para los proveedores de servicios y otros operadores de red.

NFV les ofrece a los proveedores de servicios y operadores la oportunidad de reducir los costos de infraestructura de red mientras aceleran la configuración y el despliegue de nuevos servicios de red.

Este nuevo retorno de servicios de red más flexible y basado en software les permite a los proveedores de servicios y operadores acelerar rápidamente nuevos servicios de red según las necesidades de situaciones y clientes específicos, lo que reduce el retorno de semanas o meses a días e incluso minutos.

Esta agilidad ofrece una importante ventaja competitiva, ya que permite a los operadores de red perseguir nuevos mercados y oportunidades económicamente inviables con el hardware y el software de red tradicionales con mayor rapidez.

NFV puede reducir o eliminar el hardware propietario específico de una aplicación de la infraestructura de red. En lugar de requerir que un operador de red despliegue nuevos componentes de hardware para cada nuevo servicio o aplicación, el modelo de NFV proporciona la misma funcionalidad utilizando dispositivos virtuales que se ejecutan en servidores x86 convencionales.

Conceptualmente, un operador puede desplegar un dispositivo virtual bajo demanda y actualizar la funcionalidad a través de actualizaciones de software a lo largo del tiempo.

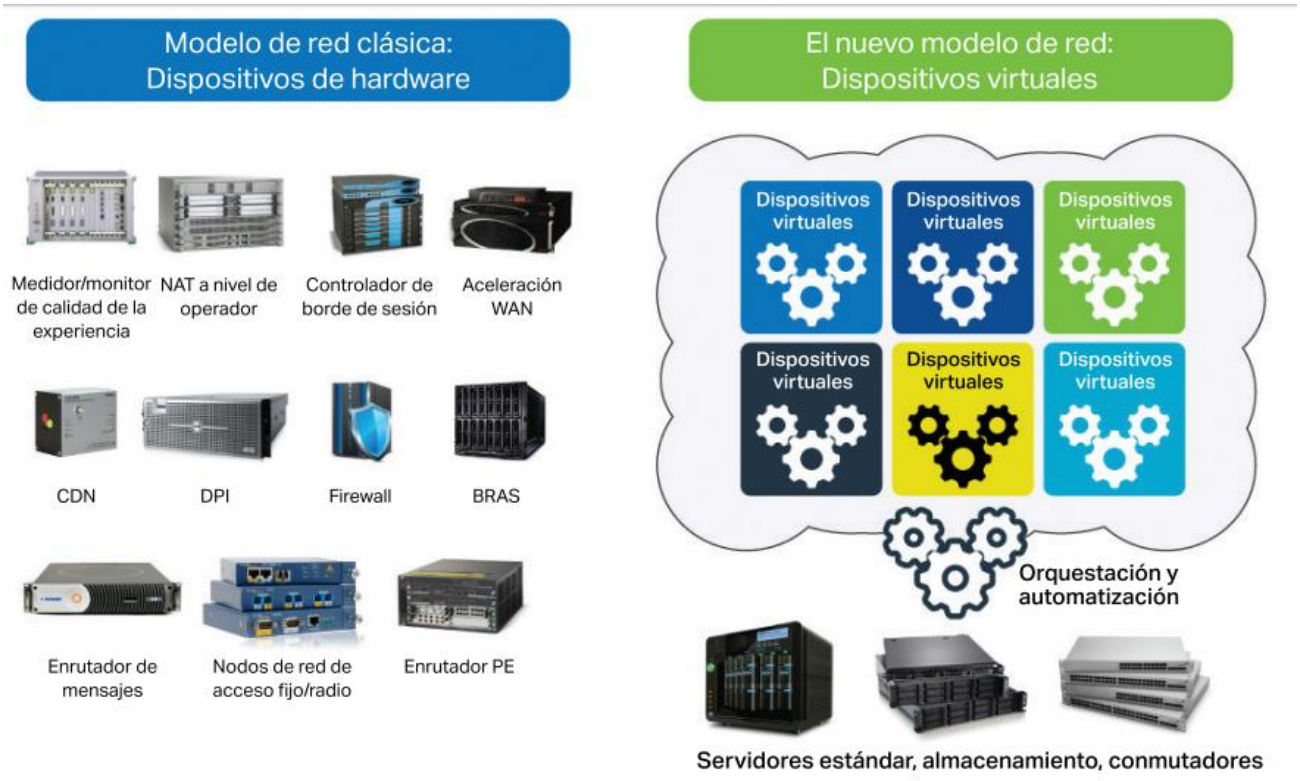


Figura 10. Cambio de paradigma: Red Clásica vs Red virtualizada

En la Figura 10 se muestra el cambio de paradigmas en la Ingeniería de Redes: se pasa de hardware propietario y de uso específico a NaaS ejecutándose en servidores estándar de la industria.

Los firewalls, los enrutadores PE (Provider-Edge), la inspección profunda de paquetes (DPI) y el cifrado son algunos ejemplos en los que en la actualidad un operador típicamente despliega hardware independiente, a menudo de diferentes proveedores. Cada elemento de hardware se enfrenta a su propio ciclo de obsolescencia, requiere sus propias actividades de certificación y necesita ser administrado, a menudo de formas únicas. La administración de estos múltiples proveedores y funciones es complicada, y la escalabilidad y el soporte son esencialmente demasiado costosos. Además, para la mayoría de los proveedores de servicios el rápido lanzamiento de nuevas aplicaciones es difícil o imposible.

En el modelo de NFV, los dispositivos virtuales que residen en servidores físicos o virtuales, reemplazan los dispositivos de red basados en hardware dedicados. Las funciones de operación y administración se manejan a través de un sistema de orquestación que

coordina los dispositivos virtuales en funcionamiento de una red. Al igual que las máquinas virtuales, los dispositivos virtuales se seleccionan en función de las necesidades del cliente final y se despliegan según sea necesario cuando y donde se requieran.

La escalabilidad para adaptarse a los cambios de las necesidades del cliente se basa en la carga del software en los servidores apropiados. Además, cuando el dispositivo virtual ya no es necesario, el espacio en el servidor se puede liberar para usar por otra aplicación. Este uso compartido de recursos ayuda a los proveedores de servicios a disminuir los costos generales.

NFV simplifica la arquitectura de la red física mientras que mejora la capacidad de aumento y adaptación al cambio tecnológico. Además, promete no solo mejorar el aumento y la agilidad de las operaciones de un proveedor de servicios, sino también reducir los costos.

Los gastos operativos también se reducen a través de menores requerimientos de espacio físico y energía, y en menor medida, a través de requisitos de pruebas de calificación de proveedores e interoperabilidad más cortas.

En lugar de los costos de capital de hardware, se paga por las funciones virtuales basadas en software que el cliente requiere.

Con NFV, los operadores de red pueden abordar la solución a las necesidades de TI como una visita al supermercado: comprar lo que se necesita, cuando se necesita, a una selección de diferentes proveedores.

Los proveedores de servicios pueden adquirir y desplegar rápidamente solo la cantidad de recursos de red requeridos según las demandas de los clientes. Han quedado atrás los días en los que un proveedor de servicios tenía que aprovisionarse de una gran cantidad de equipos de red de un solo propósito a cuenta de la creciente demanda de los clientes, que podía o no materializarse.

Menos riesgo y gasto se traduce en más retorno y ganancias para el proveedor de servicios. Luego de que el operador de red despliegue un dispositivo de hardware genérico en las

instalaciones del cliente, el cliente puede comprar funciones de red virtuales (VNF) que crearán los dispositivos virtuales necesarios para las operaciones de red.

El ecosistema de VNF incluye una amplia variedad de componentes de múltiples proveedores. Algunos ejemplos de VNF incluyen la funcionalidad vRouter, la seguridad y el cifrado, el equilibrio de carga, las cajas decodificadoras virtuales (vSTB), la optimización de la WAN y el monitoreo de desempeño.

Una vez seleccionadas, las VNF se controlan y operan a través de lo que ETSI definió como la función Gestión y Orquestación (MANO).

La función MANO incluye la distribución de VNF en todos los hosts, la orquestación de la funcionalidad de las VNF y la gestión del ciclo de vida de las VNF.

La Figura 11 muestra una selección de proveedores que ofrecen VNF y dispositivos virtuales.

Figura 11. Proveedores de NFV en la Industria

En la Figura 12 se muestran las empresas que participan activamente en el desarrollo de OpenStack y garantizan la compatibilidad de sus IOS (Internetwork Operating System) con el mismo, con miras hacia SDN.

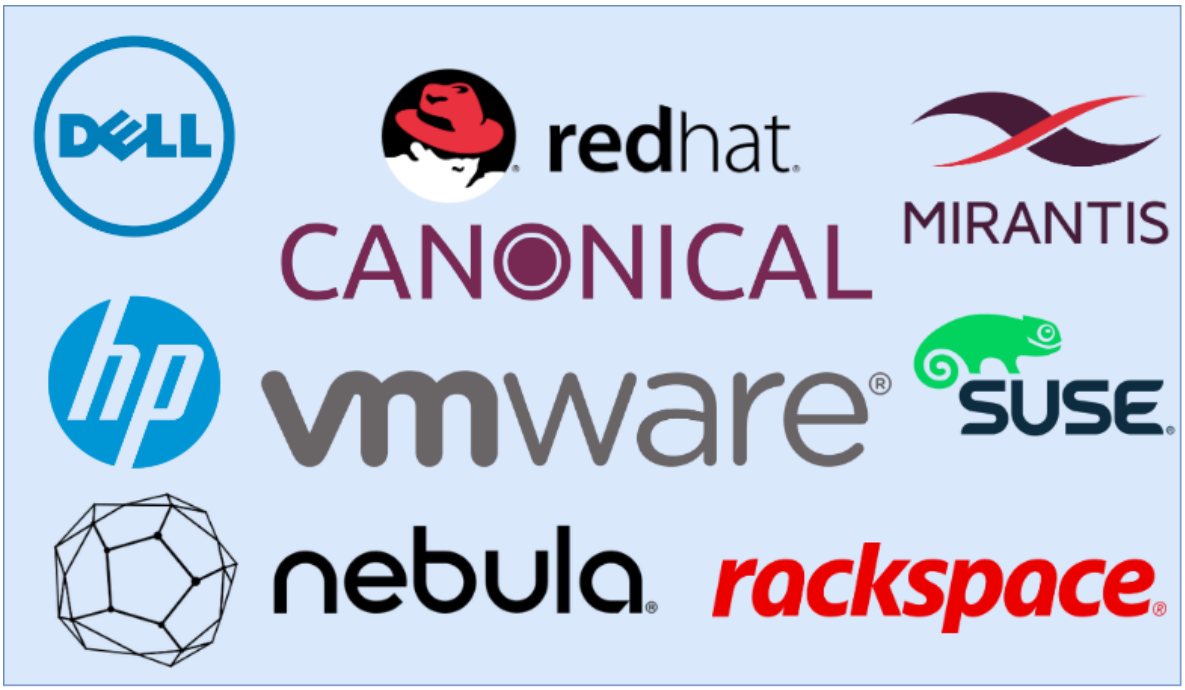


Figura 12. Empresas que participan activamente en el desarrollo de OpenStack (elaboración propia)

La función MANO de NFV se compone de tres áreas funcionales que realizan todas las tareas relacionadas con el ciclo de vida de una VNF: orquestador de NFV (NFVO), administrador de VNF (VNFM) y administrador de infraestructura virtualizado (VIM).

El NFVO aporta nuevos servicios de red y las VNF proporcionan la gestión de recursos globales, la validación y la autorización de solicitudes de recursos de infraestructura de la VNF.

El VNFM controla instancias de VNF específicas, coordinando las solicitudes de recursos de infraestructura entre la instancia de la VNF y los sistemas de gestión de elementos de red relacionados. El VIM controla y administra la infraestructura de NFV, que incluye recursos de computación, almacenamiento y red.

Los organismos de normalización y consorcios del sector adaptaron y dirigieron las vías de comercialización específicas de SDN y NFV. La organización de normalización de Open Networking Foundation (ONF) desarrolló las especificaciones de SDN, mientras que el ISG para NFV del ETSI desarrolla y adapta las especificaciones de NFV.

Tanto las especificaciones de SDN como las de NFV se consideran marcos para los proveedores de redes que desean que los dispositivos y el software interactúen con otros proveedores que cumplen con los estándares, mientras que los proveedores individuales pueden apartarse de las especificaciones según sea necesario para diferenciar los productos, ofrecer capacidades más allá de las especificaciones o asegurar la compatibilidad con el equipo de red heredado.

El ISG para NFV del ETSI incluye proveedores de software de redes, proveedores de servicios, operadores de red, organizaciones tecnológicas nacionales de numerosos países, instituciones académicas y de investigación y grupos de usuarios relacionados.

La filosofía de gestión del ISG para NFV del ETSI se basa en estándares abiertos y utiliza un modelo de consenso para las decisiones de desarrollo y definición relacionadas con NFV.

En este sentido el ISG para NFV del ETSI es más un modelo de cooperación del sector que una organización de normalización tradicional que tiende a ser menos transparente acerca de las negociaciones y los compromisos que forman el estándar.

En la actualidad, muchos vendedores y clientes de NFV se benefician de las muy utilizadas API de NFV que no forman parte de las especificaciones de NFV, pero son cruciales para el éxito de NFV.

En este sentido, los estándares son definitivamente importantes, pero la meta es una arquitectura de red de NFV operativa que vaya más allá de las funciones y las capacidades de las especificaciones “oficiales” de NFV.

El ISG para NFV del ETSI anunció recientemente una colaboración con el organismo de normalización Metro Ethernet Networking (MEF) para integrar NFV como la arquitectura de servicios de red de Carrier Ethernet 2.0, un estándar en desarrollo para Ethernet de escala de operador.

VNF

Las VNFs son la base del paradigma NFV. A continuación, se realiza un análisis de las características más importantes de las mismas, las cuales deben ser tenidas en cuenta como parte del procedimiento de evaluación.

Estructura interna de las VNFs

Una VNF puede estar compuesta por uno o múltiples componentes llamados Componentes VNF (VNFC). En este caso, un VNFC es una entidad de software desplegada en un contenedor de virtualización.

Una VNF formada por un conjunto de uno o más VNFC se presenta al exterior como un único sistema integrado. La misma VNF puede ser desarrollada de manera diferente por cada proveedor de VNFs.

Modelos de balanceador de carga

Existen diferentes tipos de balanceo de carga en las VNF:

- Balanceador de carga interno a la VNF: el balanceado de carga puede ser un VNFC dentro de la VNF que se encarga de distribuir y recolectar flujos entre varias instancias de VNFC.
- Balanceador de carga externo a la VNF: balanceador de carga externo el cual puede ser una VNF que se encarga de distribuir y recolectar flujos entre varias instancias de VNF.
- Balanceador de carga de la Infraestructura de Red: un balanceador de carga proveído por la NFVI que se encarga de distribuir y recolectar flujos entre varias instancias de VNF.

Modelos de escalabilidad

Una característica muy importante a tener en cuenta de las VNF es el modelo de escalabilidad que éstas presentan. Existen varios modelos de escalabilidad, generalmente se identifican tres:

- Auto escalabilidad: el VNFM desencadena la escalabilidad de VNF acorde con las reglas en el descriptor VNF (VNFD), basado en el monitoreo de la utilización de recursos en la VM de la VNF, el EM, el VIM o localmente generados.
- Escalabilidad en demanda: una instancia VNF o su EM monitoriza el estado de los componentes de la instancia VNF y desencadena una operación de escalado a través de una solicitud explícita al VNFM de añadir o remover instancias de VNFC, o bien, incrementar o disminuir recursos de una o más instancias de VNFC.
- Escalabilidad basada en solicitudes de gestión: escalabilidad desencadenada manualmente o debido al OSS/BSS en correspondencia con las reglas definidas en el VNFD.

Independencia de hardware

Se pueden identificar dos clasificaciones en cuanto a la independencia con respecto al hardware de la NFVI subyacente, éstas son:

- Lista para ejecutarse en equipamiento estándar industrial: Independencia completa de la infraestructura física, permitiendo a la VNF ejecutarse en cualquier infraestructura física apropiada.
- Parcialmente lista para ejecutarse en equipamiento estándar industrial: Independencia parcial de la infraestructura física, pues algunos VNFC solo pueden ejecutarse en tipos específicos de hardware, de acuerdo a las características del usuario.

Las funciones de red que dependan completamente de hardware específico quedan fuera del estudio de NFV, porque como ya fue mencionado, la NFV se basa en desacoplar las funciones de red del hardware subyacente.

Virtualización

Esta característica es importante para ver la relación de la VNF con varios contenedores y tecnologías de virtualización. Algunas funciones de red que se ejecutan en ambientes virtualizados en la forma de una VNF están diseñadas para contenedores de OS (Operating System), mientras otras son diseñadas para virtualización de hipervisor. A continuación, se muestra la plataforma de virtualización que puede o no soportar las VNFs:

- Agnóstico al hipervisor: la VNF es capaz de ejecutarse en un ambiente de VM/Hipervisor; su software no tiene dependencia del ambiente de VM/Hipervisor; por lo tanto, puede ejecutarse sin cambios en diferentes hipervisores.
- Dependiente del hipervisor: la VNF es capaz de ejecutarse en una VM entre el ambiente lógico conocido por la VNF y la infraestructura física. La VNF depende del entorno VM/Hipervisor sobre el cual se encuentra ejecutada.
- Contenedores de OS: el software de la VNF es diseñado para el uso en tecnologías de contenedores de OS para su despliegue.
- Parcialmente virtualizada: algunos componentes de la VNF pueden ser virtualizados y otros no.

Elasticidad

La elasticidad de las VNFs se refiere al grado de crecimiento que pueden soportar las mismas, es decir, la cantidad de recursos de la infraestructura y número de VNFC soportados por las VNFs. Esta propiedad es importante para conocer si las VNFs implementadas en el entorno a evaluar permiten realizar operaciones de elasticidad, así como el tipo de escalabilidad que usan, horizontal o vertical. A continuación, se muestra la clasificación atendiendo a las operaciones de elasticidad:

- Sin elasticidad: La VNF requiere un conjunto fijo de recursos que no pueden cambiarse.
- Elasticidad solo mediante operaciones de escalamiento horizontal: la VNF es capaz de realizar operaciones de escalamiento agregando o eliminando separadamente instancias de VNFC específicos.

- Elasticidad en ambas dimensiones: la VNF tiene VNFCs que pueden ser escalados vertical y horizontalmente.

Operaciones de migración

La migración de recursos de las VNFs se refiere a la propiedad que tienen éstas de soportar que sus recursos sean movidos, por ejemplo, a otra infraestructura. Esta propiedad describe la habilidad de una VNF para ejecutar operaciones de migración de sus recursos:

- Sin soporte para migración “en caliente”: la VNF no soporta que sus recursos sean migrados durante su tiempo de asignación.
- Soporte para la migración “en caliente”: la VNF permite que sus recursos sean migrados a diferentes recursos de la NFVI usando tecnologías de hipervisor.
- Otros mecanismos de migración: La VNF permite otros mecanismos, ejemplo, apagado y reinicio en hardware diferentes; para cambiar dinámicamente la asignación de recursos.

Gestión de la aplicación

Esta propiedad describe qué tipo de aplicación de gestión debe ser usada para gestionar la VNF:

- Sin gestión: la VNF no requiere operaciones de gestión durante su ciclo de vida. Todas las conFiguraciones deben realizarse durante la conFiguración.
- Interfaz de gestión estandarizada: La VNF provee una interfaz estandarizada de gestión para ser utilizada por un EM u otro paquete de software OSS/BSS.
- Interfaz de gestión propietaria: la NFV provee una interfaz de gestión no estandarizada para ser utilizada por un EM del mismo proveedor de VNF.

OpenStack

OpenStack es un sistema que ofrece una infraestructura como servicio (IaaS), en la que los usuarios podrán albergar sus sistemas virtualizados. En la Figura 13 se muestra el logotipo característico de OpenStack.

Una de las principales características del sistema es su modularidad, que otorga una gran flexibilidad y escalabilidad. Gracias a esto, se pueden distribuir sus componentes en los servidores de un centro de datos de la manera que más convenga.

Figura 13. Logotipo de OpenStack

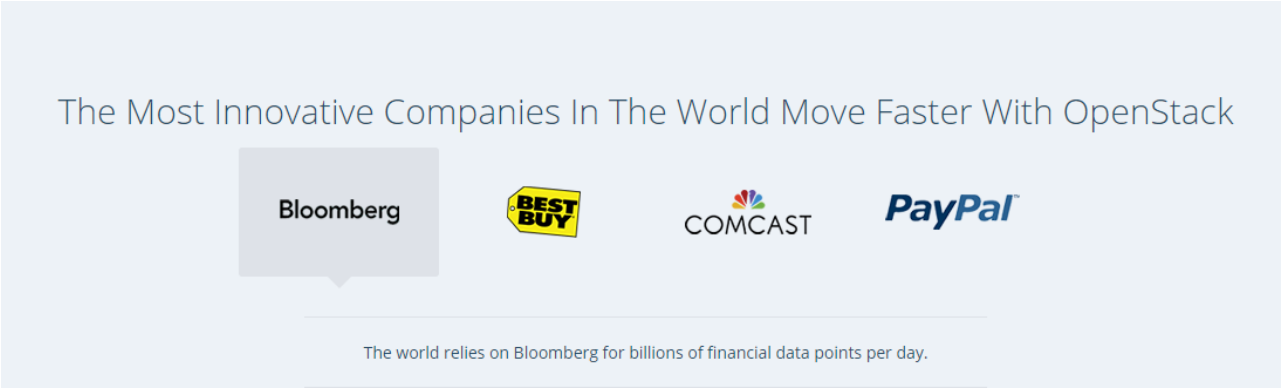


Figura 14. Grandes empresas que han desplegado sus aplicaciones/servicios propietarios en base a OpenStack

En la Figura 14 se muestran algunas empresas que han desarrollado sus propias aplicaciones y/o servicios, e incluso han desplegado su infraestructura basándose en la arquitectura OpenStack.

En la Figura 15 se muestra que, para la intercomunicación entre los diferentes módulos, OpenStack utiliza mensajería basada en el estándar *Advances Message Queuing Protocol* (AMQP), que como su nombre lo indica, es un protocolo de nivel de aplicación según el modelo de interconexión de sistemas abiertos (comúnmente conocido como modelo OSI).

Esta mensajería es gestionada mediante RabbitMQ, que es un servicio de negociación de mensajería para aplicaciones (aunque acepta otros gestores de colas).

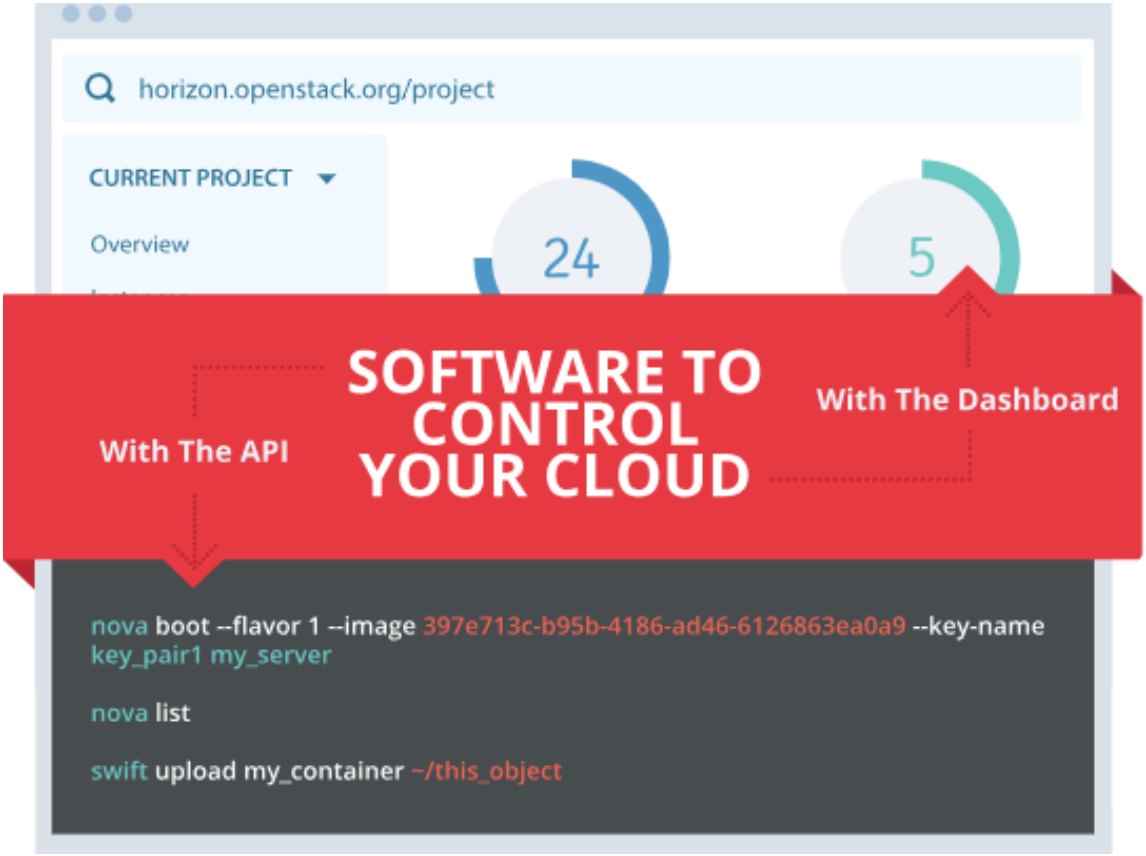


Figura 15. Intercomunicación entre los diferentes módulos de OpenStack

OpenStack tiene sus módulos englobados en dos grupos: los principales (entre los que se encuentran los que realizan las funciones de intercomunicación con el usuario, los que realizan las tareas de computación, los que realizan las tareas de red y los de almacenamiento) y los módulos complementarios (denominados *Shared Services)*. Lo anterior se muestra en la Figura 16.

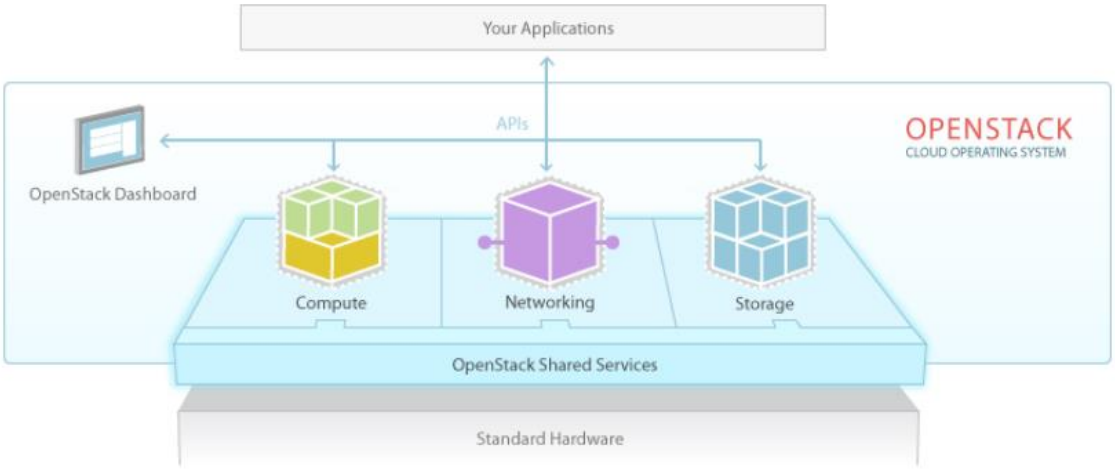


Figura 16. OpenStack Shared Services

A continuación, se describen los módulos principales:

- Compute: es el bloque principal del IaaS. Es el encargado de asignar los recursos a las diferentes instancias de máquinas virtuales que se crean en el entorno e identificar a los diferentes usuarios y módulos que intentan interactuar con el sistema. Los módulos que los integran son Nova (Computación) y Keystone (Autenticación).
- Storage: este bloque se dedica a gestionar el almacenamiento del sistema. Por un lado, tenemos el almacenamiento de los usuarios (imágenes, documentos, etcétera.) y por otro, el de las imágenes de las máquinas virtuales instaladas en el sistema (máquinas ya instaladas y listas para ser lanzadas en el sistema). Existen dos niveles de gestión de la información de los usuarios:
    - Nivel de objeto: se refiere a la gestión de almacenamiento a nivel de objetos concretos. Por ejemplo, un fichero, que es replicado en los diferentes servidores que componen el sistema.
    - Nivel de bloque: es la parte encargada de ofrecer la persistencia de la información al sistema. Gestiona la creación, agregación y desagregación de los dispositivos de bloques a los servidores.

Los módulos que lo componen son Glance (Servicio de imágenes de las máquinas virtuales), Cinder (Almacenamiento a nivel de bloque) y Swift (Almacenamiento de objetos).

- Networking: es la parte que separa el sistema Cloud del exterior. Nos permite crear redes virtuales con todos sus componentes (Switch, Router, etcétera). Esto nos permite crear una arquitectura completa de red, brindándonos una seguridad extendida. El módulo principal que lo gestiona es Neutron, aunque hay que destacar que el módulo Nova (Computación), incluye internamente virtualización de red, aunque a un nivel básico (solo permite la interconexión entre instancias lanzadas).

- Dashboard: es el bloque que permite a los usuarios y al administrador interactuar con el sistema. Tiene tres modos de interacción:
    - Línea de comandos
    - Web (si instalamos Horizon)
    - RESTful API.

  Dashboard tiene como módulo principal OpenStack Client (Línea de Comandos), aunque en este mismo bloque también podemos incluir el módulo Horizon (Interfaz Web), cuya imagen se presenta en la Figura 17, aunque según OpenStack está incluido en los *Shared Services*.

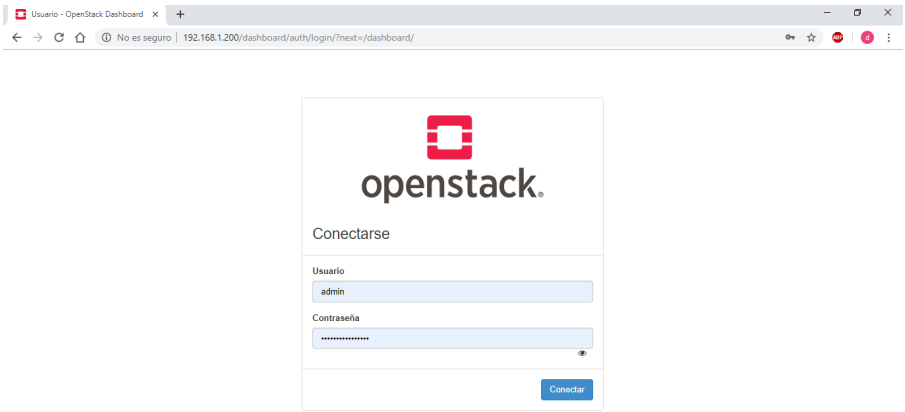


Figura 17. Interfaz web Horizon de OpenStack

En la Figura 18 se muestran los bloques que componen OpenStack, es decir, su arquitectura.

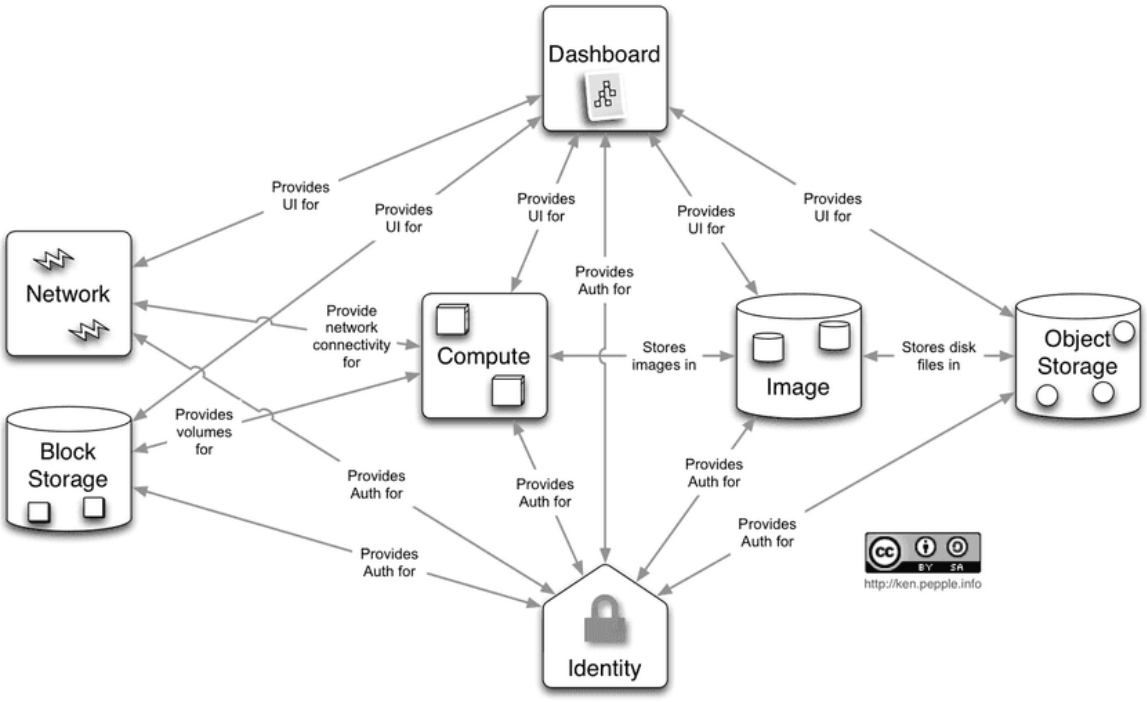


Figura 18. Bloques que componen OpenStack.

Respecto a los módulos complementarios (Shared Services), se trata de módulos que están preparados para la integración con el sistema.

Dependiendo del objetivo del sistema, se puede necesitar estos módulos. Este bloque está compuesto por:

- Ceilometer (Telemetría): este módulo permite monitorizar el sistema, tiene funciones de benchmarking y de estadísticas de rendimiento.
- Heat (Orquestación): permite implantar aplicaciones desarrolladas para la ejecución en la nube utilizando plantillas para su despliegue.
- Trove (Base de datos): este módulo permite ofrecer bases de datos relacionales y no relacionales como servicio.
- Designate (Servicio DNS): ofrece servicios de resolución de nombres integrado en el sistema cloud (SaaS).

- Zaqar (Servicio de Mensajería): es un servicio que mensajería para desarrolladores web. Combina las ideas del servicio pionero Amazon SQS con soporte adicional para eventos de broadcast.
- Barbican (Gestión de claves): módulo dedicado al almacenamiento y gestión de secretos tales como contraseñas, claves de encriptación y certificados X.509. Está orientado para ser una potente herramienta para todos los entornos, incluyendo los sistemas cloud.
- Congress (Gobierno): permite ofrecer Políticas de Sistema como servicio. Se encarga de automatizar la definición y aplicación de políticas en sistemas cloud, debido a que este tipo de sistemas son dinámicos.
- Sahara (Elastic MapReduce): este módulo nos permite crear un clúster de tipo Hadoop o Spark, mediante unos sencillos pasos.
- Ironic (Bare-metal): este módulo brinda otro modo de ofrecer los servicios cloud. En vez de virtualizar los sistemas, lo que hace es ofrecer una parte del sistema sin virtualizar, con la ventaja de no sufrir la sobrecarga que nos da el HyperVisor.
- Manila (Sistemas de Ficheros Compartidos): permite ofrecer almacenamiento compartido. Este almacenamiento compartido puede utilizarse desde las diferentes instancias que están en funcionamiento en el cloud o también podría ofrecerse como servicio.
- Magnum (Contenedores): este módulo permite hacer despliegues automatizados de sistemas con aplicaciones distribuidas mediante un contenedor (actualmente permite utilizar Docker Swarm y Kubernetes).
- Murano (Catálogo de Aplicaciones): permite a los desarrolladores diseñar aplicaciones preparadas para su ejecución en entornos cloud.

Como se ha mencionado anteriormente, dependiendo del objetivo que tenga el sistema que se quiera implementar, necesitaremos unos módulos u otros.

Se revisarán algunas conFiguraciones que utilizan las grandes organizaciones:

- **Computación de alto rendimiento**: como se comentaba en párrafos anteriores, el CERN utiliza este sistema para realizar los cálculos para el colisionador de hadrones. Para este uso utilizan Nova (Computación), Glance (Servicio de imágenes de las máquinas virtuales), Cinder (Almacenamiento a nivel de bloque), Keystone (Autenticación), Ceilometer (Telemetría), Heat (Orquestación) y Horizon (Interfaz Web).
- **Web Hosting**: este servicio consiste en ofrecer a los clientes servicios para publicar sus páginas web. Los módulos serían Nova (Computación), Neutron (Red), Glance (Servicio de imágenes de las máquinas virtuales), Keystone (Autenticación), Ceilometer (Telemetría) y Horizon (Interfaz Web).
- **Clouds públicos**: en este tipo de servicio se ofrece una infraestructura (IaaS) para que los usuarios puedan instalar sus servicios. Los módulos para este tipo de conFiguraciones serían Nova (Computación), Cinder (Almacenamiento a nivel de bloque), Swift (Almacenamiento de objetos), Neutron (Red), Glance (Servicio de imágenes de las máquinas virtuales), Keystone (Autenticación) y Designate (Servicio DNS).
- **Servicios Web y Comercio electrónico (eComerce)**: dedicado a dar servicios y a la venta online. Para este objetivo se utiliza Nova (Computación), Cinder (Almacenamiento a nivel de bloque), Neutron (Red), Glance (Servicio de imágenes de las máquinas virtuales), Keystone (Autenticación), Horizon (Interfaz Web) y Trove (Base de Datos).
- **Big Data**: otro de los servicios que, junto con *Cloud Computing*, son de los que más destaca actualmente. El *Big Data* consiste en la gestión, el procesamiento y almacenamiento de grandes volúmenes de información. Para esta finalidad los módulos indicados serían Nova (Computación), Glance (Servicio de Imágenes de las máquinas virtuales), Keystone (Autenticación), Neutron (Red), Horizon (Interfaz Web), Sahara (Elastic MapReduce) e Ironic (Bare-metal).

- **DBaaS**: Base de datos como servicio (Database as a Service) como su nombre indica, consiste en ofrecer un servicio de base de datos. Para este fin se usaría Nova (Computación), Cinder (Almacenamiento a nivel de bloque), Neutro (Red), Glance (Servicio de imágenes de las máquinas virtuales), Keystone (Autenticación), Swift (Almacenamiento de objetos), Horizon (Interfaz Web), Ironic (Bare-Metal), Trove (Base de Datos) y Designate (Servicio DNS).
- **Procesamiento de video y distribución de contenidos**: este tipo de servicios está actualmente muy extendido. Dedicado a ofrecer streaming, los cuales deben ser procesados y entregados al cliente para su visualización. Para este tipo de servicios se usarían los módulos Nova (Computación), Neutron (Red), Keystone (Autenticación) y Swift (Almacenamiento de objetos).

Cabe destacar que muchas empresas no utilizan OpenStack con sus módulos oficiales, sino que realizan una combinación entre los módulos y sus propias tecnologías, como es el caso de VMware, que utiliza toda la plataforma de OpenStack con su sistema de virtualización.

Por último, mencionar que OpenStack dispone de APIs (Application Programming Interfaces) para interactuar con el sistema. Cada módulo dispone de su propia API. Los principales métodos de comunicación mediante API son:

- cURL: este tipo de API funciona como una línea de comandos que permite enviar peticiones HTTP y recibir respuestas mediante HTTP.
- Línea de comandos OpenStack: como se ha comentado anteriormente, es el método principal para intercomunicarse con el sistema.
- Clientes REST: OpenStack dispone de una API RESTful, por lo que con cualquier cliente REST podemos interactuar con el sistema.
- *Software Development Kit*: los SDK nos permiten escribir programas para crear y gestionar los recursos en nuestro sistema cloud. Dispone de SDK para los lenguajes Java, Node.JS, Python, Ruby, .NET y PHP.

Principales servicios de OpenStack

Un servicio OpenStack puede alojarse en un nodo independiente o compartir el alojamiento según el diseño del arquitecto Cloud. Cada servicio a su vez se divide en varios componentes o sub-servicios. Por ejemplo, el Nodo de Cómputo es el referente del servicio Nova Compute y aloja el sub-servicio nova-compute.

El resto de sub-servicios como nova-api y nova-scheduler se alojan en el nodo Controlador. A medida que avanza el desarrollo de OpenStack, los servicios se actualizan o se reemplazan totalmente, provocando el cambio de nombre en el servicio. A continuación, se describirán los principales servicios de OpenStack:

Horizon

Provee una interfaz o cuadro de mando sobre el resto de servicios a los usuarios finales y al administrador. Estará incluido en este proyecto para visualizar la interfaz principal de OpenStack.

Características principales:

- Es una aplicación web basada en Django
- Se encuentra implementado a través de mod_wsgi en Apache
- Tiene una pequeña base de datos SQLite3, pero la mayoría de los datos son provistos por otros servicios
- Utiliza las APIs estándares de OpenStack para comunicarse con el resto de servicios
- Horizon es la implementación estándar del Dashboard, pero es fácilmente adaptable para cambiar la visualización y funcionalidad del Dashboard.

Nova Compute

Nova Compute transforma bajo demanda los pedidos de usuarios en máquinas virtuales o contenedores. Es el servicio responsable del ciclo de vida de las instancias, es decir permite lanzar, reiniciar, suspender o parar instancias. Está diseñado para escalar de forma horizontal, es decir, que podemos ir añadiendo más nodos de Cómputo si la carga de trabajo lo justifica. Constituye una parte fundamental y estará incluido en nuestra instalación de OpenStack.

Características principales:

- El cliente nova-api se encarga de aceptar y responder las peticiones del usuario final, ya sea a través del API de OpenStack Compute, del API de Amazon EC2 o de Horizon. A su vez, interacciona con KeyStone para lograr autenticación y puede hacer petición a glance-api para comprobar la disponibilidad de imágenes para las instancias.

  Inicia la mayoría de actividades como el lanzamiento de una instancia. Existe otro subservicio asociado llamado nova-api-metada que acepta peticiones de meta-datos de las instancias, pero normalmente se usa en despliegues con nova-network.
- El componente nova-compute es el encargado de administrar la virtualización, creando y finalizando las instancias de VMs a través de la Api del hipervisor correspondiente. Por ejemplo, tenemos XenAPI para hipervisor XenServer, libvirt para KVM o EMU y VMware API para VMware.

  Básicamente, nova-compute acepta acciones de la cola y ejecuta una serie de comandos como lanzar la instancia y actualizar su estado en la cinder-api para montar volúmenes en dichas instancias. Finalmente nova-compute gestiona el almacenamiento efímero de las instancias que representa el disco interno del Nodo de Cómputo.

  En los últimos años se ha avanzado en el desarrollo de la virtualización basada en contenedores. Nova ya puede crear instancias de contenedores LXC o Docker a través de módulos especiales, pero el soporte de funcionalidades es muy bajo en comparación con la virtualización estándar.
- El componente nova-schedule toma pedidos de instancias desde la cola y determina en cuál Nodo de Cómputo debería ejecutarse, actualizando el estado en la base de datos. Es decir, se encarga de la planificación de los recursos de cómputo.
- El componente nova-conductor media en las interacciones entre nova-compute y la base de datos. No se debe desplegar en el mismo nodo donde corra nova-compute por motivos de seguridad.
- El componente nova-consoleauth autoriza tokens para usuarios que deseen acceder a proxis de consola. Como proxis de consola existen nova-novncproxy, nova-

spicehtml5proxy y nova-xvpnvncproxy. En nuestro proyecto se usará nova-novncproxy.

- El componente nova-novnc proxy provee un proxy para acceder a instancias en ejecución a través de VNC. Soporta navegadores que funcionen como clientes novnc.
- El componente nova-cert provee de certificados X509, usados principalmente para la API de EC2.
- La base de datos almacena los estados de instancias en ejecución (run-time) y disponibilidad para nuevas instancias (build-time): instancias en uso, tipos de instancias disponibles, redes y proyectos disponibles, etcétera.
- La cola funciona como un eje central para el paso de mensajes entre los subservicios. Normalmente se implementa con RabbitMQ, pero se puede usar otra cola basada en el protocolo AMQP como Apache Qpid o ZeroMQ.
- Un usuario al crear una instancia debe elegir un sabor (flavor) que representa un tipo de conFiguración de recursos virtuales disponibles. Por ejemplo, el número de vCPUs (virtual CPUs) o el tamaño del almacenamiento efímero.
- Los componentes nova-volume y nova-network han sido reemplazados por los nuevos servicios Cinder y Neutron respectivamente.

Keystone Identity

Keystone provee autenticación y autorización para los usuarios, tenants y para el resto de servicios de OpenStack. Además, provee de un catálogo de servicios disponibles y sus API end-points. Constituye una parte fundamental y estará incluido en nuestra instalación de OpenStack.

La Figura 19 muestra los componentes de KeyStone y las relaciones con componentes del resto de servicios.

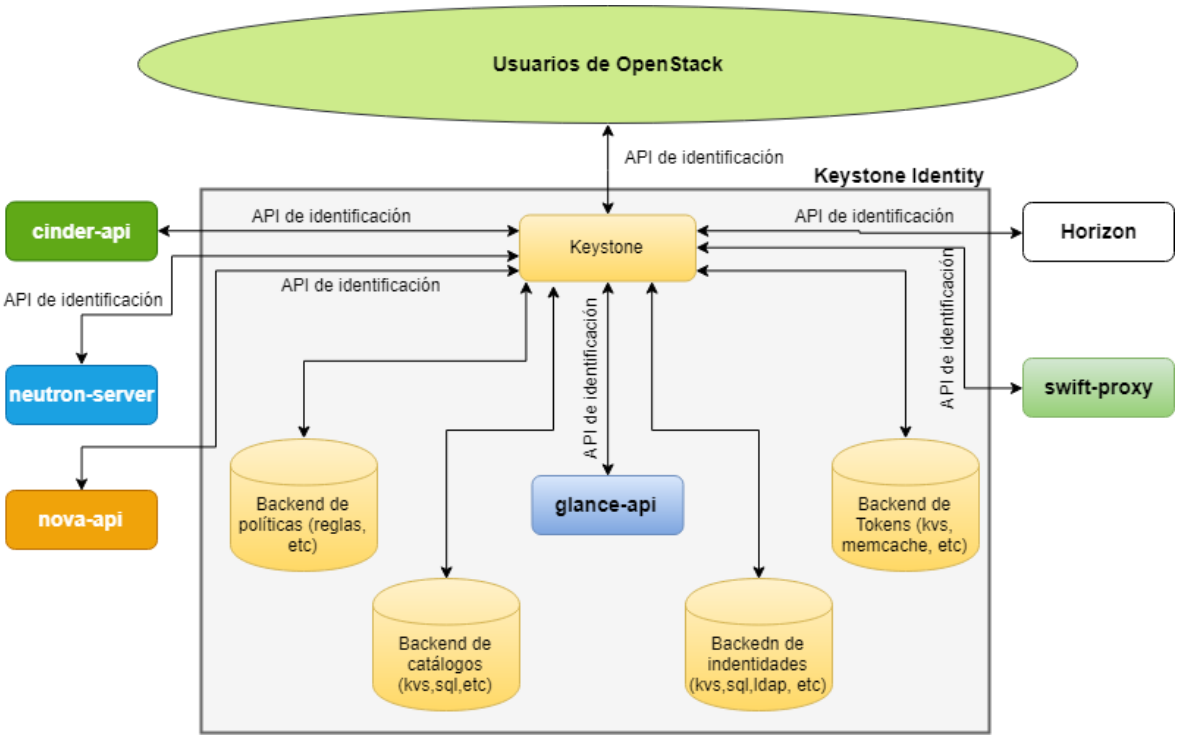


Figura 19. Componentes y relaciones del servicio KeyStone (elaboración propia)

Los siguientes conceptos están intrínsecamente relacionados con KeyStone y su comprensión es fundamental para entender el servicio:

- Usuario: Representación digital de una persona o sistema que usa OpenStack. KeyStone valida las peticiones del usuario a través de confirmación de las credenciales. Se pueden asignar tokens a un usuario para acceder a un recurso compartido, por ejemplo, para el acceso a la consola VNC. Un usuario puede ser asignado a un tenant en particular y restringirse por los permisos de dicho tenant.
- Credenciales: Datos que confirman la identidad del usuario. Por ejemplo: usuario y password, usuario y API key o un token de autenticación.
- Token: Línea de caracteres alfanuméricos usados para acceder a los recursos y APIs de OpenStack. Un token puede ser revocado luego de un tiempo y es válido durante un tiempo limitado.
- Tentant: Usuario o grupo de usuarios que comparten acceso común a una instancia con privilegios específicos. En dependencia del operador del servicio, un tenant puede ser un cliente, proyecto, cuenta, organización o suscriptor. En el contexto

interno de la arquitectura Cloud, existe el tenant “service” que identifica en KeyStone a todos los servicios de OpenStack.

- API end-point: Dirección accesible desde la red donde se encuentra el API del servicio, normalmente una dirección URL.
- Rol: Define los privilegios que tiene un usuario frente a un tenant asociado. Es decir, un usuario puede ser administrador de un tenant y usuario sin privilegios en otro tenant determinado. Es posible definir más roles, pero no es usual, así que normalmente tenemos un administrador y usuarios sin privilegios.

Características principales de KeyStone:

- En la infraestructura Cloud, KeyStone se encarga de autenticar el resto de servicios respondiendo a las peticiones que le llegan a su API. Cuando desplegamos un nuevo servicio éste debe ser registrado en la base de datos de KeyStone.
- Provee de un catálogo de servicios disponibles a los usuarios y sus API end-points para consumo de otros servicios y de los usuarios.
- Gestiona usuarios, roles, tenants (proyectos), y a qué tenants (proyectos) pertenecen.
- Constituye un punto centralizado para integrar las políticas de OpenStack, catálogos de servicios, tokens y la autenticación.
- Cada función de KeyStone (políticas, catálogo de servicios, tokens, autenticación) tiene un backend vinculable con distintas posibilidades de usar el servicio. KeyStone soporta backends estándares como LDAP, SQL o KVS (Key-Value Stores).

Neutron Network

Neutron se encarga de gestionar las redes virtuales entre instancias y la comunicación con redes externas. Provee de “conectividad en red” como servicio para interfaces virtuales (vNICS) creadas por NOVA. Su nombre inicial fue Quantum, pero debido a problemas de copyright tuvo que cambiar su nombre a Neutron. Constituye una parte fundamental y estará incluido en nuestra instalación de OpenStack.

La Figura 20 muestra los componentes de Neutron y las relaciones con componentes del resto de servicios.

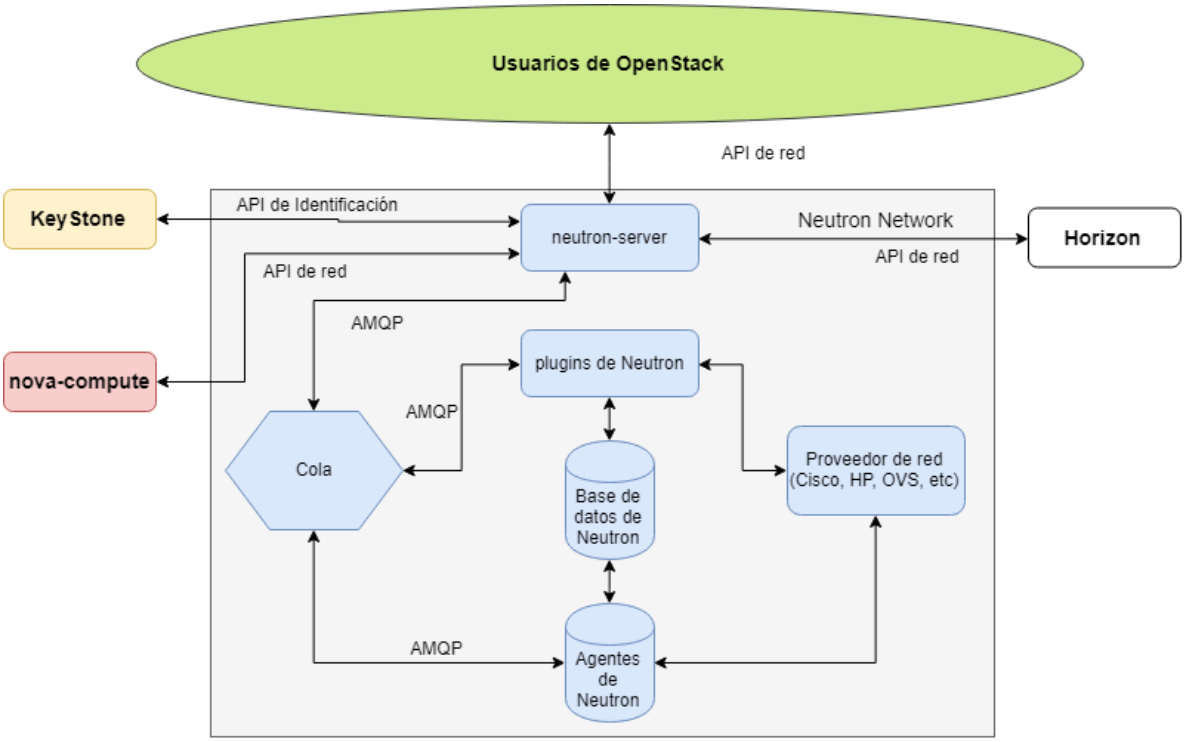


Figura 20. Componentes y relaciones del servicio Neutron (elaboración propia)

Neutron utiliza los siguientes conceptos:

- Plug-in: La implementación original de gestión de red en OpenStack se hacía a través del subservicio nova-network de Nova y presentaba un modelo muy básico de aislamiento a través de VLAN e IPtables en Linux.

  Con el desarrollo de Neutron se introduce el concepto de plug-in: implementación "enchufable" de back-end del API de red de OpenStack. Un plug-in puede usar cualquier tecnología que le permita implementar la lógica del API de red.

  Por ejemplo, algunos plug-ins usan VLANs e IPTables de Linux, mientras que otros usan OpenFlow o túneles "L2-in-L3". Existen plug-ins propietarios que dan soporte a equipamientos físicos (Cisco, HP, etcétera) para que gestionen las redes virtuales.

Por otra parte, algunos plugins tienen asociado a un agente específico en diferentes nodos del sistema Cloud que le ayudan a cumplir su función. Por ejemplo, el plug-in OVS necesita un agente llamado “neutrón-plugin-openvswitch-agent” instalado en el Nodo de Red y el Nodo de Cómputo.

El plug-ing ML” (Modular Layer 2) requiere especial atención debido a la versatilidad que aporta gracias a su estructura modular. Dicho plug-in se comporta como un framework para nuevos sub plug-ins o drivers conocidos como “Mechanism Driver” del plug-in ML2.
Es decir, un programador puede implementar funciones de capa 2 desarrollando un plug-in completo o escribiendo un driver para el plug-in ML2 que requiere menos código y esfuerzo. Finalmente, el plug-in ML2 permite a Neutron el uso simultáneo de diferentes variedades de tecnologías de red de capa 2 que se encuentran en la mayoría de centros de datos (VLANs, Túneles GRE, Túneles VXLAN).

En la Figura 21 se muestra la arquitectura modular del plug-in ML2 y el soporte de tecnologías L2 para OpenStack.

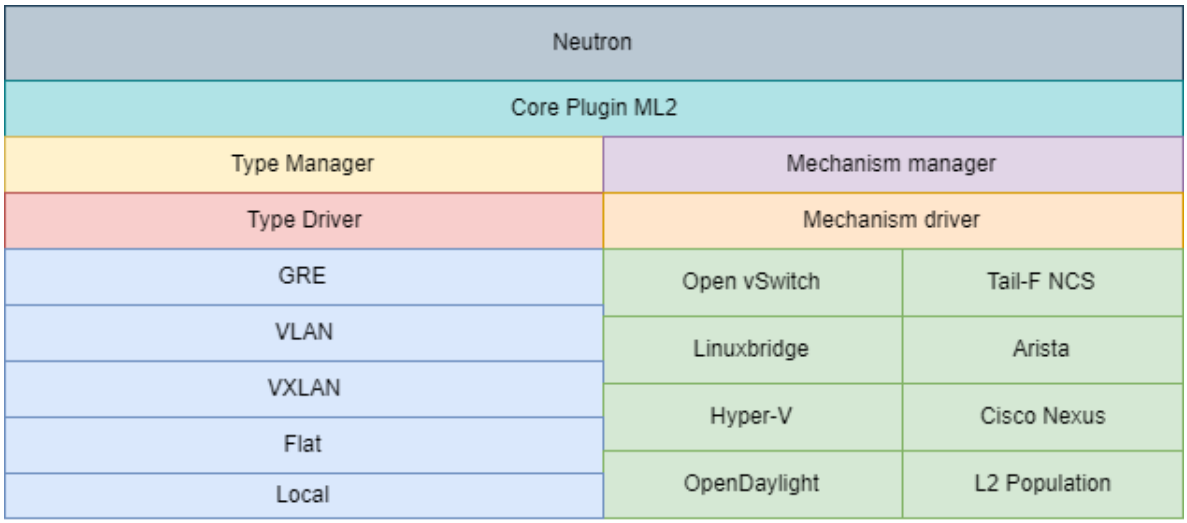


Figura 21. Arquitectura modular del plug-ing ML2

- Los “Mechanism Driver” pueden seguir tres modelos: Basado en agentes como Open vSwitch y Linuxbridge, basado en controlador como OpenDaylight y basado en Switches ToR como Arista y Cisco Nexus.

  Para nuestro proyecto usaremos el “Type Driver GRE” para redes virtuales internas y “Type Driver Flat” para la red virtual externa.

- Agente: Un agente normalmente se encuentra asociado a un plug-in para realizar una función de red concreta. Los agentes se pueden instalar en el mismo nodo del plug-in o en otro nodo, en dependencia de la arquitectura de los sistemas Cloud. Existen dos agentes comunes para todos los plug-ins. El “neutrón-l3-agent” brinda enrutamiento L3 y servicio NAT para proveer de acceso externo a las instancias, mientras que el “neutrón-dhcp-agent” provee de servicio DHCP para las redes internas.
- Red: Representa un dominio de colisión aislado de capa 2 y virtual. En el contexto Cloud, muchas veces se virtualiza una red como un switch virtual o lógico.
- Subred: Representa un bloque de direcciones IPv4 o IPv6 que podrán ser asignadas a las instancias en una red anteriormente creada.
- Puerto: Representa un puerto del switch virtual o lógico de una red determinada.
- vNIC: Representa una interfaz virtual de una instancia.

Características principales de Neutron:

- Neutron controla todos los aspectos de red de la infraestructura virtual de red (VNI “Virtual Networking Infrastructure”) y solo el acceso a la infraestructura física de red (PNI, Physical Networking Infrastructure) en un sistema OpenStack.
- Los plug-ins y agentes de Neutron son los que realmente ejecutan tareas de red específicas como el direccionamiento IP, conexión/desconexión de puertos y creación de redes y subredes.

- Otros plugins y agentes pueden añadir funcionalidades avanzadas en las redes virtuales para ser consumidas por los clientes. Como ejemplo tenemos, firewalls, balanceadores de carga y VPNs.
- Todas las instalaciones tienen al menos una red externa, que no es realmente virtual. Dicha red externa representa un grupo de las direcciones IP disponibles en la red "física" externa. El resto de redes de Neutron son consideradas internas y están asociadas a un Tenant determinado.
- Se pueden asignar direcciones IP de redes externas a puertos en redes internas. En términos de Cloud, si un dispositivo está conectado a una subred, dicha conexión también se conoce como puerto.
- El componente neutrón-server acepta peticiones a su API y las dirige al plug-in adecuado. Se comunica con KeyStone para lograr la autenticación y con nova-compute para desplegar redes virtuales donde se conectan las instancias.
- En la Figura 22 se muestra una conexión ya creada entre la vNIC de la instancia y el puerto de la red virtual.

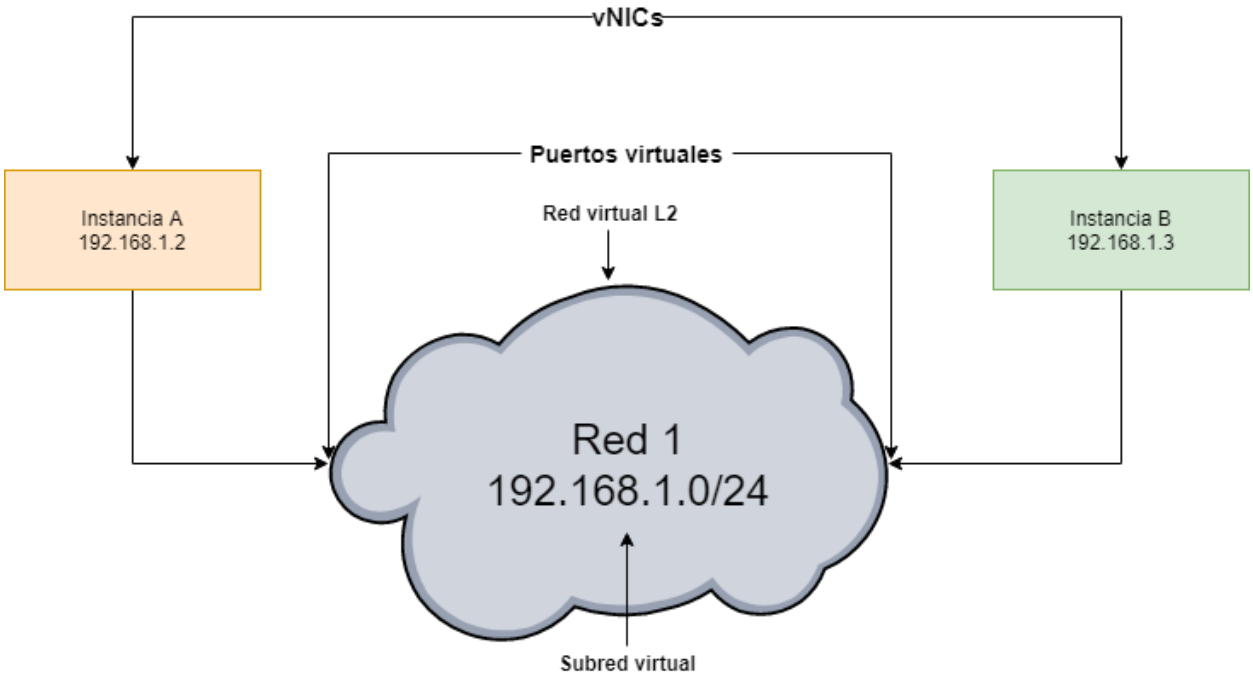


Figura 22. Redes virtuales e instancias (elaboración propia)

- A continuación se explicarán los pasos asociados a dicha conexión.
    1. Un tenant crea una red interna (Red 1).
    2. El tenant asocia una subred con dicha red (192.168.1.0/24).
    3. El tenant levanta una instancia, especificando una vNIC conectada a “Red 1”.
    4. Nova contacta con Neutron para que cree un puerto en la red “Red 1”.
    5. Neutron crea el puerto y le asigna una IP. Normalmente en redes internas, la asignación se hace a través del agente DHCP.
    6. Si el tenant decide eliminar la instancia, la elimina.
    7. Nova contacta con Neutron para que destruya el puerto anteriormente creado. Neutron destruye el puerto y devuelve su antigua IP al grupo de IPs disponibles para asignar.
- Neutron soporta “Grupos de seguridad” (Security Groups) que permiten al administrador definir las reglas de firewall para dichos grupos. Una instancia puede pertenecer a uno o más “Grupos de seguridad”. Las reglas más comunes son el bloqueo de puertos y bloqueo de determinados tipos de tráfico.
- Todas las instalaciones de Neutron usan un plug-in núcleo (core plug-in) y un plug-in que implementa los “Grupos de seguridad “(security group plug-in). Si no se necesitan los “Grupos de Seguridad”, se debe usar el “No-Op sercurity group plug-in”.
- La cola funciona como un eje central para el paso de mensajes entre neutrón-server, plug-ins y agentes. Se implementará la cola con RabbitMQ.
- La pequeña base de datos sirve para almacenar el estado de la red y es necesaria para algunos plug-ins.
- Neutron soporta una arquitectura de red SDN, pero delega la implementación exacta a un plug-in determinado, como el plug-in del controlador SDN OpenDaylight.

Swift Object Storage

Swift se encarga de almacenar y recuperar objetos de datos a través del "Sistema de almacenamiento de objetos".

La Figura 23 muestra los componentes de Swift y las relaciones con componentes del resto de servicios.

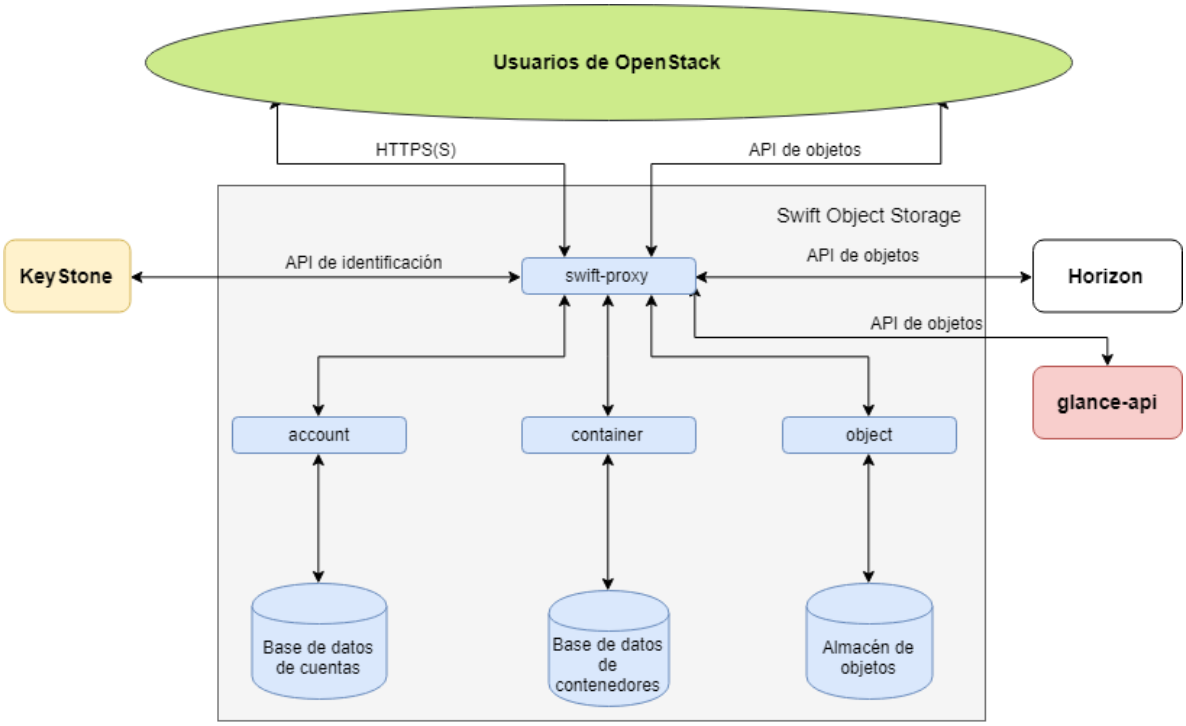


Figura 23. Componentes y relaciones del servicio Swift (elaboración propia)

Se pasa a explicar las características que definen a los objetos y a su sistema:

- La jerarquía del sistema de objetos es muy diferente a la jerarquía de un sistema de ficheros. En general, el sistema de objetos se divide en Cuenta, Contenedor y Objetos.
- Una Cuenta representa el máximo nivel de jerarquía y define un espacio de nombre para los contenedores. En OpenStack una cuenta es sinónimo de proyecto o cliente.
- Un Contenedor define un espacio de nombres para objetos. Un objeto con el mismo nombre en dos contenedores diferentes representa dos objetos.

- Un Objeto es una entidad que almacena información como un documento, imagen o fichero. Posee meta-datos extendidos asociados a los datos que contiene y tienen un identificador único que permite su recuperación sin conocer su ubicación física.
- Es posible comparar el almacenamiento de objetos con el estacionamiento de un hotel. El cliente intercambia las llaves de su auto por un recibo y no conoce dónde se estacionará su coche o cuántas veces será movido de lugar. En este ejemplo, el identificador único de los objetos está representado por el recibo.

Características principales de Swift

- El componente swift-proxy acepta peticiones a través de la API de objetos de OpenStack y a través de una API REST genérica accesible para los clientes. Las peticiones pueden ser subidas de ficheros, modificación de meta-datos o creación de contenedores. Por otra parte, el swift-proxy puede servir ficheros y un listado de objetos al navegador del cliente. Swift-proxy se autentica a través de KeyStone y normalmente se comunica con glance-api ya que puede almacenar sus imágenes.
- El servidor de cuentas (account-server) maneja las cuentas relacionadas con el sistema de objetos.
- El servidor de contenedores (container-server) maneja los contenedores relacionados con el sistema de objetos. Los contenedores de objetos también son conocidos como carpetas.
- El servidor de objetos (object-server) maneja los objetos en los nodos de almacenamiento.
- Swift utiliza procesos internos como autorreplicación, autorrepación de datos y balanceo de carga para garantizar la consistencia y disponibilidad de datos en el clúster de almacenamiento.

Cinder Block Storage

Cinder provee de almacenamiento en bloque a las instancias alojadas en la nube.

La Figura 24 muestra los componentes de Cinder y las relaciones con componentes del resto de servicios.

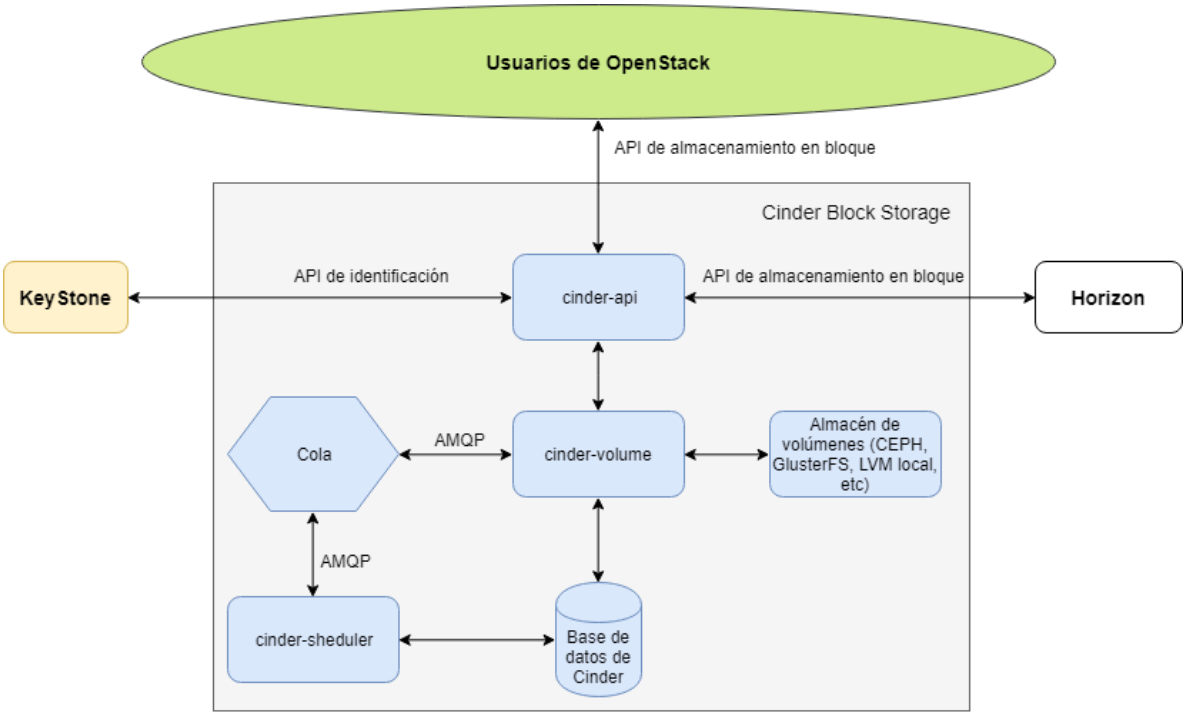


Figura 24. Componentes y relaciones del servicio Cinder (elaboración propia)

Se pasa a explicar las características que definen el almacenamiento en bloque:

- El almacenamiento en bloques también se conoce como almacenamiento de volúmenes. Los usuarios interactúan con este tipo de almacenamiento vinculando volúmenes a las instancias en ejecución.
- Los volúmenes son persistentes, es decir, podemos desvincular un volumen de una instancia y vincularlo a otra y los datos se mantienen intactos. Usualmente se compara el almacenamiento en bloques con la funcionalidad de una memoria USB o un disco duro externo.
- Los volúmenes se usan en escenarios de alto nivel de desempeño como bases de datos de aplicaciones.

Características principales de Cinder:

- El API de almacenamiento en bloque de OpenStack permite la manipulación de volúmenes, elegir un tipo de volumen disponible y hacer capturas (snapshots) de ellos.
- El componente cinder-api acepta las peticiones del API y las enruta hacia el componente cinder-volume para ser procesadas. Es el responsable de la autenticación a través de KeyStone.
- El componente cinder-volume lee y escribe en la base de datos de Cinder para mantener el estado de los volúmenes. También interactúa con el componente cinder-sheduler a través de la cola de mensajes. Finalmente, cinder-volume se comunica con el "driver" (controlador" de back-end utilizado en el almacén de objetos.
- El componente cinde-scheduler selecciona el proveedor óptimo de almacenamiento de bloques para crear un volumen determinado. Su función es similar a la realizada por nova-sheduler.
- La cola funciona como un eje central para el paso de mensajes entre cinder-volume y cinder-sheduler.
- La base de datos guarda el estado de los volúmenes.
- Cinder usar por defecto LVM de back-end para proveer los volúmenes.
- Se puede cambiar el back-end del almacén de volúmenes usando drivers de terceros como CEPH, Red Hat Storage (GlusterFS), IBM XIV, HP Leftland, 3Par, etcétera. En este caso cinder-volume usará el protocolo iscsi para comunicarse con el API del driver específico.

Glance Image Storage

Glance provee de un catálogo y repositorio de imágenes para ser usadas por las instancias, constituye una parte fundamental y estará incluido en nuestra instalación de OpenSatck.

La Figura 25 muestra los componentes de Glance y las relaciones con componentes del resto de servicios.

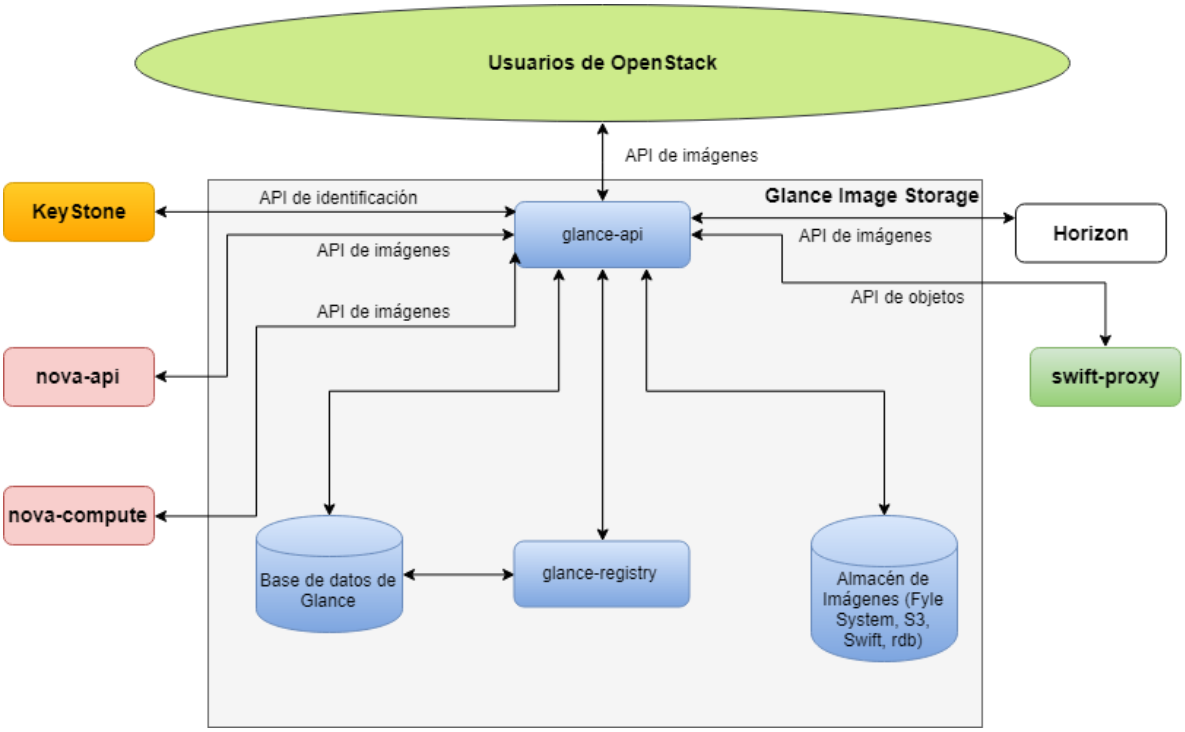


Figura 25. Componentes y relaciones del servicio Glance (elaboración propia)

Se pasa a explicar las características de las imágenes:

- Una imagen es un archivo único que contiene los contenidos y la estructura completa de un medio de almacenamiento.
- Las imágenes son utilizadas como medio de distribución de sistemas operativos y de snapshots de los mismos. Podemos definir una instancia como un sistema operativo en ejecución y un “snapshot” como la captura congelada de una instancia.

Características principales de Glance:

- El componente glance-api acepta las llamadas a su API para descubrir, almacenar y ofrecer imágenes.
- El componente glance-registry almacena, procesa y recupera los meta-datos de las imágenes (tamaño, tipo, etcétera).
- La base de datos de Glance almacena los meta-datos de las imágenes.
- Glance utiliza un repositorio o almacén de imágenes. Dicho almacén usualmente es Swift en un sistema OpenStack, pero Glance soporta además: sistemas de ficheros normales, Amazon S3, RADOS, etcétera.

Ceilometer Telemetry

Ceilometer monitoriza y recopila métricas de los servicios de OpenStack con propósitos de escalabilidad, facturación, benchmarking y estadísticos.

- La función de monitorización consiste en asignar agentes que realicen un sondeo al resto de servicios de OpenStack, las peticiones son del tipo: carga de CPU usada por el cliente, congestión en la red actual, almacenamiento libre, etcétera. A su vez, un colector realizará otro sondeo a los agentes para recopilar los datos actualizados. Si ciertos datos superan un umbral determinado (carga de CPU muy elevada) se dispara una alarma hacia el agente que redireccionará la alarma hacia un agente de alarmas.
- La función de recopilación de métricas se refiere al constante almacenamiento de datos con fines estadísticos.
- El componente *polling-agent* se encarga de realizar el sondeo hacia el resto de servicios OpenStack y construir métricas.
- El componente *notification-agent* se encarga de escuchar las notificaciones desde la cola de mensajes y convertirlas en eventos y muestras.
- El componente *collector* se encarga del sondeo hacia los *polling-agent* y *notification-agent* para almacenar los datos de forma más centralizada.

- El componente api permite a los clientes hacer peticiones al *collector* para visualizar los datos.
- El componente *alarming* evalúa y notifica las alarmas generadas por los *polling-agent* según las políticas de alarmas.

Heat Orchestration

Heat permite la orquestación de stacks de aplicaciones en la nube. Dichos stacks pueden estar compuestos por servicios de distintos proveedores Cloud.

No estará incluido en esta instalación.

Sus características son:

- Los Stacks se describen con un lenguaje descriptivo de plantillas con formato nativo HOT o con formato AWS CloudFormation de Amazon.
  Las plantillas se usan a través de la API REST de orquestación de OpenStack o a través de una API compatible con AWS CloudFormation de Amazon.
- Incluye la orquestación de los diferentes recursos necesarios y sus dependencias.
- Permite el escalado automático de stacks basados en métricas conFigurables que recoge de la API de Ceilometer.

OpenStack vs VMware

OpenStack está claramente orientado a la nube y vSphere sigue siendo un modelo tradicional de infraestructura de redes virtuales. OpenStack es similar a VMware Cloud, el cual ha quedado muy retrasado respecto a OpenStack y cuyas licencias son demasiado costosas.

Curiosamente, es tanto el potencial que VMware ha visto en OpenStack, que han sacado su propia distribución de OpenStack para integrarla en VMware vSphere. Es decir, se pueden desplegar clústers de vSphere con todas las posibilidades de OpenStack. Asimismo, en la Figura 26 se muestra la relación que mantiene WMware con OpenStack.

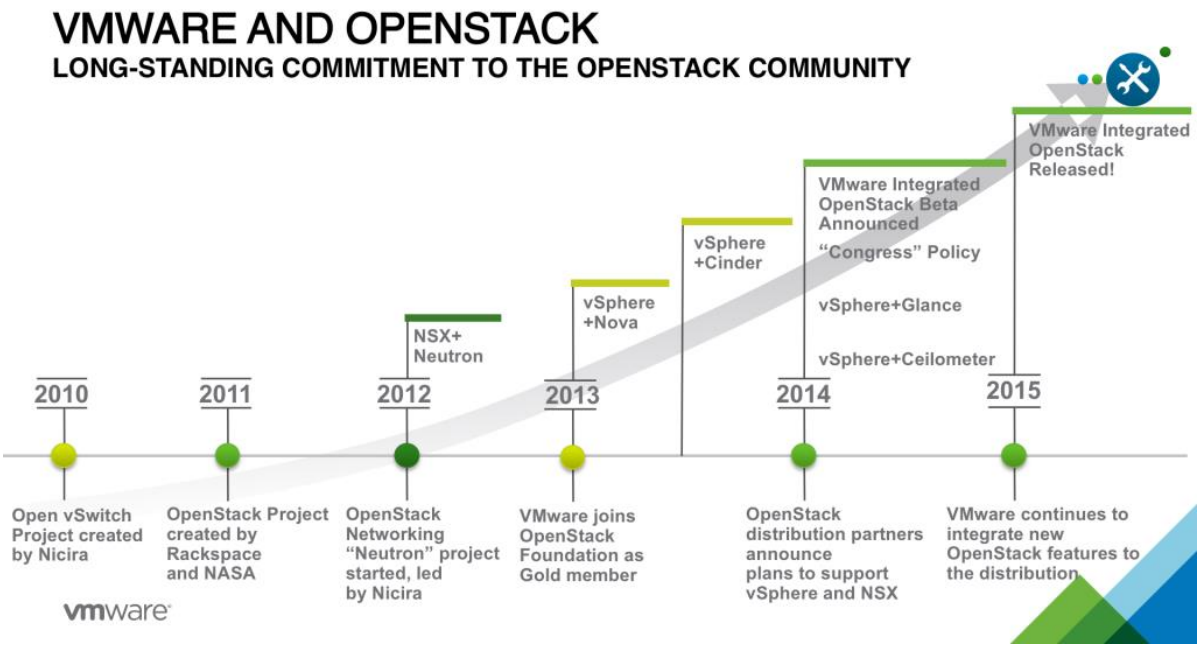


Figura 26. Relación VMware con OpenStack

CAPÍTULO II: MARCO TEÓRICO

Introducción

Se considera a NFV una potente alternativa para las redes de telecomunicaciones. Se plantea que reduce el CAPEX[22] por las funciones virtuales de red (VNF) que se ejecutan sobre equipo de propósito general de la industrial de las telecomunicaciones con altas prestaciones, es decir, servidores.

La reducción de OPEX[23] se logra por los ahorros de los requerimientos de espacio, potencia y ambientación de los equipos; y también por la simplificación del despliegue y gestión de los NS. NFV brinda agilidad y flexibilidad por la capacidad de escalar las VNFs[24] de acuerdo a las cambiantes demandas de la red.

El hecho de ejecutar funciones de red en software, permite a los administradores de red desplegar nuevos servicios más rápido que nunca, también reduce los riesgos al probar y desplegar estos nuevos servicios.

El presente capítulo está dedicado a profundizar los fundamentos teóricos de NFV y el estado del arte de su evaluación. En especial, se realiza su definición y se analizan los principales componentes de su arquitectura. Además, se revisan las principales actividades de estandarización de esta tecnología y los proyectos más importantes desarrollados en torno a la misma.

Definición de la NFV

En una red tradicional, cada función distinta se implementa normalmente como un dispositivo especializado basado en hardware propietario.

Tales dispositivos incluyen invariablemente una cantidad sustancial de software, pero el software y el hardware no se pueden separar, lo que los hace altamente dependientes uno del otro.

[22] Gastos en capital, por sus siglas en inglés

[23] Costo permanente por operación de un producto, por sus siglas en inglés

[24] Funciones virtuales de red, por sus siglas en inglés

La tecnología NFV transforma el enfoque clásico de las redes tradicionales, hacia un enfoque donde el software se encuentra desacoplado del hardware subyacente como se muestra en la Figura 27.

Figura 27. Cambio de enfoque en el paradigma de redes.

El concepto de NFV es transferir las funciones de red de los dispositivos de hardware propietario a aplicaciones basadas en software ejecutándose sobre servidores x86.

A través de NFV, las funciones de red pueden ser instanciadas como VNFs en varias locaciones, como son Centros de Datos (DC), nodos de red, y locales de usuarios finales según lo requiera la red, sin la necesidad de instalar nuevo equipamiento.

Arquitectura de la NFV

El NFV ISG ha definido un *framework*[25] de arquitectura de alto nivel funcional para NFV. Este framework de arquitectura identifica los bloques funcionales y los principales puntos de referencia entre estos bloques como se muestra en la Figura 28.

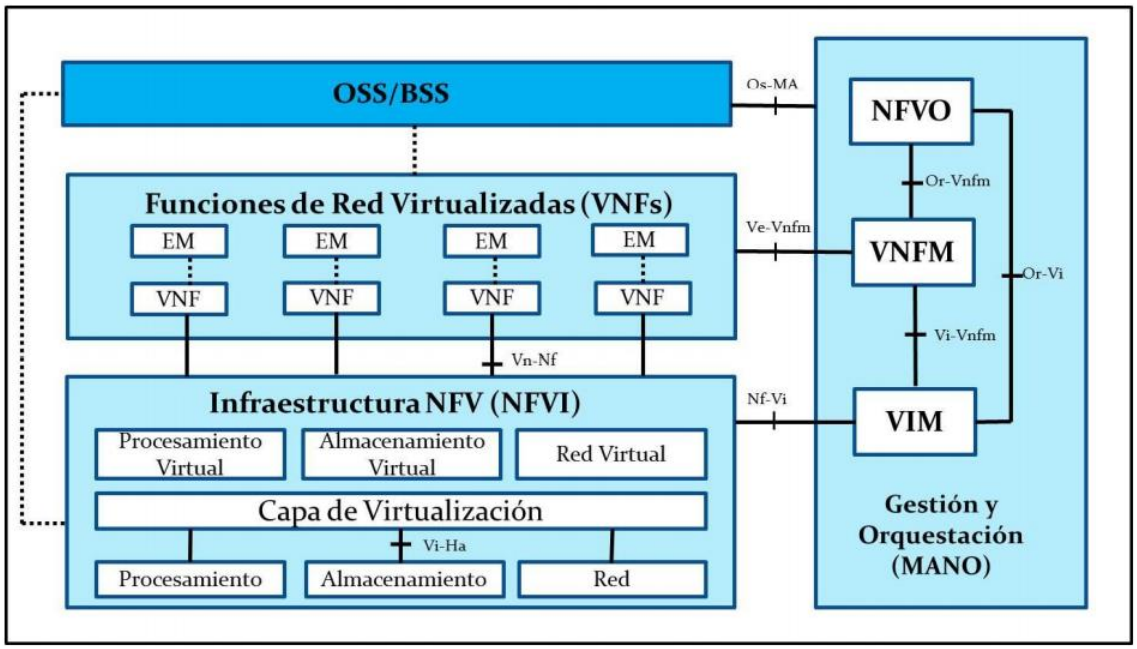


Figura 28. Arquitectura de la Virtualización de Funciones de Red (NFV)

De acuerdo al ETSI, la arquitectura NFV se encuentra formada por tres elementos claves: la Infraestructura de la Virtualización de las Funciones de Red (NFVI), las VNFs y el bloque de Gestión y Orquestación (MANO) NFV.

Infraestructura NFV

La NFVI es la combinación de recursos de hardware y software que conforman el entorno en el que las VNFs son desplegadas, gestionadas y ejecutadas. La NFVI se puede encontrar distribuida a través de varias localizaciones, como los Puntos de Presencia de la NFVI (NFVI-PoPs[26]). La red que provee conectividad entre estas localizaciones es considerada parte de la NFVI.

[25] Es una estructura conceptual y tecnológica de soporte definido, normalmente con módulos de software concretos

[26] Puntos de presencia, por sus siglas en inglés

Características de la infraestructura

En la NFVI se analizan los recursos de hardware de procesamiento, almacenamiento y red. También es muy importante conocer las características de la capa de virtualización y las VMs como expresión de los recursos virtualizados.

Características de los recursos de hardware

NFV utiliza equipos COTS[27]. Las características fundamentales que identifican el hardware COTS son:

- Equipo de propósito general: no diseñado ni optimizado para ninguna aplicación en particular.
- Integración: alta integración de los recursos de cómputo, almacenamiento y red.
- Alta eficiencia: capacidad de soportar un gran número de VMs y mejoras en el rendimiento de las mismas. Optimización de los puertos de E/S y de las tarjetas de red.
- Simple: rápido despliegue e introducción en la red. Facilidad de mantenimiento y reparación.
- Ágil: arquitectura innovadora que soporta mejoras futuras y expansión de los servicios.
- Seguridad: aislamiento entre VMs y la red. Puertos y periféricos gestionados y controlados centralmente.

El equipo de hardware COTS comúnmente basa su procesador en las soluciones de tipo x86-64 y tienen soporte para diversas técnicas de virtualización, incluida la virtualización completa asistida por hardware. Estas capacidades lo hacen un elemento vital dentro de la NFVI.

Entre las características más notables que deben ser analizadas como parte del estudio de la NFVI se encuentran: proveedores, capacidades de rendimiento, procesamiento,

[27] Producto de caja, por sus siglas en inglés

seguridad, mantenimiento, compatibilidad con plataformas y productos existentes, costos en tiempo y dinero de instalación y mantenimiento.

Específicamente, para la caracterización de los recursos de procesamiento, algunos de los parámetros a tener en cuenta son:

- Familia, modelo y versión.
- Arquitectura del grupo de instrucciones (ARM, x86, etcétera).
- Número de núcleos físicos y lógicos.
- Velocidad.
- Tamaño, nivel y tipo de caché.
- Políticas de lectura y escritura.

Algunos parámetros para caracterizar los recursos de almacenamiento son:

- Tipo, familia y versión.
- Capacidad de almacenamiento.
- Velocidades de lectura y escritura.
- Accesibilidad aleatoria o secuencial de la información.
- Volatilidad de la información.
- Habilidad para cambiar la información

Algunos parámetros para caracterizar los recursos de red son:

- Proveedores.
- Cantidad de enlaces físicos y lógicos.
- Número de controladores de interfaz de red (NIC).
- Número de puertos.

Características de la solución de virtualización

La mayoría de las plataformas de virtualización y consolidación de los Centros de Datos tradicionales y Nubes se encuentran basadas en tecnologías de virtualización de hardware: paravirtualización y virtualización completa, especialmente la Virtualización Asistida por Hardware (HVM).

Otra característica de los hipervisores es la arquitectura que presentan. Para sistemas de arquitectura x86, los hipervisores son clasificados en dos arquitecturas principales:

- Arquitectura hospedada: Instala y ejecuta la capa de virtualización como una aplicación sobre un sistema operativo, y soporta un amplio rango de conFiguraciones de hardware.
- Arquitectura nativa (bare-metal): Instala la capa de virtualización directamente en el hardware de un sistema basado en x86. Al tener acceso directo a los recursos de hardware, sin necesidad de pasar por un sistema operativo, ésta es más eficiente que una arquitectura hospedada.

Características de los recursos virtualizados

Algunas características de estos recursos pueden ser:

- Velocidad de CPU virtual.
- Capacidad de memoria de acceso aleatorio (RAM) y almacenamiento virtual.
- Velocidades de operaciones de lectura y escritura en memoria y almacenamiento virtual.
- Cantidad de enlaces virtuales.
- Cantidad de redes de Área Local Virtuales (VLAN).
- Cantidad de NIC virtuales.

Para caracterizar las VMs implementadas se deben tener en cuenta los siguientes aspectos:

- Tipo de VM (de sistema o de proceso).
- Cantidad de recursos asignados.
- Independencia de hardware.
- Aislamiento.

Recursos de hardware

Los recursos de hardware físicos incluyen los recursos de procesamiento, almacenamiento y red. Éstos son los recursos compartidos que las VNFs necesitan para sus necesidades de procesamiento, almacenamiento y conectividad. El hardware de procesamiento está constituido en su mayoría por servidores disponibles comercialmente. El almacenamiento puede encontrarse directamente en los servidores o Almacenamiento Conectado en Red (NAS) y en Red de Área de Almacenamiento (SAN). Los recursos de red incluyen funciones de conmutación, enrutamiento y enlaces físicos o inalámbricos.

Capa de Virtualización

La capa de virtualización, comúnmente referida como hipervisor, abstrae los recursos de hardware de la NFVI, separando el software del hardware subyacente. En resumen, la capa de virtualización es responsable de:

- Abstraer y hacer una partición lógica de los recursos físicos de hardware.
- Habilitar el software para la virtualización, de forma que este pueda disponer de los recursos de infraestructura.
- Proveer los recursos virtualizados sobres los cuales se ejecutarán las VNFs.

La capa de virtualización debe proveer un ambiente de alta eficiencia para el despliegue de las VNFs. Además, permite que las mismas puedan ser desplegadas sobre diferentes modelos de equipamiento de una variedad de proveedores, eliminando la dependencia ante los mismos.

La funcionalidad de esta capa está típicamente implementada mediante las técnicas de virtualización y el uso de hipervisores, de forma tal que las VNFs son desplegadas en una o varias máquinas virtuales (VM). Las especificaciones del NFV ISG no se restringen al uso particular de una solución para la capa de virtualización, se manejan soluciones basadas en hipervisores.

Se recomienda que las VMs tengan acceso a los recursos de hardware, para un mejor rendimiento, sin que esto signifique que la instanciación de las funciones virtuales depende del hardware subyacente, o componentes del mismo.

Un requerimiento importante es que la operación y funcionamiento de las VNFs no dependa de la solución establecida en la capa de virtualización, con el objetivo de asegurar la transparencia operacional y el despliegue de la NFV sobre un amplio número de soluciones y técnicas de virtualización.

Recursos Virtualizados

Los recursos virtualizados son los mismos componentes de procesamiento, almacenamiento y comunicación que proveen los servidores físicos, pero en este caso se encuentran separados del equipamiento por la operación de la capa de virtualización. Estos componentes que integran la capa de recursos virtualizados, y que conforman el ambiente de funcionamiento de las VNFs son: procesamiento virtual, dispositivos de entrada – salida (E/S) virtuales, memoria virtual y almacenamiento virtual.

La capa de recursos virtualizados está entonces integrada por todos los componentes que la capa de virtualización se encargó de agrupar y que ahora están disponibles para ser compartidos y utilizados por las VNFs.

Todos estos recursos, ya virtuales, deben ser capaces de proveer una plataforma con excelente rendimiento para las funciones virtuales y sus servicios, así como brindar la automatización y dinamismo para la ejecución de las acciones del DC.

Funciones de Red Virtualizadas

Una función de red es un bloque funcional dentro de una infraestructura de red que tiene bien definidas interfaces externas y comportamiento funcional.

Una VNF es una implementación de una función de red desplegada sobre recursos virtuales, por ejemplo, una VM. Una sola VNF puede estar compuesta por múltiples componentes internos, por tanto, puede ser desplegado sobre múltiples VMs, en cuyo caso cada VM contiene un solo componente de la VNF. Sin embargo, en otros casos la VNF completa puede ser desplegada en una sola VM.

Gestor de Elementos

El Gestor de Elementos (EM) es la entidad de gestión para los elementos de red. Ayuda a conFigurar los elementos de la red; recolecta y analiza información de fallos y funcionamiento, además de tomar acción en los mismos. Gestiona los aspectos de seguridad y contabilidad de las VNFs. Él mismo puede ser una VNF y realizar funciones de gestión para una o varias VNFs.

El EM tiene puntos de integración con el VNFM, dando soporte a la gestión del ciclo de vida. También con OSS/BSS, dando soporte a conFiguraciones de aplicaciones, gestión y activación de clientes que ofrecen las VNFs.

Gestión y Orquestación

De acuerdo al ETSI, el *framework* NFV MANO provee las funcionalidades requeridas para el aprovisionamiento de VNFs y operaciones relacionadas como las conFiguraciones de las VNFs y la infraestructura en que estas funciones se ejecutan. Incluye la orquestación y gestión del ciclo de vida de recursos físicos y/o software que soporta la virtualización de la infraestructura y gestión del ciclo de vida de las VNFs.

También incluye bases de datos usadas para almacenar la información y modelos de datos, los cuales definen propiedades del despliegue y ciclo de vida de VNFs, servicios y recursos. NFV MANO se centra en todas las tareas de gestión específicas de la virtualización, necesarias en el entorno NFV.

Orquestador NFV

El Orquestador NFV (NFVO) tiene dos responsabilidades fundamentales: la orquestación de los recursos NFVI, y la gestión del ciclo de vida de NS. Los NS son un conjunto de funciones de red y están definidos por las especificaciones funcionales y de comportamiento.

En orden de cumplir sus responsabilidades, el NFVO consume servicios expuestos por otras funciones, por ejemplo, funciones del Gestor de Infraestructura Virtualizada (VIM). Similarmente, los servicios provistos por el NFVO pueden ser consumidos por funciones que son autenticadas y propiamente autorizadas, ejemplo funciones del Sistema de Soporte a la Operación (OSS) y Sistema de Soporte al Negocio (BSS).

Gestor VNF

El Gestor de VNFs (VNFM) es la entidad que gestiona las VNFs. Es responsable de la gestión del ciclo de vida de instancias de VNFs. Cada instancia VNF debe tener asociado un VNFM, el cual puede gestionar una o múltiples instancias VNFs del mismo o diferente tipo. Muchas de las funciones del gestor de VNF se consideran genéricas y aplicables a cualquier tipo de VNF, sin embargo, el *framework* debe soportar los casos donde las instancias VNF demanden funcionalidades específicas, las cuales deben ser especificadas en el paquete VNF.

Gestor de Infraestructura Virtualizada

El VIM es responsable del control y le gestión de los recursos de cómputo, almacenamiento y red pertenecientes a la NFVI. Usualmente se encarga de todos los recursos dentro de un NFVI-PoP, a través de múltiples NFVI-PoPs o de un subconjunto de recursos dentro de un NFVI-PoP. También puede estar especializado en cierto tipo de recursos, ejemplo solo cómputo, o solo almacenamiento; o ser capaz de manejar cualquier elemento perteneciente a la NFVI.

Sistema de Soporte a la Operación/Sistema de Soporte al Negocio (OSS/BDSS)

Los OSS/BDSS son la combinación de las funciones de los operadores y las de soporte al negocio, que no están explícitamente contenidas en el *framework* NFV MANO, pero que intercambian información con entidades del mismo. Pueden proveer gestión y orquestación

de sistemas tradicionales y tener visibilidad completa de los servicios suministrados por redes legadas, en una red de cualquier operador.

Interfaces de referencia

Las interfaces o puntos de referencia son aquellas interfaces que conectan los diferentes componentes de la arquitectura NFV. A continuación, se realiza una breve explicación de las mismas.

- Capa de virtualización-Recursos de hardware (Vi-Ha): esta interfaz conecta la capa de virtualización a los recursos de hardware para crear un entorno de ejecución para VNFs, y recoger información relevante sobre el estado de los recursos de hardware para la gestión de las VNFs sin depender de ninguna plataforma de hardware.
- VNF-NFVI (Vn-Nf): representa el entorno de ejecución ofrecido por la NFVI a la VNF. No asume ningún protocolo de control específico. Está en el ámbito de aplicación de NFV con el fin de garantizar tanto el ciclo de vida del hardware de manera independiente, como el rendimiento y los requisitos de portabilidad de la VNF.
- NFVO-VNFM (Or-Vnfm): se utiliza, entre otras, para las solicitudes relacionadas con los recursos, por ejemplo, autorización, validación, reserva y asignación por parte del VNFM; el envío de la información de conFiguración para el VNFM, de modo que la VNF se pueda conFigurar apropiadamente para funcionar dentro del Gráfico de Reenvío de VNF (VNFs FG) en el NS y la recopilación de información sobre el estado de las VNFs necesarias para la gestión del ciclo de vida del NS.
- VIM – VNFM (Vi-Vnfm): esta interfaz de referencia se utiliza para las solicitudes de asignación de recursos, la conFiguración de recursos de hardware virtualizados y la información de cambio de estado.
- NFVO VIM (O-Vi): se utiliza para la reserva de recursos y/o asignación de las solicitudes del orquestador, la conFiguración de recursos de hardware virtualizado y la información de cambio de estado.
- NFVI-VIM (NF-Vi): esta interfaz se utiliza para la asignación específica de los recursos virtualizados en respuesta a las solicitudes de asignación de recursos, para la

transmisión de información del estado de los recursos virtualizados y para la conFiguración de recursos de hardware.

- OSS/BSS-NFV MANO (Os-Ma): esta interfaz se utiliza para las solicitudes de gestión del ciclo de vida de NS, el intercambio de políticas de gestión, el análisis de datos intercambiados, la transmisión de los registros contables y de uso de NFV, la capacidad de la NFVI y los intercambios de información de inventario.

Todo el sistema NFVI es impulsado por un conjunto de metadatos que describen los servicios, las VNFs y los requerimientos de la infraestructura NFV, por los que sistemas de gestión y orquestación NFV pueden tomar las medidas respectivas.

El modelo arquitectónico de referencia de la primera Figura muestra la interacción de los distintos bloques a través de los puntos de referencias descritos anteriormente, estos puntos hacen, además, que las entidades de la red puedan ser claramente separadas para proporcionar un ecosistema NFV abierto e innovador.

Casos de uso

Los casos de uso son los escenarios donde puede ser usada la NFV. El ETSI NFV ISG ha definido nueve casos de uso potenciales para NFV. Actualmente se están considerando otros dos escenarios en los que puede ser aplicada esta tecnología: Internet de las Cosas (IoT) y las Redes de Información Centradas (ICN). La mayoría de estos casos de uso pueden ser considerados los entornos NFV en los cuales será imprescindible realizar evaluaciones de rendimiento.

Infraestructura NFV como servicio (NFVIaaS)

La NFVI como servicio (NFVIaaS) es la capacidad de un proveedor de servicio (SP) de ofrecer su NFVI como un servicio, por ejemplo, a otros SPs. Permite una oferta de servicio comercial adicionar para directamente apoyar y acelerar el despliegue de NFV. La NFVI deberá proveer capacidades computacionales comparables a Infraestructura como Servicio (IaaS) de Computación en la Nube como ambiente de ejecuciones temporales, así como soportar los servicios de conectividad dinámica de la red que se pueden considerar comparables a Red como Servicio (NaaS). NFVI también puede ser ofrecido como servicio de un departamento a otro dentro del mismo SP.

Funciones de Red Virtualizadas como servicio (VNFaaS)

Las funciones de Red Virtualizadas como Servicio (VNFaaS) son análogas a Software como Servicio (SaaS) de Computación en la Nube, el cliente solo paga por acceso al servicio y no a la infraestructura que tiene los servicios, desconoce de la infraestructura que soporta los servicios que está contratando. Tiene como ventaja el eficiente uso de licencias de software, gestión y datos centralizados.

Plataforma de Red Virtual como Servicio (VNPaaS)

En Plataforma de Red Virtual como Servicio (VNPaaS) el SP provee un conjunto de herramientas de infraestructura de red y cómputo, así como algunas VNFs como plataforma para la creación de redes virtuales. La empresa cliente que usa este servicio, utiliza ese conjunto de herramientas para desarrollar su propia red virtual. VNPaaS es similar a VNFaaS, pero difieren en la escala del servicio y el alcance de control permitido al cliente, pues en este caso el cliente sí puede tener cierto acceso a la infraestructura. VNPaaS provee un servicio a mayor escala, típicamente provee una red virtual, en lugar de una sola VNF. También permite al cliente introducir sus propias instancias VNFs.

- Gráficos de Reenvío de VNF.
- Virtualización del Núcleo de la red Móvil y del Subsistema Multimedia IP (IMS).
- Virtualización de la Estación Base Móvil.
- Virtualización del entorno hogar.
- Virtualización de las Redes de Entrega de Contenido.
- Virtualización de las Funciones de Redes de Acceso Fijas.
- IoT.
- ICN.

Flujo de red detallado en OpenStack

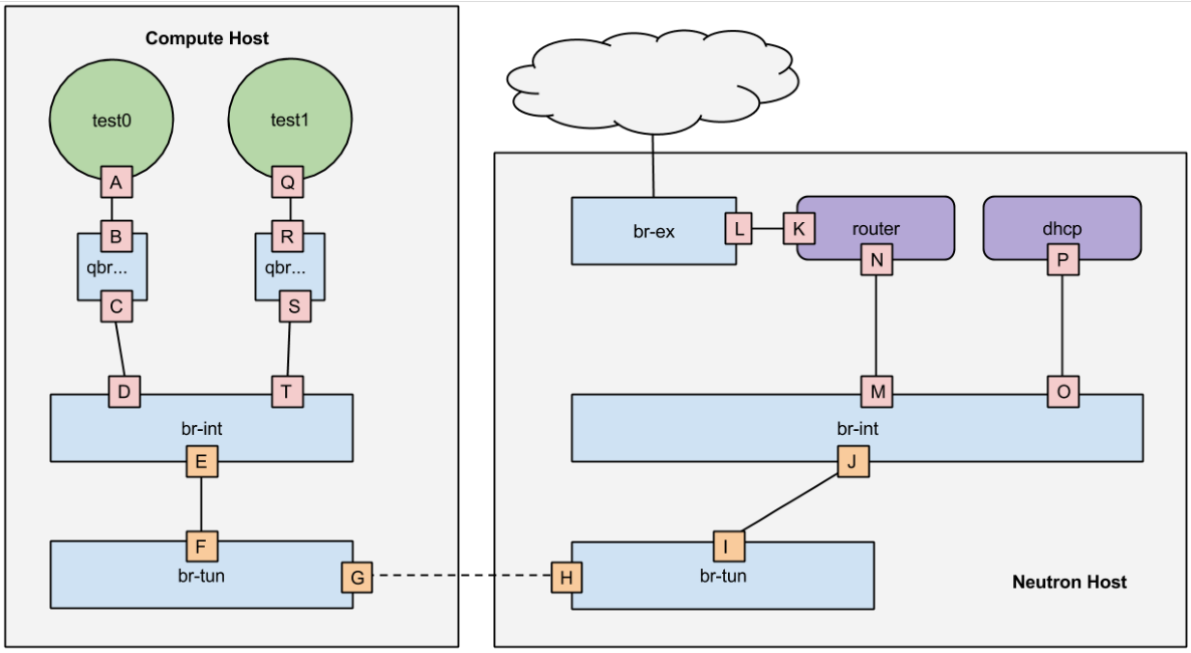


Figura 29. Flujo de red detallado

El esquema de la Figura 29 muestra el flujo de red que siguen los paquetes de una instancia en nuestra instalación OpenStack inicial, es decir, con el plug-in ML2 OpenVswitch y túneles GRE.

- Nodo de cómputo: Redes de las instancias (A, B, C): Desde la instancia sale un paquete por la interfaz virtual eth0, que está conectada a una interfaz TAP (tap0) en el nodo de Cómputo. Tap0 está conectada a un switch Linux (brctl) que actúa como firewall y por tanto es llamada “Switch firewall”.

  Idealmente, el tap0 debería conectarse directamente con el switch de integración (br-int) pero no se puede debido a la forma en que los grupos de seguridad de OpenStack están implementados. OpenStack usa reglas IPTables en los dispositivos TAP para implementar los grupos de seguridad y OpenVswitch no es compatible con reglas IPtables que son aplicadas directamente en dispositivos TAP que están conectadas a un puerto de un switch OpenVswitch y por tanto, se necesita el switch firewall intermedio. Finalmente, en switch firewall hay conectada otra interfaz tap1 que también estará presente en el br-int.

- Nodo de Cómputo: Switch de integración (D, E): El Switch de integración “br-int” ejecuta un “VLAN tag” al tráfico que sale de las instancias y un “VLAN untag” al tráfico que se dirige a las instancias. Cada red interna Tenant creada con Neutron tiene su propia VLAN ID. Br-int se conecta con br-tun a través de una interfaz especial de OpenvSwitch conocida como “patch-tun” (patch-tun es un concepto similar a un veth, pero no se visualiza con ifconfig).

  Br-int se comporta como un switch normal que utiliza “MAC-learning”. En caso de que la instancia “test0” envíen un paquete con dirección MAC-destino a la instancia “test1”, br-int la encaminará directamente si ya conocía dicha MAC-destino.
- Nodo de Cómputo: Switch túnel (F, G): El switch túnel (br-tun) traduce el tráfico etiquetado con VLANS que viene desde el br-int en tráfico GRE etiquetado con un “Tunel ID” específico. Cada VLAN ID tiene asociado solo un TUNNEL ID, por tanto, para cada red interna tenant se tiene un TUNNEL ID y un VLAN ID. También realiza la operación inversa, es decir, todo el tráfico que viene por el túnel GRE con el TUNNEL ID 1 se traduce en tráfico VLAN ID 1 enviado al br-int. La traducción es realizada por reglas OpenFlow instaladas en br-tun.

  En caso de que una instancia se levante en un nuevo nodo de Cómputo, el switch br-tub de dicho nodo creará un nuevo túnel GRE entre su br-tun y cada br-tun existente anteriormente. En otras palabras, se formará una red completamente mallada de túneles GRE entre switches br-tun de todos los nodos de Cómputo y de Red.

  Br-tun también se comporta como un switch normal que usa “MAC-learning”. Con el añadido de mirar los VLAN ID y reenviar el paquete solo por los TUNNEL ID asociados.

  Como se tienen reglas OpenFlow en el switch br-tun se puede decir que Neutron es una especie de “Pseudo-Controlador SDN”, ya que a través del plug-in ML2 y el agente OpenVswitch se ordena a un switch OpenVswitch utilizar reglas OpenFlow para controlar el tráfico. Para observar las tablas de flujo OpenFlow se puede

ejecutar el comando “ovs-ofctl dump-flows br-tun” en los nodos de Red y de Cómputo.

- Nodo de Red: Switch túnel (H, I): El tráfico llega al nodo de Red a través del túnel GRE unido al switch br-tun. En este caso, el br-tun del nodo de Red tiene reglas OpenFlow similares al br-tun del nodo de Cómputo, pero en este caso las reglas separan el tráfico destinado al servidor dhcp (MAC_DST=DHCP_MAC) del tráfico destinado al router virtual (MAC_DST=DHCP_Router) ya que ambos componentes se ejecutan en espacios de red diferentes. Como en el caso anterior, br-tun se conecta a br-int a través de un “patch-tun”.
- Nodo de Red: Switch de Integración: Este switch sirve para conectar las instancias a los servicios de red como los routers virtuales y servidores DHCP.
- Nodo de Red: Servidor DHCP (O, P): Cada red privada interna en la cual se habilita DHCP tiene un servidor DHCP ejecutándose en el nodo de Red. El servidor DHCP es una instancia de dnsmasq ejecutándose dentro de un espacio de nombres de red. Se conecta al br-int a través de una interfaz TAP.
- Nodo de Red: Router (M, N): Un router virtual es un espacio de nombres de red con una tabla de rutas y reglas IPtables que garantizan el enrutamiento entre subredes. Dentro del router tenemos dos interfaces TAP, una se conecta al br-int y la otra se conecta al br-ex como Gateway. Dentro de este espacio de nombres tenemos una tabla NAT netfilter responsable de asociar las direcciones IPs flotantes de las instancias con las IP privadas internas. Por defecto también se tiene una regla SNAT que permite a las instancias el acceso externo aún sin usar una IP flotante.
- Nodo de Red: Tráfico externo (K, L): El tráfico externo viaja hacia br-ex a través de una interfaz TAP en el espacio de nombres del router virtual.

Flujo de peticiones para levantar una instancia

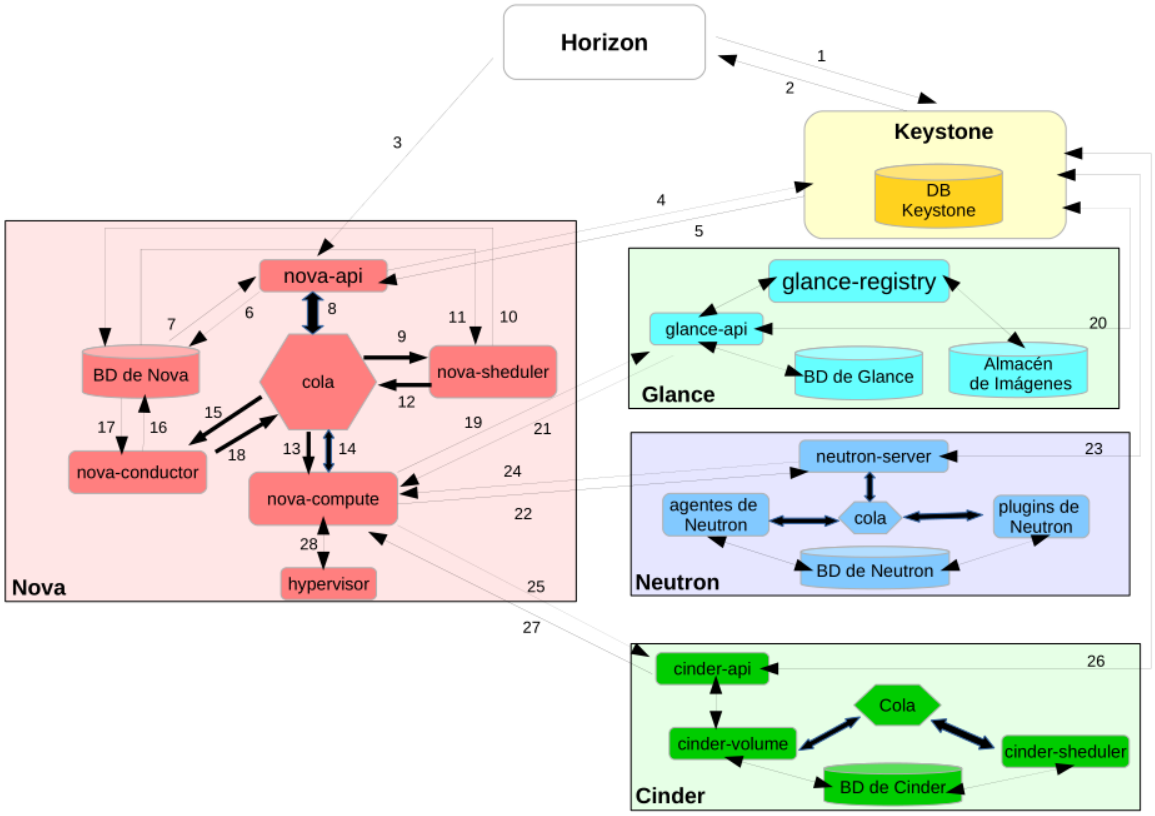


Figura 30. Levantamiento de una instancia

Uno de los casos de uso más importante para entender el funcionamiento distribuido de OpenStack es el flujo de peticiones que se establece para levantar una instancia. La Figura 30 muestra un esquema con las peticiones que se producen al lanzar una petición desde Horizon para levantar una instancia.

1. El usuario (tenant) introduce sus credenciales en Horizon y éste las envía a través del API REST de KeyStone para lograr la autenticación.
2. KeyStone autentica las credenciales y en la respuesta envía un token de autenticación que posteriormente podrá ser enviado en una petición a otros servicios.
3. El usuario ejecuta la opción “Lanzar instancia” y Horizon envía una petición a la API REST de nova-api.

4. Nova-API recibe la petición de Horizon y envía otra petición a KeyStone para validar el token y conocer los permisos que tiene el usuario sobre los recursos de Nova.
5. KeyStone valida el token y envía un paquete con los roles y permisos que tiene el usuario.
6. Nova-API interactúa con la base de datos de Nova.
7. La base de datos crea una nueva entrada para la instancia.
8. Nova-API envía el mensaje “rpc-call” a nova-scheduler para obtener el “HOST ID” donde correrá la instancia y actualizar su entrada.
9. Nova-scheduler recoge el mensaje desde la cola.
10. Nova-scheduler interactúa con la base de datos de Nova para encontrar el mejor host disponible, luego de un proceso de filtrado y ponderación.
11. Luego de filtrar y ponderar los nodos de Cómputo disponibles nova-scheduler obtiene el “HOST ID” del mejor nodo disponible.
12. Nova-scheduler envía el mensaje “rpc.cast” hacia nova-compute para lanzar la instancia en el host obtenido en el paso anterior.
13. Nova-compute recoge el mensaje de la cola.
14. Nova-compute envía el mensaje “rpc.call” hacia nova-conductor para actualizar la información de la instancia: “HOST ID”, y “flavor” (RAM, CPU, almacenamiento).
15. Nova-conductor recoge el mensaje desde la cola.
16. Nova-conductor interactúa con la base de datos de Nova.
17. Nova-conductor recoge la información de la instancia actualizada de la base de datos.
18. Nova-compute recoge la información de la instancia desde la cola.
19. Nova-compute realiza una llamada a la API REST de glance-api pasando el token del usuario y el “Image ID” para obtener los meta-datos de la imagen como el URI donde se encuentra.
20. Glance-api valida el token en KeyStone.
21. Nova-compute recoge los meta-datos de la imagen (URI) y la descarga en su almacenamiento local.

22. Nova-compute realiza una llamada a la API REST de neutron-server (quantum-server) pasando el token del usuario para conFigurar la red y la instancia pueda obtener una dirección IP.
23. Neutron-server (quantum-server) valida el token en KeyStone.
24. Nova-compute obtiene la información de red.
25. Si el servicio de Cinder está disponible nova-compute realiza una llamada a la API REST de cinder-api pasando el token del usuario y el tamaño de disco indicado en Horizon para vincular un volumen a la instancia.
26. Cinder-api valida el token en KeyStone.
27. Nova-compute obtiene la información del almacenamiento en bloques.
28. Nova-compute genera una petición con todos los datos recopilados y la envía al Hipervisor para que levante la instancia (usando una API determinada, libvirt para KVM)

Tecnologías de almacenamiento

En este apartado se mencionarán los tipos de almacenamiento y los backends Open Source que se tienen disponibles para OpenStack. Como una primera clasificación se tiene:

1. Almacenamiento efímero: En un almacenamiento efímero, los datos del usuario se pierden cuando la instancia se termina. Si sólo desplegamos Nova, el usuario sólo tendrá el almacenamiento efímero.
2. Almacenamiento persistente: El almacenamiento persistente permite que los datos del usuario se guarden al terminar una instancia. OpenStack soporta dos tipos de almacenamiento persistente:
    a) Almacenamiento de objetos: El objeto almacena una pieza de información con metadatos extendidos y está identificado por un ID único que permite su recuperación. Los metadatos se definen a conveniencia como la confidencialidad, el objetivo, tipo, etcétera.

       Se usa principalmente para almacenar información estática como fotos, videos, backups de discos, imágenes ISO, contenido web estático, etcétera.

Es la tecnología de almacenamiento más escalable y accesible (normalmente se accede mediante una API REST).

Por otra parte, el almacenamiento de objetos es “eventualmente consistente”, es decir, no garantiza que una petición de lectura devuelva la versión más actualizada, por lo que es idónea para un tipo de datos estáticos que no se modifican a menudo.

Como ejemplo de uso, se puede citar las imágenes subidas a Facebook y las canciones de Spotify. En OpenStack, Swift controla el almacenamiento de objetos, implementado en un clúster o red de almacenamiento distribuido.

b) Almacenamiento de bloques: En el almacenamiento de bloques, los ficheros se dividen en pequeños bloques de datos, cada uno con su propia dirección, pero sin metadatos asociados.

   A diferencia del caso anterior, el almacenamiento de bloques permite editar una parte de un fichero (un grupo de bloques determinados) sin tener que editar la parte restante. Puede ser accedido fácilmente por el sistema operativo como un volumen de disco montado.

   Se usa principalmente para información transaccional como sistemas de bases de datos.

   Es considerada “fuertemente consistente”, es decir, garantiza que una petición de lectura devuelva la versión más actualizada.

   Como ejemplo de uso, se citan las redes de almacenamiento internas de grandes corporaciones con arquitectura SAN. En OpenStack, Cinder controla el almacenamiento de bloques.

Tecnologías de Colas de mensajes

En los servicios Nova, Cinder y Neutron, se utiliza AMQP como tecnología de mensajes. La cola de mensajes proporciona las siguientes ventajas a un sistema distribuido como son Nova, Cinder y Neutron dentro de OpenStack:

- Redundancia: Si se produce el fallo de un servicio mientras procesa una petición, dicha petición no se perderá, ya que la cola almacena el mensaje hasta que el mismo sea procesado por completo.
- Naturaleza asíncrona: En muchas ocasiones, los sistemas distribuidos necesitan que los mensajes se vayan acumulando para procesarlos en lotes, la cola irá manteniendo los mensajes hasta que el servicio decida su procesamiento de acuerdo a su programación.
- Garantía de entrega y ordenamiento: Se garantiza que el orden en el que llegan los mensajes será el orden en el que sean procesados a través del tipo de cola FIFO.
- Disponibilidad: Si parte de la arquitectura falla, los mensajes no se perderán, ya que estos seguirán en la cola hasta que se les indique lo contrario, al mismo tiempo, la cola podrá seguir recibiendo mensajes para el procesamiento posterior a la recuperación del sistema.
- Elasticidad: Si el sistema llega al tope de capacidad de recepción de peticiones y se vuelve incapaz de responder por un anormal flujo de mensajes, el hecho de tener una cola o buffer de peticiones permitirá balancear, filtrar y normalizar el flujo.
- Desacoplamiento: La cola actúa como una capa intermedia de comunicación entre los servicios brindando flexibilidad en la definición de servicios. Cada servicio tan solo debe mantenerse alineado con los requerimientos de la interfaz de la cola de mensajes.
- Escalabilidad: Al agregar más unidades de procesamiento al sistema (más nodos para el mismo servicio), la cola de mensajes se encargará de balancear la carga entre todos los nodos.

Servicios en desarrollo de OpenStack

OpenStack está en constante desarrollo, por lo tanto, existen nuevos servicios que se están gestando:

- Trove (DBaaS): Tiene como objetivo proveer bajo demanda bases de datos de forma escalable, tanto para base de datos relacionales como no relacionales.
- Sahara (DPaaS): Tiene como objetivo proveer bajo demanda clústers para procesar grandes grupos de datos, usando Hadoop o Spark.
- Ironic (BMaaS): Tiene como objetivo proveer bajo demando servidores físicos en vez de máquinas virtuales (MaaS).
- Zaqar (MsgaaS); Tiene como objetivo proveer bajo demanda colas de mensajes, principalmente pensado para que los desarrolladores puedan conectar sus aplicaciones móviles y SaaS.
- Designate (DNSaaS): Tiene como objetivo proveer bajo demanda servidores DNS.
- Manila (SFSaaS): Tiene como objetivo proveer bajo demanda sistemas de ficheros compartidos.

Parámetros de rendimiento

Latencia

La latencia de red se mide en milisegundos (ms), cuanto más bajo es el número, la latencia se comporta mejor y, por consiguiente, la red también. Definir qué constituye una baja latencia depende en gran medida del sistema utilizado. Por ejemplo, una conexión Ethernet con una latencia superior a 10 ms, producirá una caída de rendimiento notable si supera los 150 ms.

En un escenario ideal, cada conexión tendría una latencia cero, sin embargo, existen un indeterminado número de variables que es poco probable alcanzar dicha latencia.

Incluso en el escenario perfecto, el acto de transferir un paquete de datos de un nodo a otro a la velocidad de la luz, conocida como propagación, producirá cierto retraso. Además, cuanto mayor sea el tamaño del paquete más tiempo necesitará para viajar a través de la red.

Por otra parte, también está la influencia de la infraestructura de red y el hardware. Las conexiones basadas en medios físicos producirán diversas latencias dependiendo del tipo de línea utilizada, ya sea coaxial o de fibra óptica, y si el paquete tiene que viajar a través de una conexión de red inalámbrica, agregará un retraso considerable al proceso.

Throughput

En redes de telecomunicaciones, es la tasa media de éxito de la entrega de mensajes en un canal de comunicaciones. Estos datos pueden ser enviados a través de un enlace físico o lógico, o pasar por un determinado nodo de la red. El rendimiento (throughput) se mide en bits por segundo (bit/s o bps), y en ocasiones, en paquetes de datos (paquetes de datos por segundo).

Jitter

Es un efecto de las redes de datos no orientadas a conexión y basadas en conmutación de paquetes. Como la información se discretiza en paquetes, cada uno de ellos puede seguir una ruta distinta para llegar a la dirección de destino.

Se define técnicamente como la variación en el tiempo en la llegada de los paquetes, causada por congestión de red, pérdida de sincronización o por las diferentes rutas seguidas por los paquetes para llegar al destino.

Las comunicaciones en tiempo real (como VoIP) son especialmente sensibles a este efecto. En general, es un problema frecuente en enlaces lentos o congestionados.

El jitter entre el punto inicial y final de la comunicación debería ser inferior a 100 ms (milisegundos), para que pueda ser compensado de manera apropiada ya sea mediante la implementación de un “jitter buffer” u otras técnicas.

Utilidades para medición de rendimiento

Iperf3®

Es una herramienta para mediciones activas del máximo ancho de banda alcanzable en redes IP. Soporta el ajuste de varios parámetros relacionados con el tiempo, protocolos y buffers. Por cada medición, indica el máximo ancho de banda (throughput), la tasa de transmisión (bitrate), paquetes perdidos (packet loss) y otros parámetros.

La versión 3 de ésta herramienta, es conocida como iPerf3®, es un rediseño de la versión original desarrollada por el equipo DAST (Distributed Applications Support Team, por sus siglas en inglés) dentro del NLANR (The National Laboratory for Advanced Network Research, por sus siglas en inglés). Es una versión implementada desde cero, con el objetivo de ser más pequeña, con un código base más simple y, además, una versión en forma de librería para que pueda ser utilizada en otros programas.

iPerf3® también tiene un cierto número de características que pueden ser encontradas en otras herramientas similares como son "nuttcp" y "netperf". Esto incluye, un modo de copia cero y una salida opcional en formato JSON. Es importante señalar que iPerf3 no es retrocompatible con la versión original de iPerf®.

Es desarrollado principalmente por ESnet/Lawrence Berkeley National Laboratory. Está liberado bajo una licencia "three-clause BSD".

Wireshark

Antes conocido como Ethereal, es un analizador de protocolos "open Source" utilizado para realizar análisis de paquetes y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta didáctica.

Ésta herramienta intercepta el tráfico y lo convierte en un formato legible para las personas, lo cual hace que sea más fácil identificar el tráfico que está cruzando por la red, con qué frecuencia y la latencia que hay entre determinados saltos. WireShark admite más de 2000 protocolos de red, la mayoría de ellos son inusuales o antiguos, los profesionales de TI encuentran una gran utilidad en el análisis de identidades IP. La mayoría de los paquetes son TCP, UDP e ICMP.

Dado el gran volumen de tráfico que circula por una red típica, las utilidades de WireShark ayudan a filtrarlo. Los filtros de captura solo recopilan los tipos de tráfico que son requeridos para el análisis, y los filtros de visualización ayuda a acercarse al tipo de tráfico que se desea inspeccionar.

CAPÍTULO III: MARCO METODOLÓGICO

El marco metodológico está referido a un momento que alude al proceso de investigación, con el objeto de ponerlo de manifiesto y sistematizarlo; con el propósito de descubrir, describir y analizar los supuestos del estudio y de reconstruir los datos, a partir de los conceptos teóricos operacionalizados. De acuerdo a lo mencionado, se define el Marco Metodológico, que será descrito a lo largo de éste capítulo.

Se presenta la metodología utilizada para el desarrollo del proceso de investigación, con la finalidad de que este procedimiento pueda ser replicado en todos los segmentos correspondientes a la capa de acceso de la Red Institucional de ESIME Zacatenco (en adelante, Red Politécnica), e incluso, a *posteriori* en algunos o todos los segmentos de la Red Institucional del Instituto Politécnico a nivel nacional.

La metodología propuesta consta de tres partes muy importantes, como se muestra en a Figura 31:

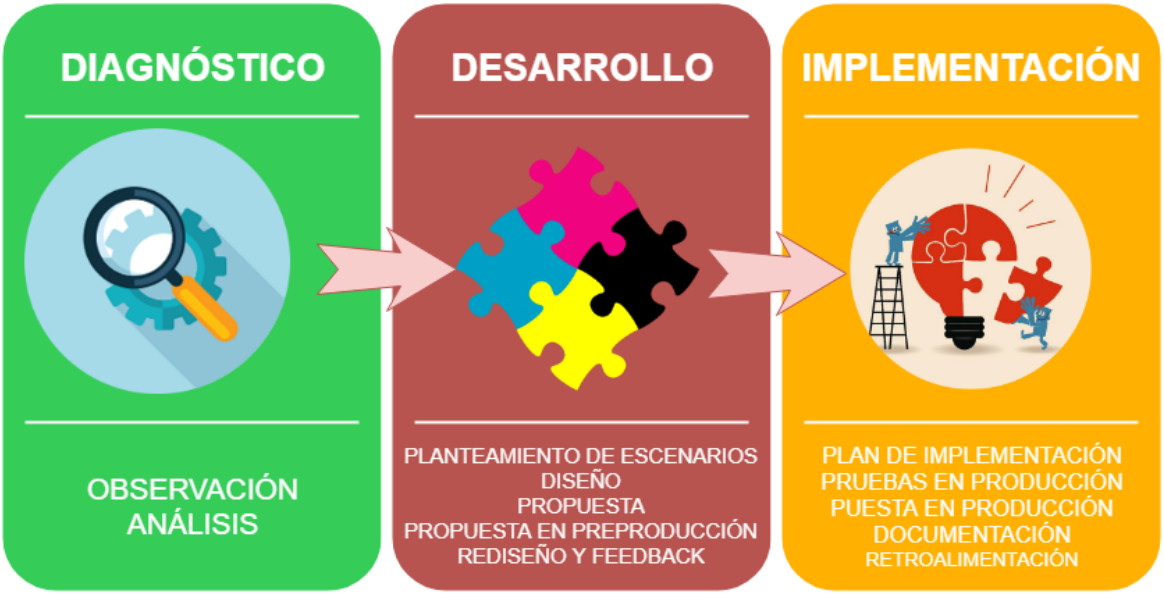


Figura 31. Etapas de la Metodología del Proyecto de Investigación (elaboración propia)

La aplicación de una metodología permite que las tareas relacionadas con dicho proyecto se realicen con el menor esfuerzo y menor uso de recursos posible. Lo anterior se menciona porque se tienen objetivos establecidos, tanto generales y específicos, mismos que deben cumplirse.

Por lo tanto, el hecho de definir una metodología resulta necesario para cualquier tipo de proyecto de investigación que involucre una implementación o modificación en un sistema que ya se encuentra en producción.

Diagnóstico

Es la primera etapa, y en ella se definen la topología física, topología lógica e incluso problemas de la red de datos de la Red Institucional del Instituto Politécnico Nacional, por lo tanto, es muy útil para establecer objetivos preliminares que hagan posible la aplicación de la metodología que tiene por objeto la implementación de equipos de red virtualizados bajo IPv4 en la capa de acceso correspondiente al segmento de la ESIMEZ. Una vez finalizado el diagnóstico, se tiene una visión total del comportamiento de la red de datos en cuestión, lo cual marca el punto de inicio de la segunda parte de la metodología, que consiste en el desarrollo del proceso de implementación, pasando desde pre-producción hasta producción. La etapa de diagnóstico, consta de dos sub-etapas, mismas que se detallan a continuación.

Observación

Se examina el funcionamiento del sistema bajo investigación, y a partir de ella se pueden hacer diagnósticos preliminares de corto alcance, para definir la problemática, éste es un paso muy importante dentro del proceso de investigación, ya sea de tipo exploratoria y/o experimental. En este caso, es de ambos tipos.

Para esta investigación, la observación estará presente a lo largo del desarrollo del proyecto, por lo que será de gran utilidad en esta etapa que consiste en la búsqueda de equipos de red que integran la Red Institucional del Instituto Politécnico Nacional y así tener los elementos necesarios para definir cómo está constituida su topología física y direccionamiento lógico.

Como es evidente, ésta etapa es la que demanda mayor inversión de tiempo y recursos, puesto que la Dirección de Cómputo y Comunicaciones (DCyC) no cuenta con la documentación suficiente acerca de la red que administra.

Además, en la Unidad de Informática (UDI) de ESIME Zacatenco, que es la encargada de la administración de la red de datos de ESIMEZ, tampoco existe una documentación previa de la red de datos que se administra, es por ello que se parte de cero para hacer la observación de la "red completa de la Red Institucional del Instituto Politécnico Nacional".

Durante la etapa de observación se obtiene información característica de la red, es decir; sus interconexiones, direccionamiento lógico, puertos abiertos, parámetros de operación, servicios y/o aplicaciones de red que provee, además de las mediciones correspondientes, para finalmente ser sometidas a un proceso de análisis.

En resumen, esta etapa consiste especialmente en la captura de datos de toda la red, desde información técnica, documentación antigua y/o obsoleta, diagramas, etcétera; hasta la problemática señalada por los administradores de la red, tanto del DCyC como de la UDI. Por lo tanto, todos los datos que sean adquiridos en ésta etapa serán importantes para analizar.

Análisis

El resultado de la etapa de observación, será un caudal de información en bruto que requiere un procesamiento, y el siguiente paso, es el análisis de la misma. Éste análisis permitirá conocer en qué estado se encuentra la red ESIMEZ, así como de sus características.

Entre las características más importantes de la red que se buscan conocer, se enuncian los siguientes:

- Topología física
- Direccionamiento lógico
- Equipos
    - Licencias

    - Ciclo de vida
    - Características, propiedades y conFiguración
    - Rendimiento
- Administración
- Normatividad: Lineamientos y estándares

Al obtener la información antes listada, se detectarán deficiencias, incongruencias, problemas de diseño, problemas de compatibilidad, pérdidas de rendimiento en equipos, falta de servidores de bases de datos, falta de servidores web y de otras necesidades que permiten plantear la propuesta para implementar la virtualización de funciones de red (NFV) en la red de datos de ESIMEZ.

Cuando se finaliza el análisis de la red, y como resultado del mismo, se identifican las carencias para implementar NFV en ella, será tiempo de iniciar con el desarrollo del rediseño de la red, y la propuesta para la implementación de funciones de red virtualizadas en la red ESIMEZ, ya que será posible proponer un escenario y rediseño de la red con una base de argumentos sólidos, tanto a nivel técnico como a nivel de necesidades y problemas que la red presenta.

Desarrollo

Con los obstáculos técnicos y/o de hardware para la implementación de NFV plenamente identificados, es momento de formular una propuesta con base en la información obtenida en la etapa del diagnóstico.

Durante ésta etapa se estudian las distintas posibilidades existentes para implementar NFV en la red de datos de la ESIMEZ, para posteriormente proponer escenarios que sean viables y/o factibles, tanto a nivel técnico, de hardware y económico, con el objetivo de realizar pruebas experimentales con los mismos, de ésta forma se podrá determinar cuál de todos los escenarios es la mejor opción costo-beneficio, y a partir de ella realizar un diseño y posteriormente una propuesta de implementación ágil de acuerdo con los equipos actuales, las necesidades de la red y el "*knowledge*" y "*background*" en redes y cloud computing de los administradores de la red.

La etapa de desarrollo, consta de cinco sub etapas (Planteamiento de Escenarios, Diseño, Propuesta, Puesta en preproducción y Feedback), como se observa en la Figura 32:

Figura 32. Partes que componen la etapa de Desarrollo (elaboración propia)

Planteamiento de escenarios

Como todo problema, existen múltiples soluciones para el mismo, por esta razón se hace necesario plantear varias soluciones, para estar en condiciones de determinar cuál de ellas es la más viable y factible, de acuerdo al hardware con el que se cuenta.

En este caso, las soluciones para este proyecto de investigación son infinitas, ya que las mismas dependerán de distintos factores, a nivel de software de OpenStack® y a nivel de recursos humanos y técnicos de:

- Administradores de la Red
- Personal Técnico
- Equipos de Networking
- Estructura y topología de la red
- Potencia de cálculo de servidores estándar
- Presupuesto
- Entre otros

Al plantear los distintos escenarios, se podrán tener las distintas opciones para elaborar la propuesta final para la implementación. La técnica de creación de escenarios permite combinar distintas variables para prever situaciones posibles y de esta forma realizar una planificación flexible. La simulación de los diversos escenarios contempla anticipar combinaciones entre comportamientos de las variables externas a la red y decisiones internas de los administradores.

Al finalizar esta etapa se elige la mejor opción, teniendo en cuenta que la misma es aquella cuyo despliegue es más ágil, flexible, escalable y con la mejor relación costo-beneficio para que, a partir de ella, se realice la propuesta formal para la implementación de virtualización de funciones de red en la red de datos de la ESIMEZ.

Diseño de escenarios

En el diseño de la propuesta formal, se debe tomar en cuenta las mejores características de todos y cada uno de los escenarios generados en el punto anterior a fin de combinar las mejores características de cada uno con el objetivo de crear una sola propuesta. Una vez creada y elegida la mejor propuesta para la implementación de NFV dentro de la red de datos de la ESIMEZ, se debe realizar el diseño y plantear la propuesta formal que permitirá que el diseño antes mencionado sea posible.

De la opción elegida, se tendrá que plantear escenarios de implementación, con esto, se podrá definir cuál es la mejor estrategia para la implementación de NFV, los escenarios podrán ser: simulaciones, emulaciones y/o maquetas.

Estos escenarios permitirán conocer los alcances de la propuesta y del diseño planteados, por lo tanto, será posible determinar si es necesaria la adquisición de nuevos servidores, cambiar la topología de red, realizar cambios en la programación de OpenStack, reconFigurar equipos a nivel de capa 2 y 3, respectivamente; entre otras acciones que modifiquen el estado actual de la red.

Finalizando el diseño de la opción más viable para NFV, se tiene que realizar la propuesta formal a los administradores de la red para que se aprueben o propongan modificaciones, o se tengan que solicitar las autorizaciones correspondientes antes de ser implementado.

Selección de escenario viable

En este punto, con la anuencia de los administradores y tomando en cuenta sus modificaciones en caso de tenerlas, se selecciona el escenario con la mejor relación costo-beneficio. Es decir, aquel escenario que permite reutilizar el hardware de los servidores físicos ya existentes, y que, además promete entregar buenos resultados en las características pruebas técnicas a las que será sometido durante su evaluación.

Evaluación de escenario viable

Al tratarse de un proyecto sumamente complejo, se han planteado numerosos escenarios al principio de ésta metodología, sin embargo, ahora se ha elegido la mejor opción, misma que debe ser evaluada de acuerdo a parámetros técnicos que se han establecido anteriormente como son: la latencia, el jitter y el throughput.

De acuerdo a los resultados obtenidos, se propone clasificar los escenarios en algunas de las siguientes tres situaciones:

- **Escenario optimista**: ¿Qué es lo mejor que puede pasar? ¿Se deben tomar medidas si esto ocurre? No se trata de fantasear, pero sí de considerar que el proyecto esté a plena marcha, aunque su probabilidad de ocurrencia no sea clara. Se recomienda analizar el impacto en la red y en sus usuarios; para anticipar qué hacer en caso de que se dé este escenario positivo.
- **Escenario moderado**: Este escenario es una situación intermedia, puede ser que la red funcione bien por momentos y de pronto haya una caída. Ésta situación también debe ser considerada, para que no haya sorpresas tanto negativas como positivas.
- **Escenario pesimista**: Es cuando el despliegue de la propuesta sale mal. A nadie le gusta pensar de forma negativa, sin embargo, el responder a esta pregunta permite calcular los riesgos y prepararse para el caso de que suceda lo peor. En este punto, es necesario establecer cursos de acción que permitan mantener la red o minimizar los tiempos fuera de servicio si la red se inclina hacia este escenario.

Propuesta en preproducción

Si la propuesta ha superado la evaluación, es momento de implementarla en un entorno de "preproducción". En esta fase de llevan a cabo las pruebas finales del proyecto antes de su paso final al entorno de producción, donde se pondrá en funcionamiento en un escenario real.

Para ello, en primer lugar, se transfiere el proyecto una vez que ha sido validado en el entorno de integración, para ser sometido a un conjunto exhaustivo de pruebas, de diversos tipos:

- Funcionales y estructurales
- De rendimiento
- De tolerancia a fallos
- De seguridad
- De disponibilidad

Si estas pruebas, conocidas como de aceptación, resultan satisfactorias, entonces el proyecto está listo para su paso al entorno de producción.

Sin embargo, si las pruebas de aceptación no son satisfactorias, se abre paso a la fase de "Feedback" o retroalimentación.

Retroalimentación

Se recurre a este paso cuando se detectan fallas en la ejecución del entorno de preproducción del proyecto. En esta etapa, se enumeran o se enlistan todas las fallas, errores e incidencias que presenta el proyecto en preproducción, a fin de solucionarlas y reenviar el proyecto a una reevaluación, a fin de tenerlo completamente operativo para su traspaso al entorno de producción.

Implementación

En esta etapa, se realizará un programa de implementación, dado que es sumamente complicado y difícil realizar modificaciones en una red de telecomunicaciones que se encuentra en fase de producción.

Por lo que se hace necesario, que se creen sub fases de implementación que no comprometan el funcionamiento de la red, ya que como se ha mencionado, es una red en producción, y se mantendrá así mientras se realicen los trabajos de implementación.

Las etapas de implementación se observan en la Figura 33.

Figura 33. Pasos de la etapa de implementación (elaboración propia)

Plan de Implementación

En esta etapa, es importante crear un plan para integrar gradualmente la propuesta realizada previamente, es decir, realizar una transición suave entre tecnologías y los nuevos paradigmas de red asociados a la propuesta. Este plan debe contemplar la continuidad del servicio de red o en el peor de los escenarios, la menor afectación a los usuarios de la red, por esto, se hace crítico el seccionamiento de la propuesta.

Este plan detallará lugares donde se realizarán los trabajos de implementación y las personas que se encargarán de ella, así como de los pasos a seguir en caso de contingencia, es decir, en caso de que haya errores ya sean humanos o técnicos y que la implementación no sea exitosa.

Finalmente, el plan debe ser revisado y autorizado por los administradores de la red, dado que ellos junto con el equipo encargado de la implementación, serán los responsables de auditar cada una de las etapas del plan, y dar a conocer si existe algún inconveniente para continuar con la implementación.

Puesta en pre-producción

Es el paso más importante mediante el cual se evalúa el funcionamiento y rendimiento de los escenarios propuestos que son análogos a los que ya se encuentran en producción en la red de datos de la ESIMEZ. De esta manera es posible verificar, el funcionamiento, o no funcionamiento, la compatibilidad o la incompatibilidad de esta tecnología con los escenarios reales. Aquí se analizan diferentes parámetros de rendimiento propios de una red de telecomunicaciones y se detectan algunas fallas con el fin de asegurar un servicio de alta disponibilidad en la red de datos de la ESIMEZ para su posterior puesta en producción.

Puesta en producción

Cuando ya se ha comprobado el correcto funcionamiento de cada uno de los escenarios, así como de su estabilidad ante escenarios de tráfico real, es necesaria la revisión y autorización de los administradores de la red, quienes deben estar en todo momento atentos a cada cambio que se presente en la red y no tengan inconvenientes en cuanto a su implementación.

De esta forma, será posible hacer cambios en una red que se encuentra en producción sin tener caídas en el servicio, poniendo gradualmente en producción cada una de las partes del escenario propuesto formalmente, para que impacten de forma positiva y directamente al usuario final.

Para finalizar, si las conclusiones obtenidas al haber puesto en producción los escenarios previstos son positivas, y con el fin de agilizar dicha implementación para los administradores de red, entonces: será posible replicar esta metodología en cada uno de los segmentos que conforman la capa de Acceso de la Red Institucional del Instituto Politécnico Nacional.

Documentación

Debido a que no existe documentación actualizada sobre la estructura tanto de la Red Institucional del Instituto Politécnico Nacional, así como de la ESIMEZ, y añadiendo que el paradigma de NFV se encuentra en sus etapas intermedias (lo que brinda posibilidades a la comunidad académica e investigadora de desarrollar nuevas arquitecturas, sistemas, aplicaciones, modelos de gestión, implementación y evaluación); se recomienda documentar todo lo realizado en cada una de las etapas de la metodología, esto significa, que los procedimientos que se llevaron a cabo durante el diagnóstico de la red, el análisis de los escenarios propuestos para implementación, el análisis de las especificaciones de los servidores x86, el análisis de las normas y estándares, las conFiguraciones de los equipos, etcétera.; pueden ser mejorados y superados en el futuro.

CAPÍTULO IV: DESARROLLO DE LA PROPUESTA

Topología física y direccionamiento lógico de la Red Nacional IPN MX

Nota: Por cuestiones de confidencialidad y de seguridad de la red IPN MX, se manejarán direcciones IP distintitas para preservar las ubicaciones lógicas de los servidores y servicios de red de misión crítica.

Como se menciona en los objetivos particulares, primero es necesario analizar y definir la topología física y el direccionamiento lógico de la red Institucional del Instituto Politécnico Nacional en toda su extensión.

Para ello se hace uso de la herramienta "Nmap", misma que es una aplicación de código abierto que sirve para efectuar el rastreo de puertos escrita originalmente por Gordon Lyon y cuyo desarrollo se encuentra hoy a cargo de una comunidad. Nmap posee varias funciones para escanear redes de computadoras, incluyendo detección de equipos, servicios y sistemas operativos.

Para éste proyecto, únicamente se utilizaron las herramientas de detección de equipos, lo cual incluye su direccionamiento lógico, como se verá a continuación en los siguientes diagramas.

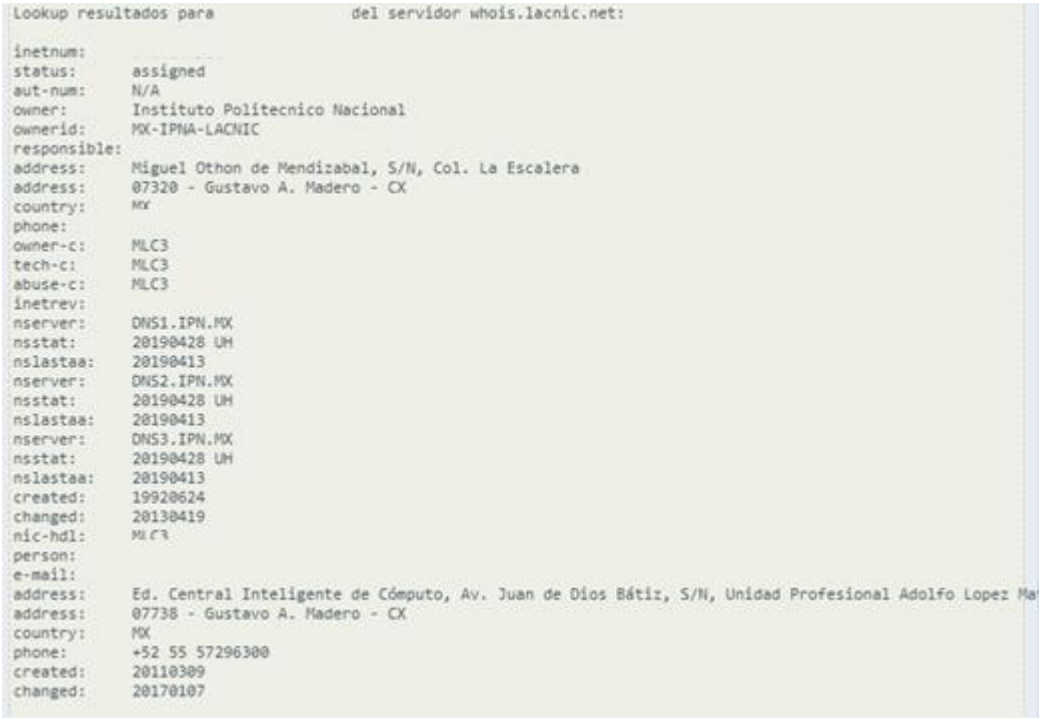
En la Figura 34, se muestra un ejemplo de la ejecución de Nmap para la dirección de "loopback":

```
notwist@notwist:~$ nmap localhost

Starting Nmap 4.20 ( http://insecure.org ) at 2007-04-02 15:50 CEST
Interesting ports on localhost (127.0.0.1):
Not shown: 1691 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
631/tcp  open  ipp
3306/tcp open  mysql

Nmap finished: 1 IP address (1 host up) scanned in 0.213 seconds
notwist@notwist:~$
```

Figura 34. Nmap ejecutándose en modo texto sobre un sistema operativo Linux.

La Red Institucional del Instituto Politécnico Nacional maneja la dirección de red: **128.168/16**, misma que fue asignada por la IANA (Internet Assigned Numbers Authority) a través de LACNIC (Registro de Direcciones de Internet de América Latina y Caribe, por sus siglas en inglés).

Al realizar una petición “whois.lacnic.net” acerca de la dirección IP 128.168/16, se obtienen los resultados, que se muestran en la Figura 35:



```
Lookup resultados para              del servidor whois.lacnic.net:

inetnum:
status:      assigned
aut-num:     N/A
owner:       Instituto Politecnico Nacional
ownerid:     MX-IPNA-LACNIC
responsible:
address:     Miguel Othon de Mendizabal, S/N, Col. La Escalera
address:     07320 - Gustavo A. Madero - CX
country:     MX
phone:
owner-c:     MLC3
tech-c:      MLC3
abuse-c:     MLC3
inetrev:
nserver:     DNS1.IPN.MX
nsstat:      20190428 UH
nslastaa:    20190413
nserver:     DNS2.IPN.MX
nsstat:      20190428 UH
nslastaa:    20190413
nserver:     DNS3.IPN.MX
nsstat:      20190428 UH
nslastaa:    20190413
created:     19920624
changed:     20130419
nic-hdl:     MLC3
person:
e-mail:
address:     Ed. Central Inteligente de Cómputo, Av. Juan de Dios Bátiz, S/N, Unidad Profesional Adolfo Lopez Ma
address:     07738 - Gustavo A. Madero - CX
country:     MX
phone:       +52 55 57296300
created:     20110309
changed:     20170107
```

Figura 35. Información que proporciona LACNIC para la dirección de red asignada al IPN MX.

De acuerdo a la Figura 35, como primera aproximación se observa que la red del IPN a nivel nacional tiene 16 bits para realizar técnicas de segmentación de red (Subnetting o VLSM). Además, se informa que la red IPN cuenta con tres DNS (Domain Name System).

Por lo tanto, con una primera búsqueda en Internet, se obtienen los datos mencionados en el párrafo anterior, e incluso información de contacto acerca de dicha red, y a dónde llega el enlace dedicado de internet para proveer de conectividad a toda la red del Instituto Politécnico Nacional.

A continuación, se realiza un escaneo, para ello se utilizó un script dada la facilidad que provee el entorno Linux, con ello se evita el tener que ejecutar Nmap “n” número de veces para cada “m” número de segmentos que componen la Red Nacional del Instituto Politécnico Nacional.

Los resultados del escaneo se observan en la tabla que se encuentra en el ANEXO I, y de acuerdo a lo que se planteó al inicio de este proyecto, se parte de lo general hacia lo específico, por lo tanto, se toman en cuenta las direcciones IP marcadas en color rojo, dichas direcciones son las pertenecientes a la Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Zacatenco que, en este punto, es nuestro caso de estudio.

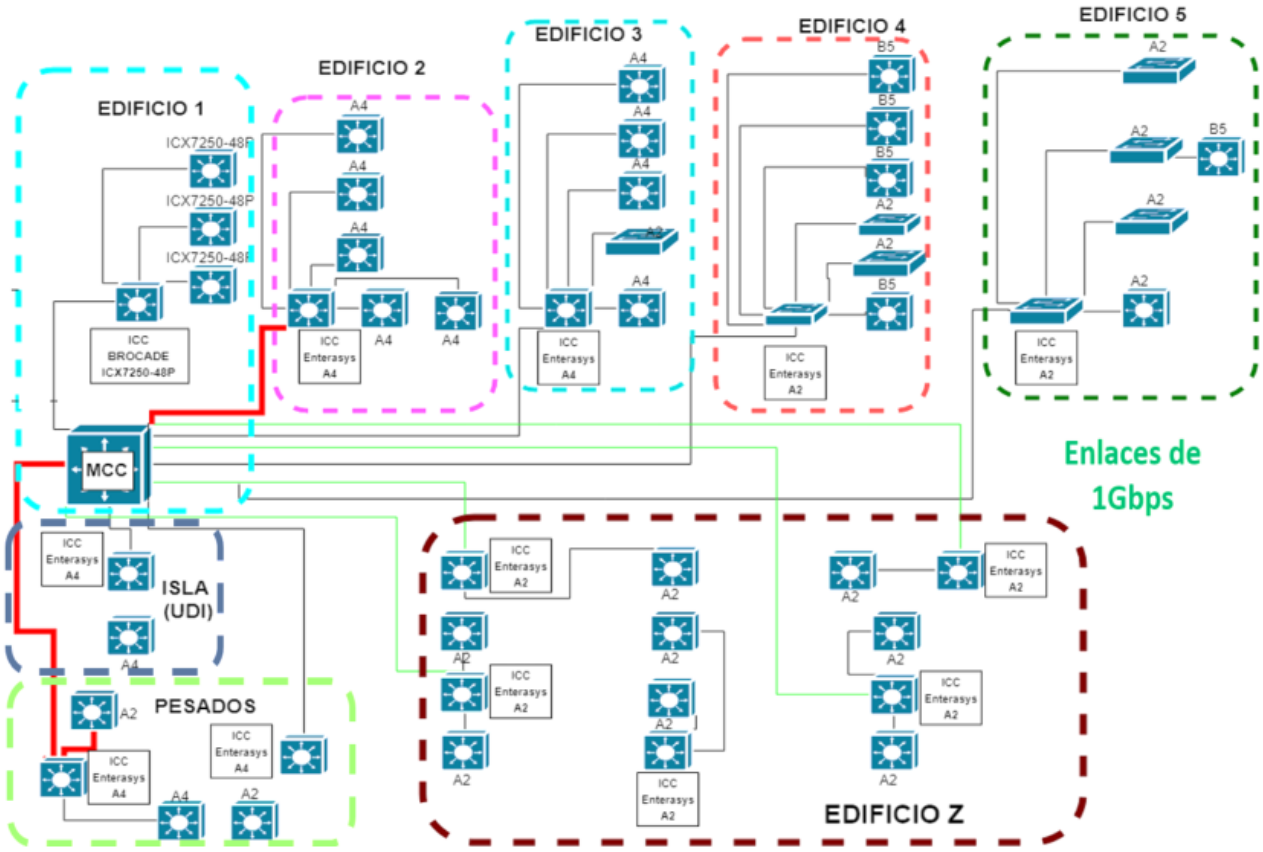
Revisando a detalle el ANEXO I, se encuentra que el direccionamiento es para CIDR=24, por lo tanto, las direcciones de red utilizables se encuentran en el rango de 128.168.1.0/24 hasta 128.168.254.0/24.

Profundizando en el anexo antes mencionado, las direcciones de red que forman parte de éste análisis son las que se muestran en la Figura 36:

128.168.31.0/24	**ESIMEZ**
128.168.32.0/24	**ESIMEZ**
128.168.33.0/24	**ESIMEZ**
128.168.34.0/24	**ESIMEZ**
128.168.35.0/24	**ESIMEZ**
128.168.36.0/24	**ESIMEZ**
128.168.37.0/24	**ESIMEZ**

Figura 36. Segmentos de red asignados por la Dirección de Cómputo y Comunicaciones a ESIME Zacatenco

De acuerdo a los resultados obtenidos producto de la observación durante el recorrido realizado por las instalaciones de ESIMEZ y al posterior análisis de los mismos, se obtuvo la Figura 34:



28

Figura 37. Diagrama General de la Red de Datos de ESIMEZ (diciembre 2018).[29]

Partiendo de la información mostrada en la Figura 33, y apoyándose en el archivo en bruto, producto del escaneo con Nmap, es posible reconstruir la topología física de la red ESIMEZ. Es una actividad que demanda tiempo, y el resultado se muestra en la Figura 38.

[28] Obtenida de la tesis de Maestría "Propuesta de despliegue del protocolo IPv6 en la red de Telecomunicaciones de la ESIME Zacatenco" del M.C. Octavio Eduardo García Barragán.

[29] ICC= Punto de conexión intermedia. MCC= Punto de conexión principal

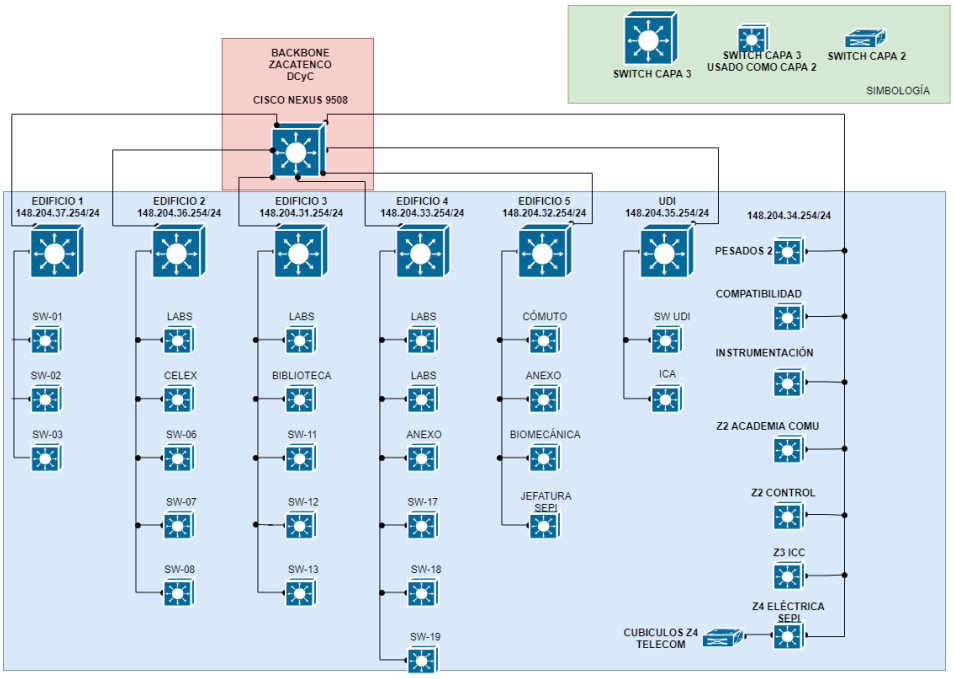


Figura 38. Topología física tipo estrella de la capa de distribución de la red ESIMEZ (abril 2019)

En la Figura 38, queda definida la topología física de la capa de distribución de la ESIMEZ, por supuesto, que debajo de ésta capa se encuentra la capa de acceso, misma que no será detallada en este momento y no es del alcance de este trabajo. El direccionamiento IP podrá consultarse en los ANEXOS II, III, IV, V, VI, VII, y VIII respectivamente, en cada una se detalla el equipo de red conectado a los respectivos dispositivos de "switching" de capa 2.

Antes del despliegue de OpenStack

El despliegue de ésta tecnología conlleva el uso de nuevas herramientas como las que se mostrarán a continuación, sin embargo, no sólo se trata de simplemente adoptar las nuevas herramientas, sino que habría que plantearse la adopción paulatina de la metodología DevOps (Desarrollo y Operaciones, por su acrónimo en inglés).

DevOps

Es un acrónico inglés de development (desarrollo) y operations (operaciones o sistemas), que se refiere a una metodología de trabajo que se centra en la comunicación, colaboración e integración entre desarrolladores de software y los profesionales en las tecnologías de la información.

Los principios fundamentales de esta metodología son:

- Maneja tu infraestructura como tu software
- Utiliza software de control de versiones para las conFiguraciones y especificaciones
- Todo debe ser legible
- Automatizar la conFiguración de nuestro sistema

Herramientas que se pueden usar:

- Creación de entornos de desarrollo ligeros y portables: Vagrant
- Software de control de versiones: Git, Subversion, Bazaar.
- Automatización de la conFiguración de sistemas: Ansible, Pupplet, Chef.
- Integración continua: Jenkins.

Git

Es un software de control de versiones distribuido diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Git es uno de los sistemas de control de versiones más utilizado hoy en día por su rapidez y la gran cantidad de funcionalidades que ofrece.

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. GitHub permite alojar de forma gratuita un número indeterminado de repositorios públicos y es ampliamente utilizado hoy en día por grandes proyectos de software, así como proyectos desarrollados por un solo individuo.

La mayoría de los scripts que se usarán para implantar la infraestructura de prueba de OpenStack están alojados en repositorios en GitHub, por lo tanto, se tendrá que clonarlos para descargarlos a nuestros equipos.

Vagrant

Es una aplicación libre desarrollada en Ruby que nos permite crear y personalizar entornos de desarrollo livianos, reproducibles y portables. Vagrant nos permite automatizar la creación y gestión de máquinas virtuales. Las máquinas virtuales creadas por Vagrant se

pueden ejecutar con distintos gestores de máquinas virtuales (oficialmente VirtualBox, VmWare e Hyper-V), en este proyecto se utilizarán máquinas virtuales en VirtualBox.

El objetivo de Vagrant es aproximar los entornos de desarrollo y producción, de esta manera el desarrollador tiene a su disposición una forma muy sencilla de desplegar una infraestructura similar a la que se va a tener en entornos de producción. A los administradores de sistemas les facilita la creación de infraestructuras de prueba y desarrollo.

Ansible

De forma general, se le llama gestores de conFiguración a las aplicaciones software que nos permiten la conFiguración y el mantenimiento de un sistema informático. En el caso específico de sistemas operativos se trata de gestionar automáticamente todas las tareas de conFiguración de un sistema, automatizando la edición de ficheros de conFiguración del mismo, creación de usuarios y autenticación, de forma que se pueda hacer de forma automática y masiva.

En este caso, se decantó por Ansible, que es un sistema de orquestación escrito en Python, que nos permite automatizar la conFiguración de una máquina. El principal motivo de esta elección es que es posiblemente el sistema más sencillo de aprender y fácil de instalar.

Caracterización de la infraestructura mínima OpenStack

Redes

Tipos

- Pueden utilizarse redes básicas Gigabit Ethernet
- Se pueden utilizan combinaciones de varias interfaces físicas Gigabit Ethernet en modo “bonding” para eliminar cuellos de botella.
- En caso necesario, ya sea por el tamaño del Cloud, o por la utilización de modos de almacenamiento distribuido, se puede usar por utilizar redes 10 GBE
- De manera alternativa a la utilización de redes 10 GbE, principalmente en la red de almacenamiento, se puede optar por utilizar “Fibre Channel”.
- También está soportado “Infiniband”, aconsejable para conseguir altas prestaciones.

Dispositivos de interconexión

La opción más sencilla y asequible es utilizar switches convencionales que simplemente conecten los nodos del cloud. Toda la conFiguración de redes virtuales puede hacerse por software.

Otra opción es utilizar dispositivos de mayor potencia de procesamiento de red, que trabajen de forma coordinada con el software y controladores específicos para conseguir mayor rendimiento y control como en el caso de algunos dispositivos Cisco o Juniper.

Si se quiere aumentar el tamaño del Cloud, se debe estudiar la posibilidad de realizar un cambio íntegro de toda la arquitectura de red, hacia "spine-leaf".

Servidores

- El nodo controlador no necesita características especiales.
- Para los nodos de cómputo será necesario utilizar procesadores con extensiones de virtualización, ya sean Intel o AMD.
- La cantidad de RAM de los nodos de computación condicionará el número de instancias a ejecutar en cada nodo.
- En los nodos de cómputo, es necesario disponer de procesadores multinúcleo y en el caso de Intel, de la tecnología HT (Hyper-Threading), por ejemplo:
    - AMD Opterom Interlagos o Abu-Dhabi como mínimo
    - Interl Xeon Ivy Bridge o Haswell como mínimo
- El nodo de red conecta todas las redes virtuales con el exterior y por tanto debe tener interfaces de red ajustadas al tamaño del Cloud para no provocar cuellos de botella

Almacenamiento

Unidades

- SATA HD, SAS HD y SSD:
    - SATA HD (7200 o 1000 rpm): Hasta 4 TB de almacenamiento, bus de 3 GBps y hasta 200 IOPS.
    - SAS HD (10000 o 15000 rpm): Hasta 900 GB de almacenamiento, bus de 6 Gbps y hasta 500 IOPS.

    - SSD: Ya hay unidades SSD en el mercado de 1 o 2 TB, bus de 3 0 6 GBps y hasta 100 000 IOPS.
- A igual tamaño, los SSD son relativamente más caros, aunque pronto igualarán los precios de los discos rígidos.
- Las tasas de transferencia secuencial son del mismo orden, pero las aleatorias son mucho mayores en SSD
- Enterprise SAS o SATA son más fiables todavía
- SSD tienen un menor consumo energético

Equipos específicos

- Es habitual en los centros de datos utilizar equipos específicos para el almacenamiento.
- Una gran parte de los fabricantes de estos equipos (EMC, NetApp, etcétera.) pertenecen a la OpenStack Foundation y por tanto sus dispositivos están soportados o lo estarán en OpenStack.
- Pueden utilizarse estos dispositivos para almacenamiento de volúmenes con Cinder y/o almacenamiento distribuido o no de los sistemas de ficheros de las instancias.
- La mayor parte de estos dispositivos no incluyen soporte de almacenamiento de objetos directamente.

Servidores

- Como alternativa a lo anterior, se puede optar por utilizar servidores de almacenamiento en los que se instala algún sistema operativo y se conFigura el almacenamiento por software.
- En estos casos se suelen utilizar servidores con chasis de varias bahías para discos duros extraíbles en caliente (hot swap) y un backplane que permita conectarlos todos a la placa base.
- También se puede optar por utilizar controladoras RAID hardware o conFigurar el RAID por software (en este caso, es imprescindible que la controladora de disco soporte el modo JBOD).

Implementación de la infraestructura de prueba OpenStack

Como resultado de la recopilación bibliográfica, se encontraron formas diferentes de instalar OpenStack en un entorno de pruebas en el equipo “Workstation”.

Dentro de todas las opciones, las principales son las siguientes:

- Devstack
- RDO
- OpenStack-Ansible

En la tabla 2, se muestra el resumen de las tres opciones anteriormente mencionadas:

Tabla 2. Comparativa entre las diferentes opciones para el despliegue de OpenStack.

	DevStack	RDO	OpenStack-Ansible
Instalación	En máquina física o en máquina virtual	En máquina física	En 4 máquinas virtuales
Versión OpenStack	Rocky	Rocky	Rocky
Servicios OpenStack	Se pueden conFigurar de una manera fácil los servicios que se requieren ofrecer	Todos	Todos
Vagrant-Ansible	Para la instalación en una máquina virtual existen repositorios donde encontramos las recetas	No	Si
Ventajas	La instalación y conFiguración es muy sencilla. Se pueden instalar distintas versiones.	Fácil de instalar y muy depurado. Buen rendimiento. Puede ser perfecto para un entorno de pruebas.	Entorno de pruebas muy cercano al real, con nodo de red, nodo de almacenamiento, etcétera. Es muy adecuado para luego cambiarlo a un entorno real
Inconvenientes	No es totalmente real porque el controlador tiene todos los componentes. La instalación sobre una máquina virtual ofrece menos rendimiento, lo ideal es instalarlo sobre una máquina física.	No es totalmente real porque el controlador tiene todos los componentes (incluso el nodo de cómputo). Es difícil modificarlo para hacerlo funcionar en un entorno real.	Al utilizarse sobre máquinas virtuales tiene peor rendimiento y es más exigente en cuanto a requisitos de hardware.

DevStack

Es un conjunto de script bash que nos permiten instalar OpenStack de forma automática. Se tienen varias formas de realizar la instalación:

- En una máquina física
- En una máquina virtual

La conFiguración de los servicios de OpenStack que se van a instalar se define en un fichero de conFiguración de una forma muy sencilla. La instalación se realiza en un único nodo (aunque se puede realizar en multi-nodo), y se puede seleccionar la versión de OpenStack que se va a instalar, en la actualidad es la versión Rocky. Se debe tener en cuenta que la instalación sobre una máquina virtual ofrece menos rendimiento, lo ideal es instalarlo sobre una máquina física.

Para la instalación en una máquina virtual se tienen varios repositorios GitHub, que facilitan la tarea utilizando las herramientas de Vagrant (https://github.com/xiaohanyu/vagrant-ansible-devstack)y Ansible (https://github.com/lorin/devstack-vm).

En la tabla 3, se muestran los requerimientos para su instalación:

Tabla 3. Requerimientos para instalación de OpenStack mediante DevStack

REQUERIMIENTOS CON DEVSTACK	
EN MÁQUINA FÍSICA	**EN MÁQUINA VIRTUAL**
• Equipo necesario: RAM 2 GB y CPU VT-x/AMD-v • Ubuntu 18.04 LTS instalado y actualizado • Git instalado • El usuario que hace la instalación debe tener privilegios de superusuario	• Equipo necesario: RAM 3 GB y CPU VT-x/AMD-v • Git instalado • Virtualbox instalado • Vagrant instalado • Box precise64 • Ansible instalado

RDO

Es una distribución mantenida por la comunidad de usuarios que permite ejecutar OpenStack en sistemas Red Hat y derivados (Fedora y CentOS).

Utiliza la herramienta PackStack, que mediante el uso de manifiestos de puppet y un simple fichero de conFiguración es capaz de instalar OpenStack en uno o varios nodos.

En la tabla 4 se observan los requisitos para la instalación:

Tabla 4. Requisitos de instalación de OpenStack mediante RDO.

REQUISITOS
• Software: RHEL 6.4 o equivalente • Hardware: Equipo con extensiones de virtualización por hardware en el procesador y al menos 2 GB de RAM libre

En el ANEXO IX, se presenta la forma para instalar OpenStack mediante RDO.

Ansible

Como se mencionó en la Tabla 2, ésta técnica es para su instalación en máquinas físicas, sin embargo, *dadas las limitaciones técnicas*, la infraestructura está compuesta por cuatro máquinas virtuales y la conFiguración de cada una de ellas se realiza por medio de Ansible.

Éste método es el que se selecciona para el desarrollo del proyecto dado que es el más parecido al entorno real, además de que Ansible es una herramienta muy poderosa para el despliegue y autoconFiguración de los diferentes nodos que componen OpenStack.

La tabla 5 muestra los requisitos para el despliegue de OpenStack mediante Ansible:

Tabla 5. Requisitos mínimos para instalación de OpenStack con Ansible

REQUISITOS PARA DESPLIEGUE DE OPENSTACK CON ANSIBLE
• Equipo necesario: RAM 4 GB y CPU VT-x/AMD-v • Git instalado • Virtualbox instalado • # apt-get install python-netaddr • Vagrant instalado • Box precise64 • Ansible instalado

En el ANEXO XI se muestra la forma para instalar OpenStack mediante Ansible.

Entorno de pruebas OpenStack

OpenStack puede funcionar perfectamente en equipos y dispositivos convencionales, aunque también soporta gran cantidad de dispositivos más especializados. Sin embargo, dadas nuestras limitaciones técnicas y a la nula disponibilidad de servidores físicos para pruebas en ESIMEZ, se optó por utilizar un equipo "Workstation" con las principales características:

- CPU 8 núcleos@2.4 GHz con soporte para Intel VT-x
- SSD 256 GB
- 32 GB RAM DDR3 1600 MHz tipo ECC
- 2 x NIC 1 GbE

El entorno de desarrollo consiste en la instalación de OpenStack en modo "single node" en un servidor virtual que ejecuta CentOS 7 x64 en su versión mínima, es decir, sin entorno gráfico.

El procedimiento utilizado para la instalación se encuentra en el ANEXO IX de este documento. El servidor utilizado tiene las características que se muestran en la Figura 39:

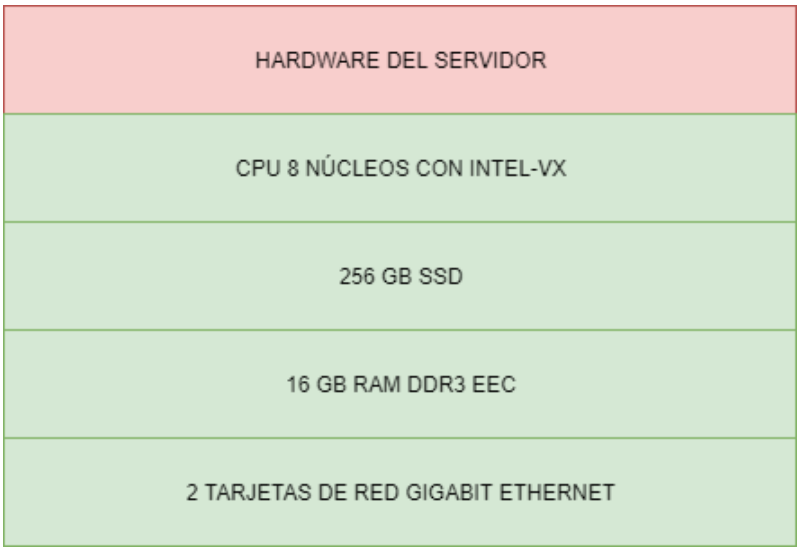


Figura 39. Hardware del servidor utilizado para virtualización

Las características del equipo virtual que ejecuta OpenStack sobre CentOS 7 en modo “single node” se muestran en la Figura 40:

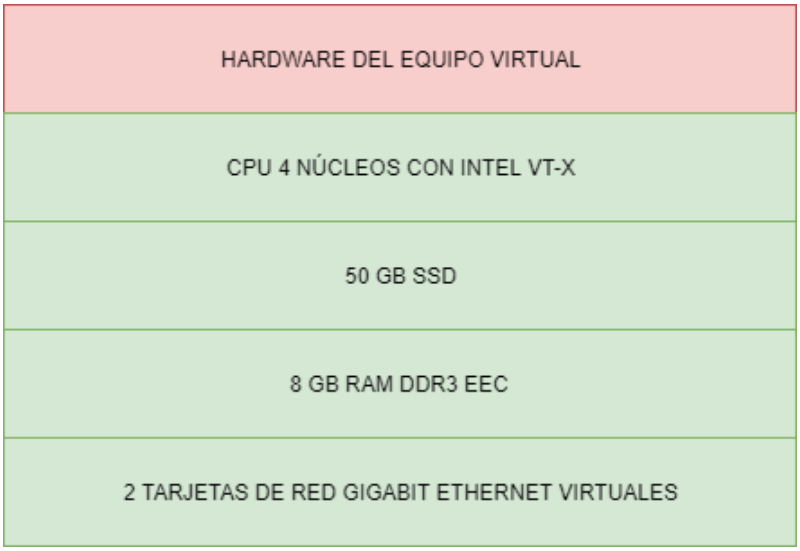


Figura 40. Características del equipo que ejecuta OpenStack sobre CentOS 7

El proyecto OpenStack está definido como una plataforma de Cloud Computing realizada con software libre para desplegar nubes públicas y privadas.

Desarrollada con el objetivo de ser fácil de implementar, masivamente escalable y con muchas prestaciones. OpenStack proporciona una solución de Infraestructura como Servicio (IaaS) a través de un conjunto de servicios interrelacionados.

OpenStack es un conjunto de componentes que pueden combinarse en función de las características y necesidades de cada caso. Por lo tanto, hay algunos componentes fundamentales y otros opcionales.

OpenStack tiene una gran flexibilidad y permite implementar tanto a nivel de cloud privado como público.

Cada componente de OpenStack es totalmente autónomo y funcional y utiliza el protocolo AMQP de gestión de colas para comunicarse con el resto de componentes (Rabbit) y una API web RESTful para comunicare con procesos “externos” o con los mismos usuarios.

A continuación, en la Figura 41 se observa la arquitectura de OpenStack que se utilizará para el desarrollo del proyecto:

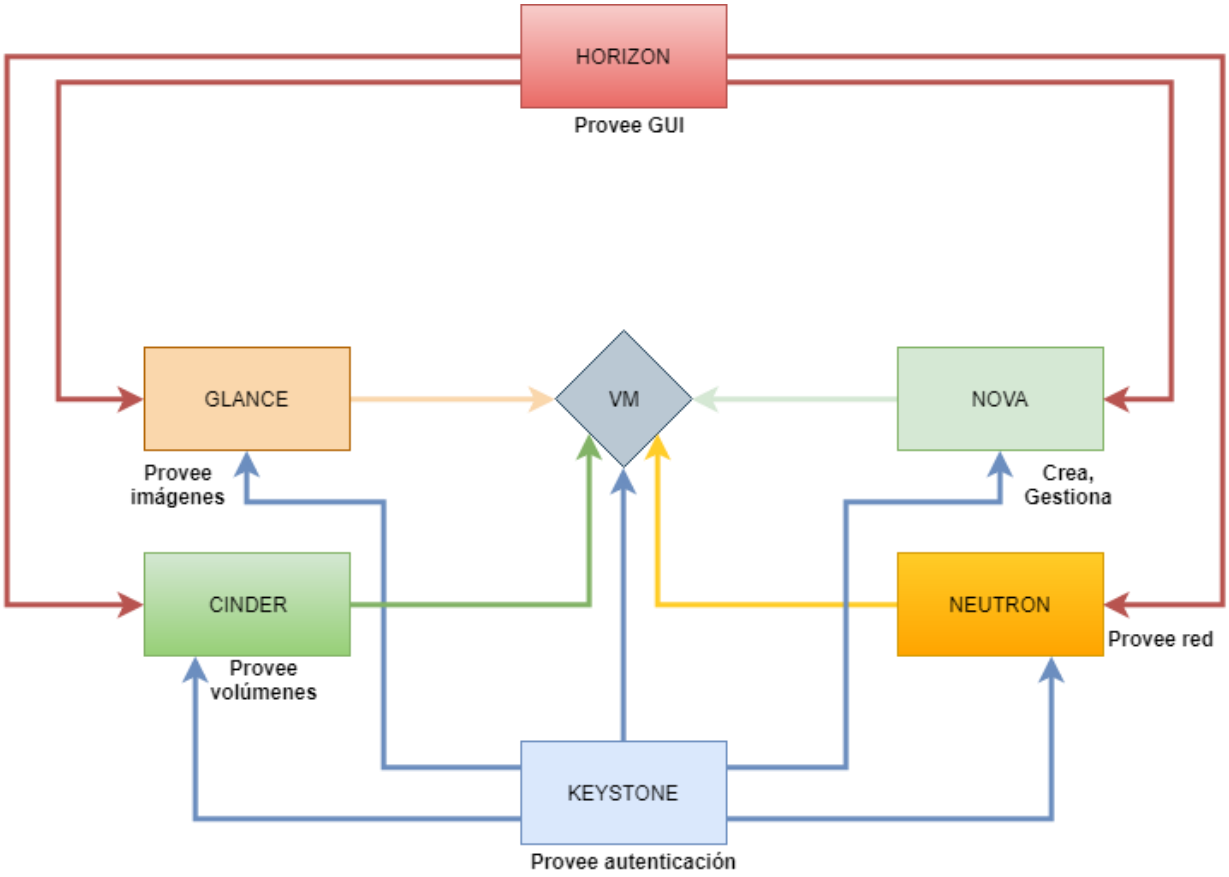


Figura 41. Interrelación de los módulos de OpenStack utilizados para el Proyecto NFV

Como se ha descrito en el párrafo anterior, el esquema de pruebas constará de un solo nodo, que contendrá a los tres nodos de un despliegue básico de OpenStack: Controlador, Cómputo y Red. Esto se observa en la Figura 42.

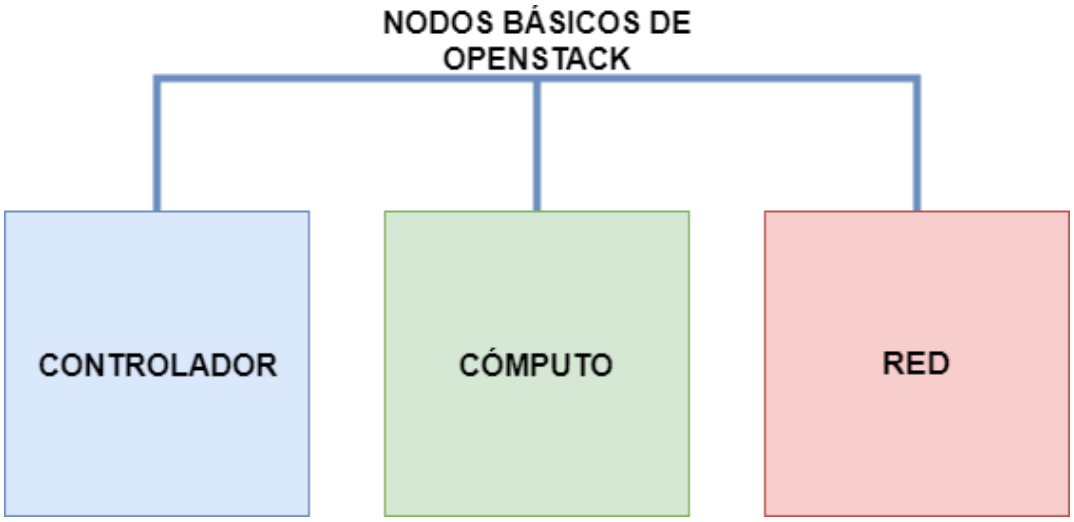


Figura 42. Nodos que componen un despliegue inicial de OpenStack

OpenStack utilizará dos redes independientes, la primera denominada la red "Management" que se trata de la red interna utilizada por los componentes para su comunicación entre ellos y la red "Provider" la cual es la encargada de proveer de acceso a internet tanto a los nodos como a las instancias.

Se añadió una tercera red, para realizar VLANs y VXLANs dentro de la red virtual, estando ésta red de forma experimental. En la Figura 43 se muestra el nodo de cómputo y el nodo de control, este último es el que proporciona la inteligencia a la red.

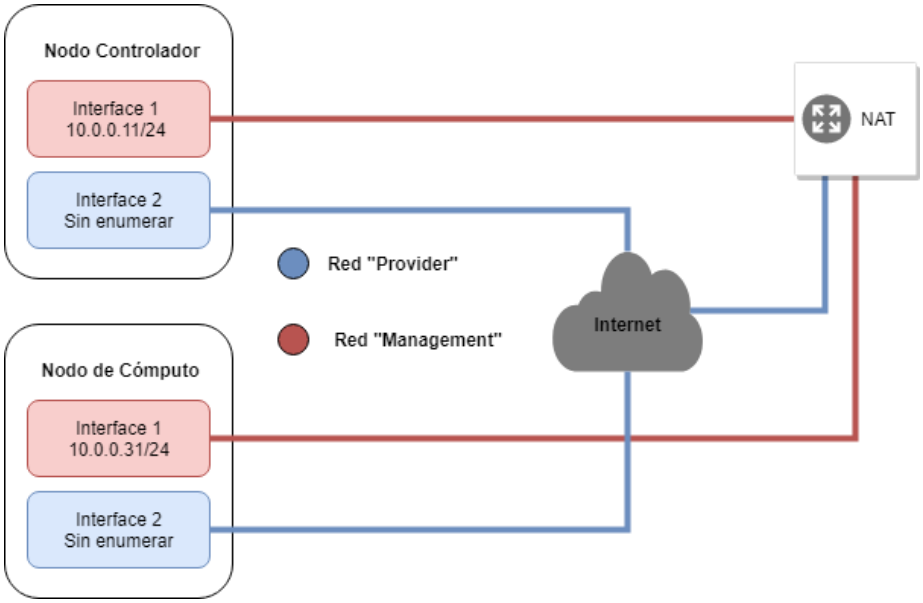


Figura 43: Principales redes dentro de OpenStack (elaboración propia)

Capacidades de servidores recomendados

Aunque la mayoría de los entornos incluyen Identidad, Servicio de imagen, Cómputo, al menos un servicio de red y el Dashboard, el servicio de almacenamiento de objetos puede operar de forma independiente. Se debe utilizar una cuenta con privilegios administrativos para conFigurar cada nodo.

En un entorno real (no virtualizado), los siguientes requisitos mínimos deben ser compatibles con un entorno de prueba de concepto con servicios centrales y varias instancias de CirrOS:

- Nodo del Controlador (Controller): 1 procesador, 4 GB de memoria RAM y 5 GB de almacenamiento
- Nodo de Cómputo (Compute): 1 procesador, 2 GB de memoria RAM y 10 GB de almacenamiento.

A medida que aumenta la cantidad de servicios y máquinas virtuales de OpenStack, también aumentan los requisitos de hardware para obtener el mejor rendimiento. Si el rendimiento se degrada después de habilitar servicios adicionales o máquinas virtuales, se debe considerar agregar recursos al entorno.

Para minimizar el desorden y proporcionar más recursos para OpenStack, se recomienda una instalación mínima de una distribución Linux.

Además, se debe instalar una versión de 64 bits (x64) de su distribución en cada nodo. Esto con el fin de que el sistema operativo sea capaz de soportar más de 4096 MB de RAM.

Una única partición de disco en cada nodo funciona para la mayoría de las instalaciones básicas. Sin embargo, se debe considerar LVM (Logical Volumen Manager) para instalaciones con servicios opcionales como Block Storage.

Para la primera instalación y como propósito de prueba, se decide construir cada host como una máquina virtual (VM).

Los principales beneficios de las máquinas virtuales son los siguientes:

- Un servidor físico puede admitir varios nodos, cada uno con casi cualquier número de interfaces de red.
- Capacidad para tomar instantáneas periódicas durante el proceso de instalación y "retroceder" a una conFiguración de trabajo en caso de un problema.

Sin embargo, las máquinas virtuales reducirán el rendimiento de sus instancias, especialmente si su hipervisor y/o procesador carecen de soporte para la aceleración de hardware de las máquinas virtuales anidadas.

Para el caso específico de ESIMEZ, se considerarán los dos recursos principales de almacenamiento, que son CPU y Memoria, para ello se toman en cuenta los aspectos listados en el ANEXO XVI.

Uso de OpenStack

Al llegar a este punto, después de la instalación de OpenStack, es necesario recordar los siguientes conceptos previos que se han venido mencionado a lo largo de este escrito:

- Token de autenticación: OpenStack utiliza Autenticación basada en Tokens. Un token es una cadena aleatoria que se obtiene tras autenticarse en un sistema (con nombre de usuario y contraseña, por ejemplo) y que permite al usuario utilizar recursos de forma segura sin la necesidad de volver a autenticarse.
- Imagen: Imagen del sistema operativo preconFigurado que se utiliza como base para crear instancias. Dentro del Cloud habrá diferentes imágenes para cada tipo de instancia que se quiera utilizar.
- Instancia: Instancia de una imagen que se crea a demanda del usuario en uno de los nodos del cloud.
- IP fija: Dirección IP con la que se crean una instancia en una red y que se utiliza para comunicación interna. La dirección IP fija no cambia durante la vida de la instancia.
- IP flotante: Dirección IP asociada a una instancia en un momento dado para poder acceder a ella desde fuera. Una IP flotante puede asignarse a otra instancia diferente

cuando se considere necesario por el administrador o las necesidades de la red así lo requieran.

- Grupo de seguridad: Reglas de firewall que controlan el acceso a las instancias.
- Par de claves SSH: Par de claves RSA pública/privada utilizadas para acceder por ssh a las instancias desde fuera del Cloud.
- Endpoint: URL completa para la utilización de una determinada API. Pueden definirse URLs diferentes para las API pública, interna y de administración.

Cómo se usa OpenStack

Creación y acceso a una instancia

Es el caso más básico, a continuación, de forma breve se enuncian los pasos a seguir para completar el objetivo.

- El usuario se autentica en Keystone y obtiene un token de sesión y la lista de los endpoints de los servicios.
- Solicita a Glance las imágenes disponibles.
- Solicita a Neutron las redes disponibles.
- Solicita a Nova que arranque una instancia de una de las imágenes disponibles, en una de las redes y que inyecte posteriormente su clave ssh pública.
- Solicita una IP flotante a Neutron y la asocia a la IP fija de la instancia creada.
- Accede por SSH a la instancia utilizando la clave privada correspondiente generada previamente.

Trabajo con Redes

Para el apartado medular de este trabajo es muy importante conocer la terminología que se va a utilizar:

- Red: Red Dominio aislado de capa 2. Sería el equivalente a una VLAN. Las redes externas solo pueden ser definidas por el administrador.
- Subred: Bloque de direcciones IPv4 que se asignan a las máquinas virtuales que se conectan a ella.
- Router: Dispositivo de capa 3 para conectar redes.
- Puerto: Puerto virtual de un switch o router.
- IP fija: Dirección IP con la que se crean una instancia en una red y que se utiliza para comunicación interna.
  La dirección IP fija no cambia durante la vida de la instancia.
- IP flotante: Dirección IP asociada a una instancia en un momento dado para poder acceder a ella desde fuera.
  Una IP flotante puede asignarse a otra instancia diferentes cuando se requiera.

Redes con Horizon

Horizon es la interfaz web que interconecta todos los elementos de OpenStack, y mediante ella se pueden realizar las siguientes acciones:

- Crear una red privada y una subred asociada
- Crear un router, conectarlo a la red externa y a la red anterior
- Crear una instancia en la nueva red

Redes con OpenStack CLI

CLI es la línea de comandos, y para utilizarla se requiere de la instalación del paquete “python-neutronclient”. Los comandos básicos para la creación de redes desde CLI se detallan en la Figura 44:

```
#Listar redes
$ nova net-list
#Listar subredes
$ nova subnet-list
#Listar routers
$ nova router-list
# Establecer puerta de enlace de un router
$ neutron router-gateway-set ROUTER_ID EXTERNAL_NETWORK_ID
# Conectar un router a una red privada
$ neutron router-interface-add ROUTER_ID SUBNET_ID
```

Figura 44. Comandos de OpenStack CLI para la creación de redes

Conceptos avanzados: Despliegue automatizado de escenarios de red

Esto requiere conocimientos avanzados en Redes Definidas por Software, así como de altas habilidades en el manejo del lenguaje de automatización ANSIBLE. Se mencionó que gracias a ANSIBLE es posible automatizar el despliegue de una red mediante un script en OpenStack.

Escenario 1

En el ANEXO XV se muestra el script escrito en ANSIBLE que permite el despliegue automático de la red mostrada en la Figura 45, en cuestión de segundos o minutos, dependiendo de la potencia de nuestro servidor.

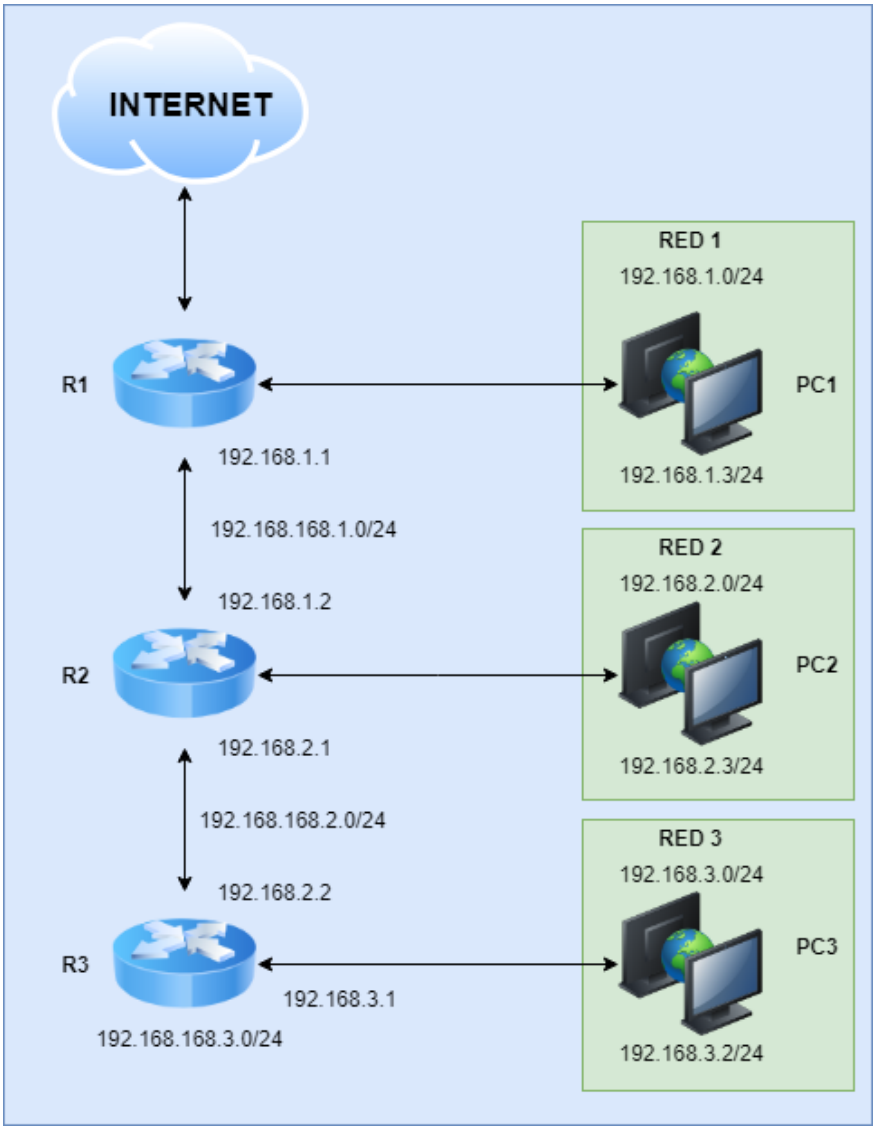


Figura 45. Red de prueba para despliegue automatizado (ESCENARIO 1)

Escenario 2

Se utiliza un script escrito en ANSIBLE (basado en el script del anexo XV) que permite el despliegue automático de la red mostrada en la Figura 46.

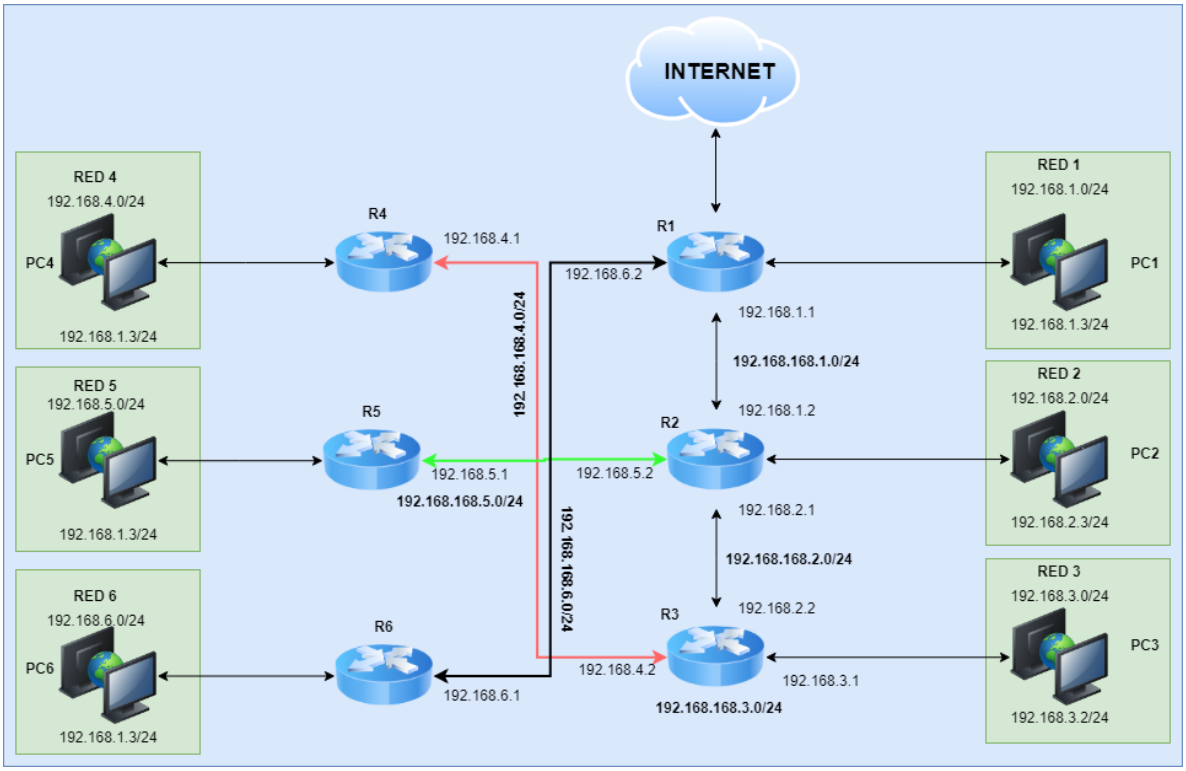


Figura 46. Se plantea el Escenario 2 para despliegue automático

El escenario plantea un poco más de complejidad respecto del escenario 1, además pone a prueba el poder de cómputo del servidor que ejecuta Neutron. Consta de 6 subredes con salidas a Internet, mismas que se segmentan en subredes de 254 hosts virtuales cada una.

En este escenario no se añade redundancia, ni balanceo de carga.

Escenario 3

Se utiliza un script escrito en ANSIBLE (basado en el script del anexo XV) que permite el despliegue automático de la red mostrada en la Figura 47. Cabe destacar que el escenario 4 es similar al escenario actual de la ESIME Zacatenco, mismo que podemos observar en la Figura 38 del CAPÍTULO IV.

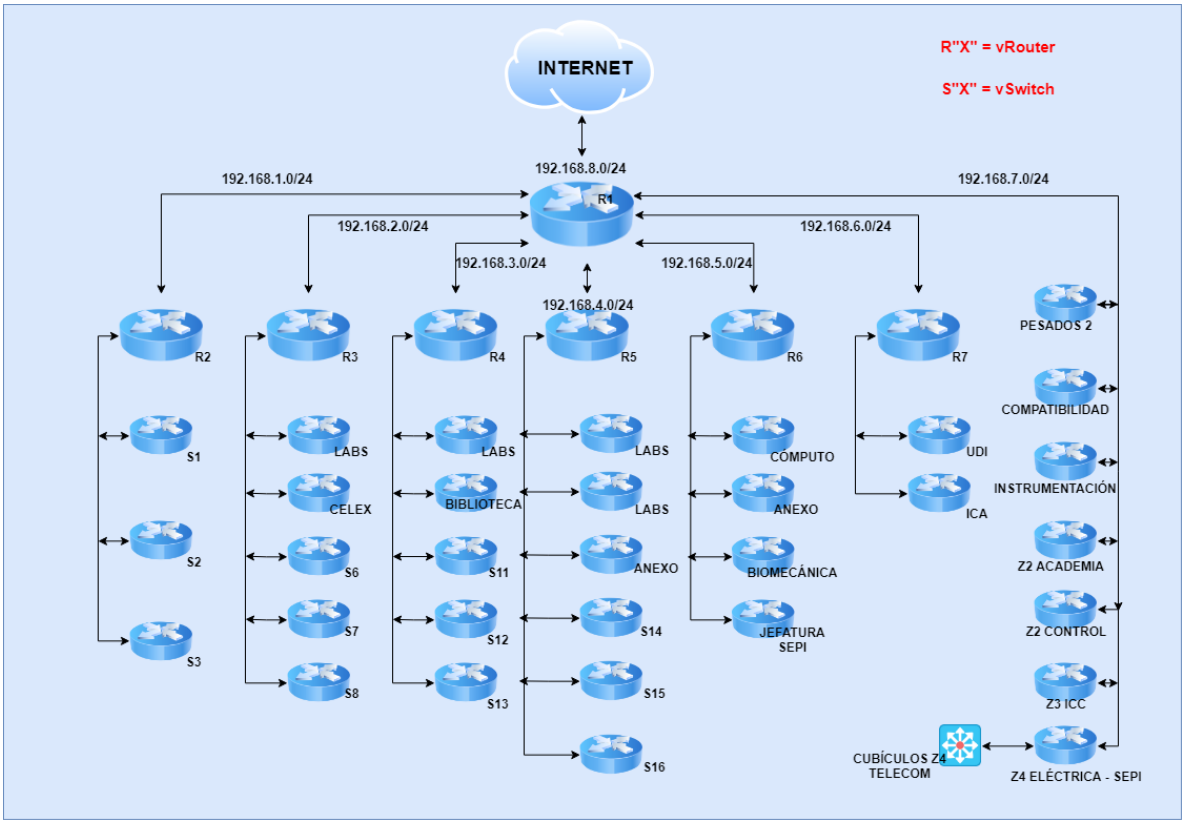


Figura 47. Escenario 3: Red semejante a la que actualmente opera en la ESIME Zacatenco

El escenario 3 representa la capa de distribución que actualmente se encuentra instalada en la ESIME Zacatenco. Los escenarios 1 y 2 son de prueba, éste es el primer escenario similar a uno que se encuentra en producción que será desplegado de forma automática mediante un script en OpenStack.

Escenario 4

Se utiliza un script escrito en ANSIBLE (basado en el script del anexo XV) que permite el despliegue automático de la propuesta final de red virtualizada para la ESIME Zacatenco utilizando OpenStack.

Ésta propuesta interconecta todos los switches multicapa en una topología de anillo, incrementando con esto la redundancia y obteniendo una alta disponibilidad. Además, se incorpora un balanceador de carga, de tal forma que cualquier host de la red de la ESIME Zacatenco podrá obtener la mejor ruta rumbo hacia el backbone de la red Politécnica y hacia Internet.

La red con alta disponibilidad y balanceo de carga que se propone, se muestra en la Figura 48.

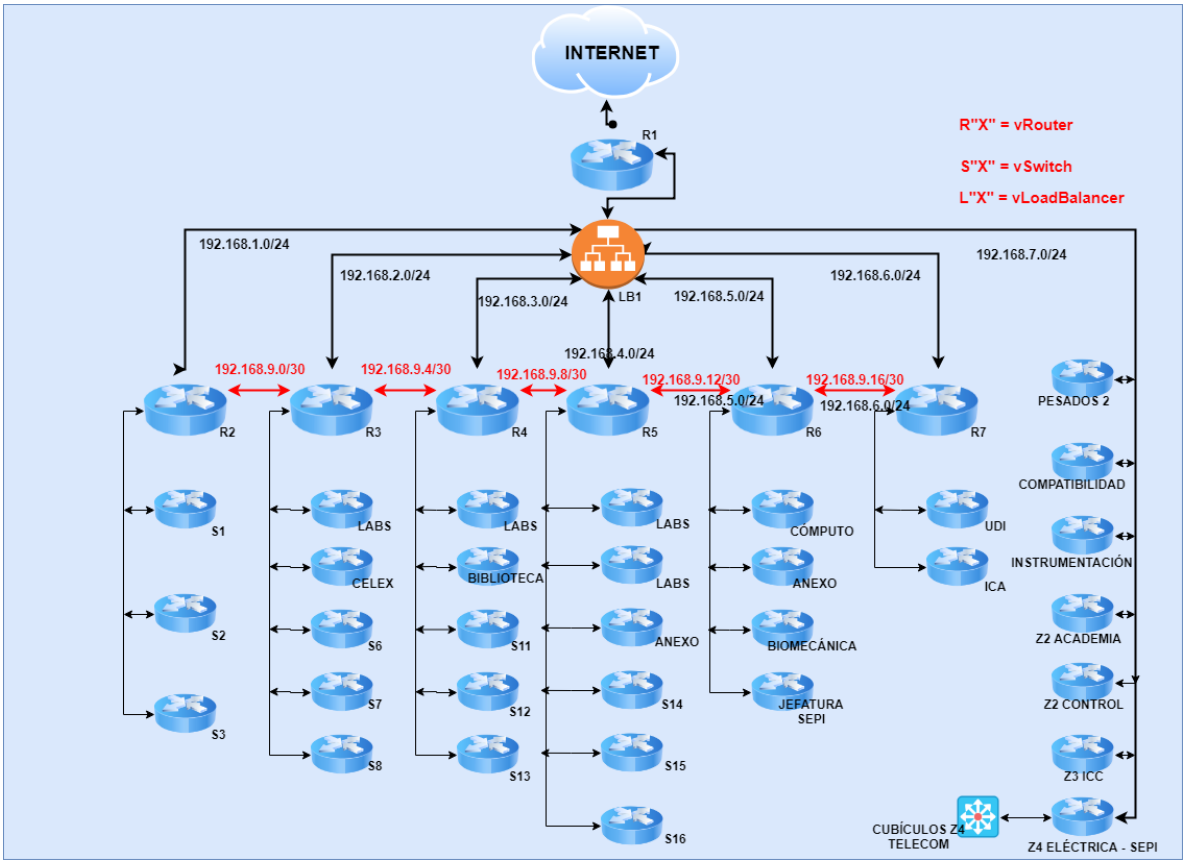


Figura 48. Escenario 4: Red con alta disponibilidad y balanceo de carga

Escenario 5

En la Figura 49, se muestra un escenario con dos servidores físicos ejecutando OpenStack, se incorporan puentes de red (bridges) e interfaces de red Ethernet agrupadas (bonding). De forma adicional, se hace uso de VLANs para separar el tráfico de red, y al mismo tiempo limitar las tormentas de broadcast. Para conectar ambos servidores físicos, se utilizó un Switch Cisco Catalyst 2950 (utilizado como switch de capa 2 sin programar).

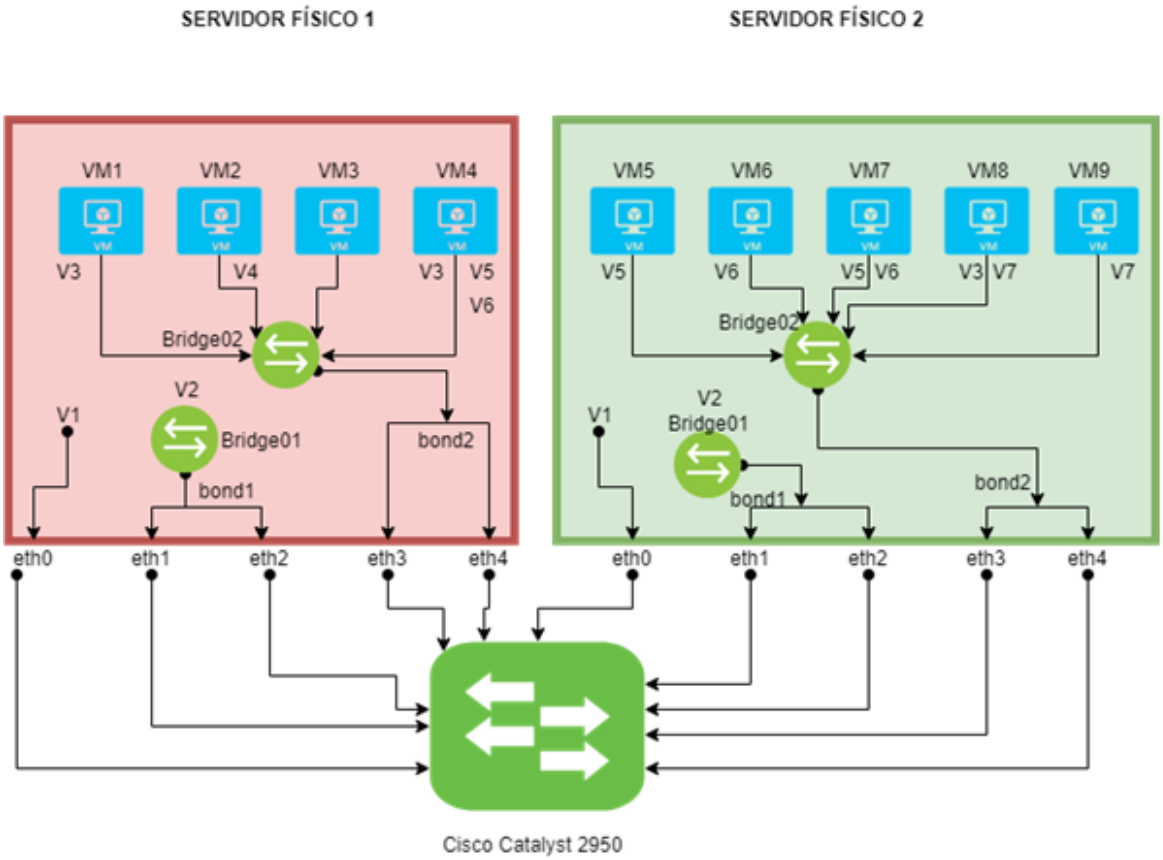


Figura 49. Escenario 5: Despliegue real básico de OpenStack

El escenario 5, se ejecuta en un entorno real utilizando dos servidores físicos (no hay hardware virtualizado), con interfaces Gigabit Ethernet. Haciendo uso del análogo de EtherChannel de Cisco, las interfaces “bonding”. Además, se hace uso de puentes de red (bridges) con switches virtuales del tipo OpenvSwitch gestionados por OpenStack.

CAPÍTULO V: PRUEBAS Y RESULTADOS

Por cada escenario propuesto, se ejecutarán pruebas de latencia, ancho de banda y jitter (Interpacket Delay Variation, IPDV). Esto permitirá evaluar el rendimiento de cada escenario y del servidor que ejecuta OpenStack con cada uno de ellos.

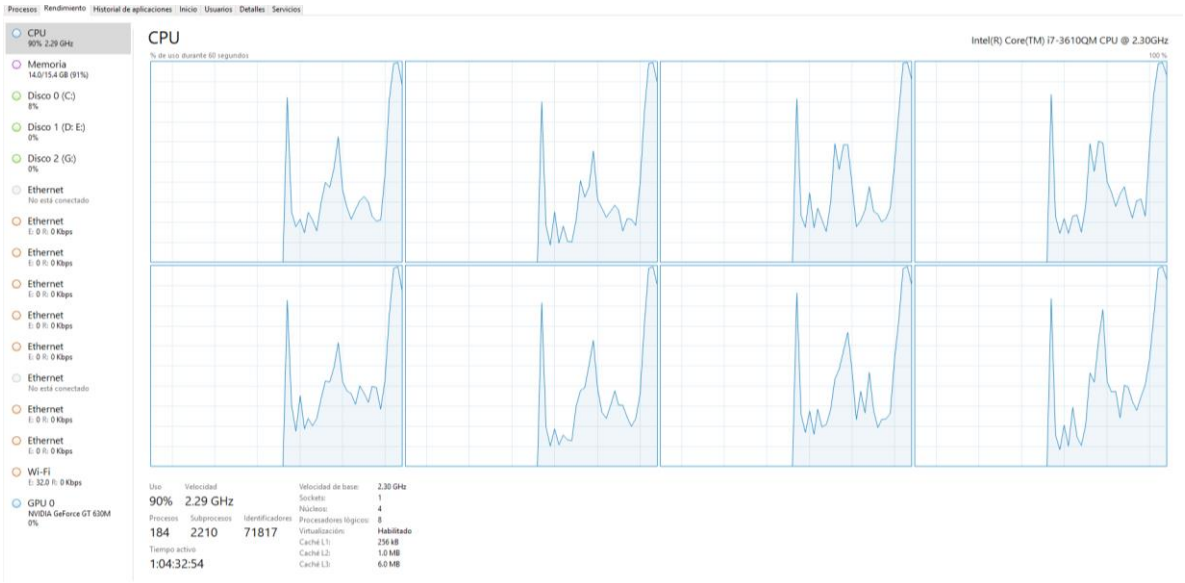


Figura 50. Se observa consumo elevado de recursos de hardware del servidor durante la ejecución de OpenStack

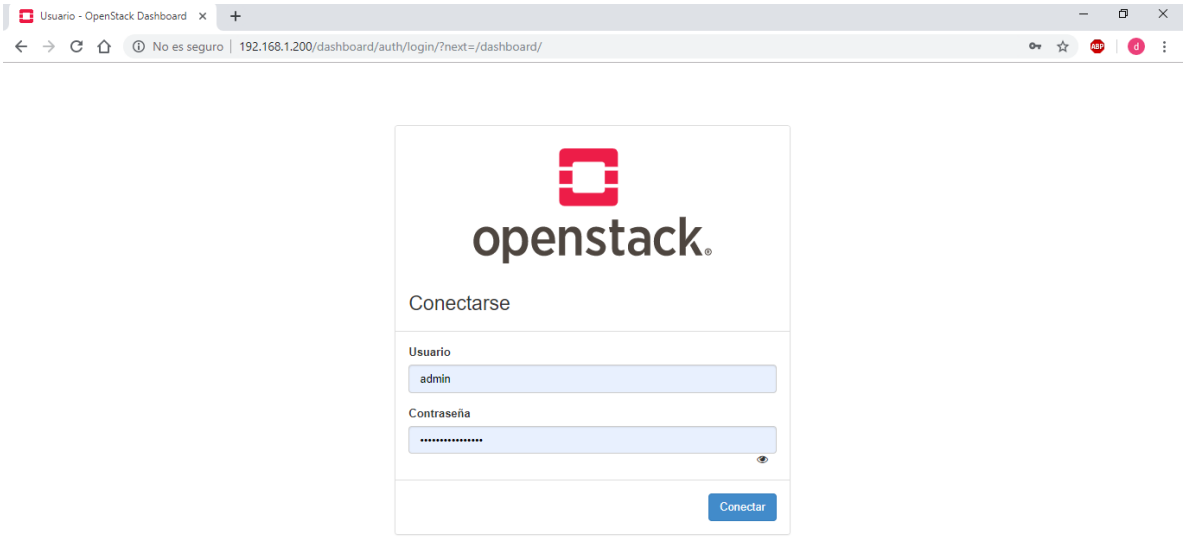


Figura 51. Dashboard Horizon de OpenStack

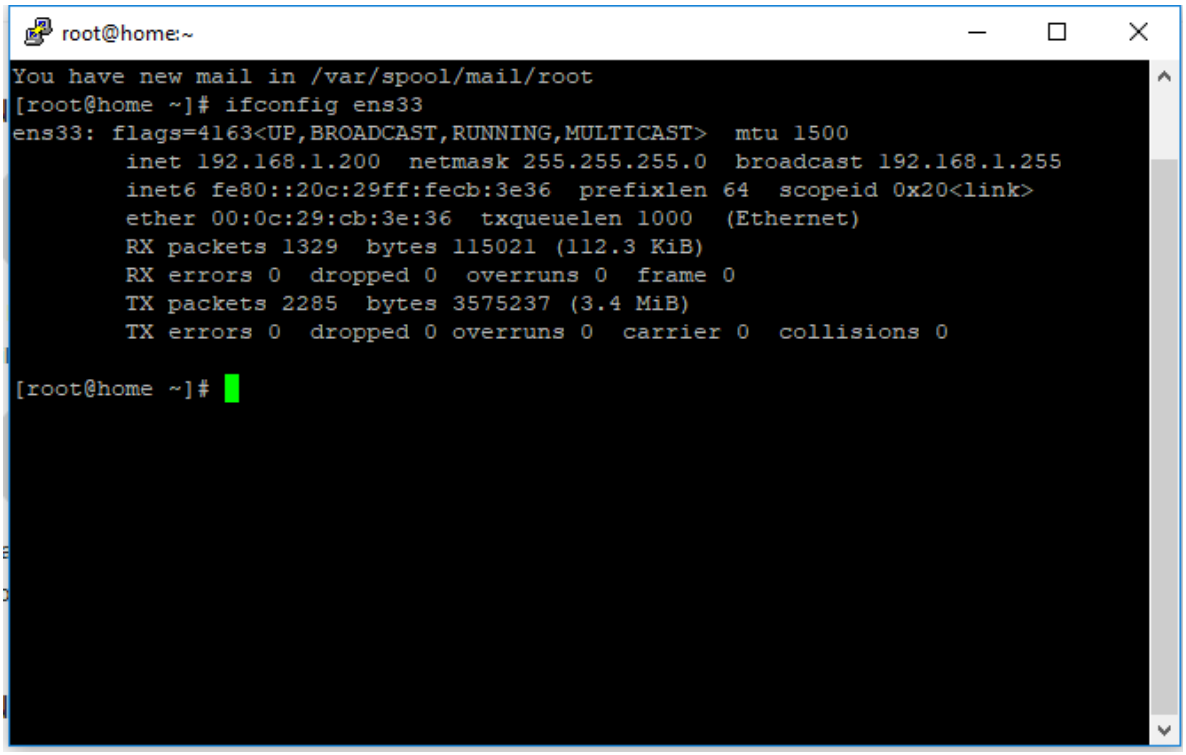


Figura 52. Acceso a OpenStack vía SSH

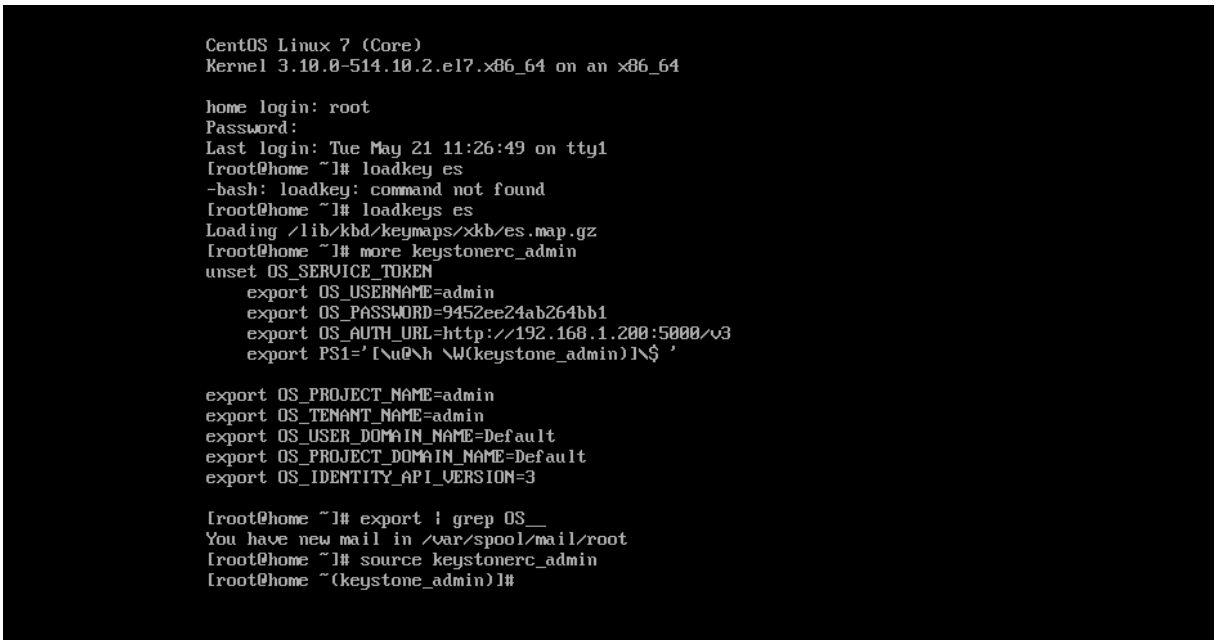


Figura 53. Autenticación e inicio de OpenStack sobre CentOS 7

Escenario 1

Latencia

En la Figura 54 se muestran resultados de latencias entre máquinas virtuales, la máquina virtual 1 (PC1) y la máquina virtual 3 (PC3), bajo un entorno virtualizado.

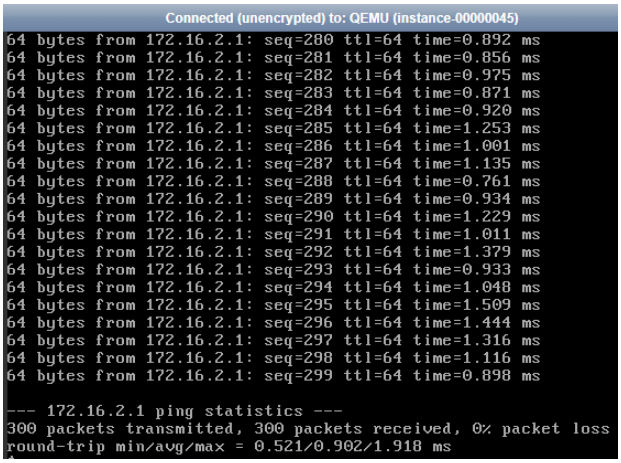


Figura 54. Pruebas de latencia para el escenario 1

En la Figura 55 se muestran resultados de latencias entre máquinas virtuales, la máquina virtual 2 (PC2) y la máquina virtual 1 (PC1), bajo un entorno virtualizado.

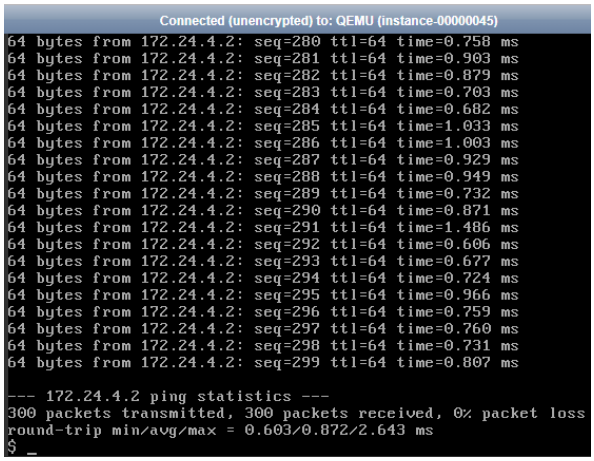


Figura 55. Pruebas de latencia para el escenario 1

Jitter

En la Figura 56 se obtiene un resultado de 0.095 ms de Jitter, con 228 datagramas perdidos, lo cual representa el 0.26% del total de datagramas enviados, la prueba es para un enlace de 100 Mbps.

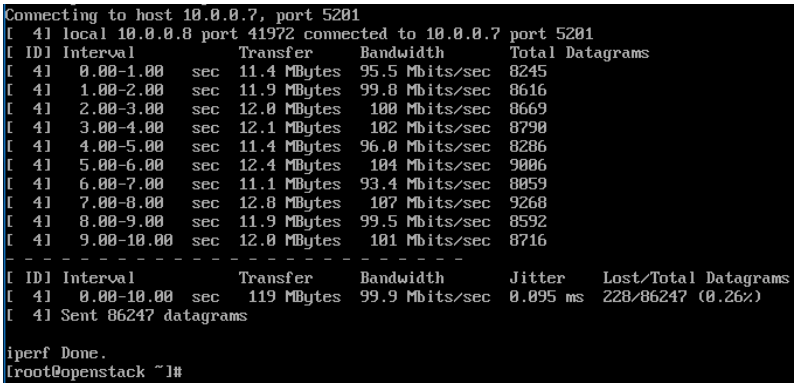


Figura 56. Medición de Jitter para una conexión 100BASE-T

En la Figura 57 se obtiene un resultado de 0.079 ms de Jitter, con 574 datagramas perdidos, lo cual representa el 0.55% del total de datagramas enviados, la prueba es para un enlace de 1 Gbps.

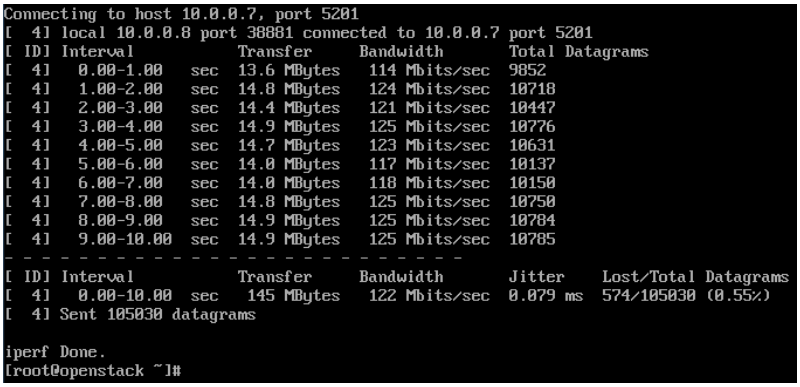


Figura 57. Medición de Jitter para un enlace 1GbE

En la Figura 58 se obtiene un resultado de 0.086 ms de Jitter, con 390 datagramas perdidos, lo cual representa el 0.42% del total de datagramas enviados, la prueba es para un enlace de 10 Gbps.

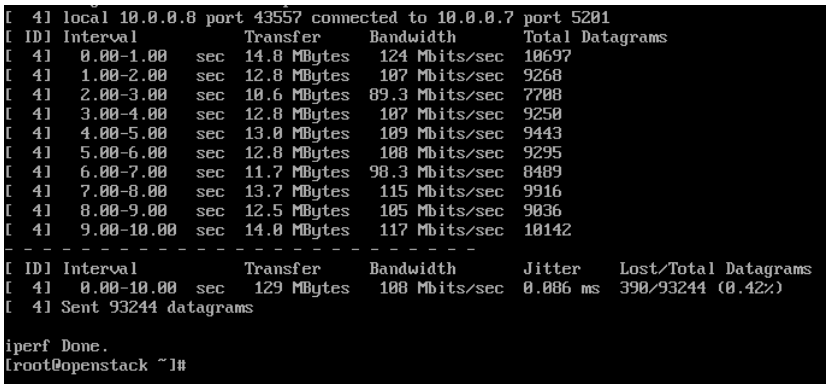


Figura 58. Medición de Jitter para una interfaz 10GbE

Throughput

Para la medición del Throughput (ancho de banda máximo utilizado del canal de comunicaciones), se utilizó la herramienta cliente-servidor llamada Iperf3. Para hacer más fiable la prueba, se ejecutó durante 60 segundos. Los resultados se muestran a continuación.

En la Figura 59 se muestra el máximo Throughput usando paquetes TCP de 85 KB.

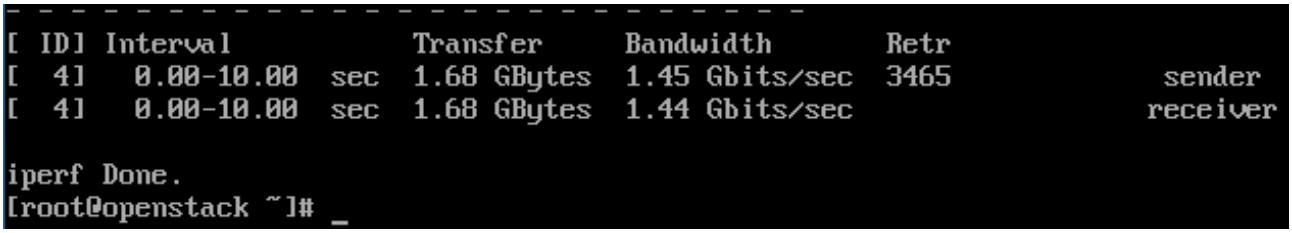


Figura 59. Máximo ancho de banda obtenido al utilizar una ventana TCP de 85 KB.

En la Figura 60 se obtiene un resultado de 99 Mbps de Throughput usando paquetes TCP, con 180 datagramas retransmitidos, la prueba es para un enlace de 100Mbps.

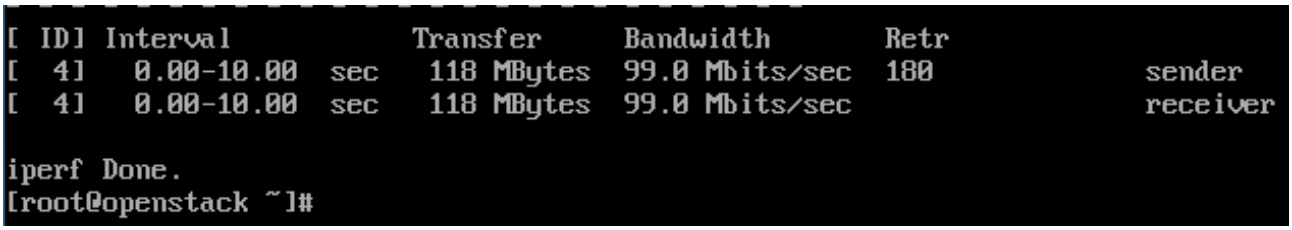


Figura 60. Máximo ancho de banda obtenido para un enlace 100Base-T

En la Figura 61 se obtiene un resultado de 996 Mbps de Throughput usando paquetes TCP, con 2250 datagramas retransmitidos, la prueba es para un enlace de 1Gbps.

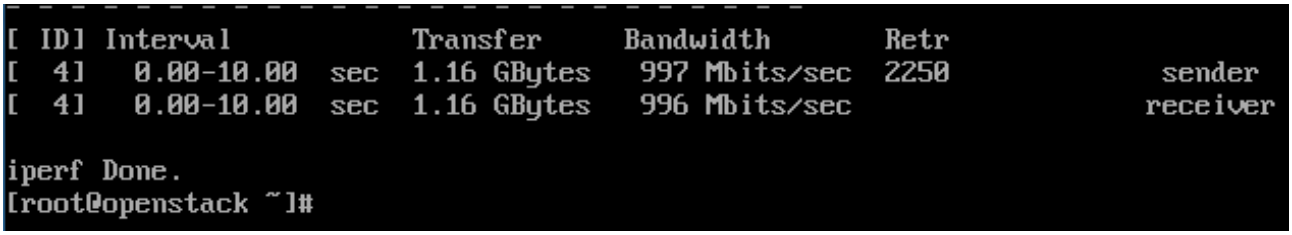


Figura 61. Máximo ancho de banda obtenido para un enlace de 1 GbE

Escenario 2

Latencia

En la Figura 62 se muestran resultados de latencias entre máquinas virtuales, la máquina virtual 1 (PC1) y la máquina virtual 6 (PC6), bajo un entorno virtualizado.

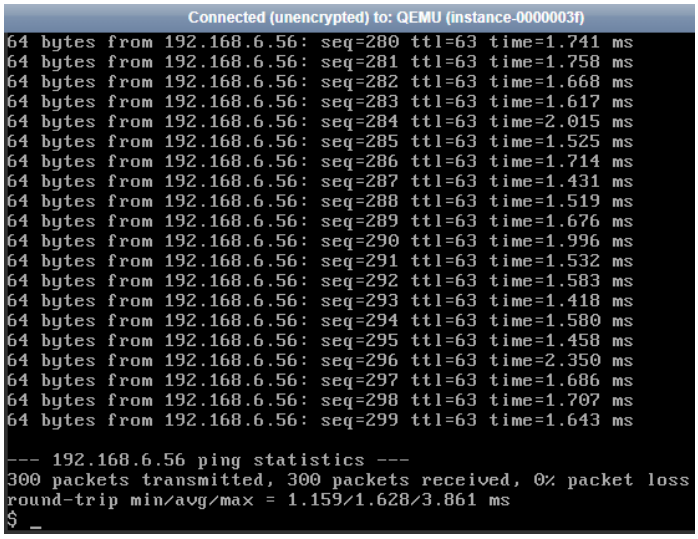


Figura 62. Pruebas de latencia para el escenario 2 (PC1-PC6)

En la Figura 63 se muestran resultados de latencias entre máquinas virtuales, la máquina virtual 3 (PC3) y la máquina virtual 4 (PC4), bajo un entorno virtualizado.

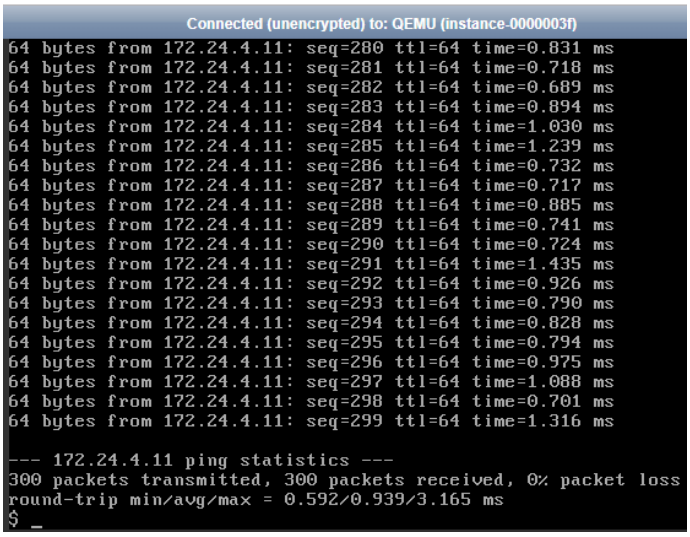


Figura 63. Pruebas de latencia para el escenario 2 (PC3-PC4)

En la Figura 64 se muestran resultados de latencias entre máquinas virtuales, la máquina virtual 2 (PC2) y la máquina virtual 4 (PC4), bajo un entorno virtualizado.

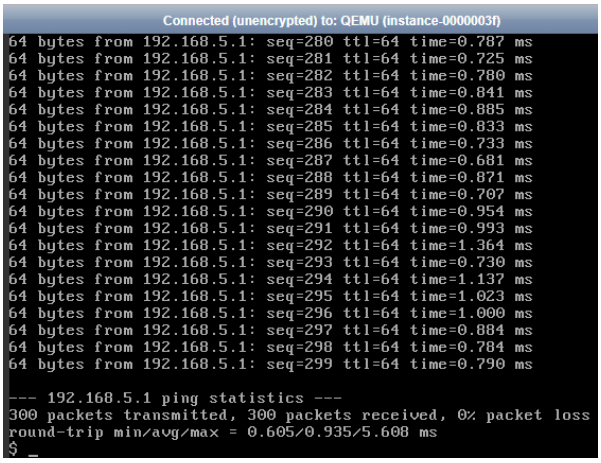


Figura 64. Pruebas de latencia para escenario 2 (PC2-PC4)

En la Figura 65 se muestran resultados de latencias entre máquinas virtuales, la máquina virtual 1 (PC1) y la máquina virtual 6 (PC6), bajo un entorno virtualizado.

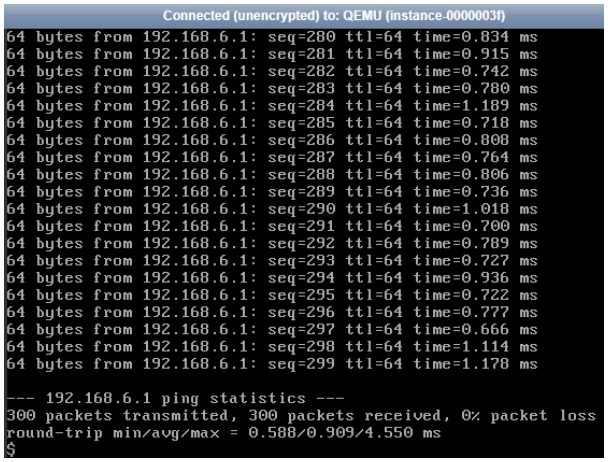


Figura 65. Pruebas de latencia en escenario 2 (PC1-PC6)

Jitter

En la Figura 66 se obtiene un resultado de 0.091ms de Jitter, con 380 datagramas perdidos, lo cual representa el 0.44% del total de datagramas enviados, la prueba es para un enlace de 100 Mbps.

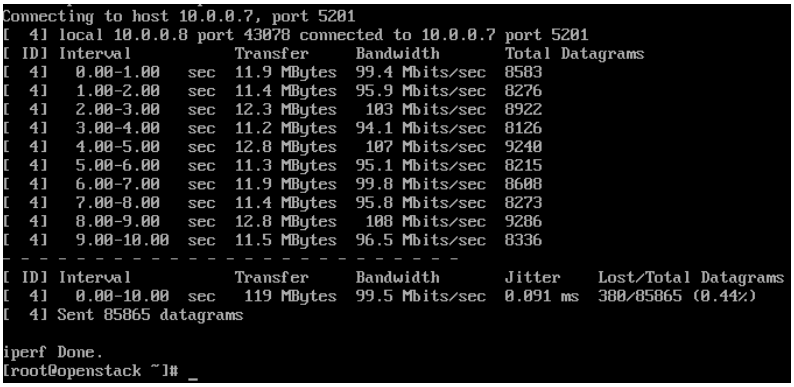


Figura 66. Medición de Jitter para una conexión 100BASE-T

En la Figura 67 se obtiene un resultado de 0.099 ms de Jitter, con 529 datagramas perdidos, lo cual representa el 0.52% del total de datagramas enviados, la prueba es para un enlace de 1 Gbps.

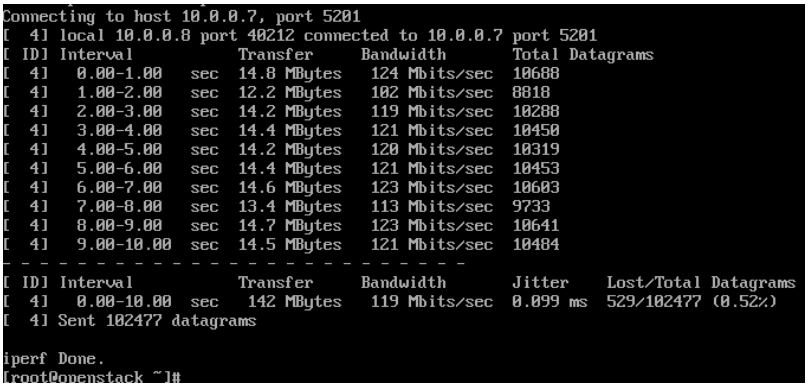


Figura 67. Medición de Jitter para un enlace 1GbE

En la Figura 68 se obtiene un resultado de 0.095 ms de Jitter, con 702 datagramas perdidos, lo cual representa el 0.67% del total de datagramas enviados, la prueba es para un enlace de 10 Gbps.

```
Connecting to host 10.0.0.7, port 5201
[  4] local 10.0.0.8 port 40210 connected to 10.0.0.7 port 5201

[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec  13.9 MBytes   116 Mbits/sec  10056
[  4]   1.00-2.00   sec  14.2 MBytes   119 Mbits/sec  10297
[  4]   2.00-3.00   sec  14.9 MBytes   125 Mbits/sec  10756
[  4]   3.00-4.00   sec  14.8 MBytes   125 Mbits/sec  10750
[  4]   4.00-5.00   sec  14.9 MBytes   125 Mbits/sec  10756
[  4]   5.00-6.00   sec  14.9 MBytes   125 Mbits/sec  10775
[  4]   6.00-7.00   sec  13.7 MBytes   115 Mbits/sec  9957
[  4]   7.00-8.00   sec  14.7 MBytes   123 Mbits/sec  10613
[  4]   8.00-9.00   sec  14.8 MBytes   124 Mbits/sec  10742
[  4]   9.00-10.00  sec  14.9 MBytes   125 Mbits/sec  10782
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec   146 MBytes   122 Mbits/sec  0.095 ms  702/105479 (0.67%)
[  4] Sent 105479 datagrams

iperf Done.
[root@openstack ~]#
```

Figura 68. Medición de Jitter para una interfaz 10GbE

Throughput

Para la medición del Throughput (ancho de banda máximo utilizado del canal de comunicaciones), se utilizó la herramienta cliente-servidor llamada Iperf3. Para hacer más fiable la prueba, se ejecutó durante 60 segundos. Los resultados se muestran a continuación.

En la Figura 69 se muestra el máximo Throughput usando paquetes TCP de 85 KB.

```
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.68 GBytes  1.44 Gbits/sec  3285             sender
[  4]   0.00-10.00  sec  1.67 GBytes  1.44 Gbits/sec                   receiver

iperf Done.
[root@openstack ~]#
```

Figura 69. Máximo ancho de banda obtenido al utilizar una ventana TCP de 85 KB.

En la Figura 70 se obtiene un resultado de 99.1 Mbps de Throughput usando paquetes TCP, con 135 datagramas retransmitidos, la prueba es para un enlace de 100Mbps.

```
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec   118 MBytes  99.1 Mbits/sec  135              sender
[  4]   0.00-10.00  sec   118 MBytes  99.1 Mbits/sec                   receiver

iperf Done.
[root@openstack ~]# _
```

Figura 70. Máximo ancho de banda obtenido para un enlace 100Base-T

En la Figura 71 se obtiene un resultado de 997 Mbps de Throughput usando paquetes TCP, con 2295 datagramas retransmitidos, la prueba es para un enlace de 1Gbps.

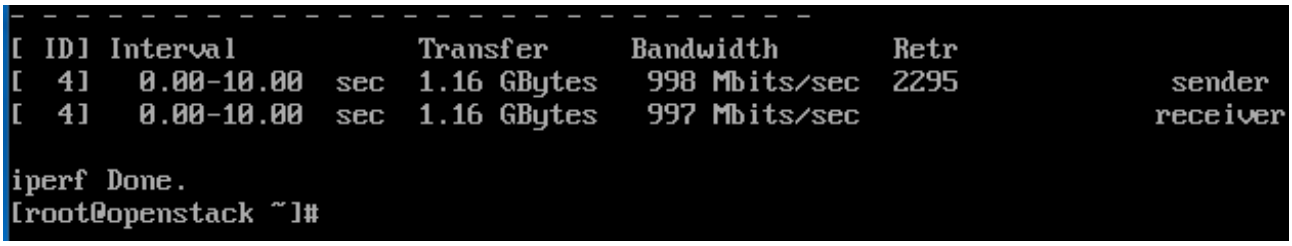


Figura 71. Máximo ancho de banda obtenido para un enlace de 1 GbE

Escenario 3

Latencia

Se muestran resultados de latencias entre máquinas virtuales conectadas mediante switches y routers virtuales, del escenario 3. En este escenario se virtualiza la capa de distribución de la red de la ESIME Zacatenco, por lo tanto, se realizarán pruebas de latencia entre diferentes equipos virtuales de diferentes segmentos de red.

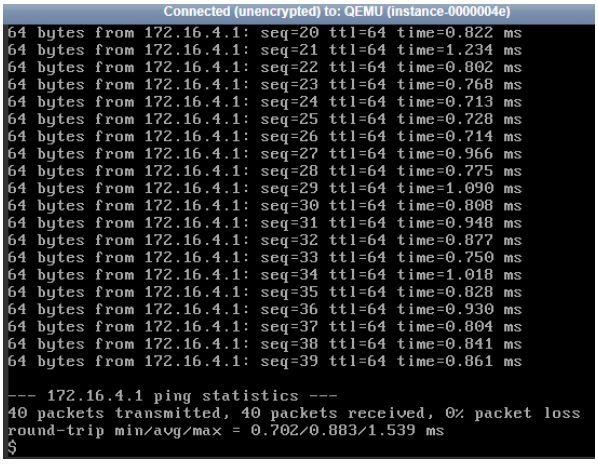


Figura 72. Pruebas de latencia entre clientes interconectados por los equipos R1 y R5.

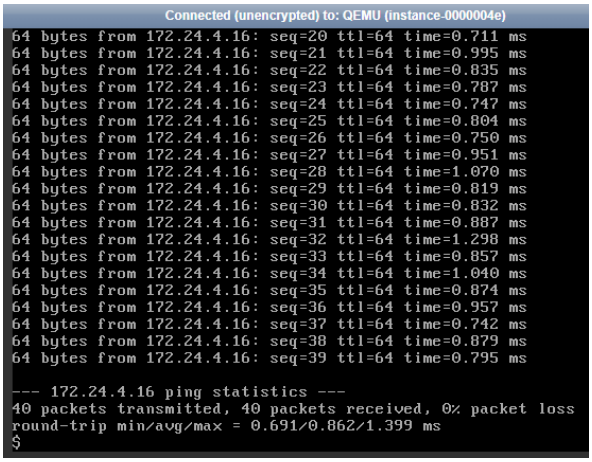


Figura 73. Pruebas de latencia clientes interconectados por los equipos CELEX y PESADOS2.

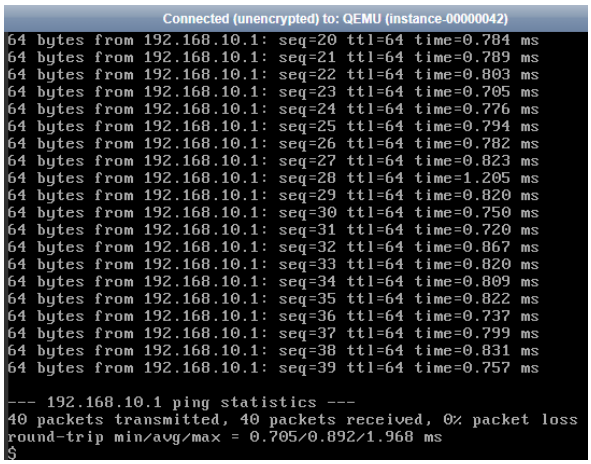


Figura 74. Pruebas de latencia entre los clientes interconectados por los equipos R2 y Z4 ELECTRICA SEPI.

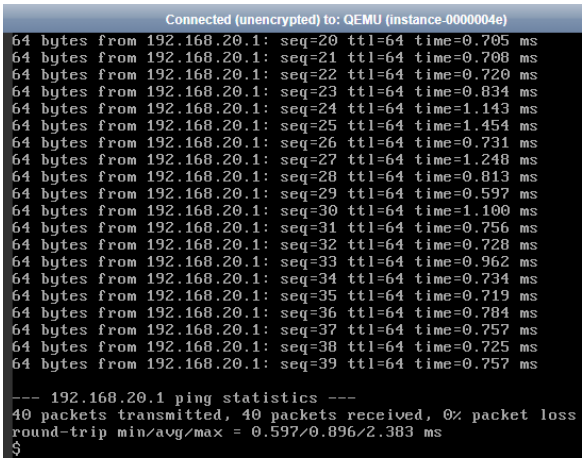


Figura 75. Pruebas de latencia entre los clientes interconectados por los equipos S16 y Z3 ICE.

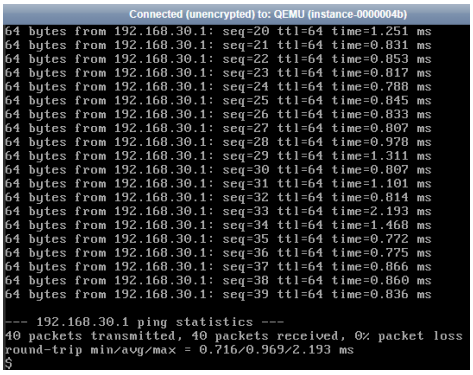


Figura 76. Pruebas de latencia entre los clientes interconectados por los equipos R1 y R5.

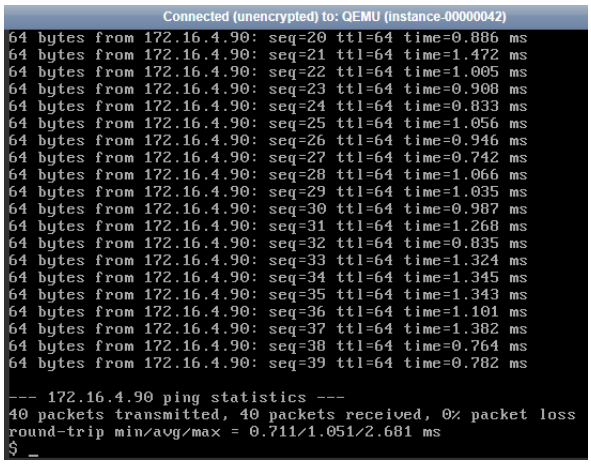


Figura 77. Pruebas de latencia entre los clientes interconectados por los equipos BIBLIOTECA y CÓMPUTO.

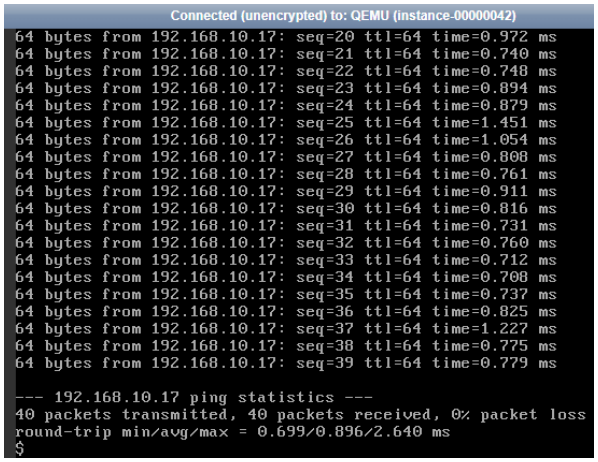


Figura 78. Pruebas de latencia entre clientes interconectados por los equipos S12 e ICA.

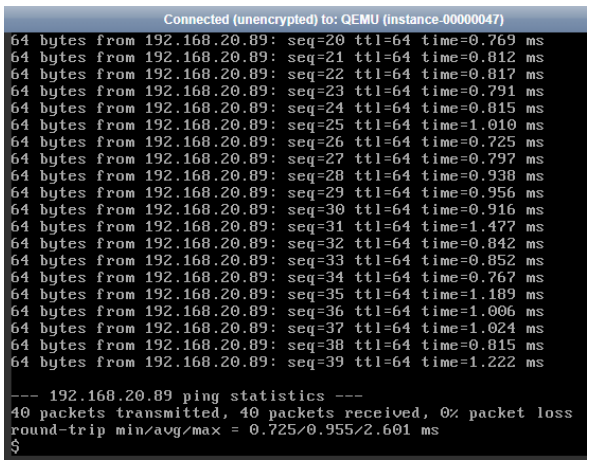


Figura 79. Pruebas de latencia entre clientes interconectados por los equipos S3 y JEFATURA SEPI.

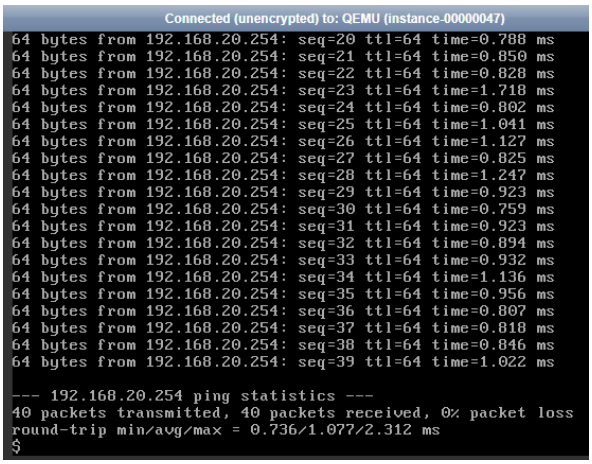


Figura 80. Pruebas de latencia entre clientes interconectados por BIOMECÁNICA y R7.

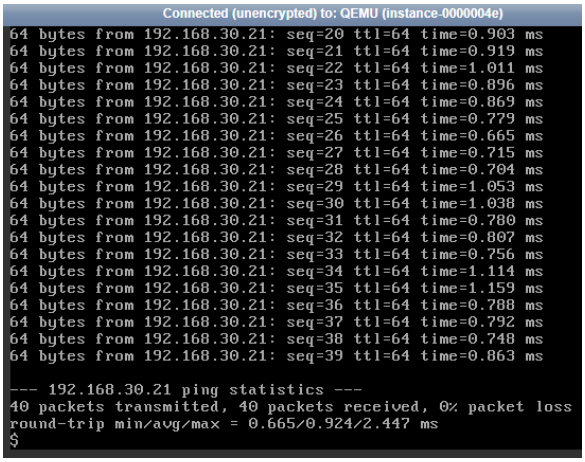


Figura 81. Pruebas de latencia entre clientes interconectados por CUBÍCULOS Z4 TELECOM y S16.

Jitter

En la Figura 82 se obtiene un resultado de 0.087 ms de Jitter, con 246 datagramas perdidos, lo cual representa el 0.29% del total de datagramas enviados, la prueba es para un enlace de 100 Mbps.

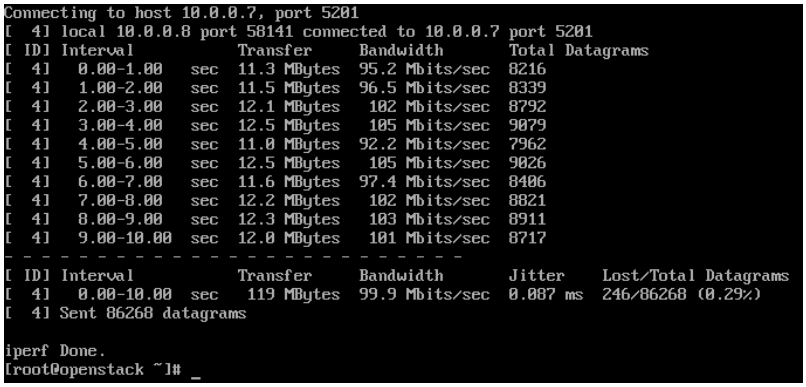


Figura 82. Medición de Jitter para una conexión 100BASE-T

En la Figura 83 se obtiene un resultado de 0.092 ms de Jitter, con 639 datagramas perdidos, lo cual representa el 0.62% del total de datagramas enviados, la prueba es para un enlace de 1 Gbps.

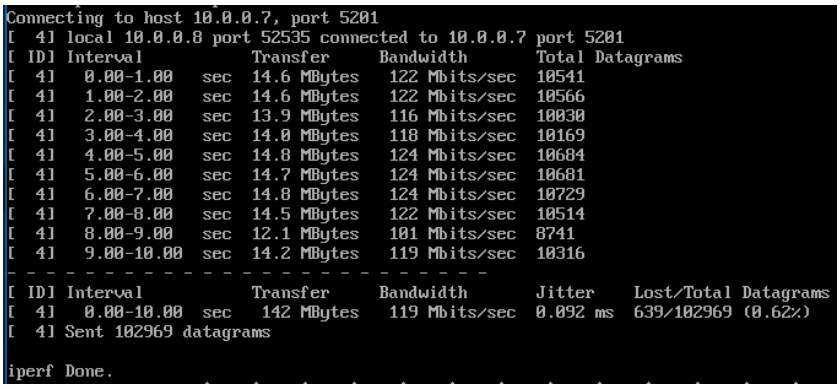


Figura 83. Medición de Jitter para un enlace 1GbE

En la Figura 84 se obtiene un resultado de 0.055 ms de Jitter, con 460 datagramas perdidos, lo cual representa el 0.44% del total de datagramas enviados, la prueba es para un enlace de 10 Gbps.

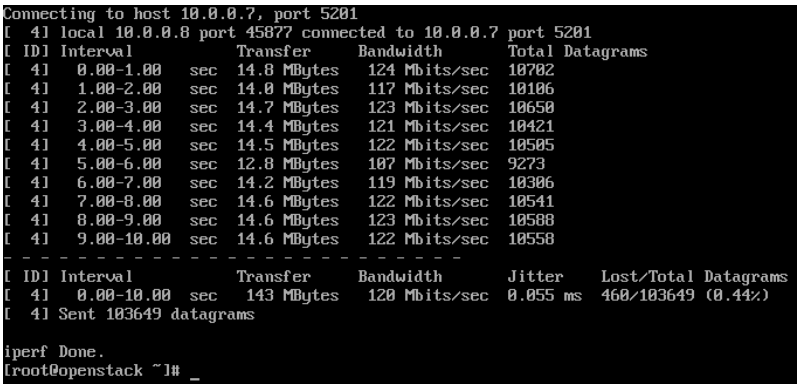


Figura 84. Medición de Jitter para una interfaz 10GbE

Throughput

Para la medición del Throughput (ancho de banda máximo utilizado del canal de comunicaciones), se utilizó la herramienta cliente-servidor llamada Iperf3. Para hacer más fiable la prueba, se ejecutó durante 60 segundos. Los resultados se muestran a continuación.

En la Figura 85 se muestra el máximo Throughput usando paquetes TCP de 85 KB.

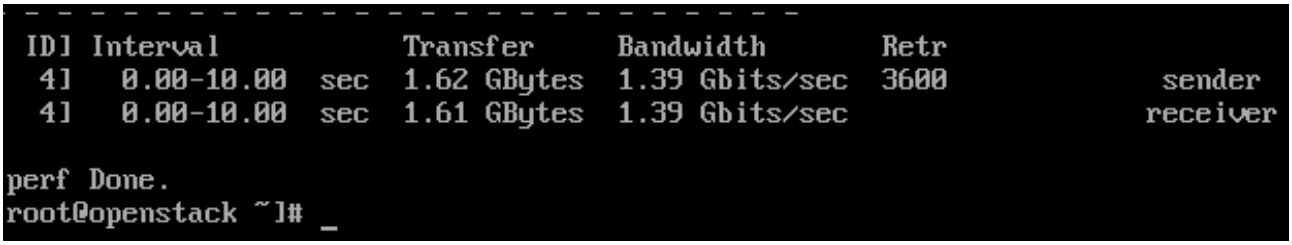


Figura 85. Máximo ancho de banda obtenido al utilizar una ventana TCP de 85 KB.

En la Figura 86 se obtiene un resultado de 99 Mbps de Throughput usando paquetes TCP, con 135 datagramas retransmitidos, la prueba es para un enlace de 100Mbps.

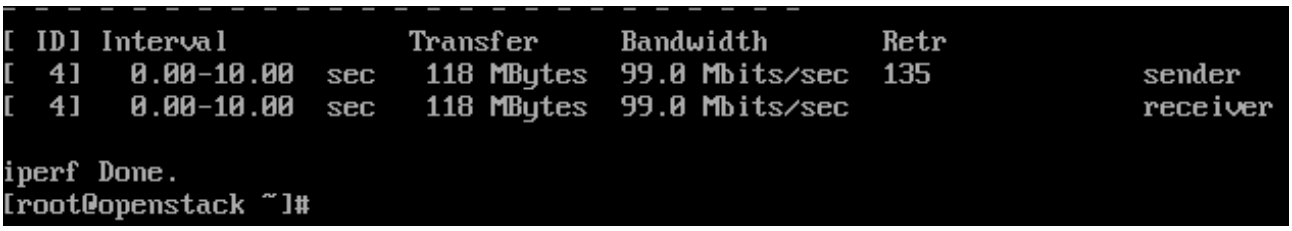


Figura 86. Máximo ancho de banda obtenido para un enlace 100Base-T

En la Figura 87 se obtiene un resultado de 996 Mbps de Throughput usando paquetes TCP, con 2430 datagramas retransmitidos, la prueba es para un enlace de 1Gbps.

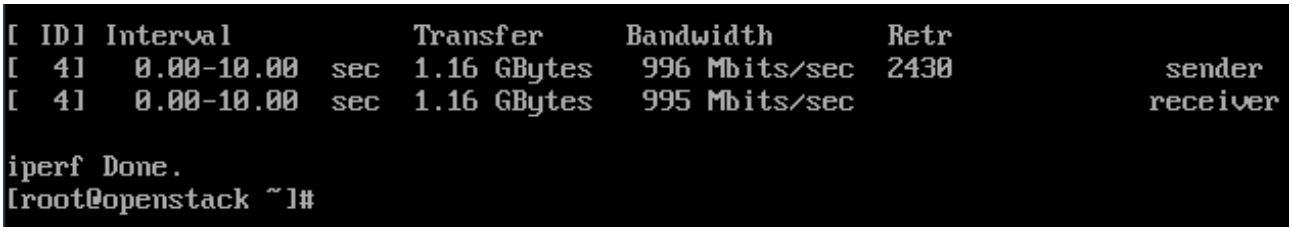


Figura 87. Máximo ancho de banda obtenido para un enlace de 1 GbE

Escenario 4

Latencia

Se muestran resultados de latencias entre máquinas virtuales conectadas mediante switches y routers virtuales, con las modificaciones propuestas para éste escenario, el cual incluye alta disponibilidad y un balanceador de carga. Se virtualiza la capa de distribución de la propuesta de red de la ESIME Zacatenco, por lo tanto, se realizarán pruebas de latencia entre diferentes equipos virtuales de diferentes segmentos de red.

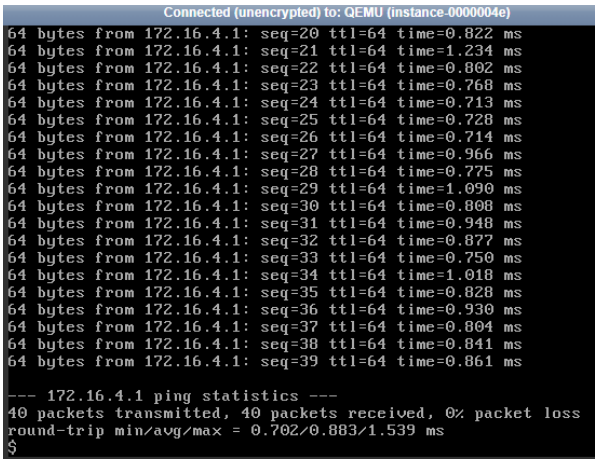


Figura 88. Pruebas de latencia entre clientes interconectados por los equipos R1 y R5.

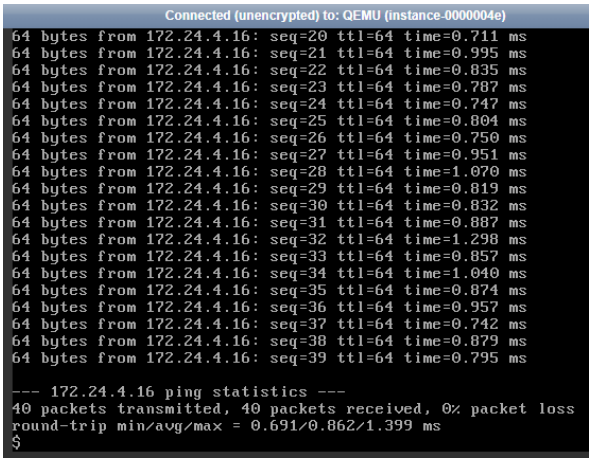


Figura 89. Pruebas de latencia clientes interconectados por los equipos CELEX y PESADOS2.

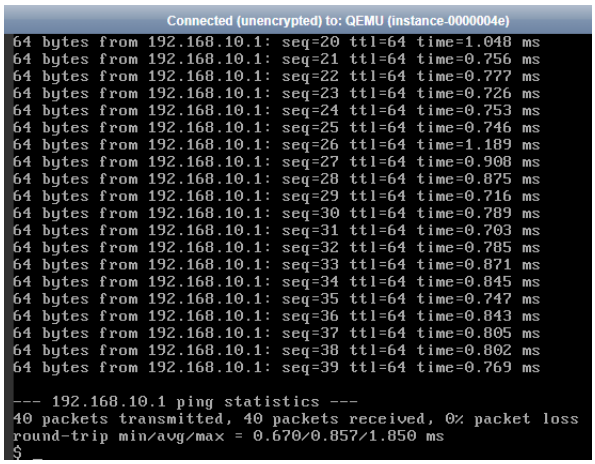


Figura 90. Pruebas de latencia entre los clientes interconectados por los equipos R2 y Z4 ELECTRICA SEPI.

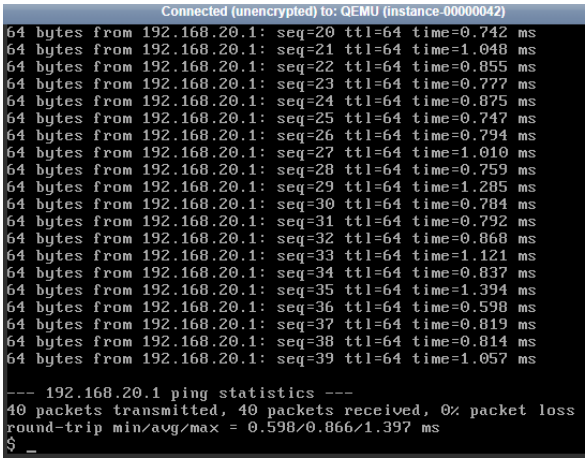


Figura 91. Pruebas de latencia entre los clientes interconectados por los equipos S16 y Z3 ICE.

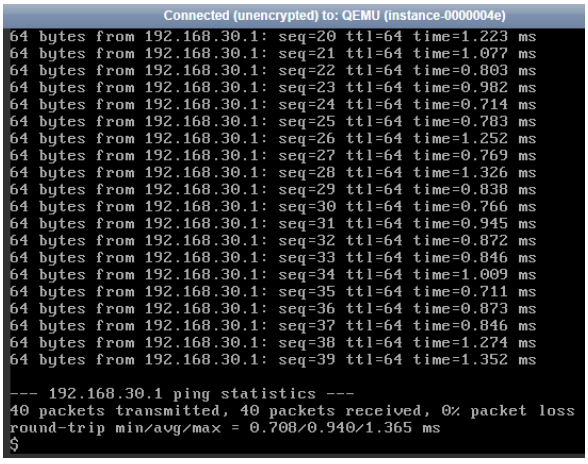


Figura 92. Pruebas de latencia entre los clientes interconectados por los equipos R1 y R5.

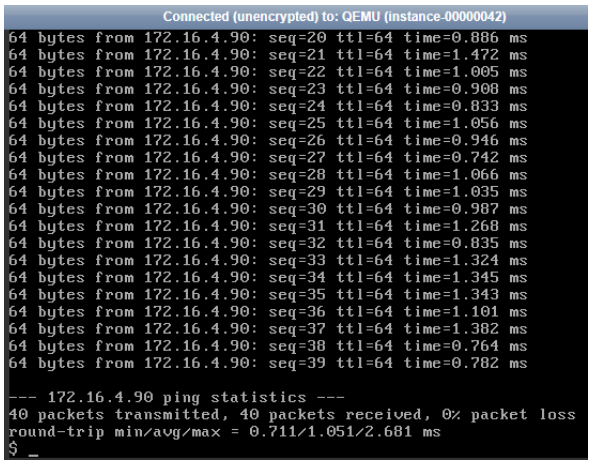


Figura 93. Pruebas de latencia entre los clientes interconectados por los equipos BIBLIOTECA y CÓMPUTO.

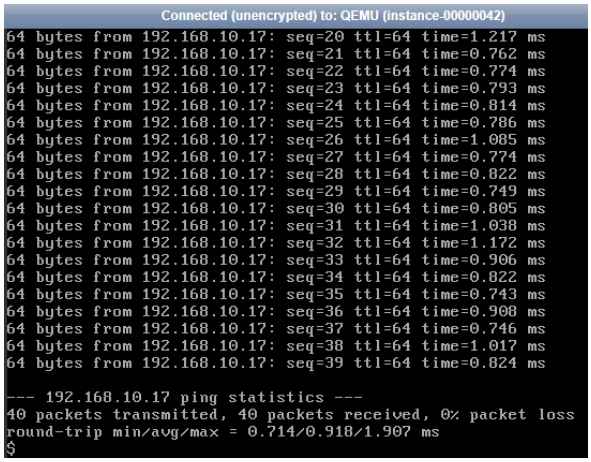


Figura 94. Pruebas de latencia entre clientes interconectados por los equipos S12 e ICA.

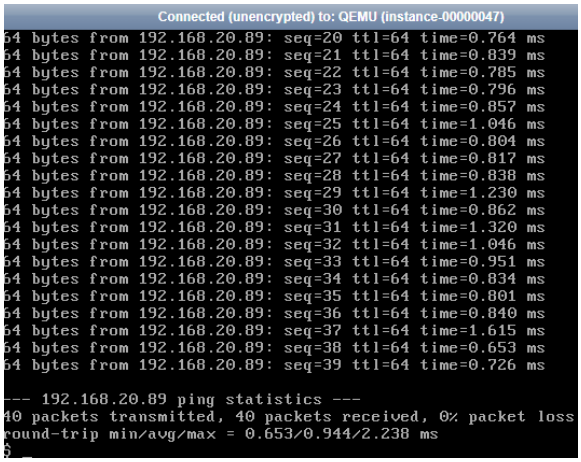


Figura 95. Pruebas de latencia entre clientes interconectados por los equipos S3 y JEFATURA SEPI.

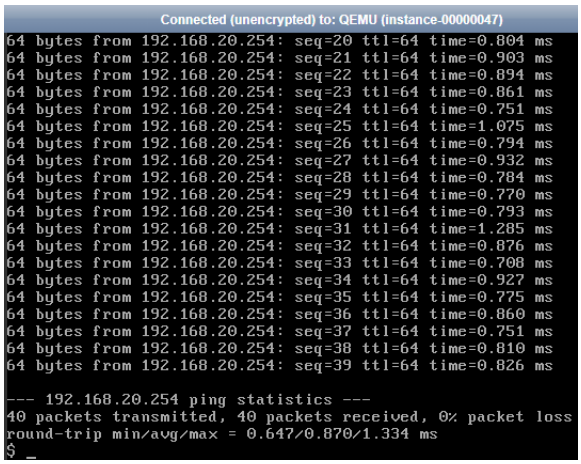


Figura 96. Pruebas de latencia entre clientes interconectados por BIOMECÁNICA y R7.

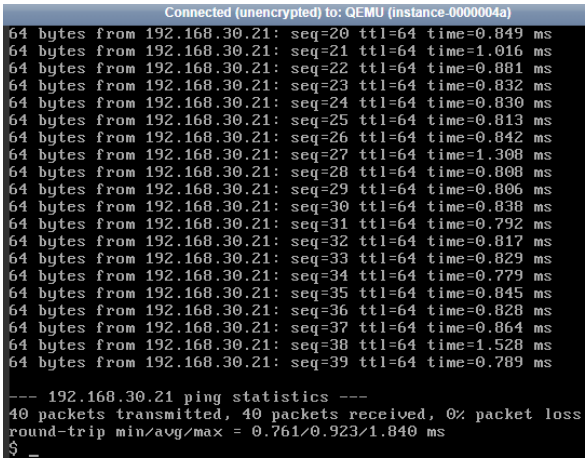


Figura 97. Pruebas de latencia entre clientes interconectados por CUBÍCULOS Z4 TELECOM y S16.

Jitter

En la Figura 98 se obtiene un resultado de 0.081 ms de Jitter, con 394 datagramas perdidos, lo cual representa el 0.40% del total de datagramas enviados, la prueba es para un enlace de 100 Mbps.

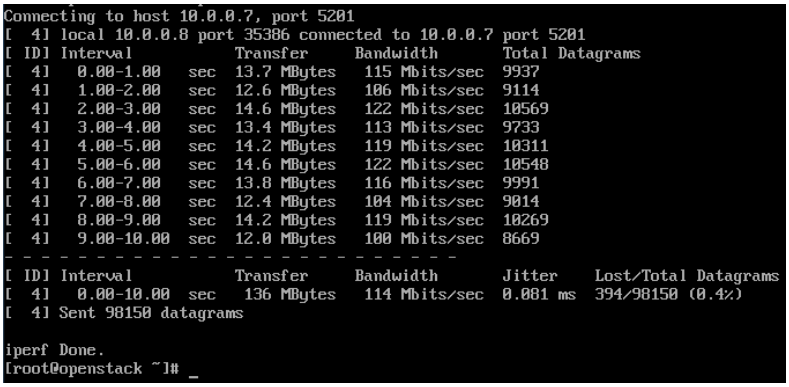


Figura 98. Medición de Jitter para una conexión 100BASE-T

En la Figura 99 se obtiene un resultado de 0.085 ms de Jitter, con 499 datagramas perdidos, lo cual representa el 0.49% del total de datagramas enviados, la prueba es para un enlace de 1 Gbps.

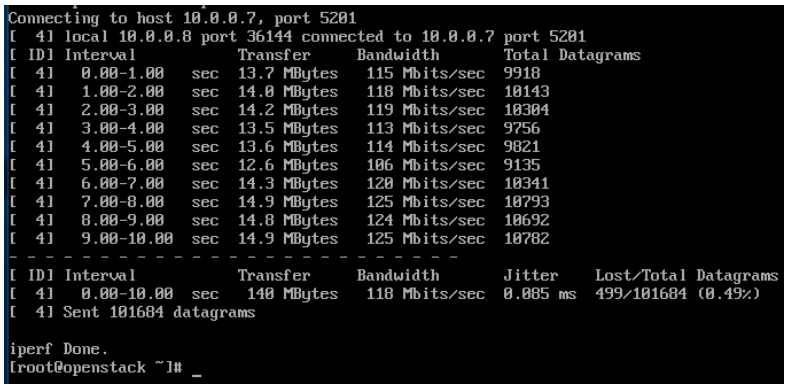


Figura 99. Medición de Jitter para un enlace 1GbE

En la Figura 100 se obtiene un resultado de 0.092 ms de Jitter, con 565 datagramas perdidos, lo cual representa el 0.55% del total de datagramas enviados, la prueba es para un enlace de 10 Gbps.

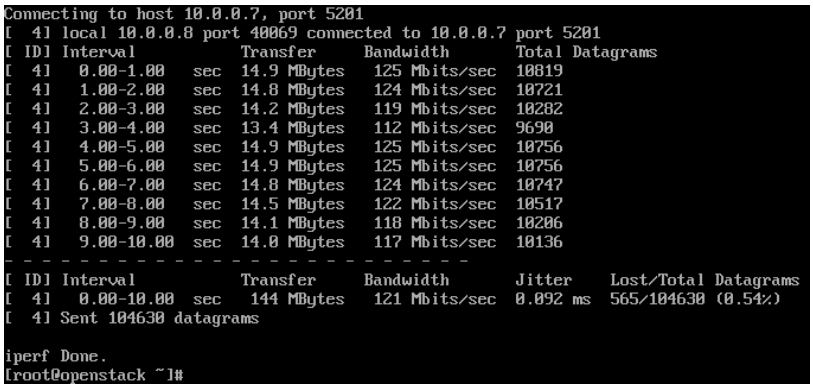


Figura 100. Medición de Jitter para una interfaz 10GbE

Throughput

Para la medición del Throughput (ancho de banda máximo utilizado del canal de comunicaciones), se utilizó la herramienta cliente-servidor llamada Iperf3. Para hacer más fiable la prueba, se ejecutó durante 60 segundos. Los resultados se muestran a continuación.

En la Figura 101 se muestra el máximo Throughput usando paquetes TCP de 85 KB.

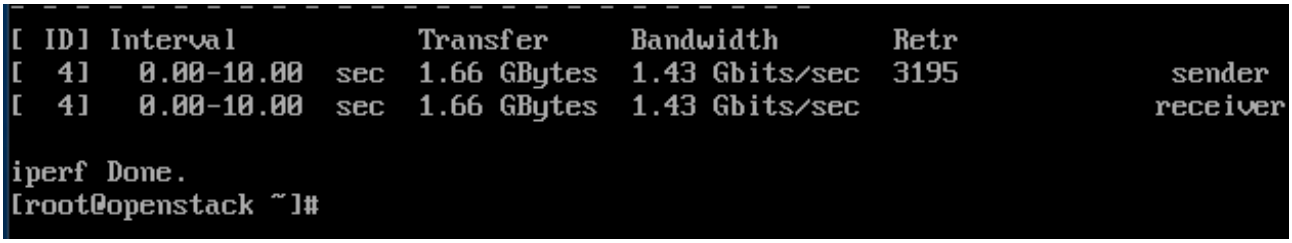


Figura 101. Máximo ancho de banda obtenido al utilizar una ventana TCP de 85 KB.

En la Figura 102 se obtiene un resultado de 99.1 Mbps de Throughput usando paquetes TCP, con 135 datagramas retransmitidos, la prueba es para un enlace de 100Mbps.

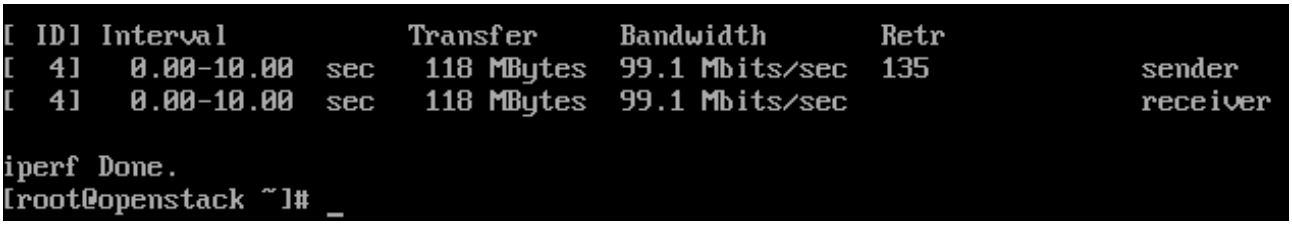


Figura 102. Máximo ancho de banda obtenido para un enlace 100Base-T

En la Figura 103 se obtiene un resultado de 999 Mbps de Throughput usando paquetes TCP, con 2295 datagramas retransmitidos, la prueba es para un enlace de 1Gbps.

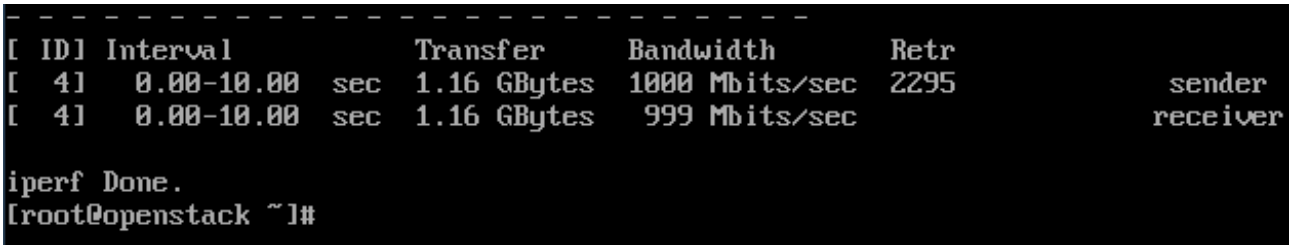


Figura 103. Máximo ancho de banda obtenido para un enlace de 1 GbE

Escenario 5

Latencia

En la Figura 104 se muestran resultados de latencias entre máquinas virtuales, la máquina virtual 1 y la máquina virtual 3 no se alcanzan porque están en diferentes VLANS.

Prueba	Resultado
Ping (VM1-VM2)	0.194 ms (0.187-0.205)
Ping (VM1-VM3)	Destination Host Unreachable

Figura 104. Pruebas de latencia para el escenario 5

Jitter

En la Figura 105 se obtiene un resultado de 0.108 ms de Jitter, con 194 datagramas perdidos, lo cual representa el 0.67% del total de datagramas enviados, la prueba es para un enlace de 100 Mbps.

```
Connecting to host 10.0.0.11, port 5201
[  4] local 10.0.0.31 port 51695 connected to 10.0.0.11 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec  20.8 MBytes   174 Mbits/sec  2656
[  4]   1.00-2.00   sec  23.0 MBytes   193 Mbits/sec  2945
[  4]   2.00-3.00   sec  21.1 MBytes   177 Mbits/sec  2697
[  4]   3.00-4.00   sec  21.7 MBytes   182 Mbits/sec  2781
[  4]   4.00-5.00   sec  23.3 MBytes   195 Mbits/sec  2980
[  4]   5.00-6.00   sec  23.5 MBytes   197 Mbits/sec  3002
[  4]   6.00-7.00   sec  23.5 MBytes   197 Mbits/sec  3008
[  4]   7.00-8.00   sec  23.1 MBytes   194 Mbits/sec  2955
[  4]   8.00-9.00   sec  23.3 MBytes   196 Mbits/sec  2985
[  4]   9.00-10.00  sec  23.2 MBytes   195 Mbits/sec  2971
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec   226 MBytes   190 Mbits/sec  0.108 ms  194/28979 (0.67%)
[  4] Sent 28979 datagrams

iperf Done.
ipn@compute:~$
```

Figura 105. Medición de Jitter para una conexión 100BASE-T

En la Figura 106 se obtiene un resultado de 0.038 ms de Jitter, con 0 datagramas perdidos, lo cual representa el 0% del total de datagramas enviados, la prueba es para un enlace de 1 Gbps.

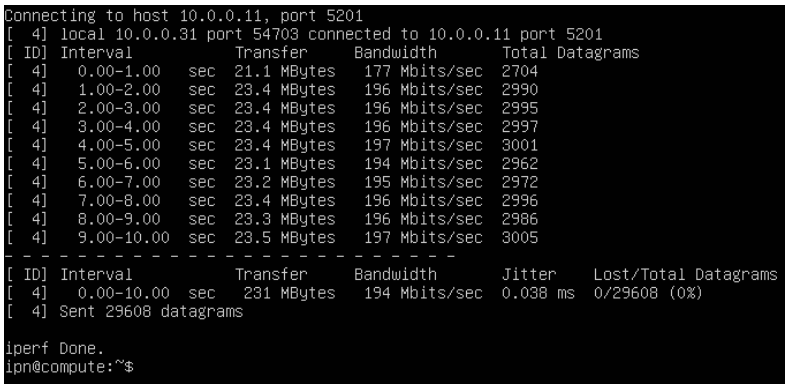


Figura 106. Medición de Jitter para un enlace 1GbE

En la Figura 107 se obtiene un resultado de 0.059 ms de Jitter, con 162 datagramas perdidos, lo cual representa el 0.55% del total de datagramas enviados, la prueba es para un enlace de 10 Gbps.

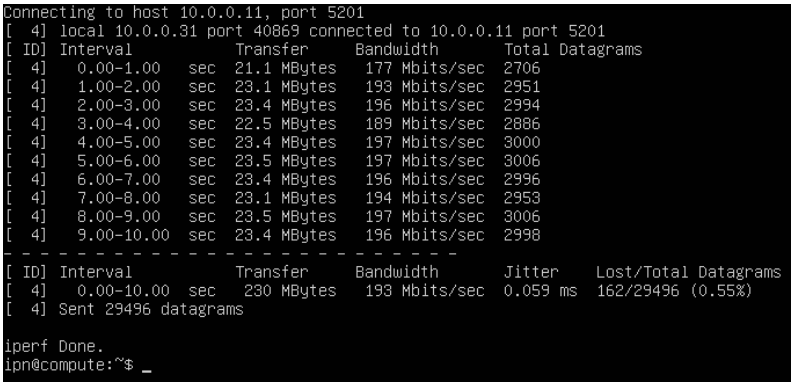


Figura 107. Medición de Jitter para una interfaz 10GbE

Throughput

Para la medición del Throughput (ancho de banda máximo utilizado del canal de comunicaciones), se utilizó la herramienta cliente-servidor llamada Iperf3. Para hacer más fiable la prueba, se ejecutó durante 60 segundos. Los resultados se muestran a continuación.

En la Figura 108 se muestra el máximo Throughput usando paquetes TCP de 85 KB.

```
------------------------------------------------------------
Client connecting to 10.0.0.11, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.31 port 37058 connected with 10.0.0.11 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  2.91 GBytes  2.50 Gbits/sec
ipn@compute:~$
```

Figura 108. Máximo ancho de banda obtenido al utilizar una ventana TCP de 85 KB.

En la Figura 109 se obtiene un resultado de 87.5 Mbps de Throughput usando paquetes TCP, con 0 datagramas retransmitidos, la prueba es para un enlace de 100Mbps.

```
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec   626 MBytes  87.5 Mbits/sec    0             sender
[  4]   0.00-60.00  sec   626 MBytes  87.5 Mbits/sec                  receiver

iperf Done.
ipn@compute:~$
```

Figura 109. Máximo ancho de banda obtenido para un enlace 100Base-T

En la Figura 110 se obtiene un resultado de 876 Mbps de Throughput usando paquetes TCP, con 0 datagramas retransmitidos, la prueba es para un enlace de 1Gbps.

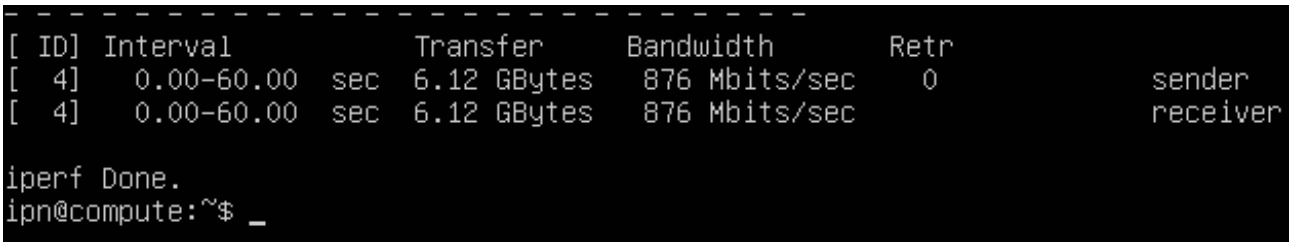


Figura 110. Máximo ancho de banda obtenido para un enlace de 1 GbE

CONCLUSIONES

De acuerdo con las actividades desarrolladas en este trabajo para la obtención de datos acerca de la red Institucional del Instituto Politécnico Nacional a nivel nacional, es decir, se examinaron todas las unidades del IPN, y con todo el proceso de análisis, desarrollo, propuesta y evaluaciones, basado en el método científico: se diseñó una METODOLOGÍA a medida que agilizará las pruebas de concepto, puestas en pre-producción y algunas recomendaciones para el despliegue y la puesta en producción de la Virtualización de Funciones de Red (NFV), como primer paso para la migración final hacia las Redes Definidas por Software.

De acuerdo a la metodología propuesta, y después de ejecutar el proceso de observación y análisis que en la misma se detalla, se obtiene que: NFV posee un gran potencial para contribuir a la disminución de CAPEX y OPEX, reducir la complejidad de las redes, insertar servicios en la red de manera rápida, ofrecer flexibilidad en la entrega de servicios y simplificar los procedimientos operacionales.

Sin embargo, no se puede ignorar el hecho de que es una tecnología que posee aún muchos desafíos. Entre ellos se destacan:

1. Escalabilidad y rendimiento

   Puesto que NFV tiene como objetivo dar soporte a una gran cantidad de servidores a bajo costo, ofreciendo diversos recursos de infraestructura. Lo complejo radica en cómo garantizar que la utilización de estos recursos sea escalable, lo cual representa un gran desafío a enfrentar por los desarrolladores e investigadores del tema.

   También es de gran importancia que las especificaciones de rendimiento estén debidamente especificadas y bien definidas para los recursos de la infraestructura donde se despliegue la virtualización de funciones de red. Es importante porque el rendimiento y el comportamiento pueden no ser tan buenos como los dispositivos intermedios dedicados.

   Además, es imprescindible tratar de mantener la más baja degradación del rendimiento posible, mediante el uso de los hipervisores adecuados, así como de las

tecnologías emergentes de software, por lo que de esta manera se reducen al mínimo la latencia, la sobrecarga de procesamiento en los nodos de cómputo y el rendimiento.

2. Portabilidad e interoperabilidad

   Dado que la meta es la implementación de los conceptos de virtualización de funciones de red en entornos de producción, la interoperabilidad con otras redes físicas o virtuales y la coexistencia entre funciones de red virtualizadas y funciones de red físicas es de extrema importancia.

3. Seguridad

   La virtualización del servidor genera problemas relacionados con la seguridad, mediante la aplicación de funciones de red en un conjunto de routers o servidores por ejemplo, puede resultar en la aparición de agujeros de seguridad en los sistemas operativos de red, interfaces virtuales y servicios que interconectan las funciones de red virtualizadas.

   En varios casos, los clientes (tenants) pueden requerir gestionar los servicios en diversos niveles de operación de acuerdo a sus SLAs. De esta forma, la virtualización de funciones de red debe ofrecer una exposición segura de su interfaz de programación de aplicaciones (API) y que se comporte en niveles independientes para que la gestión individual de un cliente no interfiera en el funcionamiento de funciones de red de otro cliente.

4. Elasticidad

   El entorno de virtualización de funciones de red no está exento de errores, por lo tanto, es esencial que las funciones de red se vuelvan a crear cuando se produce un error. Dependiendo del tipo de servicio que es dependiente de estas funciones de red, el proceso de recrearlas puede ser automática o manual.

5. Gestión y Orquestación

   La implementación de virtualización de funciones de red crea un entorno donde las funciones de red ganan movilidad, y de acuerdo con la demanda y los requisitos, se pueden insertar en diferentes contextos y entornos, esto ocasiona que se pueda

ganar agilidad y consistencia, por lo tanto, es de extrema importancia la gestión y orquestación de las funciones de red.

6. Automatización

   La escalabilidad de la virtualización de funciones de red, es totalmente dependiente de su automatización. Sin ella, la NFV presenta los mismos problemas existentes para escalar las infraestructuras físicas, perdiendo así uno de sus principales beneficios.

7. Simplicidad

   Se debe garantizar que la virtualización de las plataformas de red sea mucho más simple de operar que las plataformas actuales. Con OpenStack aún se debe hacer más desarrollo en este punto, ya que sólo el proceso de instalación es sumamente complicado, al igual que el proceso de conFiguración.

Es importante recalcar que la NFV no se reduce sólo a aminorar costos, se trata también de ofrecer servicios de la manera más rápida y confiable posible, así como de reconFigurarlos dinámicamente para entregarlos de forma personalizada y también lograr una adecuada calidad de experiencia para el usuario.

Ahora, la evaluación llevada a cabo en una red tradicional (RED IPN), es vital también para los entornos virtualizados, teniendo en cuenta las diferencias entre ambos. Al evolucionar las redes hacia NFV, el modelo tradicional de funciones de red y la soluciones de un solo vendedor, cambia hacia un grupo de funciones virtualizadas, con mayor dinamismo y mayor gestión automática del ciclo de vida. De aquí, que las metodologías y herramientas tradicionales de prueba, necesitan también evolucionar para cubrir el ancho espectro de conFiguraciones y escenarios.

Respecto de la evaluación llevada a cabo en la red Institucional del Instituto Politécnico Nacional, basada en la metodología personalizada que se propone se obtiene lo siguiente:

8. Graves problemas de seguridad en determinados puntos de la red, que no se pueden revelar en este trabajo y que le corresponderá solucionar en su momento a la DCyC

9. En algunas secciones de la red, la Topología física no cumple con los estándares de cableado estructurado, es decir, no cumplen con la norma TIA/EIA 568B.
10. Los servidores físicos con los que cuenta el Instituto Politécnico Nacional, cumplen con los requisitos mínimos mencionados en este trabajo para el despliegue de NFV.
11. Los equipos de red físicos que actualmente operan en la red Politécnica están ya obsoletos, y los que se han adquirido recientemente para la capa de distribución no cuentan con sus licencias correspondientes, por lo que se cuentan con equipos que soportan anchos de banda de hasta 40 Gbps y que actualmente están operando a 1 Gbps.
12. Existen cuellos de botella en gran parte de la red Politécnica, en el núcleo se tienen equipos que operan a 80 Gbps, en la capa de distribución están los equipos sin licencia limitados a 1 Gbps y en la capa de acceso se encuentran los equipos cuyo límite es de 100 Mbps. Por lo tanto, aunque en gran parte de la red de la ESIME Zacatenco está conectada mediante fibra óptica, el rendimiento no se incrementa dadas las limitaciones técnicas de los equipos de Networking.
13. En la mayoría de los casos, los administradores no cuentan con la capacitación profesional suficiente para la operación, conFiguración y troubleshooting (solución de fallas) de los equipos de red que administran, así como de conocimiento básico en ciberseguridad. Esto conlleva a que aún existan tormentas de broadcast por las mañanas, que la red sea vulnerable a pruebas de penetración, que existan puertos abiertos, y que las listas de control de acceso (ACLs) ya sean estándares o extendidas no sean lo suficientemente concretas para poder limitar el acceso por SSH a los dispositivos de red que operan en la capa de distribución, por poner un ejemplo. Lo anterior no se contemplaba en los alcances de este trabajo.
14. Esto no se contempla en los alcances ni objetivos de este trabajo, sin embargo, es importante mencionar que la red Politécnica a nivel nacional sólo cuenta con un balanceador de carga, lo cual es preocupante dado que ésta red tiene servidores web en gran parte de sus unidades, por lo que se recomienda un análisis exhaustivo del tráfico TCP para hallar puntos críticos y el despliegue de balanceadores de carga.

15. Es urgente el despliegue del Protocolo de Internet versión 6, primero porque representa un avance significativo en aspectos de seguridad y segundo porque la estructura de la red Politécnica permite que se inicie la migración en cualquier momento, ya que solo basta con conFigurar el Router ubicado en el Edificio 1 para que opere completamente en IPv6 con enrutamiento OSPF versión 3.
16. Evaluar entornos NFV es crucial para su éxito. Su despliegue dentro de la Red Politécnica solo puede ser exitoso si se demuestra que es tan confiable como las implementaciones existentes.
17. Las pruebas y monitoreo de la NFVI y las VNFs son un componente clave para la puesta en marcha de una red productiva de cualquier entorno NFV.
18. Las evaluaciones son necesarias para entender las características de desempeño, fiabilidad, capacidad, escalabilidad y convergencia del todo el sistema, además que proveen visibilidad de la red y de los procesos que en ella se ejecutan.
19. Dadas las grandes limitaciones técnicas en cuanto a potencia de cómputo con las que se realizó éste trabajo, sólo se realizaron pruebas de latencia, ancho de banda y retraso de paquetes; y no se pudieron realizar las modernas pruebas que son propias de un entorno NFV-SDN, mismas que se listan a continuación:
    a. Single node L2 instance-to-instance performance
    b. L2 instance-to-instance performance
    c. L2 concurrent performance
    d. Single node L3 east-west instance-to-instance
    e. L3 east-west instance-to-instance performance
    f. L3 east-west concurrent performance
    g. Single node L3 north-south instance-to-instance
    h. L3 north-south instance-to-instance performance
    i. L3 north-south concurrent performance
    j. Neutron QoS testing
20. Mediante la metodología propuesta en este trabajo, es posible replicarla en cada una de las unidades del IPN distribuidas en todo el territorio nacional (Escuelas,

Centros de Investigación, Edificios de Gobierno, etcétera.) para la prueba de concepto, puesta en pre-producción y la puesta en producción de un entorno NFV que beneficie a todos los usuarios de la red Politécnica.

Trabajos futuros

Se propone seguir realizando pruebas con equipo de cómputo más potente, así como ir capacitando al personal encargado de la administración de las Unidades de Informática como del propio Departamento de Cómputo y Comunicaciones para reducir la curva de aprendizaje y el cambio de paradigma no sea tan brusco, debido a que la migración hacia SDN como objetivo final pasando por NFV, es inevitable.

Por ejemplo, el caso del SAES en periodos de inscripción-reinscripción, sería interesante que al menos la ESIME Zacatenco se interesara en desplegar NFV en ese tipo de situaciones, para los encargados del proceso de admisión sería tan sencillo como solicitar los servicios DBaaS, NaaS, por citar algunos, y el administrador únicamente con unos cuantos clics añadiría más servidores virtuales de bases de datos, crear una red de alta disponibilidad entre ellos, anunciarlos en la red, dejarlos en operación durante la contingencia y una vez finalizado el periodo en el que saturan los servidores físicos, respaldar la información contenida en las respectivas bases de datos, para finalmente eliminar los servidores de bases de datos virtuales creados inicialmente.

De igual forma, se propone realizar un manual que pueda ser utilizado en cualquier dependencia del Instituto Politécnico Nacional que esté interesada en implementar NFV en su red de telecomunicaciones.

Para poner en marcha un proyecto de este tipo se requiere de la autorización de la Dirección de Cómputo y Comunicaciones del Instituto Politécnico Nacional.

Termino diciendo que el sistema puede ser como se requiera y como el proyecto lo necesite. Por supuesto, el diseño de éste tipo de arquitectura requiere de conocimientos avanzados de Red, Almacenamiento y Cómputo. De ahí que vuelvo a hacer hincapié en la formación de personal altamente capacitado y de la capacitación de la plantilla existente.

ANEXOS

ANEXO I. Direccionamiento de la Red Institucional del Instituto Politécnico Nacional

Revisar

https://docs.google.com/document/d/1jWzn6djFESAMcKbGRIymPYHZCifPSCA8Zcptn3MPY4U/edit?usp=sharing

ANEXO II. Direccionamiento del segmento 1 para ESIMEZ

Revisar

https://docs.google.com/document/d/1jWzn6djFESAMcKbGRIymPYHZCifPSCA8Zcptn3MPY4U/edit?usp=sharing

ANEXO III. Direccionamiento del segmento 2 para ESIMEZ

Revisar

https://docs.google.com/document/d/1jWzn6djFESAMcKbGRIymPYHZCifPSCA8Zcptn3MPY4U/edit?usp=sharing

ANEXO IV. Direccionamiento del segmento 3 para ESIMEZ

Revisar

https://docs.google.com/document/d/1jWzn6djFESAMcKbGRIymPYHZCifPSCA8Zcptn3MPY4U/edit?usp=sharing

ANEXO V. Direccionamiento del segmento 4 para ESIMEZ

Revisar

https://docs.google.com/document/d/1jWzn6djFESAMcKbGRIymPYHZCifPSCA8Zcptn3MPY4U/edit?usp=sharing

ANEXO VI. Direccionamiento del segmento 5 para ESIMEZ

Revisar

https://docs.google.com/document/d/1jWzn6djFESAMcKbGRIymPYHZCifPSCA8Zcptn3MPY4U/edit?usp=sharing

ANEXO VII. Direccionamiento del segmento 6 para ESIMEZ

Revisar

https://docs.google.com/document/d/1jWzn6djFESAMcKbGRIymPYHZCifPSCA8Zcptn3MPY4U/edit?usp=sharing

ANEXO VIII. Direccionamiento del segmento 7 para ESIMEZ

Revisar

https://docs.google.com/document/d/1jWzn6djFESAMcKbGRIymPYHZCifPSCA8Zcptn3MPY4U/edit?usp=sharing

ANEXO IX. OpenStack mediante DevStack

En máquina física

- ConFigurar NAT (Network Translation Address) para que las instancias tengan acceso a internet.
- Clonar el repositorio devstack dentro de la máquina virtual con el comando siguiente:

```
# git clone https://github.com/iesgn/devstack-rocky-release
```

- Instalar devstack (versión Rocky) en la máquina virtual (stack.sh)

```
# cd devstack-rocky-release/MF
# chmod 755 stack.sh
# ./stack.sh
```

En máquina virtual

- Como superusuario (root):

```
# apt-get install git
# apt-get install virtualbox
# wget https://releases.hashicorp.com/vagrant/2.2.4/vagrant_2.2.4_x86_64.deb
# dpkg –i vagrant_2.2.4_x86_64.deb

# apt-get install python-pip python-dev
# pip install ansible
```

- Como usuario sin privilegios se descarga el box precise64:

```
# vagrant box add hashicorp/precise64
```

- Se arranca la máquina virtual con Ubuntu 18.04 LTS.
- Se clona el repositorio devstack dentro de la máquina

```
# git clone https://github.com/iesgn/devstack-rocky-release
# cd devstack-rocky-release/MV
# chmod 0600 id_vagrant
# vagrant up
```

- Se instala devstack (versión Rocky) en la máquina (stack.sh)

```
# ./stack.sh
```

Si el proceso de instalación da algún problema, se puede optar por probar OpenStack usando una máquina virtual de VirtualBox con devstack instalada, en forma de fichero OVA. Es una máquina virtual con una interfaz de red de tipo NAT (Network Address Translation), por lo tanto, se puede trabajar con OpenStack desde la misma máquina.

Instrucciones

1. Tener el fichero OVA de Openstack
2. Importar el fichero del paso 1 a VirtualBox. La máquina está conFigurada con 5 GB de RAM, se puede modificar ésta parámetro según las necesidades.
3. Para acceder, se usa el usuario: *root* y contraseña: *|ipneutron./#*

ANEXO X. OpenStack mediante RDO

Instalación

Utilizando RDO se puede tener OpenStack Rocky funcionando en unos minutos siguiendo estos tres sencillos pasos:

```
# yum install –y http://rdo.fedorapeople.org/openstack-rocky/rdo-release-rocky.rpm
# yum install –y openstack-packstack
# packstack --allinone
```

Utilización

Podemos acceder a Horizon con la URL http://<IP-del-equipo>/dashboard y acceder con el usuario *admin* (para tareas de administración) o el usuario *demo* para utilización normal de OpenStack. Las contraseñas de ambos usuarios se encuentran definidas en los ficheros *keystonerc_admin* o *keystonerc_demo*, ubicados en el directorio desde el que se ejecutó la instrucción packstack.

La principal limitación que tiene la conFiguración automática de RDO con packstack es que no se puede acceder a las instancias desde un equipo exterior, ya que el *bridge-exterior* no está conectado a ninguna interfaz física.

Conectar el br-ex al exterior

Para conectar el bridge exterior al exterior, se siguen los pasos que se detallan a continuación:

- Se edita el fichero /ect/sysconfig/network-scripts/ifcfg-br-ex y se reemplaza su contenido por el siguiente:

```
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=X.X.X.X # La dirección inicial de eth0
NETMASK=X.X.X.X # La máscara de subred que corresponda
GATEWAY=X.X.X.X # La dirección IP de la puerta de enlace predeterminada
DNS1=X.X.X.X # La dirección IP del servidor DNS
ONBOOT=yes
```

- Se edita el fichero /ect/sysconf/network-scripts/ifcfg-eth0 y se agrega el siguiente contenido:

```
DEVICE=eth0
HWADDR=XX:XX:XX:XX:XX:XX # La dirección MAC de la interface eth0
TYPE=OVSPort
DEVICETYPE=ovs
OVS_BRIDGE=br-ex
ONBOOT=yes
```

- Se reinicia el servicio de red con el comando siguiente:

```
# service network restart
```

- Se comprueba la conectividad de la máquina con el exterior a través de br-ex

ANEXO XI. Instalación OpenStack-Ansible

- Se clona el repositorio git de openstack-ansible:

```
# git clone https://github.com/openstack-ansible/openstack-ansible
# cd openstack-ansible
# git submodule update - - init
```

- Se instala OpenStack:

```
openstack-ansible# make
```

- Durante la instalación, los pasos que se desarrollan están definidos en el fichero "*makefile*":
    - Se crean las máquinas virtuales (vagrant up en *testcase/standard*)
    - Se instalan los servicios necesarios en cada una de las máquinas que se han arrancado.
        - La receta ansible se llama *openstack-ansible/openstack.yaml*
        - Ficheros de hosts en *openstack-ansible/testcases/standard/ansible_hosts*
        - Una receta para cada servicio: *keystone, swift, glance, neutron, cinder, nova, horizon, heat y ceilometer*.
    - Se arranca una instancia de prueba (receta ansible *openstack-ansible/demo.yaml*)

ANEXO XII. FUNCIONAMIENTO DE IPERF

Una prueba mediante la utilidad Iperf se ejecuta para una duración determinada (e incluso predeterminada por el software) en un tiempo $\boldsymbol{T}$, durante la cual el cliente Iperf envía paquetes de tamaño fijo $\boldsymbol{L}$, donde el tamaño puede ser especificado por el usuario o con el valor predeterminado (expresado en bytes) de Iperf. Al finalizar la prueba, el servidor Iperf:

- Tiene el recuento total de paquetes $\boldsymbol{N}$
- Conoce la duración de la prueba $\boldsymbol{T}$

Por lo tanto, el rendimiento (throughput) lo calcula a partir de la siguiente fórmula:

$$\boldsymbol{S\ (throughput) = \frac{N \times L \times 8}{T}\ bps}$$

Para el caso de la medición del Jitter, el cliente Iperf envía tráfico a una tasa constante de bits (CBR, Constant Bit Rate), es decir, que los paquetes que se envían están espaciados de forma equidistante por el remitente, en este caso, el cliente Iperf. Por lo tanto, el servidor puede medir la demora entre paquetes recibidos consecutivamente y calcular tanto el promedio $\boldsymbol{\mu}$, como la desviación estándar $\boldsymbol{\sigma}$ de las medidas anteriormente mencionadas, de tal forma que el Jitter queda definido así:

$$\boldsymbol{\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - \mu)^2}}$$

Revisando el código fuente de Iperf3, se tiene que el algoritmo está implementado de la siguiente forma:

```
 /* jitter
measurement
    */
                                gettimeofday(&arrival_time, NULL);


                          transit = timeval_diff(&sent_time, &arrival_time);
                                   d = transit - sp->prev_transit;
                                             if (d < 0)
                                               d = -d;
                                     sp->prev_transit = transit;
                            // XXX: This is NOT the way to calculate jitter
                    //      J = |(R1 - S1) - (R0 - S0)| [/ number of packets, for average]
                                    sp->jitter += (d - sp->jitter) / 16.0;


                                                return r;
```

En Iperf2, el algoritmo está implementado así:

```
// from
RFC
1889,
Real
Time
Protocol
(RTP)
                                    // J = J + ( | D(i-1,i) | - J ) /
                                    // Compute jitter
                                    deltaTransit = transit - stats->transit.lastTransit;
                                    if ( deltaTransit < 0.0 ) {
                                        deltaTransit = -deltaTransit;
                                    }
                                    stats->jitter += (deltaTransit - stats->jitter) / (16.0);
```

Por lo tanto, se concluye que los desarrolladores, tanto de la versión 2 y 3, de Iperf, respectivamente, hacen uso de la RFC 1889 (RTP: A Transport Protocol for Real-Time Applications) para implementar sus propios algoritmos de cómputo con el objetivo de medir el Jitter.

NOTA:

Estrictamente hablando, lo que se mide actualmente en Iperf se denomina (IPDV, Inter-Packet Delay Variation) definido en la RFC 5481 (su valor absoluto). Sin embargo, la conclusión de la RFC 5481 indica, que de entre todos los algoritmos para calcular la fluctuación de fase a partir de IPDV (es decir, sincronización de tiempo), ninguno es completamente satisfactorio. Lo cual lleva a concluir que no hay un algoritmo estándar para computar el Jitter, dadas sus varias formas de definirlo y computarlo.

ANEXO XIII. LENGUAJES DE PROGRAMACIÓN REQUERIDOS PARA MANEJAR OPENSTACK

- Linux

Uno de los pilares en los que se apoya la flexibilidad y potencia del sistema operativo GNU/LINUX es su transparencia, la cual nos permite interactuar a un nivel mucho más profundo con nuestro sistema, teniendo la posibilidad de optimizar recursos o solucionar problemas "*a mano*", esto es a través de comandos de consola, edición de archivos de conFiguración y, si se cuenta con los conocimientos, modificación de código fuente.

- Java

Java es un lenguaje de programación orientado a objetos creado en 1991 y publicado en 1995 por Sun Microsystems (adquirida por Oracle en 2010), con la intención de que los programadores escribieran el código solo una vez y lo ejecutarán en cualquier dispositivo. Y esto es posible, gracias a que Java cuenta con una JVM (Java Virtual Machine) que brinda portabilidad al lenguaje, ya que hoy existen JVMs (Java Virtual Machines) para diferentes arquitecturas para todas las plataformas.

"WORA – write once, run anywhere"

- Node.JS

Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node está diseñado para construir aplicaciones en red escalables. En la siguiente aplicación de ejemplo "hola mundo", se pueden manejar muchas conexiones concurrentes. Por cada conexión el *callback* será ejecutado, sin embargo, si no hay trabajo qué hacer, Node estará en standby.

Esto contrasta con el modelo de concurrencia más común hoy en día, donde se usan hilos del Sistema Operativo. Las operaciones de redes basadas en hilos son relativamente ineficientes y son muy difíciles de usar. Además, los usuarios de Node están libres de preocupaciones sobre el bloqueo del proceso, ya que no existe. Casi

ninguna función en Node realiza I/O directamente, así que el proceso nunca se bloquea. Debido a que no hay bloqueo es muy razonable desarrollar sistemas escalables en Node.

HTTP es ciudadano de primera clase en Node, diseñado con operaciones de streaming y baja latencia en mente. Esto hace a Node candidato para ser la base de una librería o un framework web.

- Python

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de plataformas.

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, https://www.python.org/, y puede distribuirse libremente. El mismo sitio contiene también distribuciones y enlaces de muchos módulos libres de Python de terceros, programas y herramientas y documentación adicional.

Python es uno de los lenguajes preferidos en lo que concierne a Networking (programación de redes), su facilidad de sintaxis y cantidad de librerías predeterminadas como de terceros, lo hacen uno de las primeras opciones cuando se requiere desarrollar un proyecto relacionado con redes de telecomunicaciones.

- Ruby

Es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su elegante sintaxis se siente natural al leerla y fácil al escribirla. En Ruby, todo es un objeto. Se le puede asignar propiedades y acciones a toda información y código. La programación orientada a objetos llama a las propiedades *variables de instancia* y las acciones son conocidas como *métodos*.

Es considerado un lenguaje flexible, ya que permite a sus usuarios alterarlo libremente. Las partes esenciales de Ruby pueden ser quitadas o redefinidas a placer. Se puede agregar funcionalidad a partes ya existentes.

- .NET

Es un framework de Microsoft que hace énfasis en la transparencia de redes, con independencia de la plataforma de hardware y que permite un rápido desarrollo de aplicaciones.

- PHP

Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Por lo que en lugar de usar muchos comandos para mostrar HTML (como en C o PERL), las páginas de PHP contienen HTML con código incrustado que hace “algo”.

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era.

Lo mejor de usar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales.

- Django

Django es un framework web de alto nivel, escrito en Python, que ayuda al desarrollo rápido y a un diseño limpio y pragmático. Construido por desarrolladores experimentados, resuelve una buena parte de los problemas del desarrollo web de tal manera que uno se pueda enfocar en escribir su aplicación (app) sin necesidad de reinventar la rueda. Es gratis y de código abierto.

Sus características más sobresalientes son:

  - Extremadamente rápido: fue diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización lo más rápido posible.

- Muy seguro: toma en serio la seguridad y ayuda a los desarrolladores a evitar muchos errores comunes de seguridad.
- Altamente escalable: algunos de los sitios más activos en la web aprovechan la capacidad de Django para escalar de manera rápida y flexible.

- Django REST Framework

Es un kit de herramientas muy potente y flexible para crear APIs Web. Algunas de las razones para utilizar Django Rest Framework son:

- La Api de navegación web provee una gran usabilidad para sus desarrolladores
- Las políticas de autenticación incluyen paqueterías para OAuth1a y OAuth2
- Serialización que soporta tanto fuentes de datos ORM como no-ORM
- Altamente personalizable
- Amplia documentación y gran apoyo de su comunidad de desarrolladores
- Muy confiable y utilizado de forma internacional por compañías como Mozilla, Red Hat, Heroku y Eventbrite

- cURL

Es un proyecto de software consistente en una biblioteca (libcurl) y un intérprete de comandos (curl) orientado a la transferencia de archivos. Soporta protocolos FTP, FTPS, HTTP, HTTPS, TFTP, Telnet, FILE y LDAP.

También soporta certificados HTTPS, HTTP POST, HTTP PUT, subidas FTP, Kerberos, subidas mediante formulario HTTP, proxies, cookies, autenticación mediante usuario y contraseña (Basic, DIgest, NTLM y Negotiate para HTTP y kerberos 4 para FTP), continuación de transferencia de archivos, tunneling de proxy HTTP y otras prestaciones.

El principal propósito y uso para cURL es automatizar transferencias de archivos o secuencias de operaciones no supervisadas.

- Ansible

Es una plataforma de software libre para conFigurar y administrar computadoras. Combina instalación multi-nodo, ejecuciones de tareas ad hoc y administración de conFiguraciones. Adicionalmente, es caracterizado como una herramienta de orquestación. Maneja nodos a través de SSH y no requiere ningún software remoto adicional (excepto Python 2.4 o posterior para instalarlo). Dispone de módulos que trabajan sobre JSON y la salida estándar puede ser escrita en cualquier lenguaje. Nativamente utiliza YAML para describir conFiguraciones reusables de los sistemas.

ANEXO XIV. TECNOLOGÍAS DE REDES UTILIZADAS CON OPENSTACK

Dentro de las tecnologías de redes se encuentran los túneles GRE y VXLAN usados en el plug-in ML2. Por otra parte, se mencionará OpenVswitch y los nombres de espacios de red (net namespaces).

Túneles GRE

GRE significa “Generic Routing Encapsulation” y define un protocolo para el establecimiento de túneles entre dos nodos no conectados directamente. El protocolo fue originalmente desarrollado por Cisco y permite transportar paquetes de una red concreta a través de otra red diferente.

El ejemplo más común donde se usan los protocolos de túnel como GRE son para conectar oficinas de una misma organización separadas geográficamente. En este caso el túnel IP viaja cifrado y se conoce como VPN. GRE tradicionalmente se ha usado para transportar una red IP interna sobre otra red IP externa sin cifrado.

En el entorno de OpenStack los túneles GRE transportan la red virtual IP sobre una red física IP, es decir, conectan las instancias entre si y hacia los routers virtuales a través de diferentes hosts físicos. Por otra parte, el campo “tunnel ID” de la cabecera GRE permite diferenciar entre redes de distintos tenants.

Túneles VXLAN

VXLAN significa “Virtual Extensible LAN” y define un protocolo para el establecimiento de túneles o redes superpuestas (overlay networks). VXLAN se desarrolló explícitamente en entornos de virtualización de redes en sistemas Cloud. Encapsula paquetes Ethernet L2 dentro de paquetes UDP L4 a través de una técnica de etiquetado similar al VLAN. Básicamente. Es usada para crear una red plana L2 para máquinas virtuales alojadas en diferentes hosts y redes físicas.

VXLAN ejecuta la misma función que los túneles GRE, a través del plug-in ML2 conecta todas las instancias de un mismo tenant en una red virtual IP superpuesta a la red física IP. Los túneles VXLAN pueden escalar mejor que los GRE, pero también son más complejos de conFigurar.

Veth

Veth significa “Virtual Ethernet Device” y es un dispositivo que emula un enlace Ethernet virtual. Presenta dos extremos, normalmente uno en un sistema host y otro dentro de un espacio de nombres de red aislado. Todos los paquetes que entran por un extremo salen automáticamente por el otro.

OpenVswitch

OpenVswitch (OVS) es software Open Source que funciona como un switch virtual multicapa diseñado para comunicar las instancias virtuales en entornos de virtualización. Su principal uso ha sido reenviar el tráfico entre diferentes máquinas virtuales en el mismo host físico o de comunicar dichas máquinas virtuales con la red física.

Entre las funcionalidades que soporta OpenVswitch podemos destacar:

- VLAN 802.1Q, puertos troncales y de acceso
- NetFlow y sFlow
- OpenFlow 1.0 y 1.3
- STP
- QoS
- Protocolos de túnel como GRE, VXLAN, IPSec

Espacios de nombres de red

Un espacio de nombres de red (net namespaces) es un contenedor abstracto que almacena nombres o identificadores únicos. El identificador definido en un espacio de nombres está asociado a dicho espacio de nombres. El mismo identificador puede ser definido en otros espacios de nombres, pero identificará un objeto diferente. El kernel Linux da soporte para varios espacios de nombres como: PID namespaces (procesos), MNT namespaces (puntos de montaje) y NET namespaces (redes). De esta forma, los espacios de nombres de red son un grupo determinado de direcciones IP, interfaces, tablas de rutas, reglas IPTables, etcétera que comparten todos los procesos dentro del espacio de nombres de red en cuestión. Por ejemplo, un contenedor LXC utiliza un espacio de nombres de red diferente a su sistema host y cada tenant tiene su propio espacio de nombres de red para sus redes privadas.

ANEXO XV. SCRIPT PARA DESPLIEGUE AUTOMÁTICO DE RED EN OPENSTACK

```
#Script para despliegue automático de la red de la Figura 45

# Creamos tres redes

NET_ID1=$(neutron net-create red1|grep ' id '|awk '{print $4}')
NET_ID2=$(neutron net-create red2|grep ' id '|awk '{print $4}')
NET_ID3=$(neutron net-create red3|grep ' id '|awk '{print $4}')

# Creamos tres routers

neutron router-create r1
neutron router-create r2
neutron router-create r3

# Cramos una subred para cada una de las redes

neutron subnet-create --name subred1 red3 192.168.1.0/24
neutron subnet-create --name subred2 red2 192.168.2.0/24
neutron subnet-create --name subred3 red1 192.168.3.0/24

# Conectamos r1 a internet y a la red1

neutron router-gateway-set r1 public
neutron router-interface-add r1 subred1

# Creamos dos puertos en la subred1 y en la subred2, que posteriormente conectaremos a r2

PORT_ID1=$( neutron port-create red1 --fixed-ip ip_address=192.168.1.2|grep ' id '|awk '{print $4}')
neutron router-interface-add r2 port=$PORT_ID1
neutron router-interface-add r2 subred2

# Conectamos r3 a la red2 y a la red3

PORT_ID2=$( neutron port-create red2 --fixed-ip ip_address=192.168.2.2|grep ' id '|awk '{print $4}')
neutron router-interface-add r3 port=$PORT_ID2
neutron router-interface-add r3 subred3

# Añadimos rutas al servidor DHCP de la red1 para que envíe a las intancias un camino para llegar a la red2 y a
la red3.
neutron subnet-update subred1 --host_routes type=dict list=true
destination=192.168.2.0/24,nexthop=192.168.1.2 destination=192.168.3.0/24,nexthop=192.168.1.2
# Añadimos una ruta al servidor DHCP de la red2 para que envíe a las intancias un camino para llegar a la red3.
neutron subnet-update subred2 --host_routes type=dict list=true
destination=192.168.3.0/24,nexthop=192.168.2.2

# Creamos las reglas de seguridad (Permitimos IVMP y SSH)

neutron security-group-rule-create --direction ingress --protocol icmp --remote-ip-prefix 0.0.0.0/0 default
neutron security-group-rule-create --direction ingress --protocol tcp --port-range-min 22 --port-range-max 22 --
remote-ip-prefix 0.0.0.0/0 default
```

```
# Creamos una clave ssh y le cambiamos los permisos

nova keypair-add ipn > ipn.pem
chmod 600 ipn.pem

# Creamos tres instancias cirros conectadas a cada una de las redes

nova boot --flavor m1.tiny2 --image cirros \
        --security-groups default\
              --key-name mi_clave \
        --nic net-id=$NET_ID1 \
        pc1

nova boot --flavor m1.tiny2 --image cirros \
        --security-groups default\
              --key-name mi_clave \
        --nic net-id=$NET_ID2 \
        pc2

nova boot --flavor m1.tiny2 --image cirros \
        --security-groups default\
              --key-name mi_clave \
        --nic net-id=$NET_ID3 \
        pc3

# Asignamos una IP flotante a pc1

IP=$(nova floating-ip-create|awk 'NR==4'|awk '{print $2}')
nova floating-ip-associate pc1 $IP

# Añadimos una ruta de encaminamiento en r2 para que poueda acceder a la red3

neutron router-update r2 --routes type=dict list=true destination=192.168.3.0/24,nexthop=192.168.2.2

# Añadimos una ruta de encaminamiento en r3 para que poueda acceder a la red1

neutron router-update r3 --routes type=dict list=true destination=192.168.1.0/24,nexthop=192.168.2.1
```

ANEXO XVI. CONSIDERACIONES RESPECTO A CPU, RAM Y ALMACENAMIENTO DE LOS SERVIDORES FÍSICOS QUE EJECUTEN OPENSTACK PARA UN ENTORNO DE PRE-PRODUCCIÓN

Análisis de CPU

El número de CPU de máquinas virtuales (vCPU) asignados en comparación con el número de núcleos físicos de CPU disponibles, es conocida como la relación vCPU-to-core.

Una relación típica vCPU-to-core para un servidor con cargas de trabajo promedio es de aproximadamente 4:1, es decir, cuatros vCPU asignadas para cada núcleo físico.

Para este proyecto, considerando que el CPU se mantiene en un uso promedio del 60% de su capacidad, y considerando un crecimiento del 25% a futuro, es necesario un solo servidor con una instalación “todo en uno”, para soportar una carga de trabajo promedio del 60%.

Por lo tanto, para una implementación real, se requieren al menos 4 servidores físicos según la documentación de OpenStack, y cada uno de ellos debe tener al menos 16 núcleos.

Capacidad de hosts por memoria

Existen varios factores que deben ser considerados en el cálculo de los requisitos de memoria; estos factores incluyen:

1. La cantidad de memoria necesaria para soportar las cargas de trabajo actuales. Para este proyecto se han tomado los resultados del análisis obtenido en el punto anterior.
2. La cantidad de memoria necesaria para que el sistema escale en un futuro, es un factor altamente variable de acuerdo a los objetivos que se vayan planteando los administradores de la red, sin embargo, de manera promedio se usará un 25%.
3. La cantidad de memoria necesaria en caso de sobrecarga de la memoria de los equipos de red virtuales, también debe tenerse en cuenta en el cálculo de los requisitos de memoria. La cantidad de memoria requerida para una sobrecarga depende de la conFiguración de la máquina virtual. El número de CPU virtuales (vCPUs) asignados al equipo virtual, la cantidad de memoria asignada al equipo virtual y el hardware virtual conFigurado para el equipo de red virtual, tendrá un

impacto en la cantidad de memoria necesaria para uso general. De forma típica, la sobrecarga de memoria requerida para un equipo virtual es entre 20 MB y 150 MB. Lo anterior puede parecer una pequeña cantidad de memoria, pero a lo largo de decenas e incluso cientos de equipos de red virtuales, puede tener un impacto significativo en la cantidad total de memoria requerida.

4. El umbral máximo de utilización de la memoria, debe ser determinada para los recursos de memoria. Dicho umbral define el porcentaje máximo del total de recursos de memoria que podrá ser utilizada. Si el umbral máximo de utilización es 75%, un adicional de 25% necesario de los recursos de memoria será añadido con el fin de calcular el total de los recursos de memoria necesarios.

Capacidad de almacenamiento

Determinando Storage Performance requerido

El rendimiento de una unidad de almacenamiento se mide en I/O por segundo (IOPS). Una solicitud de lectura en disco o una solicitud de escritura en disco es igual a un I/O.

Los IOPS necesarios para apoyar una solicitud se calcula basándose en el porcentaje de lectura de I/O, el porcentaje de escritura de I/O y la penalidad de escritura del arreglo de unidades de almacenamiento (en caso de implementarlas) en la carga de trabajo donde será alojado.

De acuerdo a la documentación de OpenStack, la cantidad de IOPS requerido es de un mínimo de 10 discos de 15k o viceversa, es decir 15 discos de 10K. En ambos casos, los discos SAS/FC cubren esa necesidad. Sin embargo, ante la caída de precios de las unidades SSD, se recomienda el uso de éstas últimas, ya que el rendimiento en IOPS de los discos de 10k/15k es de 175 IOPS, en contraste con el rendimiento de una unidad SSD que es de mínimo 2500 IOPS.

Calculando capacidad de almacenamiento requerido

La capacidad se mide en Gigabytes (GB) o en Terabytes (TB). La capacidad debe incluir el espacio total que se necesita para apoyar el actual, el espacio necesario para soportar el crecimiento, el espacio necesario para archivos de intercambio (swap files) de máquinas

virtuales, y el espacio de holgura adicional para las instantáneas (snapshots) y otros datos de la máquina virtual.

Se debe realizar el análisis de cuánto espacio de almacenamiento se requiere para el manejo de las imágenes de las instancias que serán ejecutadas, con esto se obtiene un promedio de la capacidad de almacenamiento necesaria para soportar la carga actual de demanda de espacio, a eso se le añade un 25% de capacidad de escalamiento, una holgura del 20% que VmWare recomienda para diferentes tareas administrativas y la capacidad de almacenamiento usada en la memoria virtual de intercambio (vSwap) por cada máquina virtual alojada.

ConFiguración de Networking

En esta sección se proporcionan los detalles de la conFiguración de red para los servidores físicos y los equipos de red virtuales. En la implementación de la infraestructura virtual se usan los tipos de conexión de red: la red de máquinas virtuales y la red de administración o red Provider.

Todas las máquinas virtuales estarán conectadas a switches virtuales (OVS), en los cuales están creados los diferentes grupos que conforman la red de prueba.

GLOSARIO

API (Interfaz de programación de aplicaciones)

Una API expresa un componente de software en términos de sus entradas, salidas y operaciones para manipular y controlar un componente de software mediante programación.

CLI (Interfaz de línea de comando)

Es una manera de interactuar con un programa informático o un dispositivo de red mediante el cual el usuario emite comandos en forma de líneas de texto sucesivas.

Controlador

Un componente de software lógicamente centralizados que proporciona funcionalidad de administración de red (y control de red). Establece las políticas y reglas relativas al reenvío de paquetes y conFigura la infraestructura de red para realizar el envío.

DC (Centro de datos)

Es una instalación utilizada para albergar sistemas informáticos y componentes asociados. Incluye servidores, almacenamiento, redes, energía, aire acondicionado y seguridad.

Dominio

Un componente de red con gestión centralizada. Puede ser tan pequeño como una sola función o dispositivo de red o tan amplio como una red completa.

ETSI (Instituto Europeo de Normas de Telecomunicaciones)

Es una organización de normalización independiente y sin fines de lucro en la industria de las telecomunicaciones (fabricantes de equipos y operadores de redes) que tiene su sede en Europa. El ISG para NFV del ETSI elaboró el libro blanco original que define la NFV y definió la arquitectura de despliegue de NFV.

GUI (Interfaz gráfica de usuario)

Una GUI es una forma gráfica de interactuar con un programa informático o un dispositivo de red. La visualización es gráfica y, por lo general, los comandos se controlan con un mouse.

IaaS (Infraestructura como servicio)

Es una forma de computación en la nube donde la infraestructura informática; es decir, la computación o el almacenamiento; se proporciona a un usuario final como un servicio basado en la nube.

IP (Protocolo de Internet)

Es el principal protocolo de comunicaciones en el conjunto de protocolos de Internet para transportar datagramas a través de una red.

MANO (Gestión y Orquestación)

Es el componente de la arquitectura NFV que controla cómo se establecen las conexiones en cadena o interconexiones de una o más VNF en un servicio de extremo a extremo.

NaaS (Red como servicio)

Es una forma de computación en la nube donde la conexión en red y la conectividad se proporcionan a un usuario final como un servicio.

NE (Elemento de red)

Son dispositivos de red individuales que se administran.

NETCÉTERAONF (Protocolo de conFiguración de red)

Es un protocolo de gestión de red que se utiliza para conFigurar dispositivos de red.

NFV (Virtualización de funciones de red)

Es un concepto de arquitectura de red que utiliza las tecnologías comerciales listas para usar, incluidos los sistemas de almacenamiento y computación, para virtualizar clases enteras de funciones de nodos de red utilizadas para crear servicios de comunicación.

NFVO (Orquestación de la virtualización de funciones de red)

Es un componente de software que puede orquestar el ciclo de vida de las funciones de red virtualizadas. Esto incluye la creación y conexión en cadena de las funciones de red virtualizadas.

OpenFlow

Un protocolo de comunicaciones para controlar mediante programación el plano de reenvío de un conmutador o enrutador de red. Supone una separación del plano de control del plano de datos y dirige el flujo de paquetes a través de comandos de coincidencia de patrones o acciones.

Orquestador

Componente de software utilizado para proporcionar control de extremo a extremo a través de una red. Se considera que un orquestador ofrece mayor nivel de perspectiva que un controlador. Por ejemplo, puede proporcionar una simplificación de servicios de extremo a extremo, en la que un controlador puede centrarse en el reenvío y el control de paquetes.

PoD (Punto de suministro)

Una unidad independiente que incluye computación y almacenamiento. Pod de NFV se refiere a un componente independiente basado en x86 cuya función es ejecutar las VNF.

Python

Lenguaje de programación ampliamente utilizado, de alto nivel, multiuso, interpretado y dinámico que proporciona conceptos diseñados para facilitar la legibilidad y permitir borrar programas en pequeña y gran escala.

Recurso

Cualquier entidad que proporciona un conjunto bien definido de funcionalidades de red que se pueden conFigurar y controlar a través de una API. Un recurso puede ser un dispositivo individual o una función de red, bien o toda una red entera o un dominio de red.

REST (Transferencia de estado representacional: RESTful API)

Es un estilo de arquitectura y un enfoque de comunicaciones que suele utilizarse en el desarrollo de servicios web. REST es un protocolo de comunicaciones sin estado, cliente-servidor, caché que, en prácticamente todos los casos, utiliza el protocolo HTTP.

SDN (Redes definidas por software)

Es un enfoque de redes informáticas que permite a los administradores de red, administrar la funcionalidad abstracta de nivel superior de los servicios de red. Esto se realiza mediante la separación del plano de control y el plano de datos.

VNF (Función de red virtual)

Es una función de red que ha sido virtualizada. Una VNF es diferente a la NFV. VNF hace referencia a una instancia o implementación de una función de red en un software que está separado del hardware subyacente.

VNFI (Infraestructura de VNF)

Es la infraestructura informática en la que se ejecuta una VNF

vRouter (Enrutador virtual)

Es una versión virtualizada de la funcionalidad del enrutador

X86 (Arquitectura del procesador Intel x86)

Es una familia de arquitecturas de conjuntos de instrucciones compatibles con versiones anteriores basadas en la CPU Intel 8086. X86 se utiliza comúnmente para referirse a los servidores disponibles para NFV.

XML (Lenguaje marcado extensible)

Es un lenguaje marcado legible por humanos. XML está diseñado para ser simple y, por lo general, se utiliza para describir documentos y estructuras de datos arbitrarios.

REFERENCIAS

A Mankin, S. B. (Enero de 1995). IETF RFC1752. *The recommendation for the IP next generation Protocol*. Estados Unidos. Recuperado el Abril de 2018, de https://tools.ietf.org/html/rfc1752

Alok Shrivastwa, S. S. (2016). *OpenStack: Building a Cloud Environment.* Birmingham-Mumbai: Packet Publishing. Recuperado el Diciembre de 2018, de https://books.google.com.mx/books?id=7YJcDgAAQBAJ&printsec=frontcover&hl=es#v=onepage&q&f=false

Barragán, O. E. (2019). *Propuesta del despliegue del protocolo IPv6 en la red de telecomunicaciones de la ESIME Zacatenco.* Ciudad de México: Instituto Politécnico Nacional. Recuperado el Febrero de 2019

Ben Silverman, M. S. (2018). *OpenStack for Architects* (Second ed.). Birmingham, UK: Packet Publishing. Recuperado el Noviembre de 2018, de https://books.google.com.mx/books?id=KHxeDwAAQBAJ&printsec=frontcover&dq=openstack&hl=es&sa=X&ved=0ahUKEwiShJHKy6biAhVEYKwKHSjUDD4Q6AEIhAEwCg#v=onepage&q=openstack&f=false

Benites, A. G. (26 de Agosto de 2016). *DevCode*. Recuperado el 28 de 03 de 2019, de ¿Qué es Java y por qué es aprenderlo?: https://devcode.la/blog/que-es-java/

Denton, J. (2015). *Learning OpenStack Networking (Neutron)* (Second ed.). Birmingham, UK: Packet Publishing. Recuperado el Noviembre de 2018, de https://books.google.com.mx/books?id=cfWoCwAAQBAJ&printsec=frontcover&dq=openstack&hl=es&sa=X&ved=0ahUKEwi1rYG5zqbiAhUGI6wKHVixDys4ChDoAQg4MAI#v=onepage&q=openstack&f=false

Django® Software Foundation and contributors. (02 de Noviembre de 2017). *Django 1.8*. Recuperado el 28 de Marzo de 2019, de Django docs: https://www.djangoproject.com/

Encode OSS Ltd®. (06 de Octubre de 2017). *django RESTframework®*. Recuperado el 28 de Marzo de 2019, de Funding: https://www.django-rest-framework.org/

ESnet. (01 de Marzo de 2014). *iPerf3: A TCP, UDP, and SCTP network bandwidth measurement tool*. Recuperado el 29 de Marzo de 2019, de https://github.com/esnet/iperf

Guide, C. Q. (2019). *Shiwang Kalkhanda* (First ed.). Birmingham, UK: Packet Publishing. Recuperado el Enero de 2019, de https://books.google.com.mx/books?id=yZ2BDwAAQBAJ&printsec=frontcover&dq=centos&hl=es-419&sa=X&ved=0ahUKEwjbq63Fz6biAhUFOK0KHafjDoY4ChDoAQgxMAE#v=onepage&q=centos&f=false

Hervás, Ó. R. (2014). Software defined networking. (U. P. Catalunya, Ed.) Cataluña, España. Recuperado el Junio de 2018, de https://upcommons.upc.edu/bitstream/handle/2099.1/21633/Memoria.pdf?sequence=4

Intel Latinoamérica. (11 de Octubre de 2017). *Asistencia*. Recuperado el 03 de Marzo de 2019, de Requisitos de la tecnología de virtualización Intel®: https://www.intel.la/content/www/xl/es/support/articles/000005758/boards-and-kits/desktop-boards.html

iPerf®. (2019). The network bandwidth measurement tool. Francia. Recuperado el Marzo de 2019, de https://iperf.fr/

Kevin Jackson, C. B. (2018). *OpenStack Cloud Computing Cookbook.* Birmingham, UK: Packet Publishing. Recuperado el Enero de 2018, de https://books.google.com.mx/books?id=ryZKDwAAQBAJ&pg=PA37&dq=openstack&hl=es&sa=X&ved=0ahUKEwiShJHKy6biAhVEYKwKHSjUDD4Q6AEIXzAG#v=onepage&q&f=false

Khedler, O. (2015). *Mastering OpenStack.* Birmingham, UK: Packet Publishing. Recuperado el Enero de 2019, de https://books.google.com.mx/books?id=m1tICgAAQBAJ&pg=PA85&dq=openstack&hl=es&sa=X&ved=0ahUKEwiShJHKy6biAhVEYKwKHSjUDD4Q6AEINDAB#v=onepage&q=openstack&f=false

Markelov, A. (2016). *OpenStack Administrator Guide.* Stockholm, Sweden: Apress. Recuperado el Octubre de 2018, de https://books.google.com.mx/books?id=ekFxDQAAQBAJ&pg=PA5&dq=openstack&hl=es&sa=X&ved=0ahUKEwiShJHKy6biAhVEYKwKHSjUDD4Q6AEITTAE#v=onepage&q&f=false

Newman, D. (2014). Technology validation experiment: IPv6 and multicast support on OpenFlow. Southampton, Inglaterra. Recuperado el Marzo de 2018, de http://users.ecs.soton.ac.uk/drn/ofertie/tve_ipv6_and_ multicast.pdf

Node.js® Foundation. (27 de Julio de 2016). *Node.js®*. Recuperado el 28 de 03 de 2019, de Acerca de Node.js®: https://nodejs.org/es/about/

Oasis-open.org. (2014). OASIS Key Management Interoperability Protocol (KMIP). Recuperado el Enero de 2019, de https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=kmip

Olaya-Yandun, M. E. (2015). *Diseño e implementación de una aplicación para balanceo de carga para una Red Definida por Software (SDN).* Quito, Ecuador: Escuela Politécnica Nacional. Recuperado el Septiembre de 2018

OpenStack.org. (08 de 05 de 2019). Secrets Management. (O. Docs, Ed.) Recuperado el Marzo de 2019, de https://docs.openstack.org/security-guide/secrets-management.html#secrets-management

Openstack.org. (13 de 05 de 2019). Welcome to barbican's Developer Documentation! Recuperado el Marzo de 2019, de https://docs.openstack.org/barbican/latest/

OSRG. (2013). *RYU*. Recuperado el Septiembre de 2018, de RYU component-based software defined networking framework: https://github.com/osrg/ryu

Pradeep Kumar Singh, M. K. (2017). *Containers in OpenStack.* Birmingham, UK: Packet Publishing. Recuperado el Diciembre de 2018, de https://books.google.com.mx/books?id=NPNFDwAAQBAJ&pg=PA52&dq=openstack&hl=es&sa=X&ved=0ahUKEwiShJHKy6biAhVEYKwKHSjUDD4Q6AEIRDAD#v=onepage&q&f=false

Python Software Foundation. (22 de Julio de 2009). *Python 3.6.3*. Recuperado el 28 de Marzo de 2019, de Introducción: http://docs.python.org.ar/tutorial/3/real-index.html

R. Enns, E. (Diciembre de 2006). NETCÉTERAONF ConFiguration Protocol. (J. Networks, Ed.) Estados Unidos de América. Recuperado el Abril de 2018, de https://tools.ietf.org/html/rfc4741

Team, W. (2019). Wireshark User's Guide. Recuperado el Abril de 2019, de https://www.wireshark.org/docs/wsug_html/

Tom Fifield, D. F. (2014). *OpenStack Operations Guide: Set Up and Manage your OpenStack Cloud.* United States of America: OpenStack Foundation. Recuperado el Enero de 2019, de https://books.google.com.mx/books?id=aA5pAwAAQBAJ&printsec=frontcover&dq=openstack&hl=es&sa=X&ved=0ahUKEwiShJHKy6biAhVEYKwKHSjUDD4Q6AEIVTAF#v=onepage&q=openstack&f=false

Unipython. (23 de Noviembre de 2018). *Curso de Network*. Recuperado el 28 de Marzo de 2019, de Redes con Python: https://unipython.com/curso-de-network-o-redes/

VmWare. (Mayo de 2015). *VmWare Blogs*. Recuperado el 12 de Abril de 2018, de OpenStack Files: https://blogs.vmware.com/openstack/files/2015/05/vmware-openstack-history-1024x576.png

VmWare. (Abril de 2019). NSX Data Center. Estados Unidos de América. Recuperado el Abril de 2019, de https://www.vmware.com/products/nsx.html

Wireshark. (24 de Junio de 2008). *Wireshark Go Deep*. Recuperado el 23 de Febrero de 2019, de https://www.wireshark.org/