ESCUELA TÉCNICA SUPERIOR DE INGENIEROS

INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

ESTUDIO Y EVOLUCIÓN DE LA GESTIÓN DE RED HACIA LOS ENTORNOS CLOUD

(Study and evolution of network management towards Cloud environments)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Eduardo López Martínez

Julio – 2017

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Eduardo López Martínez
Director del TFG: José Ángel Irastorza Teja
Alberto Eloy García Gutiérrez
Título: "Estudio y evolución de la gestión de red hacia los entornos cloud"
Title: "Study and evolution of network management towards cloud environments"

Presentado a examen el día: 13 de Julio de 2017

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): García Arranz, Marta
Secretario (Apellidos, Nombre): Irastorza Teja, José Ángel
Vocal (Apellidos, Nombre): García García, José Ángel

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Trabajo Fin de Grado Nº (a asignar por Secretaría)

AGRADECIMIENTOS

A mi familia y amigos, en especial a mis padres Eduardo y Silvia y a mi hermana Raquel quienes me han estado apoyando durante todos estos años y cuyo apoyo ha sido, es, y seguirá siendo fundamental en mi vida, y por enseñarme que con esfuerzo y constancia todo se consigue.

Y mi agradecimiento a los profesores que he tenido a lo largo de mis estudios en la Universidad de Cantabria.

RESUMEN

Este trabajo consta del estudio y evolución de la gestión de redes en los entornos en la nube, en el que se contrastan las principales diferencias entre la gestión de un entorno tradicional y la gestión de un entorno en la nube y como su cambio ha venido adquiriendo un papel cada vez más importante.

Se tratarán las principales áreas de gestión de red aplicadas a un entorno de red en la nube y a modo de ejemplo se tomará la solución de Openstack para implementar una red en una nube privada y monitorizarla a través de Nagios.

ABSTRACT

This work consists of the study and evolution of network management in cloud environments, which contrasts the main differences between the management of a traditional environment and the management of an environment in the cloud and how their change has been acquiring an increasingly important role.

The main areas of network management applied to a network environment in the cloud will be discussed and as an example, the Openstack solution will be taken to deploy a network in a private cloud and monitor it through Nagios.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Acrónimos

ARP: Address Resolution Protocol

API: Application Programming Interface

CDP: Cisco Discovery Protocol

CVN: Cloud-based Virtual Networks

ICMP: Internet Control Message Protocol

ISO: International Organization for Standardization

ISP: Internet Service Provider

LAN: Local Area Network

NAT: Network Address Translation

NIDS: Network Intrusion Detection System

NMS: Network Monitoring System

SNMP: Simple Network Management Protocol

TCP: Transmission Control Protocol

VLAN: Virtual Local Area Network

VM: Virtual Machine

VNC: Virtual Network Computing

VPN: Virtual Private Network

Capítulo 1

Introducción

Esta introducción está estructurada en 3 apartados básicos en los que se presenta el trabajo realizado, para ello se explican primero las motivaciones surgidas, los objetivos perseguidos a lo largo del mismo y finalmente como se ha dividido el proyecto en los distintos capítulos y apartados.

1.1 Motivación

En un mundo globalizado e informatizado, el crecimiento exponencial del número de usuarios es notarialmente visible. Hoy en día estos usuarios buscan una mayor agilidad, movilidad, interactuar con otros usuarios, disponer fácilmente de distintos recursos y servicios y también un ahorro en los costes. Por ello se observan importantes cambios en el modelo de negocio y también a nivel de uso personal. Cuando se accede a estos servicios o recursos se pretende poder hacerlo en cualquier lugar, de forma rápida y sencilla, y esto es posible gracias a la nube.

El modelo de *Cloud Computing* es considerado un modelo de pago por uso que permite un acceso cómodo y bajo demanda a un conjunto compartido de recursos informáticos configurable, como redes, servidores, almacenamiento, aplicaciones y servicios, que se puede desplegar y utilizar de forma rápida y fácil.

Cloud Computing es una de las tendencias en el mercado TI de mayor crecimiento durante los próximos años, y es por ello por lo que se necesita

establecer unos mecanismos de gestión que permitan a las empresas un uso eficiente de este nuevo modelo.

Una de las razones fundamentales de la expansión del Cloud Computing es el ahorro de costes. Entre otras razones porque los servicios cloud evitan o minimizan las inversiones hardware y software. Sin embargo, es necesario gestionarlo adecuadamente, aspecto que no siempre se tiene en cuenta. A lo largo de este trabajo se analizará porqué es importante su gestión, comparándolo con un entorno de red tradicional y que herramientas se pueden encontrar para ello.

Esta gestión y control sobre ciertos parámetros como el rendimiento, los consumos o sobre la cantidad de elementos que forman cada una de las partes que componen la nube como servidores, switches, routers, clusters, equipos, sensores en el CPD de control de temperatura, humedad... debe ser automatizada, porque de lo contrario se vuelve insostenible y de esta forma si surge alguna incidencia se puede actuar más rápidamente y solventarlo lo antes posible, lo que se conoce como *Trouble shooting*[1].

Los proveedores de Cloud Computing se ven comprometidos a ofrecer un grado de servicio muy alto, por no decir del 100% a sus clientes. Por ello el control, el mantenimiento y la gestión se hace una parte imprescindible. Por otra parte, el cliente de estos servicios lleva a cabo una gestión de su nube, más enfocada hacia el ámbito software.

La gestión de la nube es difícil de tratar globalmente, ya que se compone de una gran cantidad de servicios diversos que se gestionan de forma distinta.

En este trabajo se abordará como gestionar los tipos de servicio más relevantes, con el objetivo de obtener una visión global de la gestión de red en la nube, que valdría como ejemplo de gestión de cualquier aplicación ejecutada desde la nube, desde el punto de vista de cinco grandes áreas funcionales de gestión y también se presentarán algunas herramientas de gestión de red más destacadas desde un entorno tradicional a uno en la nube y como prueba se implementará una red privada mediante Openstack y su monitorización a través de Nagios.

---

[1] El *Trouble Shooting* es una forma de resolución de problemas que a menudo se aplica para reparar los productos o procesos fallidos en una máquina o un sistema.

1.2 Objetivos

El objetivo de este Trabajo de Fin de Grado es estudiar como los entornos tradicionales de gestión evolucionan para adaptarse a la gestión de las redes en cloud y explicar qué es necesario gestionar en la nube, cómo se debe gestionar y qué herramientas existen para ello, profundizando particularmente en la creación de una red cloud mediante OpenStack y su monitorización con Nagios.

1.3 Estructura

Este trabajo se compone de 7 capítulos que se han estructurado de la siguiente manera:

- **Capítulo 1: Introducción:** Breve introducción al tema de este proyecto en el que se incluyen las motivaciones, los objetivos que se persiguen, así como la propia estructura del trabajo.
- **Capítulo 2: Estado del arte:** Se contextualiza la gestión de red en un entorno cloud incluyendo la evolución de la gestión desde un entorno tradicional hasta la gestión que se realiza en la nube, resaltando los conceptos más fundamentales.
- **Capítulo 3: Gestión de red en la nube:** Se trata en profundidad cómo gestionar la red en la nube, según las cinco grandes áreas funcionales de gestión, así como los principales problemas de gestión al abordar un entorno cloud.
- **Capítulo 4: Aplicaciones y herramientas de gestión de red:** Se explican algunas de las herramientas de gestión más relevantes, así como su evolución y adaptación a un modelo en la nube.
- **Capítulo 5: Implementación de un entorno cloud sobre OpenStack:** Se toma la solución de OpenStack para realizar una implementación de red en la nube y la creación de instancias en ella.
- **Capítulo 6: Creación de un entorno de gestión basado en Nagios sobre Devstack:** Se explicará cómo instalar y configurar Nagios para monitorizar la red creada en el anterior capítulo.
- **Capítulo 7: Conclusiones y líneas futuras:** Finalmente se concluye con las explicaciones, ideas y soluciones necesarias encontradas a lo largo del trabajo.

Capítulo 2

Estado del arte

En este capítulo se introducen los principales conceptos para entender la necesidad y la aparición del entorno en la nube.

Para ello se comienza dando una perspectiva histórica de cómo se ha llegado a este punto en el que se puede disfrutar de servicios en la nube y cómo ha evolucionado la gestión de red desde un entorno tradicional hasta un entorno cloud. También se clasificarán los entornos cloud siguiendo dos directrices, una primera en función de los servicios que se ofrecen y otra segunda clasificación según el tipo de despliegue de red.

Finalmente se tratará el desarrollo de estándares en la nube, así como la necesidad de su estandarización y también los principales proveedores de servicios cloud, así como las ventajas y desventajas del uso de la nube.

2.1 ¿Qué es el cloud?

Cuando se habla de *cloud* hay que decir que no se trata de una tecnología, sino una manera de acceder a unos recursos para el día a día como individuo o como empresa, aunque se basa en tecnologías variadas para implementar el modelo.

Se basa en un modelo de relación entre un consumidor y un proveedor, el primero busca un acceso inmediato a unos recursos, que se basa en un pago por

uso en función de la demanda y el proveedor es quien se encarga de proporcionar estos recursos y servicios.

El usuario es responsable del uso y consumo que hace del servicio prestado, siendo el prestador el responsable de implementarlo, ejecutarlo y entregarlo. Siempre que no sea gratuito, el usuario paga por el servicio cloud unas tarifas determinadas en función de su consumo.

El proveedor afronta una fuerte inversión para ofrecer esos servicios, que requieren un esfuerzo de automatización por su parte, ya que el consumidor busca acceder a recursos de manera automatizada sin la intervención de una persona, pero manteniendo el control sobre estos recursos. En la Figura 1 se observa la diferencia de la arquitectura entre un modelo tradicional y un modelo en la nube.

Este modelo en la nube persigue una gran escalabilidad, flexibilidad, disponibilidad, agilidad, pago por uso y una gran seguridad.

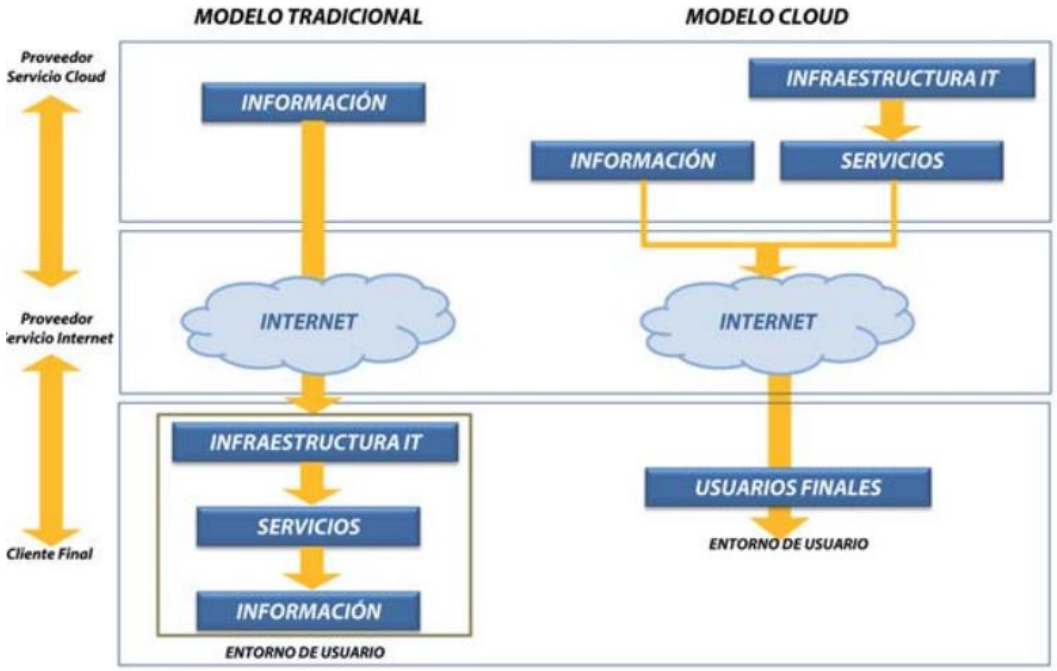

Figura 1: Modelo tradicional y modelo cloud

2.2 Perspectiva histórica

Desde una perspectiva histórica se pueden identificar seis distintas fases hasta la aparición del cloud. [1]

En una primera fase, se solían usar terminales para conectarse a unidades de proceso de gran potencia compartidos por muchos usuarios.

La fase 2 vino marcada por la aparición de los ordenadores personales, los cuales acabaron convirtiéndose en máquinas lo suficientemente potentes para satisfacer las necesidades de los usuarios en su trabajo diario, lo que permitió al usuario tener más independencia.

La fase 3 la marcó el comienzo de las redes informáticas, que permitieron que varios ordenadores se conectaran unos con otros. Cualquiera podía trabajar en un PC y conectarse a otros equipos a través de redes locales para compartir los recursos.

La fase 4 se distinguió por la interconexión entre las redes locales para establecer una red mundial. Los usuarios pudieron entonces conectarse a Internet para utilizar aplicaciones y recursos remotos.

La fase 5 trajo consigo el concepto de una malla global para facilitar compartir la potencia de cálculo y el almacenamiento de recursos, conocido como cálculo distribuido o *grid computing*[2].

En general en las fases anteriores, la gente solía tener acceso a un PC y a las redes de computadores de una manera transparente. Ahora, en la fase 6, el Cloud Computing permite disfrutar de todos los recursos disponibles en Internet en una solución escalable y de manera simple.

Cuando se habla de cloud, se refiere a un nivel conceptual, una nube en Internet, en la cual se alojan todos los recursos disponibles, tanto hardware como software, y todos los servicios existentes, que es accesible mediante un interfaz estándar que puede ser por ejemplo un navegador. A medida que los

[2] Un *grid* es un sistema de computación distribuido que permite coordinar computadores de diferente hardware y software y cuyo fin es procesar una tarea que demanda una gran cantidad de recursos y poder de procesamiento. Facilita la posibilidad de compartir, acceder y gestionar información, mediante la colaboración de varios nodos.

usuarios puedan conectarse a Internet, tienen todos les recursos web accesibles a través de su PC. La computación en la nube por lo tanto se refiere a las técnicas que permiten y facilitan este escenario.

Cuando se compara con Internet, un PC parece un terminal ligero que los usuarios puedan utilizar para acceder a la nube. Desde esta perspectiva, la computación en la nube parece un retorno al modelo del *mainframe* original según se ha explicado en la Fase 1. Por supuesto, la computación en la nube no es así de simple. A diferencia de una unidad central, que es una máquina física que ofrece potencia de cálculo finito, una nube representa todos los recursos posibles en la Internet, lo que sugiere infinito poder y capacidad. Sin embargo, lo más importante no es sólo la sensación de un retorno a los orígenes de los sistemas informáticos, debido a la similitud entre terminales ligeros que acceden al *mainframe* y los PCs que acceden a la nube, sino la evolución de la gestión en estos años.

Para ello se explica a continuación esta evolución de la gestión tradicional a la gestión en un entorno en la nube.

2.3 El cambio de la gestión tradicional

Inicialmente la gestión de los sistemas en la época del *mainframe* o de las computadoras centrales se ceñía a los departamentos universitarios o los centros de cálculo de las grandes empresas. En aquel momento el coste de los elementos hardware y software suponían el 90% del total del sistema, mientras que la gestión de estos recursos era el 10% restante. A medida que la informática ha ido ocupando un papel más importante en el entorno empresarial, la gestión de los recursos informáticos se ha ido haciendo más compleja.

El cambio cualitativo más importante fue la extensión de la informática al puesto de usuario y finalmente a los hogares. Con este paso la informática pasó de un entorno científico e ingenieril a un ámbito empresarial y popular. Durante este cambio han ido aumentando el número de aplicaciones informáticas en las empresas y el concepto de gestión TI surgió como un elemento fundamental para obtener un valor real en el uso de los sistemas.

La situación actual es que el 80% del TCO (*Total Cost of Ownership* o Coste Total de Propiedad) de los sistemas informáticos de las empresas corresponden a la gestión de los mismos, mientras que el 20% se deben al coste del hardware y del software.

La gestión se ha ido haciendo tan compleja que ha motivado la aparición de estándares y mejores prácticas de gestión del entorno TI, que permiten un uso adecuado de los sistemas, alineados con el negocio, y al mismo tiempo permiten un marco de referencia común para las empresas y para los proveedores de servicios. Entre las normas que regulan la gestión TI se pueden citar ISO/IEC-20000, que es el estándar para la provisión de servicios TI e ITIL (*IT Infrastructure Library*, un extenso conjunto de procedimientos de gestión ideados para ayudar a las organizaciones a lograr calidad y eficiencia en las operaciones de TI).

Esta tendencia a la normalización y estandarización se ha pronunciado con el avance tecnológico que ha permitido abaratar notablemente el coste del hardware al mismo tiempo que se aumentaba considerablemente la capacidad de cálculo y de proceso. Por otro lado, también se ha desarrollado bastante la virtualización, que es un elemento fundamental del Cloud Computing, el cual permite compartir servidores "virtuales" sobre la misma máquina física, contribuyendo al abaratamiento de los costes.

Los servicios de Cloud Computing pueden cambiar el modelo de negocio de una organización, como los cambios en el almacenamiento de la información. En este caso Cloud Computing facilita, gracias a sus bajos costes de almacenamiento, la gestión de grandes volúmenes de datos e imágenes que pueden ser accedidos en tiempo real y de forma segura. El ejemplo más claro es el entorno médico. En caso de emergencia, el acceso rápido y fiable al expediente completo de un paciente puede ser vital. Actualmente existen varios proyectos para unificar los expedientes médicos y hacerlos accesibles para cualquier médico desde cualquier hospital cuando la situación lo requiera. Este tipo de proyectos necesitan manejar grandes volúmenes de información de manera dinámica en situaciones puntuales. Los servicios cloud son muy útiles para estos modelos de negocio.

Pero también se pueden observar cambios en el modelo de negocio por el modo de acceso a la información, por ejemplo, hoy en día Internet está produciendo de forma paulatina y sostenida una caída importante de las ventas en los medios de prensa tradicionales. El cambio en los hábitos culturales y de acceso a la

información hace que incluso la traslación a entornos digitales de los medios de prensa escritos no acabe de funcionar. La solución pasa por la gestión de contenidos global que se pueden publicar en varios medios en Internet, que además se actualizan en tiempo real. La gestión dinámica de estos medios encaja perfectamente con la flexibilidad de un entorno cloud.

Evidentemente a medida que ascendemos en la cadena de valor del cloud la gestión se hace más compleja, ya que tiene mayor implicación en el negocio de la organización. Por ello en los niveles más bajos la gestión trata temas operativos de control de los recursos como la fiabilidad, disponibilidad, gestión dinámica de la demanda..., mientras que en los niveles más altos se debe regular muy bien el marco contractual de la relación cliente/proveedor, la seguridad en el tratamiento de los datos e incluso la viabilidad empresarial del proveedor.

Desde un punto de vista más técnico, el nivel más bajo de gestión del cloud obliga a cambiar el enfoque tradicional de la aplicación y de la gestión de la red. En un entorno en la nube no hay un número fijo de activos que deben permanecer funcionando todo el tiempo como lo podía haber en un entorno más tradicional. En su lugar, la informática es bajo demanda, donde las necesidades de recursos, tales como servidores y aplicaciones, y la capacidad fluctúa en función de la demanda.

En una gestión de red tradicional la implementación de la red y el control de la aplicación se centra en la disponibilidad y en el rendimiento de la infraestructura de red y el servidor [2]. El seguimiento de los dispositivos de red, como routers, switches y firewalls, generalmente se logra con un sistema de gestión de red (NMS) que utiliza los protocolos tradicionales como SNMP, ICMP, y WMI para recopilar estadísticas de rendimiento.

Los sistemas avanzados de gestión suelen incluir el descubrimiento de dispositivos de la red y la topología de la misma, la supervisión del rendimiento de dispositivos de red, alertas inteligentes en tiempo real e informes, monitorización de aplicaciones y muchas más características que ayudan al seguimiento y la gestión del rendimiento de la red.

Una vez que se introduce una red de la nube, se añaden características adicionales para tener en cuenta, como el rendimiento de la ISP (rendimiento de la WAN) y el rendimiento del proveedor de la nube y su ISP. Debido a que el enlace WAN es un elemento fundamental entre el cliente y el proveedor de la nube, como se puede ver en la Figura 2, es muy importante supervisar su

rendimiento. El exceso de latencia puede tener un impacto negativo en la disponibilidad del servicio y su rendimiento.

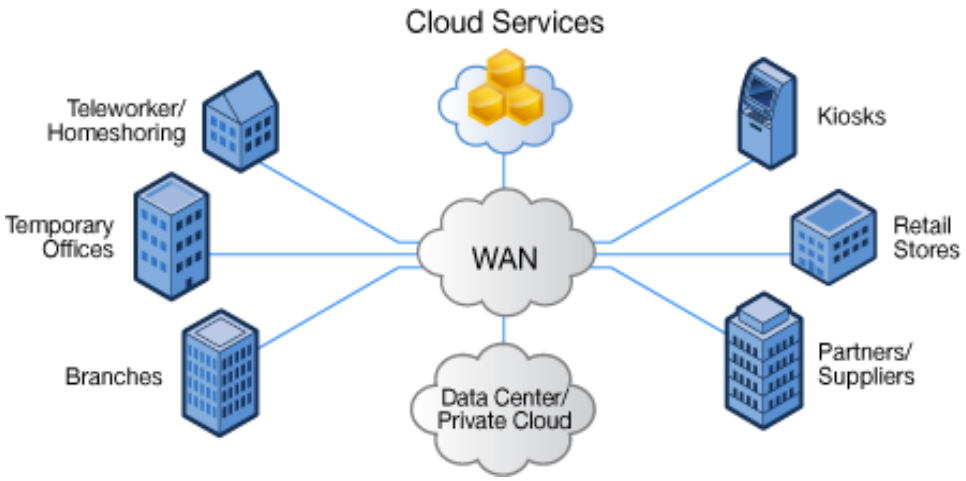

Figura 2: Enlace WAN entre la nube y el cliente

Sin embargo, no basta simplemente supervisar el rendimiento de la WAN, también es necesario entender su tráfico de red para evaluar el impacto en el rendimiento de las aplicaciones y servicios críticos. En un entorno en la nube se puede supervisar desde dentro mediante la creación de un NMS (*Network monitoring system*) en el centro de datos del proveedor de servicios, o se puede monitorizar desde fuera mediante la creación de un NMS en la red propia.

Mirando desde el exterior de la nube existe un mayor control y es posible realizar una medición más real de la experiencia del usuario final. Sin embargo, existe el riesgo de que la red o la conexión a Internet puedan fallar.

En cambio, desde el interior es posible tener una solución de monitorización de red independiente tanto de la red del cliente y del proveedor de la nube. Pero aquí el problema es que sí que se depende de la red y de la infraestructura de un tercer proveedor en la nube para monitorizar al proveedor de la nube principal, y también se depende de Internet y de la latencia que se adquiere en el proceso de vigilancia que hará que su información pueda ser más inexacta.

Otra alternativa es tener una máquina virtual en la nube junto con los servidores de la nube. Debido a que su NMS se encuentra justo al lado de los servidores y las aplicaciones que se están supervisando, se eliminan los efectos de latencia WAN y por lo tanto se reciben mediciones más precisas. El único inconveniente de este enfoque es que cuando el proveedor de la nube sufre una interrupción, entonces esta solución de monitorización sería inútil.

2.4 Clasificación en función de los servicios cloud

El modelo de computación en la nube se descompone principalmente en las siguientes opciones de servicios como se observa en la Figura 3:

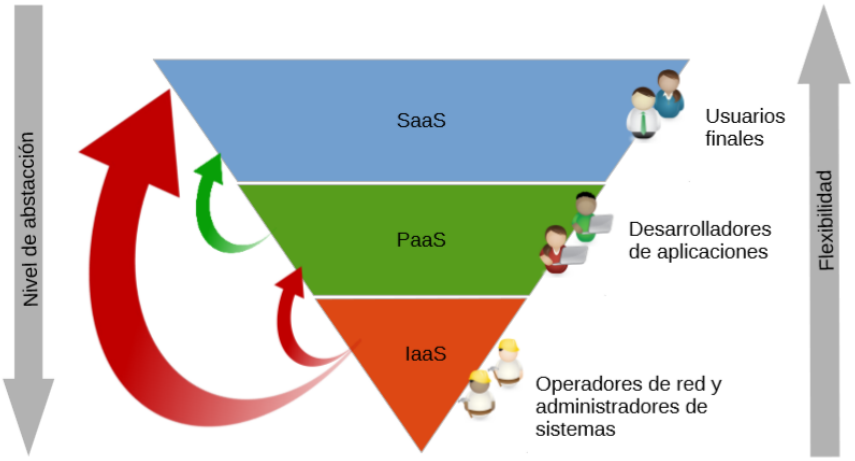

Figura 3: Servicios en la nube

- **Infraestructura como Servicio (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, etc.). Por ejemplo, mientras el operador puede correr sus aplicaciones y sistemas operativos, el proveedor se encargará de la replicación y los *backups* del sistema en general. Ejemplos de IaaS son *OpenStack* en la que se profundizara más adelante, *CloudStack*, *IBM SoftLayer* y *AWS*.

- **Plataforma como servicio (PaaS):** Se refiere a un entorno de desarrollo junto a unas herramientas y servicios asociados que se ofrece a los clientes para crear sus propias aplicaciones. Estos servicios, también conocidos como *middleware* (bases de datos, servidores de aplicaciones, servidores web, etc.) es todo aquello que es software pero que no acaba de finalizar un proceso como tal, sino que habilita un proceso. Su objetivo

consiste en dar un entorno de plataforma que permita desarrollar nuevas aplicaciones, permitiendo a un desarrollador configurar su propio sistema de programación a través de lenguajes de programación, librerías, APIs[3], servicios y otras herramientas. Ejemplos de PaaS son *OpenShift*, *CloudFoundry*, *IBM Bluemix* y *Google App Engine*.

- **Software como Servicio (SaaS):** El Software como Servicio permite al cliente final usar una aplicación que se ejecuta 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. Algunos ejemplos de SaaS son *OwnCloud*, *WordPress*, *Gmail, Dropbox* y *Salesforce*.

2.5 Clasificación en función del tipo de despliegue

Cloud ofrece libertad de elección a la hora de desplegar este tipo de red, por ello se pueden clasificar los tipos de despliegue de cloud en los siguientes:

- **Cloud pública:** En las nubes públicas los servicios ofrecidos se encuentran en servidores externos al usuario o empresa que hace uso de ellos. Como modelo de pago por uso, es habitual que el proveedor cloud cobre según el tiempo y la capacidad de los recursos.

  Tiene como ventaja un aumento de recursos sin la necesidad de instalar y mantener las máquinas en local. Así, se evita una gran inversión inicial y permite obtener una gran escalabilidad a empresas que necesiten el uso de estos recursos digitales.

[3] Las APIs son interfaces de programación de aplicaciones que se utilizan para construir aplicaciones y permiten al software solicitar los datos y los cálculos de uno o más servicios a través de una interfaz directa o indirecta

Como inconvenientes se encuentra la dependencia de terceras personas, donde habitualmente se produce el conocido "*lock-in*[4]" por parte del proveedor. Para evitar esto es recomendable elegir plataformas *OpenSource* que faciliten la migración del software o producto.

- **Cloud 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, etc. En este caso será más sencillo integrar los servicios Cloud con sistemas propietarios y en cuanto a la seguridad y a la carga de trabajo se hace cargo la empresa.

  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. Es una vía poco escalable para los inicios de una organización.

- **Cloud 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 en un momento determinado.

  Las nubes híbridas ofrecen la escalabilidad provisionada externamente y a demanda, permitiendo cubrir momentos o tareas críticas del modelo de negocio. Este tipo de nube comienza a imponerse sobre las anteriores debido a su versatilidad.

[4] Se define "*Lock-in*" como la situación en la que un cliente que utiliza un producto o servicio no puede migrar a otros de la competencia. Estas dificultades provienen, además de las medidas que adoptan los propios proveedores para retener a su clientela, de la inexistencia de estándares comunes a toda la industria.

2.6 Desarrollo de estándares

Las organizaciones que desarrollan aplicaciones basadas en la nube están interesadas en desarrollar modelos de referencia que permitan a las aplicaciones ser transferidas fácilmente de una nube a otra e interactuar con diferentes servicios basados en la nube. Por ejemplo, con un adecuado marco de interoperabilidad, una aplicación en la nube podría cambiar fácilmente de un proveedor a otro que ofrece un menor costo o una mayor gama de servicios en la nube.

Aunque quizás sea pronto para disponer de estándares estables, sí que se pueden encontrar organizaciones que trabajan en el desarrollo de los mismos, como el *Cloud Computing Foro* [3] y el *Consorcio Open Cloud* [4].

Este tipo de esfuerzos de creación de estándares resulta difícil puesto que el modelo cloud se considera un modelo relativamente nuevo. Si se compara con el inicio de Internet se puede observar que ninguna organización que tuviese una red propia quería estandarizar los protocolos de comunicación con el objetivo de conseguir la máxima dependencia de sus clientes, por lo que el envío de datos entre las redes era muy difícil. La introducción de TCP y los protocolos relacionados con Internet ayudó a remediar esta situación, pero muchas empresas con productos propietarios se han opuesto durante algún tiempo.

Hoy en día, en el entorno Cloud, se puede observar una situación similar, aunque los proveedores de servicios en la nube se han retrasado en el proceso de normalización, la capacidad de nubes diferentes para interoperar fácilmente permitiría una clase nueva e interesante de aplicaciones. Aunque la computación en la nube todavía no tiene bien establecidos puntos de referencia, en el futuro a corto plazo, como ocurrió con Internet, es previsible que se estandarice y se normalice su gestión facilitando así su implicación en la transformación de procesos empresariales y de negocio.

Actualmente el intento más serio de normalización se ha llevado a cabo con el CCOA (*Cloud Computing Open Architecture*). El modelo CCOA [5] normaliza los servicios de Cloud Computing estructurándolos en 7 niveles:

1) El nivel 1 lo constituyen los 3 tipos de actores en el mercado Cloud: El proveedor de los diferentes servicios cloud, el partner que integra las soluciones cloud en sus servicios globales y el cliente final.

2) El segundo nivel lo componen el hardware y software de virtualización y gestión de la infraestructura.
3) En el nivel 3, SOA (*Service Oriented Architecture*) se encarga de configurar la capa de orientación al servicio. Todos los servicios reutilizables para la plataforma de Cloud Computing y los servicios específicos de la aplicación deben ser definidos usando SOA para permitir la reutilización de la nube.
4) El nivel 4 lo componen la oferta de software y aplicaciones.
5) El nivel 5 establece las soluciones de negocio que se montan sobre las aplicaciones.
6) En el nivel 6 se encuentra la arquitectura de información Cloud, desde la construcción de servicios cloud hasta el despliegue de la plataforma.
7) Por último, en el nivel 7, se define el gobierno del modelo cloud, así como las mejores prácticas que permiten su gestión eficiente. Este nivel cubre la seguridad, colaboración, y las normas.

La existencia de modelos de referencia implica que los servicios Cloud han alcanzado un grado de madurez suficiente para el desarrollo de mecanismos de gestión sencillos y eficientes.

2.7 Proveedores de servicios Cloud

De entre todos los proveedores o empresas que ofrecen recursos y/o servicios a través de la nube se pueden destacar las siguientes:

- ***Amazon Web Services*:**
  Amazon [6] es el mayor proveedor de servicios cloud en la actualidad, la mayoría de estos servicios no están expuestos directamente a los usuarios finales, sino que ofrecen una funcionalidad que otros desarrolladores puedan utilizar en sus aplicaciones. Se accede a *Amazon Web Services* a través de HTTP, utilizando protocolos REST[5] y SOAP[6].

[5] REST (*Representational State Transfer*) es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP. Permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente, por lo que es más simple y convencional que otras alternativas.
[6] SOAP (*Simple Object Access Protocol*) es un protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

- ***Google:***

  Es uno de los proveedores de servicios cloud con mayor capacidad computacional. Los servicios en la nube que *Google Cloud Platform* [7] ofrece a las empresas y a los grandes usuarios son los siguientes.

  - *App Engine*: Plataforma con la que se pueden crear aplicaciones web y móviles escalables.
  - *Compute Engine*: Permite correr cargas de computación a gran escala sobre máquinas virtuales Linux alojadas en la infraestructura de Google.
  - *Cloud Storage*: Servicio de almacenamiento con la escalabilidad de la infraestructura de Google.
  - *BigQuery*: Herramienta para analizar grandes datos en la nube utilizando SQL. Proporciona un servicio de análisis de datos sin tener que instalar o mantener servidores.
  - *Cloud SQL*: Permite alojar bases de datos MySQL en la nube de Google. Proporciona un servicio totalmente gestionado capaz de mantener y administrar bases de datos.

- ***Microsoft Azure:***

  Se trata de una plataforma [8] bastante reciente, pero con una enorme capacidad. Los servicios que proporciona en la actualidad son en general los equivalentes a los proporcionados por Amazon y Google: Infraestructura, desarrollo, pruebas para aplicaciones, Big Data, servicios multimedia (*Streaming),* almacenamiento, copias de seguridad y administración de identidad y acceso.

- ***VMWare:***

  Es la empresa líder en cuanto a máquinas virtuales [9], su tecnología vCloud sirve en la actualidad como soporte a una enorme cantidad de nubes de otros proveedores y cuentan con un importante PaaS llamado "*Cloud Foundry*" que la propia empresa ofrece como servicio alojado en sus servidores, y al ser de código libre también es usada por otras compañías para ofrecer servicios PaaS. Además, VMWare ofrece a sus clientes herramientas para la construcción de su propia nube privada y también servicios IaaS basados en su tecnología vCloud.

- ***Eucalyptus:***
  Es una infraestructura *open source* para la implementación de computación en nube privada en clusters de ordenadores. Su nombre hace referencia al acrónimo "*Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems*" que puede traducirse como "Utilidad de arquitectura informática elástica para confiar sus programas a sistemas funcionales". Puede instalarse fácilmente en la mayoría de distribuciones Linux, también puede usar gran variedad de tecnologías de virtualización de hardware incluyendo hipervisores *VMware, Xen*[7] y *KVM*[8] para implementar las abstracciones de nube que soporta.

- ***OpenStack:*** Se tratará en profundidad en el capítulo 5.

2.8 Ventajas y Desventajas

El uso del empleo de servicios en la nube lleva consigo unas ventajas, así como una serie de desventajas.

Lo más probable es que las soluciones de gestión basadas en la nube consigan reducir las responsabilidades por parte de una empresa, ya que en este caso las tareas implicadas en torno a la gestión del sistema de monitorización son llevadas a cabo por un tercero, por el proveedor de servicios en la nube, y el trabajo de administración de la red interna gira en torno estrictamente a la supervisión de los elementos de la red.

2.8.1 Ventajas

Las ventajas más importantes [10] que podemos destacar al utilizar una solución de monitorización de red basada en la nube son las siguientes:

---

[7] *Xen* es un monitor de máquina virtual de código abierto desarrollado por la Universidad de Cambridge.

[8] KVM (*Kernel-based Virtual Machine* o Máquina virtual basada en el núcleo) es una solución para implementar virtualización completa con Linux

- **Facilidad de implementación:** Una de las mayores ventajas de usar una solución basada en la nube es que puede estar funcionando en un espacio de tiempo muy corto en comparación con una implementación interna, ya que, al implementar un sistema de monitorización en la nube, este ha sido completamente probado y está completo desde el principio.

- **Escalabilidad:** Los proveedores de nube pueden escalar las implementaciones de sus soluciones rápidamente, permitiendo que las organizaciones crezcan o se reduzcan sin preocuparse por las inversiones del sistema de monitorización de la red.

- **Coste:** El coste de implementar una solución gestión de red interna puede ser muy costoso, especialmente la inversión inicial. Para las organizaciones de pequeño y mediano tamaño es difícil justificar el gasto de unos servicios que posean grandes características, en cambio las opciones cloud permiten reducir considerablemente este coste.

- **Funcionalidades:** Los proveedores diseñan sus sistemas para soportar un amplio número de características con el objetivo de satisfacer los requisitos de múltiples tipos de clientes.

- **Calidad del servicio:** Los servicios cloud suelen ofrecer soporte las 24 horas al día y una respuesta inmediata en situaciones de emergencia.

- **Acceso desde cualquier lugar:** Permiten acceder a las aplicaciones y datos de forma segura desde cualquier lugar a través de una conexión a Internet, así es más fácil interactuar y colaborar con aplicaciones y los datos almacenados en la nube.

- **Copias de seguridad:** Existe la posibilidad de recuperar copias de seguridad guardadas en la nube.

Por otra parte, también este tipo de servicios y recursos en la nube, así como las soluciones de gestión tienen una serie de desventajas que también deben tenerse en cuenta, como son las siguientes.

2.8.2 Desventajas

- **Seguridad:** La seguridad de cualquier solución que requiere conectividad pública es bastante importante, en el caso de utilizar una solución de gestión de red en la nube requiere una gran confianza para ser colocado en el proveedor de la nube. A diferencia de las soluciones internas que se pueden aislar de una red pública, las soluciones en la nube son más vulnerables a un ataque, aunque los proveedores de la red utilizan sesiones cifradas; Esta sesión se establece entre el proveedor de la nube y, o bien un recopilador interno o dispositivos individuales.

- **Gestión exterior:** El hecho de tener un sistema vital que es controlado por una entidad externa tiene sus desventajas. Si el proveedor tiene un corte de luz que le afecta, lo más probable es que acabe afectando al cliente y esto difiere de la influencia que tendría en los sistemas internos.

- **Conectividad:** En las soluciones de red tradicional en una empresa, los propios sistemas se encuentran normalmente en la parte central de la red de una organización, la gran ventaja de esto es que la conectividad de los dispositivos gestionados normalmente es sencilla. Con una solución basada en la nube, la conexión a una entidad externa no va a tener dicha conectividad directa, y esto conlleva el riesgo de perder el acceso a los elementos gestionados.

- **Rendimiento:** Otro punto para tener en cuenta y que está relacionado con la conectividad es la diferencia significativa entre la disponibilidad de ancho de banda entre un sistema interno y sus elementos gestionados y por otro lado la disponibilidad de ancho de banda entre una nube externa y los elementos gestionados. Si el número de elementos que requieren gestión es grande, lo mejor es encontrar una opción en la nube que ofrezca el despliegue de un recolector interno que se encuentre dentro de las instalaciones de la organización.

Capítulo 3

Gestión de red en la nube

Las tareas de gestión de red incluyen el proceso de despliegue e integración de los diversos elementos de red, garantizando el correcto funcionamiento de los elementos de red coordinados.

Los elementos de red pueden incluir elementos de software, hardware y elementos humanos que participan en la red. La gestión de redes tiene como objetivo proporcionar la máxima disponibilidad de recursos y la fiabilidad del servicio a los hosts que utilizan la red. Las diferentes capas del sistema de red [11] serán responsabilidad o bien del proveedor de la nube, o bien del consumidor de la nube, dependiendo del tipo de nube. La Tabla 1 resume los escenarios más típicos:

Capa OSI	**Protocolos**	**IaaS**	**PaaS**	**SaaS**
Aplicación	HTTP, FTP, NFS, SMTP, SSH	Consumidor	Consumidor	Proveedor
Presentación	SSL, TLS	Consumidor	Proveedor	Proveedor
Sesión	TCP	Consumidor	Proveedor	Proveedor
Transporte	TCP	Consumidor	Proveedor	Proveedor
Red	IP, IPSec	Consumidor	Proveedor	Proveedor
Enlace de datos	Ethernet, Canal de fibra	Proveedor	Proveedor	Proveedor
Física	Cobre, fibra óptica	Proveedor	Proveedor	Proveedor

Tabla 1: Responsabilidades en cloud

La tabla es una simplificación de los distintos modelos que existen en la práctica. A partir de ella es obvio que una IaaS proporciona a los consumidores de la nube una flexibilidad considerablemente mayor en topología de red y servicios que las nubes PaaS y SaaS, posiblemente con el costo adicional de la administración de las herramientas que proporcionan tal flexibilidad.

El modelo cloud busca reducir los costes al permitir pagar por lo que realmente se utiliza. Estos beneficios juntos con otras muchas ventajas que se han explicado en el anterior capítulo motivan el cambio hacia este modelo. La entrega de los servicios cloud depende de la calidad de extremo a extremo de la conexión en la que Internet es bastante importante ya que se encarga de transportar el tráfico de las aplicaciones entre las empresas y los centros de datos públicos o privados de un servidor en la nube.

El rendimiento de las aplicaciones basadas en la nube depende no sólo del software de la aplicación en la nube, sino también la fiabilidad, eficiencia y el rendimiento de las redes en el medio.

Mejorar la calidad de la red podría lograrse mediante el despliegue de más dispositivos de red con mayor ancho de banda. Sin embargo, este enfoque no es eficiente debido a los elevados costes que supone y a las crecientes demandas de tráfico. Otro enfoque distinto seria construir soluciones de gestión de red adecuadas para utilizar mejor los recursos de red existentes. Sin embargo, la actual de gestión de redes tiene dos problemas principales que se han convertido en los puntos principales a atender [12]. Estos dos problemas son:

- **La gestión desigual de los componentes de extremo a extremo:**

  Proporcionar una red eficiente implica múltiples componentes en la ruta de extremo a extremo, como la red de los sistemas operativos de los servidores, la configuración del hardware y el control de enrutamiento de los dispositivos de red en los centros de datos, el intercambio de tráfico de los proveedores de servicios de Internet (ISP), la configuración de la red empresarial, etc.

  Cada componente puede afectar la calidad de la red en general. Sin embargo, componentes como servidores, hardware de dispositivos de red y enrutamiento de tráfico son gestionados de forma separada respecto a los diferentes sistemas, ya que tradicionalmente estos componentes se

encontraban extendidos a través de múltiples lugares. En un entorno en la nube estos componentes se concentran mucho más en los centros de datos, y la falta de integración entre los sistemas de gestión limita la mejora de la calidad de las redes.

- **Interfaces de bajo nivel para interactuar con dispositivos de red:**

  Los dispositivos de red son heterogéneos con diferentes modelos de varios proveedores. Interactuar con los dispositivos es complicado ya que las configuraciones de las APIs con los dispositivos suelen ser de bajo nivel y específicas del proveedor.

  Las personas encargadas de gestionar la red tienen que utilizar estas API de bajo nivel diariamente, y esto supone un proceso propenso a errores para configurar los dispositivos y que ejecuten los protocolos correctos con los debidos parámetros y haciendo uso de las API correctas.

  La tecnología *OpenFlow* tiene como objetivo automatizar la gestión del enrutamiento de tráfico en redes con programación de alto nivel.

Estos problemas se vuelven más graves en un entorno cloud y por ello son los principales factores que limitan la fiabilidad y eficiencia de la gestión de red.

Teniendo en cuenta el primer problema, muchos componentes están involucrados en el proceso de extremo a extremo de la ruta de las aplicaciones, estas aplicaciones se ejecutan en los servidores de los centros de datos, y estos a su vez envían tráfico a otros servidores o al exterior.

Los ISPs entregan el tráfico de aplicaciones a la red empresarial, lo que finalmente llega a los usuarios, como se puede observar de manera esquemática en la Figura 4.

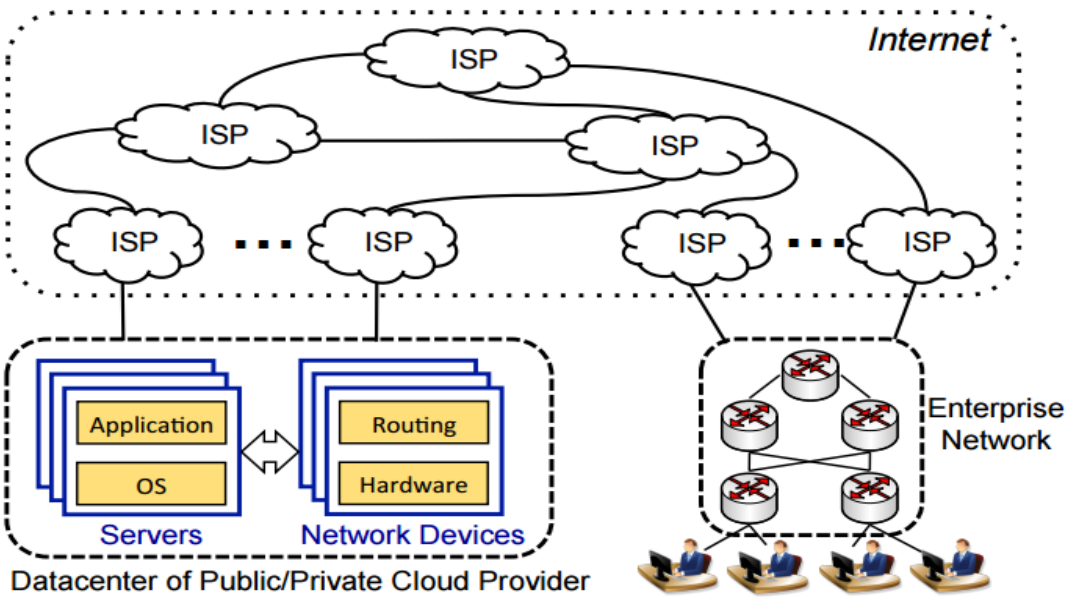

Figura 4: Estructura extremo a extremo de servicios en la nube

Tradicionalmente, los componentes existentes en todo este proceso se manejan como tres áreas separadas:

1) Aprovisionamiento de aplicaciones y servidores para colocar las aplicaciones y controlar los entornos operativos de los servidores, 2) Infraestructura de red para configurar el hardware del dispositivo desde el control de energía a la configuración de la topología, y 3) El enrutamiento del tráfico de la aplicación a través de la infraestructura de red.

En un entorno tradicional se observa una división del trabajo entre las entidades de Internet, en el pasado, la entidad que poseía las aplicaciones tenía la mayoría del tráfico dirigida directamente a los usuarios y no entre las instancias de la aplicación. Por lo tanto, se centra en el aprovisionamiento de servidores con pocas participaciones en las operaciones de red. Por el contrario, los ISPs se centran en las redes, separando la gestión de la infraestructura y el enrutamiento del tráfico porque funcionan a ritmos distintos (Los cambios de infraestructura se prolongan durante años mientras que el tráfico cambia en minutos).

En cambio, en un entorno cloud la mayoría del tráfico en los centros de datos de los proveedores de la nube viene de la comunicación interna de servidor a servidor entre las instancias de las aplicaciones. Este nuevo tipo de tráfico hace que sean necesarios grandes cambios de la arquitectura de red de los centros de datos.

El aprovisionamiento de aplicaciones, la gestión de infraestructuras y la ingeniería del tráfico se concentra en los proveedores de la nube, y la actual separación de los sistemas de gestión se convierte en el principal problema en la mejora del rendimiento de las aplicaciones. Además, las aplicaciones basadas en la nube tienen un tráfico bastante asimétrico, siendo lo normal recibir (a nivel empresarial o personal) mucho más de lo que se envía, y es por ello por lo que se busca tener un control del tráfico entrante, pero normalmente se carece de ese control directo porque la función de la ingeniería de tráfico pertenece principalmente a los ISPs. [13]

Para consolidar estos sistemas de gestión desiguales trabajan grandes proveedores de servicios cloud, como *Google* y *Microsoft* que han construido sistemas de gestión de tráfico en sus redes de área extensa combinando el conocimiento y las funciones de servidores y de redes. Estos sistemas recogen las demandas de tráfico de las aplicaciones directamente desde los servidores, y se comparan con las estadísticas de la red para obtener una solución y programar el tráfico. Los sistemas entonces aplican esta solución con la limitación de velocidad en los servidores y el control de enrutamiento en redes. Estos sistemas integrados han mejorado significativamente los niveles de utilización del ancho de banda en las redes WAN de *Google* y *Microsoft*.

Además del problema de separación, la gestión de la red también está limitada por las interfaces de bajo nivel con dispositivos heterogéneos. Cada proveedor de dispositivos desarrolla interfaces con sus dispositivos de red. A medida que los proveedores se diferencian y el hardware de sus dispositivos crece, las interfaces se vuelven más especializadas, y dado que los dispositivos especializados son más difíciles de gestionar, los operadores de red deben basarse en APIs de proveedores de bajo nivel para configurarlos. Esto hace que la operación de la red dependa del control manual de los operadores.

El uso de estas interfaces de dispositivos de bajo nivel por los operadores de red se convierte en un impedimento para automatizar la gestión de red de los centros de datos de los proveedores de la nube. Debido al cambio de la arquitectura, el número de dispositivos de red es mucho mayor en los centros de datos de los proveedores de la nube que en las redes tradicionales. El proceso de gestión tiene que ser por tanto automatizado mediante software, en lugar de por configuraciones manuales para ser escalables y sostenibles en un crecimiento futuro. Además, estos centros de datos utilizan grandes cantidades de

dispositivos de productos básicos, en lugar de hardware especializado de proveedores para ahorrar en el coste operacional y de compra.

La diferencia entre las interfaces de los proveedores y los modelos de hardware se convierte en un obstáculo para el desarrollo de soluciones de gestión de red para servicios en la nube. Los principales proveedores cloud han comenzado a reducir su dependencia de proveedores específicos y utilizan dispositivos básicos para crear sus soluciones de gestión con la ayuda de software basado en servidor. Por ejemplo, el equilibrio de carga solía ser llevado a cabo por dispositivos especializados, sin embargo, su configuración y el hardware necesario es costoso, y suele ser una fuente principal de fallos en la operación.

En cambio, *Microsoft* y *Amazon* [14] construyen sistemas de equilibrado de carga basados en software. Estos sistemas utilizan las funciones básicas de los dispositivos de red (como, por ejemplo, ECMP9) y montan el software sobre un grupo de servidores para distribuir el tráfico entre instancias de las aplicaciones. Estos sistemas consiguen mejorar en gran medida la estabilidad de los servicios en la nube, reduciendo los fallos operativos.

Para una operación eficaz, la red debe cumplir con algunos criterios básicos, como rendimiento, fiabilidad, seguridad y disponibilidad. El rendimiento garantiza un tiempo de tránsito y respuesta razonable y depende principalmente de factores como el número de usuarios, el tráfico de red y la naturaleza del software que funciona en la red. La fiabilidad garantiza la precisión en la entrega, el tiempo mínimo de recuperación y el estado de la red de ser robusto a fallos ocasionales. La seguridad es uno de los requisitos primarios de una red estable. La *Organización Internacional de Normalización* (ISO) clasifica las funcionalidades de gestión de redes en las siguientes cinco áreas funcionales [15] como se puede observar de forma esquemática también en la Figura 5.

- Gestión de la configuración
- Gestión del rendimiento
- Gestión de la seguridad
- Gestión de fallos
- Gestión de la contabilidad

[9]ECMP (*Equal-Cost Multi-Path*) es un protocolo para hacer distribución de tráfico hacia un mismo destino pero utilizando diferentes rutas. Con ECMP se puede agregar diversas rutas hacia una misma red destino, asignándoles a cada una la misma distancia y la misma prioridad.

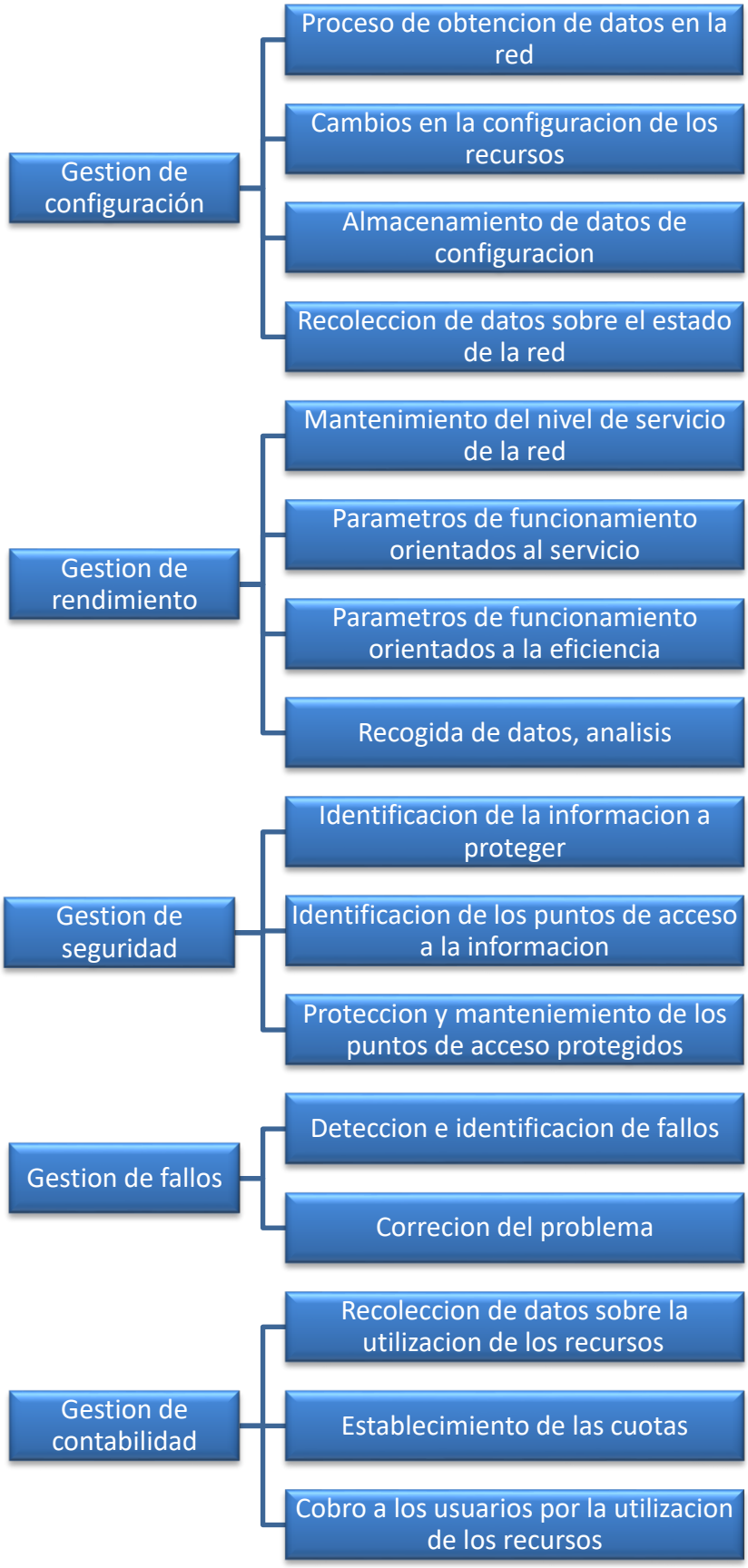

Figura 5: Áreas funcionales de la gestión de red

A grandes rasgos, la gestión de configuración se ocupa de la configuración inicial de los componentes de red y de la infraestructura de la red. La gestión del rendimiento garantiza el funcionamiento óptimo de los componentes de la red. La gestión de la seguridad es responsable de mejorar la seguridad de la red y distribuir la información relacionada con la seguridad en toda la red, las políticas de seguridad de la red deben actualizarse constantemente para evitar cualquier posible ataque de red. La gestión de fallos o problemas está pensada sobre los dispositivos defectuosos y su recuperación. La administración de la contabilidad es responsable de las funcionalidades de facturación sobre los recursos utilizados.

Hay que tener en cuenta que la gestión de red de un entorno de nube es diferente de la de un entorno de red convencional. La gestión tradicional de la red implica la gestión de los dispositivos de red y los enlaces físicos que conectan entre sí los distintos elementos de la red. Sin embargo, la gestión de red de un entorno en la nube implica la gestión de redes virtuales, esta gestión es muy importante porque los errores frecuentes pueden conducir a la caída de toda la nube, si no se mantiene adecuadamente.

A continuación, se profundizará en los requisitos necesarios para una correcta gestión de la red en la nube atendiendo a las cinco áreas funcionales de la gestión de red definidas por la ISO.

3.1 Gestión de la configuración

Un entorno en la nube implica compartir varias redes virtuales en una infraestructura de red en una nube común [16]. La computación en la nube puede ser vista como un sistema distribuido avanzado y, por lo tanto, la creación de redes autónomas no es la solución ideal para configurar la infraestructura en la nube. Cada CVN (*Cloud-based Virtual Networks*), es decir cada red virtual en un entorno cloud, está compuesta por distintas máquinas virtuales (VMs) y es probable que estas máquinas virtuales sean las interfaces de la nube, en las que los usuarios pueden obtener acceso a la nube a través de ellas.

Muchas de estas redes virtuales en la nube se agrupan para formar una nube más grande. Una máquina virtual debe ser capaz de ver el tráfico de otras máquinas virtuales que pertenecen a la misma CVN en la nube. El enlace que interconecta los recursos de la nube es puramente lógico y la infraestructura de la nube debe ofrecer una interconexión punto a punto entre los proveedores de la nube y los usuarios finales.

El proveedor de la nube se encarga de asignar sus propias direcciones IP a sus clientes [17]. Los clientes deben hacer la elección de su nube y deben seleccionar los servicios que necesitan en un proceso llamado "*on-boarding*". Además, este proveedor se encarga de proporcionar un tiempo de inactividad mínimo durante el inicio de sesión, puesto que se busca la mayor agilidad y automatización en el proceso.

Los usuarios de servicios cloud deben poder interactuar con sus propios proveedores con el propósito de administrar su CVN, esto es posible realizarlo gracias a *CVN Management Interface* (CMI) que es un canal bien definido que permite a los usuarios finales llegar a los proveedores de la nube. CMI debe permitir que los usuarios finales creen, desplieguen, modifiquen y eliminen sus CVNs.

La gestión de la configuración es una característica que muy probablemente se encuentra en las soluciones de monitorización de red alojadas, esto se debe a que esta información puede suponer un gran riesgo de seguridad si se filtra. Sin embargo, la implementación de una solución de gestión de la configuración es vital para muchas redes de grandes empresas, ya que permite a los administradores e ingenieros la capacidad de saber cómo un dispositivo está configurado actualmente y cómo se ha configurado en el pasado. Las configuraciones pasadas a menudo puede ayudar a solucionar problemas en el caso de que se produzcan después de realizar algún cambio.

Como se ha visto en este apartado, los dos puntos clave son las máquinas virtuales, las cuales componen la red virtual en la nube y a través de ellas los usuarios acceden a la nube y también la correcta gestión de las propias redes virtuales, estos dos puntos se tratan a continuación.

3.1.1 Conectarse a la VM a través de direcciones IP

Uno de los primeros pasos necesarios es conectarse a una máquina virtual (VM). Existen diferentes opciones de asignación de IP cuando se crea una VM:

- Las direcciones IP generadas por el sistema son similares a las direcciones asignadas por DHCP. Estas son en realidad direcciones IP estáticas, pero asignadas por la nube IaaS.
- Las direcciones IP reservadas son direcciones que pueden ser suministradas y administradas de forma independiente respecto a una máquina virtual. Las direcciones IP reservadas son útiles si se desea asignar múltiples direcciones IP a una máquina virtual.
- A través de una VLAN.

3.1.2 Administrar redes virtualizadas

El termino virtualización se refiere a la abstracción de los recursos de un equipo o servidor; para esto se crea una capa de software llamada *Hypervisor* o Monitor de Máquina Virtual (VMM) que permite la abstracción entre el hardware de la máquina física subyacente (*Host*), y el Sistema Operativo (SO) de la máquina virtual (*Guest*), como se refleja en la Figura 6. [18]

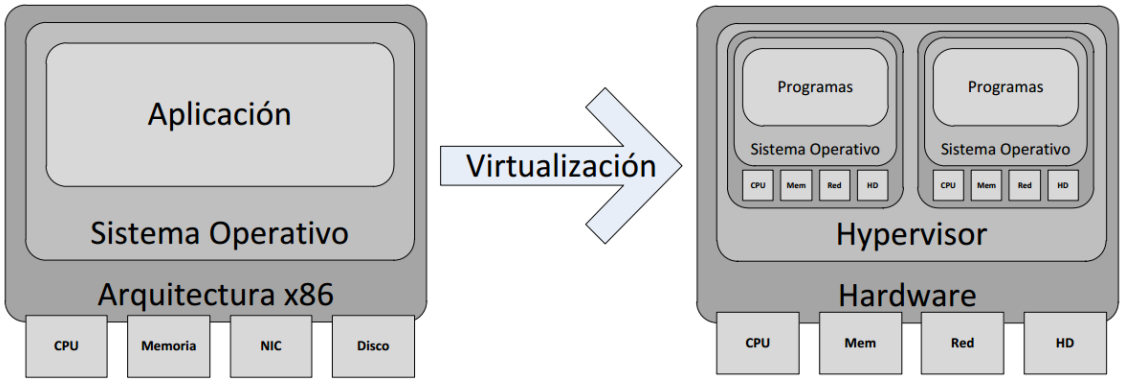

Figura 6: Sistema antes y después de virtualizar

El hipervisor controla y gestiona cuatro de los recursos principales del hardware físico (CPU, memoria, red y almacenamiento) permitiendo al usuario ejecutar concurrentemente múltiples instancias de SO en máquinas virtuales (VM) sobre un único hardware físico. A cada una de las VM, le presenta una interfaz del hardware que sea compatible con el SO elegido.

Al trabajar con sistemas de máquinas virtuales teniendo en cuenta la seguridad de la red, es necesario administrar estas redes.

Los recursos de la red pueden virtualizarse tal como otros recursos de la nube. Para hacer esto una nube utiliza conmutadores virtuales para separar una red física en particiones lógicas. Esto se puede ver ilustrado en la Figura 7.

Las VLANs pueden utilizarse como una extensión de la red privada de su empresa, y en la nube usan direcciones IP privadas para restringir la visibilidad de red de las máquinas virtuales conectadas a ellas.

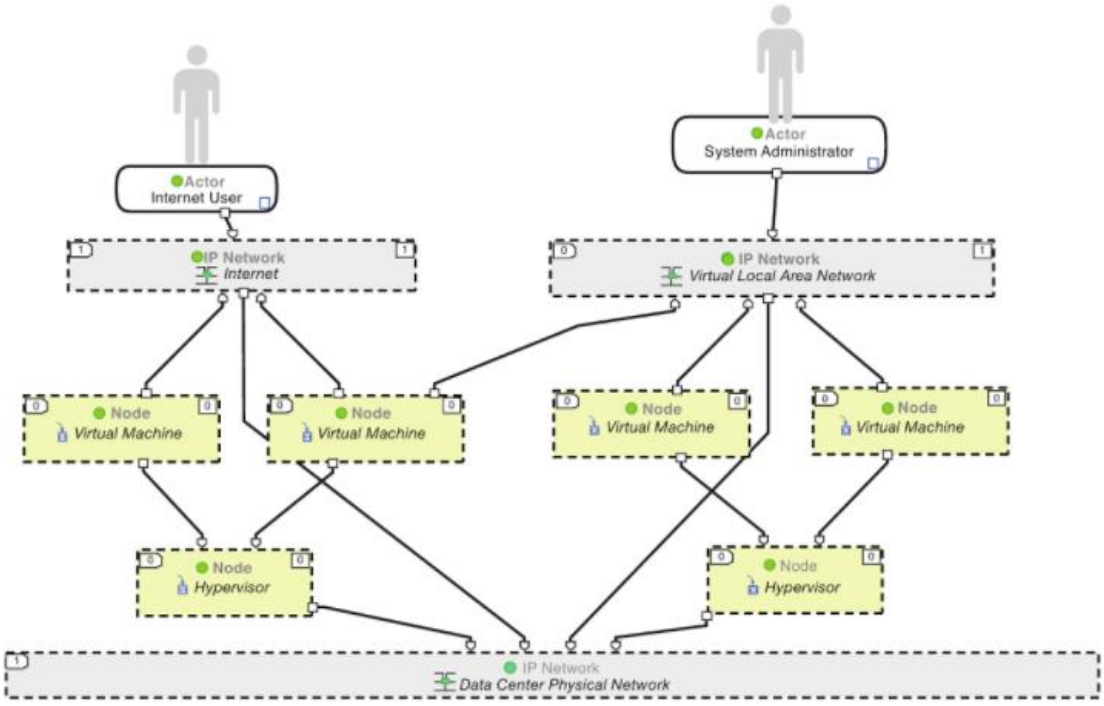

Figura 7: Redes físicas y virtuales en la nube

La IANA (*Internet Assigned Numbers Authority*) [19] reserva los siguientes tres bloques del espacio de la dirección IP para redes privadas:

- 10.0.0.0 - 10.255.255.255 (prefijo 10/8)
- 172.16.0.0 - 172.31.255.255 (prefijo 172.16/12)
- 192.168.0.0 - 192.168.255.255 (prefijo 192.168/16)

Es posible conectarse a una VLAN mediante una VPN cifrada o por enrutamiento a través de una máquina virtual con doble encauzamiento conectada a otra red, incluyendo Internet.

El hipervisor puede compartir una interfaz de red física individual con múltiples máquinas virtuales. Cada máquina virtual cuenta con una o más interfaces de red virtual. Existen tres formas en las que el hipervisor puede hacer esto:

- **Puentes:** Normalmente es el método predeterminado, en este modo el hipervisor trabaja en la capa de enlace de datos y hace que la interfaz de red virtual sea visible externamente a nivel de Ethernet.

- **Enrutamiento:** En este modo el hipervisor funciona al nivel de la capa de red y hace que la interfaz de red virtual sea visible externamente a nivel de IP.

- **Traducción de dirección de red (NAT):** Usando este método la interfaz de red virtual no es visible externamente, ni por tanto es visible en Internet la máquina virtual, pero sí le permite enviar datos de red hacia Internet. Es una manera de ocultar interfaces de red de virtualización con direcciones IP privadas mediante una dirección IP pública usada por un host o enrutador.

  El software para traducción de direcciones de red cambia la información de la dirección IP en los paquetes de red basada en la información en una tabla de enrutamiento. Los valores de verificación del paquete también deben cambiarse. La traducción de dirección de red puede utilizarse para poner en la red más servidores que el número de máquinas virtuales que haya. Esto se puede hacer mediante traducción de puertos. Esta es una de las razones por la cuales la IPv6 todavía no es usada ampliamente, incluso cuando el número de equipos excede el número de direcciones IP.

  Estas direcciones IP se pueden compartir, por ejemplo, suponiendo que se dispone de un enrutador y tres servidores manejando HTTP, FTP y correo respectivamente, es posible asignar una dirección IP pública al enrutador y direcciones IP privadas a los servidores HTTP, FTP y de correo, y direccionar el tráfico entrante como se muestra en la Tabla 2.

IP Pública	Puerto	IP Privada
9.0.0.1 (enrutador)	80, 443	192.168.0.1 (servidor HTTP)
	21	192.168.0.2 (servidor FTP)
	25	192.168.0.3 (servidor de correo)

Tabla 2: Ejemplo de traducción de direcciones de red (NAT)

3.2 Gestión del rendimiento

Dado que varias redes virtuales (CVN) se agrupan para formar una nube, es necesario que los proveedores de servicios en la nube se aseguren de que los CVN individuales estén en buenas condiciones de funcionamiento. Esto da como resultado un entorno de nube global correctamente gestionado. El rendimiento de cada CVN es inmune a otros en la nube.

El sistema de recuperación de errores en la nube debe diseñarse de tal forma que se recuperen fallos menores antes de que la nube se caiga. Por lo tanto, la gestión del rendimiento debe incluir un seguimiento coherente del estado de la nube.

Cada CVN en la nube debe ser monitorizado independientemente. La monitorización puede llevarse a cabo a través de la API. También se debe supervisar el rendimiento de las aplicaciones que se ejecutan en la nube. El sistema de detección de intrusiones en red (NIDS) se puede emplear para la supervisión pasiva de cualquier posible ataque de red. Una vez detectado un error, el método de recuperación debe basarse en si el fallo afecta sólo al CVN particular o se extiende a toda la nube.

Cada vez más a menudo se puede observar que para las pequeñas y medianas empresas es difícil permitirse tener un control y una vigilancia de los elementos de red dentro de la organización, es decir una constante monitorización de la red. Esta supervisión de la red implica que se realice un continuo seguimiento de la situación de todos los elementos dentro de una red, esto puede ser algo tan simple como usar el tráfico ICMP, por ejemplo, haciendo un ping, para verificar el estado en el que se encuentra un dispositivo. Pero partiendo de lo básico existen otras opciones más completas que ofrecen una perspectiva mucho más amplia de la red, como por ejemplo las soluciones de *Kaseya* [20], *LogicMonitor* [21] o *Monitis* [22].

Entre las distintas soluciones que se pueden encontrar para gestionar el rendimiento de la red se pueden encontrar las siguientes herramientas:

- **Detección de la topología:** En las etapas iniciales de configuración es muy importante la detección de la topología de red que se quiere monitorizar, lo que ayuda a obtener una visión completa de las conexiones entre los distintos elementos de la red.

Esta función de descubrimiento de la topología de red puede variar en complejidad, desde protocolos sencillos como el ICMP (*Internet Control Message Protocol*) o el SNMP (*Simple Network Management Protocol*) a soluciones más complejas que son capaces de analizar la información de las bases de datos del protocolo de detección de Cisco (CDP), del protocolo de detección de capa de enlace (LLDP), del protocolo de resolución de direcciones (ARP), de la LAN virtual (VLAN) y de las tablas de enrutamiento.

- **Dispositivos de monitorización:** Una de las características más básicas de cualquier solución de monitorización de redes es el seguimiento del estado de la red, que incluye la comunicación de esta información de estado en un formato fácil de usar y fácilmente entendible. Luego, esta información se compila y se puede utilizar para predecir el comportamiento futuro. Las estadísticas más comunes incluyen el tiempo de activación del dispositivo, el procesador, la memoria, el búfer, etc.

- **Informes:** La capacidad de acceder a un motor de informes bien construido es muy importante y no debe subestimarse. Mientras que todas las estadísticas que se pueden obtener a través de diversos medios son muy útiles, suelen ser difíciles de analizar. Por ejemplo, buscar un elemento de la red y descubrir que tiene una alta utilización del procesador es útil, pero ser capaz de rastrear y reportar esto con el tiempo, así como tener la capacidad para representar gráficamente cómo la utilización de este dispositivo se correlaciona con otras estadísticas, puede hacer que el trabajo de encontrar y resolver un problema, especialmente uno intermitente, sea mucho más sencillo.

- **Seguimiento y medición del nivel de servicio (SLA):** El Acuerdo de Nivel de Servicio (SLA) es básicamente una garantía del nivel más bajo de servicio que ofrecerá un proveedor. Si no se proporciona este nivel mínimo de servicio, entonces se debe compensar de alguna forma esa falta de servicio. La medición del nivel de servicio proporciona una forma automatizada para realizar un seguimiento de lo que se está proporcionado y puede basarse en la alerta de configuración para

determinar si se está cumpliendo o no. Algunos SLAs comunes se basan en torno a un ancho de banda específico o una latencia máxima que se proporciona a través de un enlace de red.

- **Recopilación de eventos:** La recopilación de eventos normalmente va relacionada con el protocolo SNMP (*Simple Network Management Procotol*). Una de las características de SNMP es que puede ser configurado para enviar *traps*[10] a un sistema de gestión. Estos traps reportan un evento del sistema; por ejemplo, si una interfaz se cae de un elemento de la red. El recopilador es una parte del sistema de gestión que gestiona estas traps a medida que se producen y los etiqueta con su respectivo nivel de gravedad, y si es necesario se puede configurar para enviar alertas a los responsables de la administración de red.

- **Monitorización ambiental:** Hay diversos factores para tener en cuenta cuando se habla de la vigilancia del medio ambiente, algunos de estos pueden ser el seguimiento de energía, calentamiento, enfriamiento, la humedad, el fuego, etc. La mayoría de las soluciones de monitorización disponibles no ofrecerán una continua supervisión ambiental sin una solución externa. Estas soluciones externas a menudo incluyen un dispositivo de control separado que tiene sensores que se conectan a él. Este dispositivo de supervisión puede configurarse para ser consultado o para proporcionar alertas (SNMP) si se produce una condición específica.

- **Análisis y recopilación de NetFlow:** NetFlow es una característica que se implementó inicialmente en dispositivos *Cisco* que permitió a los ingenieros recoger información del flujo específico de la red. Esta información puede ser utilizada para observar cómo fluye el tráfico en la red que puede ser de gran utilidad, desde la planificación de la ruta y su capacidad hasta el equilibrio de carga de las aplicaciones. Algunas soluciones que soportan la monitorización de *NetFlow* pueden ser *Traverse* [23] y *LogicMonitor* [21].

---

[10] *Trap* es un mensaje no solicitado generado por el agente, sin intervención del administrador. Ocurren cuando el agente detecta la ocurrencia de un parámetro predefinido.

3.3 Gestión de fallos

En la gestión de red convencional, la gestión de fallos se ocupa de reparar y reemplazar los dispositivos defectuosos para retener la actividad de la red.

La administración de errores en un entorno cloud se refiere a la capacidad de la red virtual basada en la nube (CVN) para resistir los frecuentes cambios en la red, como fallos inesperados o extraordinarios. La gestión de fallos desempeña un papel vital en un buen entorno en la nube.

La gestión de fallos de una CVN [24] depende de dos parámetros importantes como el Objetivo de Punto de Recuperación (RPO) y el Tiempo de Recuperación (RTO).

RPO es la métrica que determina la cantidad de datos que se perderán durante un desastre. RTO define el tiempo de inactividad mínimo para recuperar los fallos ocurridos en la red. Un entorno de nube correctamente gestionado debe poseer un nivel mínimo de tiempo de inactividad y asegurar un nivel casi nulo de pérdida de datos durante un fallo.

El uso de múltiples componentes de red idénticos de forma escalonada [25] enmascarará los fallos individuales y eliminará así el fallo a nivel de componentes. Estos componentes de red idénticos permanecerán activos todo el tiempo en la red. En caso de fallo de un componente, uno de los componentes activos idénticos toma la operación correspondiente.

Dado que estos componentes idénticos permanecen activos durante el correcto funcionamiento de los componentes de red predeterminados, este enfoque resulta en un uso ineficiente de los recursos de red.

Una característica interesante de las redes virtuales es su capacidad de soportar la gestión automatizada de fallos. Si falla la red virtual en la nube, el sistema de gestión de fallos debería poder restaurar el contenido del CVN en los servidores internos del proveedor de la nube y transferir los recursos a otro proveedor de la nube dentro de un tiempo de tránsito razonable definido por el RTO.

Dentro de algunas soluciones de monitorización de red existentes hoy en día se pueden encontrar las siguientes características en cuanto a gestión de fallos:

- **Análisis predictivo:** Los análisis predictivos funcionan mediante el control de los datos históricos que está disponible para elementos específicos y comparándolos en el tiempo para buscar unos determinados patrones. Estos patrones se pueden utilizar para prever la posibilidad de problemas en el futuro. La complejidad de este análisis puede variar en gran medida entre las distintas soluciones existentes.

- **Análisis de causa raíz:** Esta función se centra en mirar toda la información que se ha reunido, siendo más fácil de ver e interpretar los datos para encontrar el problema de origen que causó una interrupción o un evento en la red. Algunos sistemas combinan sus análisis predictivos con sus análisis de causa raíz para ayudar tanto a encontrar los problemas existentes como sus causas, y proporcionar una predicción de problemas futuros que podrían ocurrir en caso de que alguna acción no suceda.

- **Gestión de alertas:** La gestión de alertas se basa esencialmente en alertar que algo ha sido detectado en uno de los dispositivos que están siendo monitorizados.

  Esto puede considerarse como un arma de doble filo, ya que podría producirse que no solo se alerte de todos los problemas que se producen, sino que también alerte de todo lo que sucede, sin importar que sea relevante o no. Para ello, en las soluciones de gestión más modernas se pueden crear políticas que permiten usar reglas muy específicas en cuanto a qué tipo de eventos serán alertados y/o mediante qué mecanismo de alerta, como por ejemplo el correo electrónico o SMS.

3.4 Gestión de la seguridad

La seguridad es una de las principales preocupaciones de cualquier política de administración de red.

La gestión de la seguridad en una red convencional implica el control de los servicios de seguridad y la distribución de información relacionada con la seguridad en toda la red.

En un entorno de nube, la gestión de seguridad se enfrenta a muchos desafíos, dado que Internet es la infraestructura de comunicación para la nube, las amenazas de seguridad de Internet también suponen un riesgo para los entornos en la nube y son vulnerables a ataques [26] como ataques de denegación de servicio distribuidos (DDoS), *spoofing*[11] de IP, intrusiones, infecciones por gusanos y muchos otros.

El rol de seguridad más importante de la administración de la red es asegurar un uso seguro de los recursos de la nube de los proveedores de nube multinivel a los usuarios de la nube. Los entornos en la nube deben tener una política de seguridad de varios niveles que incluye la autenticación, la autorización, el cifrado de datos, la integración de datos, la seguridad de las sesiones, etc. Cada cliente tiene su propia identidad con el propósito de autenticarse. Esto garantiza que el cliente correcto tenga acceso al recurso adecuado del proveedor de la nube.

Los datos confidenciales deben ser sometidos a la gestión de identidad para hacerlos más difíciles contra las posibles intrusiones de red. Los equilibradores de carga son sugeridos para mejorar las políticas de seguridad de la red. Si no es así, al menos se debe usar un *proxy inverso*[12] que hace más difícil atacar la infraestructura de la nube. El firewall en la zona desmilitarizada (DMZ) debe ser capaz de enrutar el tráfico hacia el proveedor de la nube. Cada CVN debe ser configurado con su propio firewall para regular el tráfico entrante y saliente.

El proveedor de cloud ofrece la máxima seguridad posible, ya sea con el empleo de sistemas de cifrado, o con una medida de codificación que implementan los proveedores. La seguridad se puede contemplar desde dos perspectivas bien distintas, desde el punto de vista del usuario individual o desde un punto de vista más genérico como a nivel empresarial.

A nivel de usuario, el hecho de tener sus servicios alojados en la nube supone un cambio cultural por su parte. Dos aspectos importantes de la seguridad a nivel de usuario son la fuga de información y el robo de contraseñas. Cloud soluciona

---

[11] *Spoofing* hace referencia al uso de técnicas a través de las cuales un atacante, generalmente con usos maliciosos, se hace pasar por una entidad distinta a través de la falsificación de los datos en una comunicación.

[12] Un proxy inverso es un servidor proxy que, en lugar de permitirles el acceso a Internet a usuarios internos, permite a usuarios de Internet acceder indirectamente a determinados servidores internos.

parte de ellos aportando mayor seguridad y nuevas formas de trabajar con los datos a las empresas.

A nivel de empresa, se debe tener en cuenta una única política de seguridad de la empresa, ya que en el modelo de cloud debe perseguirse modelo integrados y unificados, siguiendo unos marcos de estandarización (ISO) y un perímetro de seguridad llevado a cada recurso que quiero proteger, lo que se conoce como "micro-segmentación".

A continuación, se explican las medidas de seguridad más empleadas en un entorno cloud, como son el uso del firewall, el empleo del protocolo SSH y el uso de VPN para las conexiones seguras.

3.4.1 Utilizar un firewall para proteger múltiples máquinas virtuales

Una de las herramientas más esenciales en la nube es la utilización y la correcta configuración de un firewall individual, es decir que está instalado en el mismo servidor que el recurso que se está protegiendo.

Una regla de firewall especifica un conjunto de criterios para un paquete de red, cuando llega un paquete de red, cada regla es verificada. Si el paquete no satisface los criterios de la regla, entonces se verifica la siguiente regla.

Es posible lograr algunas ventajas en rendimiento al utilizar las reglas de firewall del hipervisor. Sin embargo, no es posible manejarlas tan dinámicamente como las reglas del firewall local. Los firewalls de Linux pueden utilizarse para proteger servidores diferentes al servidor en el cual reside el firewall. En realidad, esta es la configuración óptima, ya que proporciona un nivel adicional de aislamiento.

Para proteger los servidores es necesario aislarlos de Internet, ubicándolos en una VLAN. La máquina virtual del firewall deberá tener dos direcciones IP, y estar configurada para el reenvío entre Internet y la VLAN. Se puede ver este modelo en la Figura 8.

En esta figura, se observa que la máquina virtual VM1 está conectada directamente a Internet. La VM2 que hace de firewall tiene dos direcciones IP, una que permite el acceso vía Internet y otra que está en la VLAN privada.

La máquina virtual VM3 tiene una dirección IP individual a la cual solo se puede acceder vía VLAN. Configurando una regla de firewall que permita el acceso de entrada desde Internet y que relacione la IP del firewall con la IP de la VM3 para un puerto determinado, se permite el acceso de entrada desde Internet únicamente para ese puerto. Por ejemplo, si este era un servidor web y un usuario intenta acceder a la dirección IP del firewall en el puerto 80, entonces esa solicitud es reenviada a un servidor web en VM3. Ningún otro puerto en VM3 está abierto.

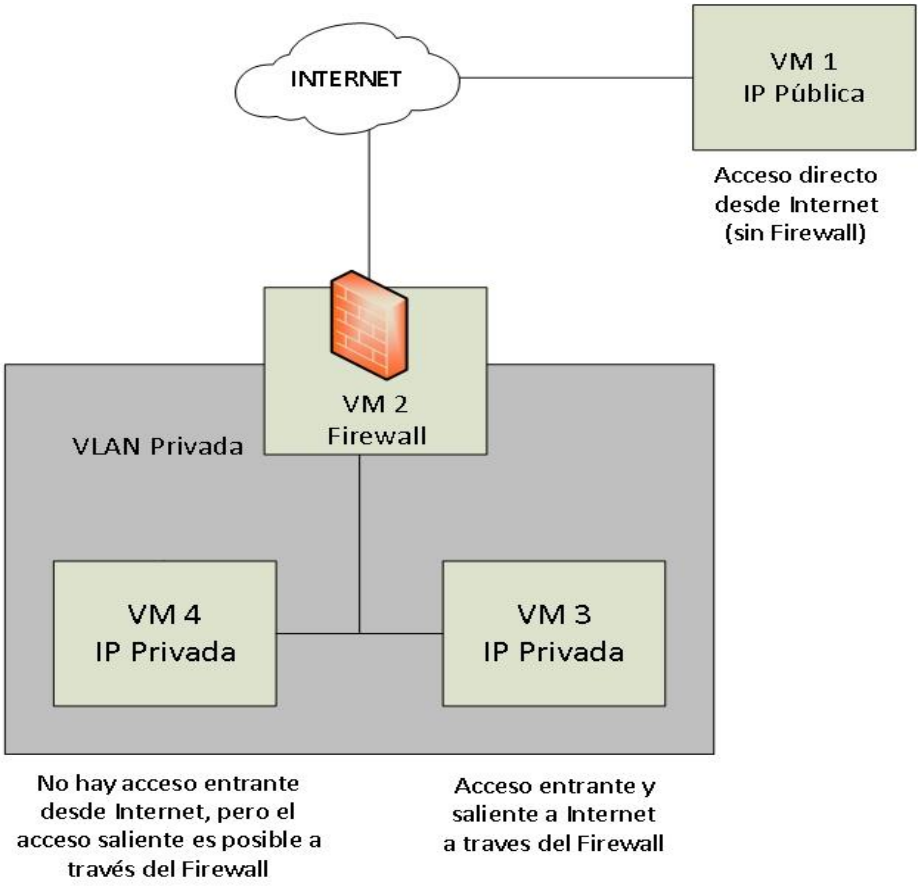

Figura 8: Diagrama del esquema de un firewall protegiendo una VLAN

Otra regla de firewall permite el acceso de salida para VM4. Esto se consigue usando puertos dinámicamente para una conexión TCP individual. Por ejemplo, si VM4 intenta abrir una conexión TCP hacia Internet, entonces el firewall puede encontrar un puerto disponible y cambiar el paquete TCP para que contenga este puerto y la dirección IP propia del firewall.

Cuando este recibe un paquete de retorno de Internet en respuesta a la solicitud de VM4, entonces hace una correlación revertida. De esta forma, la máquina virtual VM4 puede abrir conexiones hacia Internet sin ser visible en Internet.

3.4.2 Autenticar servidores y usuarios con SSH

SSH (*Secure Shell*) es una herramienta fundamental en el entorno cloud. SSH se diseñó como un reemplazo de *telnet*, es decir para administrar dispositivos remotamente, pero implementando una mayor seguridad. Las dos versiones principales de SSH son SSH-1 y SSH-2. La implementación más popular de SSH es el proyecto *open source* de *OpenSSH*. En Windows se pueden encontrar herramientas como *PuTTY*.

SSH utiliza criptografía de clave pública para autenticar la maquina remota y el usuario. Además del inicio de sesión remoto, SSH también puede utilizarse para el uso de túneles, redireccionamiento, conexiones X11 y transferencia de archivos. Los archivos se pueden copiar con el protocolo SCP (*Secure Copy Protocol)* y SFTP (*SSH File Transfer Protocol*), los cuales usan SSH.

Como ejemplo se puede plantear el siguiente escenario: Si se tienen creadas dos instancias[13] en la nube, "servidor1" y "servidor2", y se desea abrir una sesión SSH desde el primero al segundo, se debe tener la clave privada en el PC local, por lo que se necesitará copiarla en el "servidor1", para ello se puede emplear WinSCP[14]. Entonces, "servidor2" ya estará configurado para aceptar esta clave porque se ha creado con la clave pública en la lista de claves autorizadas. Es necesario crear la clave privada de manera que no sea visible para otros usuarios, si no es así SSH puede ignorarla.

3.4.2.1 Transferencia de puertos con SSH

La transferencia de puertos con SSH es un proceso donde la dirección y el puerto de un paquete son transferidos hacia un nuevo destino y el paquete es llevado sobre una conexión SSH donde se accede al destino.

---

[13] Una instancia es un conjunto de recursos de computación que estarán disponibles para una imagen en ejecución y es el símil de lo que sería un hardware real sobre el que ejecutar un sistema operativo.

[14] WinSCP es una aplicación de software libre, es un cliente SFTP gráfico para Windows que emplea SSH. Su función principal es facilitar la transferencia segura de archivos entre dos sistemas informáticos, el local y uno remoto.

Esto permite al usuario usar otro protocolo mediante túnel sobre una conexión SSH. Puede ser útil si el protocolo al que se está aplicando el túnel no es seguro o si la combinación de dirección de destino y puerto no es visible desde el origen. El cliente que utiliza el protocolo con túnel debe estar en capacidad de especificar un puerto no estándar para que esto funcione. El concepto es que se establece una sesión SSH para el servidor y luego se especifica desde que puerto de la máquina del cliente se transfieren las conexiones.

SSH también puede transferir hacia otros servidores. La transferencia de puertos puede ser importante para resolver problemas de conectividad entre la nube, la estación de trabajo y los recursos dentro de una compañía. Esto es especialmente importante cuando es necesario un acceso flexible pero seguro para tareas administrativas de bajo volumen, como mover archivos hacia ubicaciones seguras.

En aplicaciones en la nube con frecuencia es necesario conectarse con equipos Linux remotos de escritorio. Como las máquinas remotas están en Internet se debe evitar enviarles tráfico de red no asegurado y también evitar abrir servicios inseguros, como VNC[15], en Internet.

La Figura 9 muestra una representación esquemática de las conexiones de red involucradas. Un servidor VNC ejecutándose en la máquina virtual en el puerto 5901 se conecta a un monitor X11[16], que a su vez soporta un equipo de escritorio de Linux. El PC, portátil o la estación de trabajo utiliza un cliente SSH para crear una conexión con un túnel hacia el servidor SSH que se ejecuta en la máquina virtual.

Un firewall que se ejecuta en la máquina virtual evita que cualquier tráfico diferente al tráfico SSH que se ejecuta en el puerto 22, acceda a la máquina virtual. El túnel permite que un cliente VNC que se esté ejecutando en el portátil se conecte al servidor VNC, pasando a través del firewall del puerto 22.

---

[15] VNC es un programa de software libre basado en una estructura cliente-servidor que permite tomar el control del ordenador servidor remotamente a través de un ordenador cliente.

[16] X11 es el servidor gráfico que usan casi todas las distribuciones Linux y permite *forwarding* a través de SSH. Esto significa que es posible ejecutar aplicaciones gráficas de una máquina remota exportando el display a nuestro escritorio.

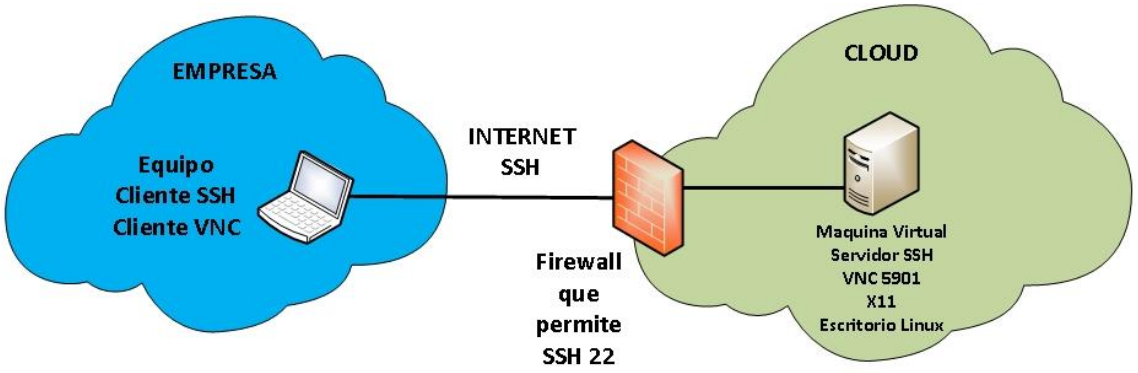

Figura 9: Diagrama esquemático de la transferencia SSH de tráfico VNC

3.4.3 Uso de redes virtuales privadas (VPN)

Las redes VPN se basan en el cifrado para crear una extensión de una red privada sobre Internet, y por ello permiten diversos escenarios de red que son de gran uso para las empresas.

Un uso tradicional de las VPN es conectar redes de área local de diferentes oficinas de una empresa, formando una red de área más amplia. Este tipo de conexiones son de sitio a sitio. Cuando las VPN se introdujeron con este propósito, reemplazaron el uso de líneas alquiladas, reduciendo en gran medida los costos para las empresas.

Otro uso tradicional de una VPN es permitir a los empleados acceder a las redes privadas de las empresas de forma remota, por ejemplo, para trabajar desde casa. En este escenario, la empresa proporciona una puerta de enlace de VPN al que se puede acceder desde Internet y el empleado instala un cliente VPN en su portátil para acceder a aplicaciones, como el correo. A esto se le asignó el término de "red virtual privada móvil", porque uno de los puntos finales, donde el empleado está ubicado, no tiene una dirección IP fija.

Cuando un cliente envía un paquete a través de una puerta de enlace de VPN se añade un encabezado de autenticación, los datos son cifrados y se añaden en una cabecera llamada *Encapsulating Security Payload (ESP)* como se observa en la Figura 10. El servidor VPN receptor descifra los datos y enruta el paquete hacia el destino, según la información del encabezado.

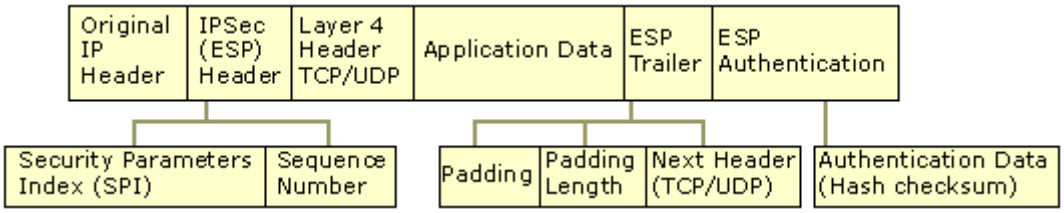

Figura 10: Encapsulating Security Payload (ESP)

El cifrado proporcionado por las VPN se realiza a un nivel muy bajo, de forma que todas las comunicaciones son cifradas. Esto puede suceder en la 2º capa OSI (capa de enlace de datos) o en la 3º capa (capa de red) y puede incluir cualquiera de los siguientes métodos: IPsec, SSL /TLS, *Datagram Transport Layer Security* (Cisco), Cifrado Microsoft *Point-to-Point* o mediante el uso de túneles SSH.

Las redes VPN pueden usar puentes para crear un área Ethernet LAN virtual. Esto tiene algunas ventajas frente al uso de enrutamiento, dado que puede funcionar con cualquier protocolo que funcione sobre Ethernet. El uso de puentes es necesario si se usan protocolos que no sean IP o intercambio de archivos Windows. No obstante, configurar una VPN con enrutamiento puede ser más fácil y permitir un mayor control sobre el acceso a los hosts y puertos específicos.

Para utilizar servicios cloud, la VPN es configurada a través de una puerta de enlace en la red corporativa, hacia una VLAN privada en la nube. Esto se conoce como una conexión de sitio a sitio. Las máquinas virtuales en la red privada en la nube no son visibles desde Internet, pero solo se puede tener acceso a ellas a través de la VPN o mediante otra máquina virtual en la VLAN, esta situación se representa en la Figura 11.

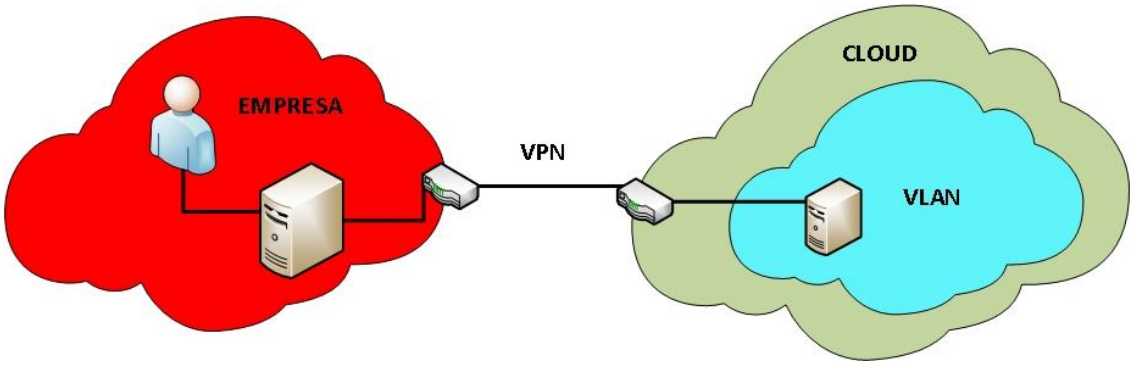

Figura 11: Uso de una VPN para extender una red corporativa

En este escenario la nube actúa como una extensión de la red corporativa. La extensión de la nube es aislada de Internet y de máquinas virtuales en la nube, por fuera de la VLAN. A diferencia del escenario de trabajo desde casa, el empleado no necesita realizar ninguna instalación especial de cliente VPN.

También es posible soportar un escenario de red que utilice la nube para soportar una VPN sin necesidad de que una empresa tenga la propiedad de una puerta de enlace VPN dentro de las instalaciones. Esto puede facilitar a los empleados que trabajen desde casa con un cliente VPN, como el cliente *OpenVPN*, como se refleja en la Figura 12.

OpenVPN es una solución de fuente abierta para cliente y servidor VPN, que puede manejar conexiones de *punto-a-punto* y de *sitio-a-sitio*. Utiliza la biblioteca de cifrado OpenSSL.

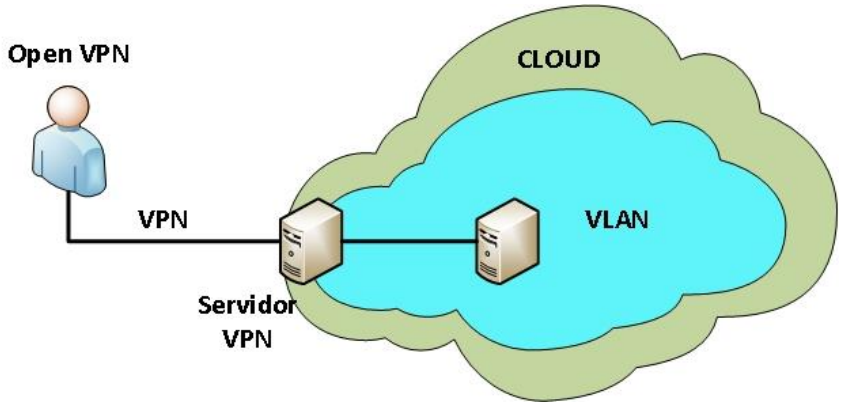

Figura 12: Uso de puerta de enlace VPN en la nube para acceder a VLAN

OpenVPN incluye software tanto de cliente como de servidor y debe instalarse en ambos. Está disponible en sistemas Linux, aunque también existe un instalador que se extrae automáticamente para Windows y también imágenes de instalación solo para el cliente hacia donde es posible dirigir a los usuarios finales.

Configurar una VPN con *OpenVPN* incluye configurar una infraestructura de clave pública, incluyendo un certificado con una clave pública y una clave privada para cada servidor y cliente, y un certificado de autoridad de certificación (CA) para firmar cada uno de estos certificados. De manera predeterminada, la VPN realiza autenticación mutua tanto de servidor como de cliente, usando estos certificados. Sin embargo, es posible configurar métodos de autenticación alternativos.

3.5 Gestión de la contabilidad

La gestión de la contabilidad en la red convencional y en el entorno de la nube tiene algunas características comunes. El papel clave de la administración contable es medir la utilización de la red de los usuarios finales y luego informar a los usuarios de la nube sobre sus cargos correspondientes o la facturación de los recursos a los que acceden [27]. Las funcionalidades de contabilidad se asignan a dispositivos basados en volumen de facturación o control en la infraestructura de la nube.

En una red convencional, se cobraría a los usuarios los recursos que posean independientemente de la utilidad de los recursos para los usuarios. Cloud en cambio, ofrece un servicio de pago por uso y por lo tanto se cobra a los usuarios sólo por los recursos utilizados por ellos. El sistema de cobro de la nube debería poder comenzar a cargar inmediatamente después del inicio de la interfaz de programación de aplicaciones (API) y debería dejar de cargar una vez que se termina la API.

Lo normal es que las aplicaciones requieran el acceso de múltiples recursos ubicados en diferentes regiones de la nube. En la aplicación de la nube, se lanzan uno o más *weblets* a la nube para expandir los recursos de la nube. *Weblets* son los micrositios o páginas web llenas de información del usuario. Estos, se comunican entre sí para intercambiar los datos y son propiedad de los usuarios de la nube que los alojan en la nube para acceder a los recursos. Durante el uso de recursos, el sistema debería poder cobrar por la transferencia de datos entre las dos instancias. Los cargos deben ser computados como datos de una instancia y datos en la otra instancia. Si un *weblet* está involucrado en la transferencia de datos, entonces está incluido en el sistema de cobro. Los *weblets* no utilizados se ignoran independientemente de su presencia en la nube.

Capítulo 4

Aplicaciones y herramientas de gestión de red

El modelo en la nube requiere, como se ha visto, de una correcta gestión y para ello es necesario hacer uso de distintas herramientas y aplicaciones, de las cuales se pueden encontrar opciones nativas o de terceros. Estas herramientas también pueden proporcionar información sobre cómo configurar los recursos, así como las relaciones entre ellos.

4.1 Herramientas de configuración

En cuanto a la gestión de la configuración se pueden utilizar servicios nativos de gestión en una plataforma de nube pública, o emplear una herramienta de terceros. Las herramientas de nube nativas involucran una mayor dependencia del proveedor de la nube pública, aumentando el riesgo de no poder utilizar otros proveedores. Por ejemplo, cuando una empresa utiliza dos o más nubes públicas, como *Amazon Web Services* (AWS) y *Google*, la herramienta de configuración nativa no funcionará bien en ambas plataformas.

Las herramientas de gestión de configuración de terceros, basadas o no en la nube, funcionan con múltiples proveedores de la nube y ofrecen diferentes capas de abstracción para reducir la complejidad de esta gestión. Sin embargo, la ventaja que tienen estas herramientas de terceros respecto a la heterogeneidad

de su uso en la nube pública, pueden perderlo en capacidad. Estas herramientas de terceros renuncian a algunas funciones que ofrecen las herramientas nativas. Por ejemplo, muchas herramientas nativas ofrecen la posibilidad de actualizar repositorios (sistemas que almacenan datos sobre los recursos que se están rastreando) en tiempo real. Las herramientas de terceros suelen requerir que se realicen manualmente este tipo de tareas, lo que implica una mayor dedicación en cuanto a tiempo y un aumento de la posibilidad de error; sin embargo, funcionan en diferentes plataformas de nube.

La mejor opción, por ahora, es utilizar múltiples herramientas de gestión de la configuración de nube, aunque sea más costoso y complejo. Se pueden encontrar multitud de herramientas de gestión de configuración de la nube de terceros entre las que se pueden destacar las siguientes:

4.1.1 Chef

Es una herramienta de código abierto para la gestión y configuración [28]. Trabaja con un modelo *cliente-servidor* como se puede observar en la Figura 13, y está pensada para entornos de trabajo heterogéneos.

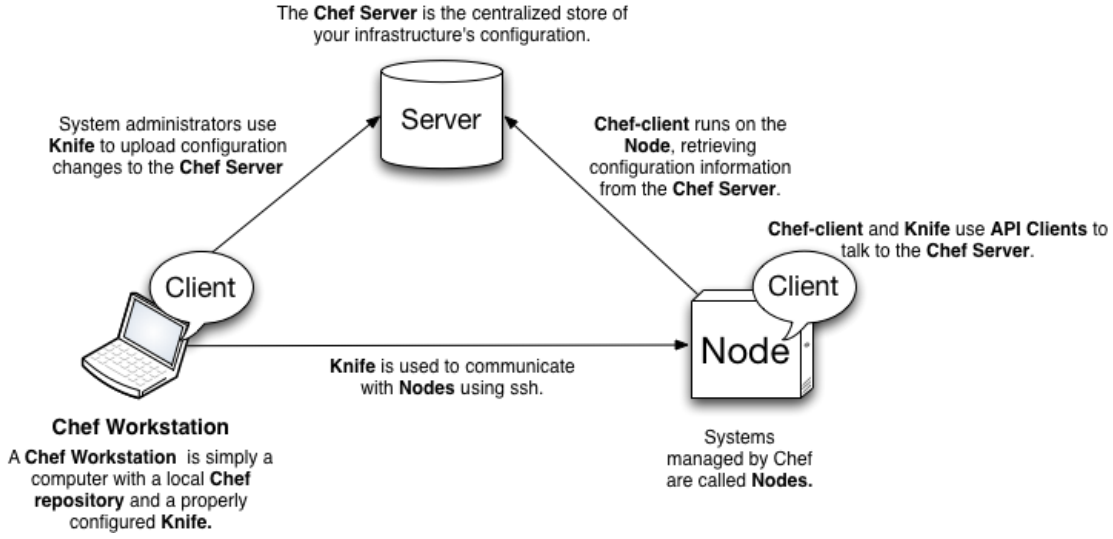

Figura 13: Modelo de gestión de Chef

Chef gestiona los procesos en los nodos y en las estaciones de trabajo usando los lenguajes de programación *Erlang* y *Ruby.* Se caracteriza por ser multiplataforma además de ser escalable y flexible, ofrece más seguridad que otras herramientas y es capaz de integrarse con las aplicaciones existentes, incluyendo varias bases de datos y directorios LDAP, y permite el descubrimiento y la provisión de los recursos públicos o privados.

Principalmente es utilizado para acelerar la tarea de configuración de los servidores de una empresa, así como su mantenimiento. Una de sus grandes ventajas es que se integra perfectamente con plataformas en la nube, desde *Amazon EC2* hasta *OpenStack, Google Cloud* o *Microsoft Azure*. Es un sistema muy práctico en todos los casos, con muchas variaciones y opciones de configuración.

Está compuesto por un servidor de gestión "*Chef server*", estaciones de trabajo "*Workstation*" las cuales se dedican a interactuar con el servidor y con los nodos, por tanto, sirven de intermediario entre los nodos y el servidor central. Los componentes de esta herramienta están estructurados de forma jerárquica, los cuales son gestionados a través de "*Chef Cookbooks*".

Como ventajas se puede destacar que posee una amplia colección de módulos y configuraciones preestablecidas, y que ofrece un control de versiones bastante robusto y funcional, además cuenta con la herramienta "*Knife*", un cliente SSH que rebaja las cargas de trabajo en instalaciones pesadas.

Como desventajas en cuanto a esta herramienta se destaca que su puesta en funcionamiento puede ser complicada, sobre todo si no se está familiarizado con Ruby, y que se no se considera una herramienta indicada para gestionar grandes paquetes de código en entornos muy complejos.

4.1.2 Puppet

Puppet fue creada con el objetivo de optimizar los procesos de configuración de forma práctica y rápida [29], automatizando tareas repetitivas, cuenta con una interfaz gráfica vía web o terminal para llevar a cabo su gestión.

La configuración de las máquinas puede ser establecida mediante código en el lenguaje propio de Puppet, esto hace que las relaciones entre los recursos y la configuración están escritas en código, lo que no solo facilita saber la configuración en sí, sino también conocer alguna configuración en específico, lo

cual contribuye a mejorar el proceso de integración consiguiendo desarrollar y probar aplicaciones con la misma configuración del sistema. Su gran ventaja es que se integra con herramientas de entornos en la nube como *Amazon Web Services*. Su arquitectura se basa en un modelo de cliente-servidor denominado "*Master-Agent*", "*Master*" hace referencia al servidor central el cual contiene el núcleo de las funciones de configuración, el cliente o "*Agent*" se encarga de gestionar los diferentes procesos en los nodos remotos, este tipo de arquitectura está pensada para usarse como un servicio que se ejecuta en cada nodo.

Si se emplea esta herramienta en un entorno en la nube, se debe hacer uso de su variante llamada "*Puppet Apply*", la cual trabaja de forma más rápida, ya que no requiere validación concreta de la ejecución, esto sin afectar a los niveles de seguridad ya que encapsula los recursos que se ejecutan por los diferentes nodos remotos. Su funcionamiento se basa en cuatro etapas que se observan en la Figura 14, primero el nodo envía información de si mismo, luego el "*master*" se encarga de enviarle la configuración y el nodo reporta que ha sido configurado.

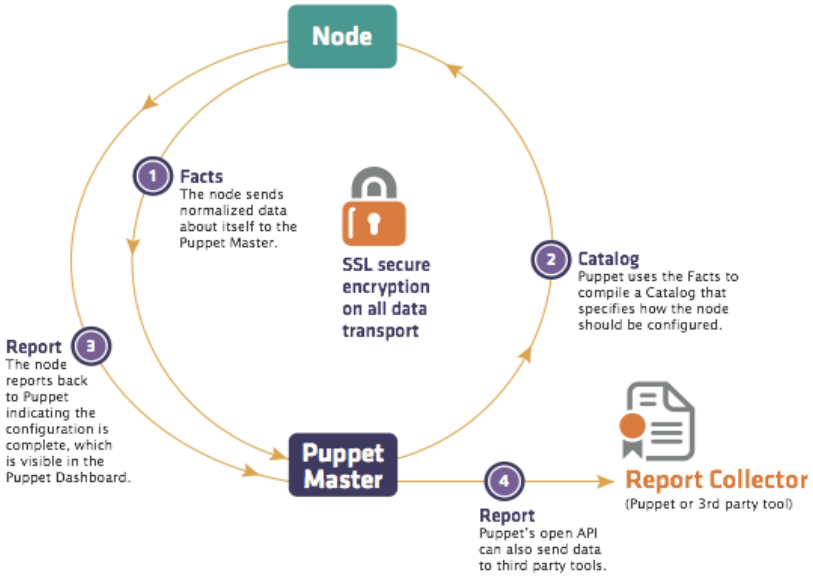

Figura 14: Funcionamiento de Puppet

Puppet no es una herramienta para la instalación desatendida del sistema operativo, sino para el despliegue de nuevos paquetes o la configuración del sistema una vez este ha sido instalado, así como para llevar a cabo estas tareas de una forma centralizada, consistente y escalable, es decir, poder desplegar la misma configuración y el mismo software en múltiples máquinas, aunque estas sean heterogéneas.

4.2 Herramientas de monitorización

Los proveedores de Cloud más importantes como *Amazon, Microsoft, OpenStack,* o *Google* ofrecen sus propios servicios de monitorización. Muchos fabricantes ofrecen "*plugins*" para integrarse con determinadas plataformas, pero no están pensados para un uso a largo plazo ya que cada plataforma amplia las API's e impone sus restricciones y condiciones con total libertad [30].

Por otro lado, la mayoría de fabricantes ofrecen métricas muy cercanas a la infraestructura, *EC2, VPC y S3* en AWS, instancias, discos, estado de máquinas, imágenes, almacenamiento, etc. Estas métricas de infraestructura son las mismas que puede obtener a través de agentes en las máquinas, de forma que no necesita una integración, simplemente seguir operando como de forma tradicional.

Como herramientas de monitorización de la red hoy en día se pueden encontrar distintas aplicaciones, muchas de ellas son de código abierto que pueden ser usadas de forma gratuita, entre las que se destacan las siguientes:

4.2.1 Nagios

Esta herramienta proporciona información en tiempo real del estado de los sistemas. [31] Su función principal es la de comprobar que, el equipo que está siendo monitoreado, responda las peticiones para evaluar su estado y en caso de que su comportamiento no sea el esperado Nagios notificará al administrador de la red de la forma que se establezca (correo electrónico, notificación sonora, etc.) mediante 5 estados identificados para los equipos y servicios que son "*Up*" "*Down*" "*Warning*" "*Flapping*" y "*Critical*".

Nagios permite ser gestionado vía web con el plugin *Centreon*, que no es más que el "entorno gráfico" de los archivos internos, por el cual se pueden agregar los equipos a monitorear, sin la necesidad de estar gestionando y configurando desde consola o editores de texto con permisos. También existe *Nagvis* que tiene la cualidad de ser manejado vía web al igual que Centreon, pero este *plugin* tiene la capacidad de mostrar de manera gráfica el estado de los equipos

y servicios que Nagios monitorea, permite crear mapas y observar el esquema de la red que se está monitorizando.

Es capaz de monitorizar los principales servicios de red como SMTP, POP3, HTTP, SNMP, etc.; además de los recursos hardware de los sistemas como la memoria, carga del procesador, ocupación de discos duros, estado de los puertos...

Se puede considerar que no es por completo una herramienta que pertenezca a la nube, pero añadiendo los *plugins* que han sido elaborados elevan su potencial, haciendo que Nagios se integre fácilmente a la red de aplicaciones en la nube y por tanto a un entorno cloud, por ello, en el capítulo 6 se monitorizará la red cloud privada mediante esta herramienta.

Nagios es un software libre bajo licencia y está disponible tanto para Linux como para Windows, aunque actualmente está cayendo en desuso cabe destacarla puesto que supuso el origen de la monitorización de redes y muchas de las nuevas herramientas actuales que han surgido han heredado su código y lo han evolucionado.

4.2.2 Cacti

Es una aplicación de código abierto [32] que recoge los datos en tiempo real de los dispositivos conectados a una red como routers, switches, servidores, tráfico de interfaces, cargas, temperatura... Es necesario que tengan habilitado el protocolo SNMP y conocer las MIBs con los distintos OIDs (identificadores de objeto) que se pueden monitorizar y visualizar, es posible también programar la colección de gráficas con las que queramos realizar el seguimiento., y permite visualizar fácilmente las estadísticas de dichos datos mediante gráficas. Esta herramienta funciona bajo entornos *Apache, PHP* y *MySQL*, por lo que permite poder usarla y gestionarla a través de un navegador.

Su funcionamiento es bastante sencillo, la aplicación sondea a cada uno de los hosts que tiene configurados solicitando los valores de los parámetros, OIDs, que tiene definidos y almacenando el valor. El período de sondeo es configurable por el administrador, éste determinará, entre otros factores, la precisión de la información a visualizar, ya que un período bajo aumentará la cantidad de datos

capturados y, por tanto, la resolución de la representación gráfica. Sin embargo, un período corto de muestreo aumentará la carga del sistema.

La ventaja de esta herramienta es que posee una gran comunidad activa de desarrolladores, que generan nuevas utilidades como plugins, scripts o plantillas y además es multiplataforma.

4.2.3 Zabbix

Es otra conocida herramienta para monitorizar la red [33], y cuya funcionalidad combina las características más interesantes de Nagios y Cacti. Su consola está basada en web, y dispone de un sistema de alertas como el de Nagios que recopila datos y los muestra gráficamente, como lo hace Cacti. Está disponible para los principales sistemas operativos y funciona con agentes que se ejecutan en sistemas supervisados, existiendo la posibilidad de no usar agente, si no usando los protocolos SNMP, SMTP y HTTP. Los agentes se pueden agregar de forma manual o por medio de autodescubrimiento.

4.3 Propias de la nube

Las herramientas también han ido evolucionando, y a la par que las herramientas tradicionales de gestión de red se han ido transformando para poder adaptarse al modelo cloud, han ido apareciendo nuevas herramientas que ya están pensadas y diseñadas para trabajar y gestionar un entorno en la nube.

Caben destacar las siguientes:

- **Pandora FMS**

  En su versión libre es capaz de monitorizar más de 10.000 nodos y cubrir sin limitaciones la monitorización de redes, de servidores y de aplicaciones. Es capaz de enviar informes y alertas.

- **GroundWork**

  Reutiliza diferentes softwares de *Nagios*, *Icinga* o *Cacti* para crear su solución global. Es una buena herramienta de monitorización de red ya que agrupa otras similares.

- **Zenoss**

  Con Zenoss es posible monitorizar almacenamiento, redes, servidores, aplicaciones y servidores virtuales sin necesidad de instalar agentes.

- **Monitis**

  Esta herramienta de monitorización de red ofrece un alto rendimiento y es compatible con la nube de Amazon y de otros proveedores. También posee un gran diseño de su interfaz, y por ello resulta fácil de utilizar y además cuenta con elementos adicionales para su personalización.

- **Icinga**

  Se integra con varias bases de datos y destaca su interfaz REST API para integrar otras aplicaciones. Está muy enfocada a redes complejas y monitorizaciones de protocolos, recursos de máquinas y servidores.

- **Op5 Monitor**

  Es capaz de monitorizar múltiples plataformas, sistemas en la nube y entornos virtuales. Esta herramienta de monitoreo está muy centrada en monitorización de hardware, tráfico de red y servicios y destaca su capacidad para grandes entornos.

- **Solarwinds**

  Destaca entre las demás por su automático mapeo de redes y nodos sin necesidad de acciones manuales. Tiene un interfaz gráfico bastante potente en el que se puede ver fácilmente la topología de red y el estado de la misma, también permite integrar máquinas virtuales en su monitorización.

Capítulo 5

Implementación de un entorno cloud sobre OpenStack

En este capítulo se explica cómo crear y gestionar un entorno cloud a través de *Openstack* y su posterior monitorización mediante *Nagios Core*.

5.1 Introducción a OpenStack

OpenStack es una solución cloud del tipo IaaS de código abierto [34].

Su objetivo es proveer una solución flexible tanto para nubes públicas como privadas, sean estas de cualquier tamaño, y para esto se requieren dos condiciones: las nubes deben ser simples de implementar y masivamente escalables.

Para cumplir con estos principios OpenStack está divido en diferentes componentes que trabajan conjuntamente. Esta integración se logra a través de interfaces de programación de aplicaciones (APIs) que cada servicio ofrece y consume.

Gracias a estas APIs, los servicios pueden comunicarse entre ellos y además se posibilita que un servicio sea reemplazado por otro de similares características siempre que se respete la forma de comunicación. Es decir, OpenStack es extensible y se ajusta a las necesidades de quien desee implementarlo.

En la Figura 15 se puede ver como se encuentra estructurado conceptualmente, así como las relaciones entre los servicios existentes en OpenStack.

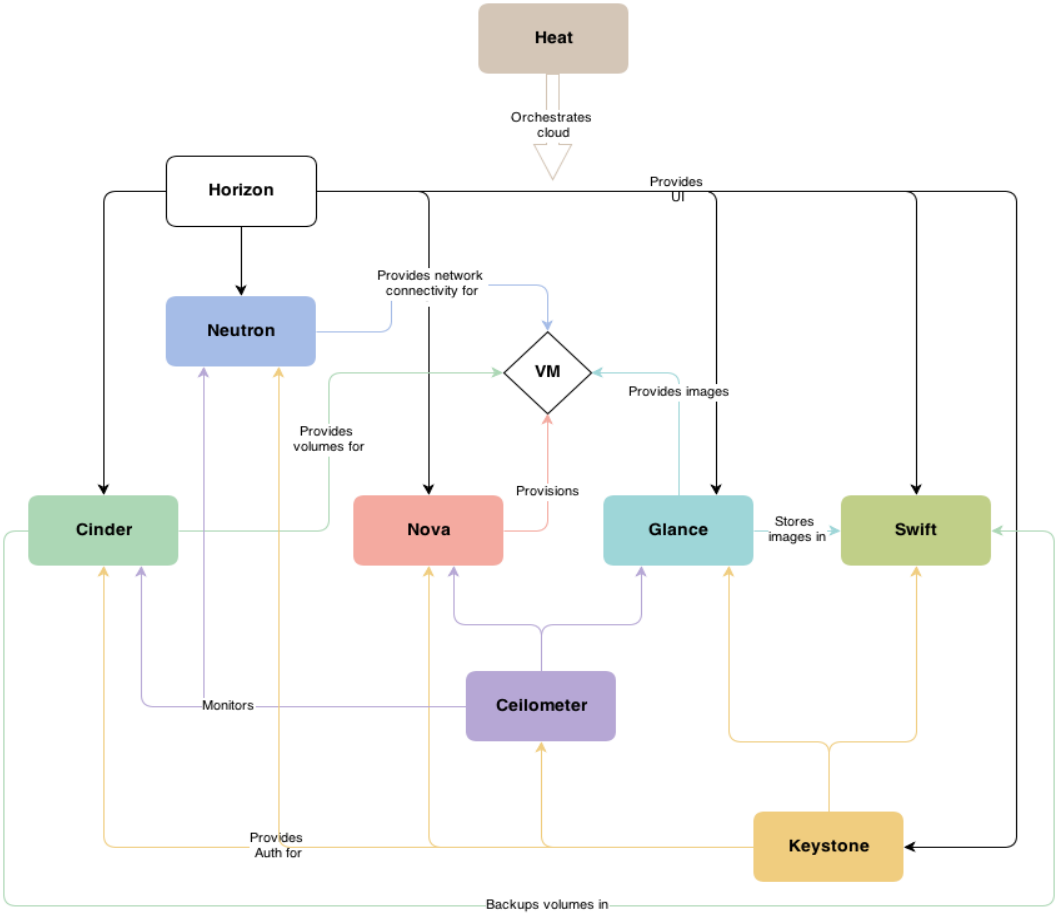

Figura 15: Relaciones entre los servicios

Los módulos de los que se compone OpenStack [35] son los siguientes:

- *"Horizon"* (*Dashboard*) Provee una interfaz a los usuarios finales y al administrador de todos los servicios.
- *"Nova"* (*Compute*) Recupera imágenes y metadatos asociados, y transforma los pedidos de los usuarios en máquinas virtuales.
- *"Neutron"* (*Network*) Provee redes virtuales como servicio entre dispositivos administrados por otros servicios de OpenStack
- *"Cinder"* (*Block Storage*) Provee almacenamiento persistente a las VMs alojadas en la nube.
- *"Glance"* (*Image*) Provee un catálogo y un repositorio para las imágenes.
- *"Swift"* (*Object Store*) Provee almacenamiento de objetos. Sirve como un contenedor en el que se pueden almacenar archivos y recuperarlos luego.
- *"Keystone"* (*Identity*) Provee autenticación y autorización para todos los servicios de OpenStack.
- *"Ceilometer"*. Mide y monitoriza el uso de la nube.
- *"Heat"* Se encarga de la organización de todos los servicios.

5.2 Creación de nube privada

Como prueba se crea una nube privada usando la solución de *Openstack* implementada a través de la instalación de *devstack* y en el que se configurará el entorno en la nube para ofrecer un servicio de hosting y red para un cliente de un servicio cloud, así como se monitorizará su rendimiento a través de Nagios. *Devstack* [36] permite realizar una instalación más sencilla de la nube Openstack en una máquina virtual, en este caso con el sistema operativo de Ubuntu 16.04 montada a través de VirtualBox. Este proceso, así como la imagen (.iso) utilizada en la máquina virtual montada en VirtualBox, está basado en la guía "*DECAMP: Lab 1. Understanding a Cloud Based Environment*" [37] en la que se explica cómo configurar los parámetros de red en VirtualBox, montar la máquina virtual e instalar sobre ella la plataforma devstack.

De manera gráfica se puede observar en la Figura 16 la pila de la arquitectura implementada que se detalla a continuación:

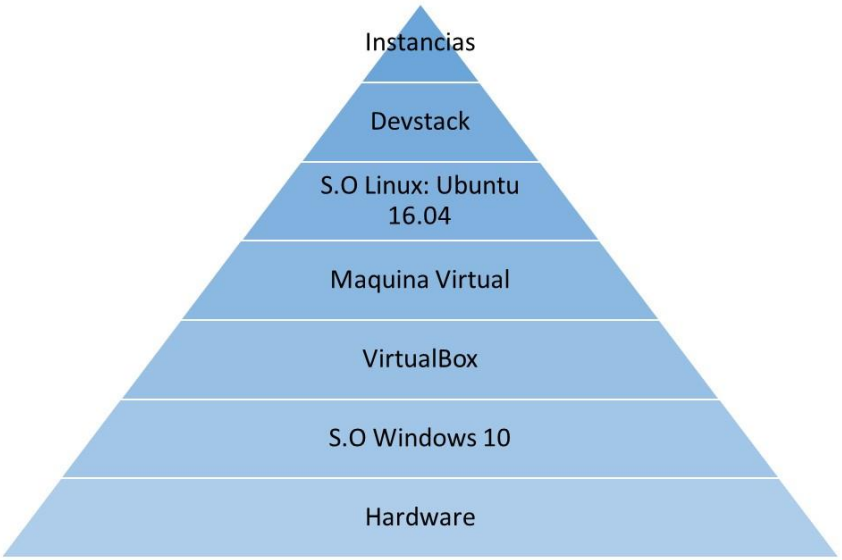

Figura 16: Pila de la arquitectura implementada

Comenzando desde el nivel más bajo hasta el nivel superior, la arquitectura implementada consta de:

- **Hardware:** Se refiere al hardware real, al conjunto de elementos físicos sobre el que se constituye el sistema informático y el que se están ejecutando todos los niveles superiores.

En este caso el hardware utilizado es el proporcionado por el portátil Lenovo G50 que consta de un procesador Intel Core i5-5200U CPU a 2.20GHz, 8 GB de memoria RAM y 500 GB de disco duro.

- **S.O Windows 10**: Este equipo cuenta con el sistema operativo de Windows 10 Pro 64 bits.

- **VirtualBox:** Oracle VM VirtualBox es un software de virtualización instalado sobre el S.O Windows 10 visto anteriormente y que permite instalar sistemas operativos adicionales, conocidos como "sistemas invitados", dentro del sistema operativo "anfitrión", cada uno con su propio ambiente virtual.

- **Máquina Virtual:** Es un software que simula a un equipo y puede ejecutar programas como si fuese uno real. Una característica esencial es que los procesos que ejecutan están limitados por los recursos y abstracciones proporcionados por ellas.

  Para el entorno de pruebas empleado en este trabajo se ha creado una máquina virtual con 60 GB de disco virtual y 4 GB de RAM.

- **S.O Ubuntu 16.04:** Sobre la máquina virtual se monta el S.O Ubuntu 16.04 a partir de la imagen "*minimal*" descargada de la siguiente dirección http://archive.ubuntu.com/ubuntu/dists/xenial/main/installer-amd64/current/images/netboot/mini.iso

- **Devstack:** Sobre Ubuntu se instala devstack, como se explicará en este capítulo para crear un entorno privado en la nube en el cual realizar pruebas.

- **Instancias:** Las instancias son las máquinas virtuales que se ejecutan en los nodos de computación. El servicio de computación, Nova, gestiona estas instancias. Se pueden lanzar cualquier número de instancias a partir de una determinada imagen. En este caso se creará una instancia con la distribución Linux de Debian, como se verá más adelante.

Para comenzar, es necesario configurar en VirtualBox dos interfaces de red, uno hace la función de NAT (*Network Address Translation*) entre la máquina virtual y el equipo real (Lenovo G50 con Windows 10, Intel Core i5 y con 8 GB de memoria RAM, los cuales 4 de ellos se dedican a continuación a la MV) y una segunda interfaz de red definida en el rango 192.168.56.X y que mediante DHCP asignará direcciones IP en el rango desde la dirección 192.168.56.101 hasta la 192.168.56.254 que se utilizará para montar la MV.

Una vez creada la MV con Ubuntu 16.04 los primeros pasos realizados son dotar de permisos al usuario "*stack*" en este caso y editar el archivo de las interfaces de red localizado en *"/ect/network/interfaces*" para añadir las siguientes líneas, necesarias para añadir el nuevo interfaz de red sobre el que se montará el entorno devstack:

```
#auto enp0s8

#iface enp0s8 inet dhcp
```

Siendo "enp0s8" el interfaz de red a utilizar, una vez hecho esto se procede a descargar, configurar e instalar devstack. Su descarga se realiza a través del siguiente enlace (https://git.openstack.org/openstack-dev/devstack)

Una vez descargado se edita el archivo de configuración "*local.conf*" siendo necesario cambiar las contraseñas poniendo todas iguales para evitar posibles errores más adelante; en la Figura 17 se puede observar como la contraseña elegida es "stack", además en el mismo archivo de configuración y como se refleja en la misma figura también es necesario cambiar la IP del Host por la que tiene el interfaz de red que utiliza devstack, en este caso la IP es la 192.168.56.102. Este archivo de configuración "*local.conf*" se puede encontrar completo en el Anexo 1.

```
# While ``stack.sh`` is happy to run without ``localrc``, devlife is better when
# there are a few minimal variables set:

# If the ``*_PASSWORD`` variables are not set here you will be prompted to enter
# values for them by ``stack.sh``and they will be added to ``local.conf``.
ADMIN_PASSWORD=stack
DATABASE_PASSWORD=stack
RABBIT_PASSWORD=stack
SERVICE_PASSWORD=$ADMIN_PASSWORD

# ``HOST_IP`` and ``HOST_IPV6`` should be set manually for best results if
# the NIC configuration of the host is unusual, i.e. ``eth1`` has the default
# route but ``eth0`` is the public interface.  They are auto-detected in
# ``stack.sh`` but often is indeterminate on later runs due to the IP moving
# from an Ethernet interface to a bridge on the host. Setting it here also
# makes it available for ``openrc`` to include when setting ``OS_AUTH_URL``.
# Neither is set by default.
HOST_IP=192.168.56.102
#HOST_IPV6=2001:db8::7
```

Figura 17: Archivo local.conf de devstack

Para su instalación tan solo es necesario ejecutar el archivo "*stack.sh*". El resultado obtenido se puede observar en la Figura 18, así como su acceso vía web en la Figura 19.

```
=========================
DevStack Component Timing
=========================
Total runtime    4449

run_process       77
test_with_retry   24
apt-get-update    12
pip_install      873
osc              655
wait_for_service  80
git_timed        462
dbsync           105
apt-get          615
=========================



This is your host IP address: 192.168.56.102
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.56.102/dashboard
Keystone is serving at http://192.168.56.102/identity/
The default users are: admin and demo
The password: stack
Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/developer/devstack/systemd.html
stack@ubuntu:~/devstack$
```

Figura 19: Instalación devstack

Figura 18: Acceso vía web a Openstack

Una vez que se accede vía web al *dashboard* de Openstack (en este caso a través de la dirección *http://192.168.56.102/dashboard*) se crea un entorno en la nube basado en dos redes y una instancia (ClientDebian1) en una de ellas, ambas redes conectadas mediante un router (R1) a la red pública.

Para poder acceder desde la máquina virtual devstack hasta las dos redes internas en la nube es necesario establecer dos rutas estáticas, una por cada red, indicando la red a la que se quiere acceder (64.64.64.0 o la 100.100.100.0), la máscara de dicha red (255.255.255.0) y el Gateway (172.24.4.10) como se indica en la Figura 20:

```
stack@ubuntu:~/devstack$ sudo route add -net 64.64.64.0 netmask 255.255.255.0 gw 172.24.4.10
stack@ubuntu:~/devstack$ sudo route add -net 100.100.100.0 netmask 255.255.255.0 gw 172.24.4.10
```

Figura 20: Rutas estáticas desde la MV hacia las redes virtuales internas

También es necesario establecer las rutas estáticas desde el propio equipo, en el caso que sea necesario acceder desde el equipo real a las redes en la nube, lo cual se realizará más tarde para conectarse a las instancias creadas a través de SSH con Putty, estas rutas son creadas de la manera que refleja la Figura 21.

```
C:\WINDOWS\system32>route ADD 64.64.64.0 MASK 255.255.255.0 192.168.56.102
 Correcto

C:\WINDOWS\system32>route ADD 100.100.100.0 MASK 255.255.255.0 192.168.56.102
 Correcto
```

Figura 21: Rutas estáticas desde el equipo hacia las redes virtualizadas

A su vez, para acceder a las interfaces del router R1 se necesita establecer unas determinadas reglas de seguridad y de acceso a estas redes, esto se puede configurar en los "*grupos de seguridad*" a través del *dashboard* en el menú de "*Red*", como en la Figura 22, que se muestra las reglas añadidas para aceptar tráfico ICMP, TCP y UDP.

☐	Dirección	Tipo Ethernet	Protocolo IP	Rango de puertos	Prefijo de IP Remota	Grupo de Seguridad Remoto	Acciones
☐	Saliente	IPv4	Cualquier	Cualquier	0.0.0.0/0	-	Eliminar Regla
☐	Entrante	IPv4	Cualquier	Cualquier	-	default	Eliminar Regla
☐	Saliente	IPv6	Cualquier	Cualquier	::/0	-	Eliminar Regla
☐	Entrante	IPv6	Cualquier	Cualquier	-	default	Eliminar Regla
☐	Entrante	IPv4	ICMP	Cualquier	0.0.0.0/0	-	Eliminar Regla
☐	Entrante	IPv4	TCP	1 - 65535	0.0.0.0/0	-	Eliminar Regla
☐	Entrante	IPv4	UDP	1 - 65535	0.0.0.0/0	-	Eliminar Regla

Figura 22: Reglas de seguridad

Para el entorno de prueba que se va a crear se ha decidido crear 1 instancia con la distribución de Linux de Debian, se ha escogido Debian puesto que es una distribución libre y presenta una gran estabilidad, además de estar mantenida por sus usuarios lo que hace que ofrezca un soporte bastante reseñable, su instalación es sencilla y se trata de una distribución rápida y ligera en memoria,

por lo que encajan perfectamente en este planteamiento, ya que la falta de recursos ha sido el principal impedimento de creación de distintas instancias.

Para descargar la imagen de Debian con la cual se implementará una instancia de *Openstack* se puede realizar de dos formas distintas, desde la consola de la máquina virtual o desde el propio *dashboard* de *Openstack*.

En el primer caso, si se hace desde la consola de la máquina virtual, se emplea el comando *wget* y a continuación la dirección donde se descargará la imagen (con formato qcow2), como indica la Figura 23, hasta que nos indica que se ha guardado correctamente.

Figura 23: Descarga de la imagen qcow2

Con la imagen ya descargada, a través de *glance* se monta la imagen para que forme parte del sistema cloud, usando el siguiente comando:

```
$ glance -os-username=admin -os-password=stack -os-tenant=admin -os-auth-url=http://localhost:5000/v2.0 image-create -name Debian -disk-format qcow2 -container-format bare < Debian.qcow2
```

Esto se puede hacer de manera más sencilla, descargando directamente la imagen (qcow2) al equipo real (Lenovo G50 en este caso) y a través del *dashboard* en el menú "*Compute\Imagenes*" añadiendo una nueva imagen, como se indica en la Figura 24.

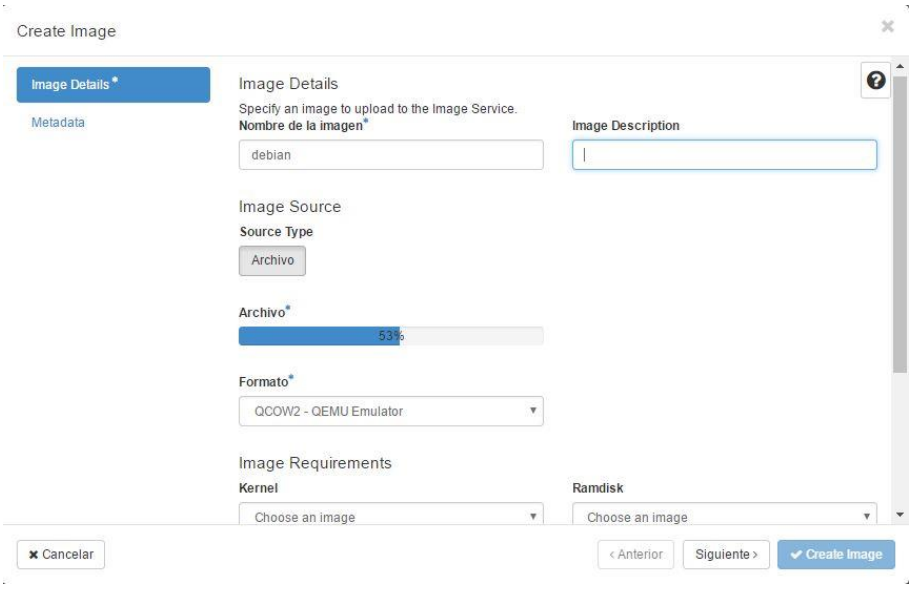

Figura 24: Creación de imagen en Openstack

Una vez que esta creada la imagen, lo cual se puede observar en la Figura 25, se necesitará un determinado sabor (*flavor*) para poder crear la instancia. Un sabor define para una instancia principalmente la capacidad de disco y la RAM, en esto caso como la MV de *devstack* tiene 80 GB de disco y 4 GB de RAM, se ha decidido crear un sabor especifico que consta de 25 GB de disco y 1 GB de RAM, al que se llama "*debianflavor*". Esto se puede hacer a través del propio *dashboard*, en el menú de "*Administrador\Compute\Sabores*". En la Figura 26 se presentan todos los sabores existentes en este proyecto.

Figura 25: Imagen debian creada

Hay que tener en cuenta que es necesario también crear un par de claves (SSH) para poder conectarse por consola a la instancia una vez este creada desde *Putty,* por ejemplo, para ello se crea el par de clave SSH como se indica en la Figura 27 y se guarda en la siguiente ruta *(/home/stack/ssh-key)*, de esta forma se puede observar que una vez creada, genera dos claves, una privada (*ssh-key*) y una publica (*ssh-key.pub*)

Nombre del sabor	VCPU	RAM	Disco raíz	Disco efímero	Disco de intercambio (swap)	Factor RX/TX	ID	Público	Metadatos	Acciones
cirros256	1	256MB	0GB	0GB	0MB	1,0	c1	Sí	no	Editar Sabor
debianflavor	1	1000MB	25GB	0GB	0MB	1,0	6	Sí	no	Editar Sabor
ds1G	1	1GB	10GB	0GB	0MB	1,0	d2	Sí	no	Editar Sabor
ds2G	2	2GB	10GB	0GB	0MB	1,0	d3	Sí	no	Editar Sabor
ds4G	4	4GB	20GB	0GB	0MB	1,0	d4	Sí	no	Editar Sabor
ds512M	1	512MB	5GB	0GB	0MB	1,0	d1	Sí	no	Editar Sabor
m1.large	4	8GB	80GB	0GB	0MB	1,0	4	Sí	no	Editar Sabor
m1.medium	2	4GB	40GB	0GB	0MB	1,0	3	Sí	no	Editar Sabor
m1.micro	1	128MB	0GB	0GB	0MB	1,0	84	Sí	no	Editar Sabor
m1.nano	1	64MB	0GB	0GB	0MB	1,0	42	Sí	no	Editar Sabor
m1.small	1	2GB	20GB	0GB	0MB	1,0	2	Sí	no	Editar Sabor
m1.tiny	1	512MB	1GB	0GB	0MB	1,0	1	Sí	no	Editar Sabor
m1.xlarge	8	16GB	160GB	0GB	0MB	1,0	5	Sí	no	Editar Sabor

Figura 26: Sabores (Flavor)

```
stack@ubuntu:/$ ssh-keygen -q -f /home/stack/ssh-key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
stack@ubuntu:/$ ls /home/stack/ | grep ssh-key
ssh-key
ssh-key.pub
```

Figura 27: Creación par de claves SSH

Ahora, se añade este par de claves al entorno de *Openstack*, para ello se emplea el comando que se observa en la Figura 28.

```
stack@ubuntu:/$ nova keypair-add key-nova --pub-key /home/stack/ssh-key.pub
```

Figura 28: Añadir par de claves SSH al entorno Openstack

A través del comando "*nova keypair-list*" se pueden ver la lista de par de claves SSH generadas que forman parte del entorno de Openstack, como se ve la Figura 29.

```
stack@ubuntu:/$ nova keypair-list
+-----------+------+-------------------------------------------------+
| Name      | Type | Fingerprint                                     |
+-----------+------+-------------------------------------------------+
| key-nova | ssh  | 66:09:05:6d:46:55:ad:20:c8:9c:46:df:b4:c4:18:1a |
+-----------+------+-------------------------------------------------+
```

Figura 29: Listado de par de claves SSH

Una vez que está creada la clave SSH, la imagen y el sabor a utilizar ya se puede lanzar una determinada instancia, pero para evitar errores de

autenticación se emplea primero el siguiente comando con las credenciales de origen para la ejecución de comandos: `source accrc/admin/admin`

Tras esto, se lanza la instancia como indica la Figura 30 a través del comando "*nova*", indicando el nombre del proyecto *(--os-project-name=admin)*, del usuario *(--os-username=admin)*, su contraseña *(--os-password=stack)*, el sabor o flavor a emplear por la instancia *(--flavor debianflavor)*, así como la imagen *(--image debian)*, la clave SSH asociada a la instancia *(--key-name keynova)*, el identificador de red en la que se desea colocar la instancia *(--nic net id)*y finalmente el nombre de la propia instancia (*ClientDebian1*)

```
stack@ubuntu:/$ nova --os-project-name=admin --os-username=admin --os-password=stack --os-tenant-nam
e=admin boot --flavor debianflavor --image debian --key-name key-nova --nic net-id=4d174204-b999-402
b-a599-4db324d9a307 ClientDebian1
+--------------------------------------+----------------------------------------------------+
| Property                             | Value                                              |
+--------------------------------------+----------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                             |
| OS-EXT-AZ:availability_zone          |                                                    |
| OS-EXT-SRV-ATTR:host                 | -                                                  |
| OS-EXT-SRV-ATTR:hostname             | clientdebian1                                      |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                                  |
| OS-EXT-SRV-ATTR:instance_name        |                                                    |
| OS-EXT-SRV-ATTR:kernel_id            |                                                    |
| OS-EXT-SRV-ATTR:launch_index         | 0                                                  |
| OS-EXT-SRV-ATTR:ramdisk_id           |                                                    |
| OS-EXT-SRV-ATTR:reservation_id       | r-g63q90oa                                         |
| OS-EXT-SRV-ATTR:root_device_name     | -                                                  |
| OS-EXT-SRV-ATTR:user_data            | -                                                  |
| OS-EXT-STS:power_state               | 0                                                  |
| OS-EXT-STS:task_state                | scheduling                                         |
| OS-EXT-STS:vm_state                  | building                                           |
| OS-SRV-USG:launched_at               | -                                                  |
| OS-SRV-USG:terminated_at             | -                                                  |
| accessIPv4                           |                                                    |
| accessIPv6                           |                                                    |
| adminPass                            | K3h2bdaGpfVB                                       |
| config_drive                         |                                                    |
| created                              | 2017-06-25T16:55:48Z                               |
| description                          | -                                                  |
| flavor                               | debianflavor (6)                                   |
| hostId                               |                                                    |
| host_status                          |                                                    |
| id                                   | 9d5f40ab-dfe6-4a75-918d-2c3ffbbc4f5d               |
| image                                | debian (0490b3d3-b967-4b6f-a341-ab5b6821a1a9)      |
| key_name                             | key-nova                                           |
| locked                               | False                                              |
| metadata                             | {}                                                 |
| name                                 | ClientDebian1                                      |
| os-extended-volumes:volumes_attached | []                                                 |
| progress                             | 0                                                  |
| security_groups                      | default                                            |
| status                               | BUILD                                              |
| tags                                 | []                                                 |
| tenant_id                            | 38ccc9f908b44678922685ca76178eef                   |
| updated                              | 2017-06-25T16:55:48Z                               |
| user_id                              | d5d9490b3bb44ee48a6ae325859539af                   |
+--------------------------------------+----------------------------------------------------+
```

Figura 30: Creación instancia ClientDebian1

A través del comando: `nova list,` empleado como en la Figura 31 se puede sacar una lista de las instancias creadas y su IP, y con el comando: `df -h` se puede observar la ocupación del disco de la MV observando en la Figura 32 que habiendo instalado devstack la ocupación del disco es de tan solo el 26%.

```
stack@ubuntu:~/devstack$ nova list
+--------------------------------------+---------------+--------+------------+-------------+--------
----------------+
| ID                                   | Name          | Status | Task State | Power State | Network
s              |
+--------------------------------------+---------------+--------+------------+-------------+--------
----------------+
| 2607abb5-dbd4-4f86-b3be-62adbf9cc7cb | ClientDebian1 | ACTIVE | -          | Running     | Clients
=100.100.100.10 |
```

Figura 31: Instancias creadas

```
stack@ubuntu:~/devstack/images$ df -h
S.ficheros     Tamaño Usados  Disp Uso% Montado en
udev             1,9G      0  1,9G   0% /dev
tmpfs            386M    40M  347M  11% /run
/dev/sda1         46G    11G   33G  26% /
tmpfs            1,9G      0  1,9G   0% /dev/shm
tmpfs            5,0M      0  5,0M   0% /run/lock
tmpfs            1,9G      0  1,9G   0% /sys/fs/cgroup
tmpfs            386M      0  386M   0% /run/user/1000
```

Figura 32: Ocupación del disco de la MV

Así finalmente, la topología de red con la que se trabaja y se utiliza para monitorizar a través de Nagios, como se verá en el siguiente apartado, queda de la forma que se observa en la Figura 33.

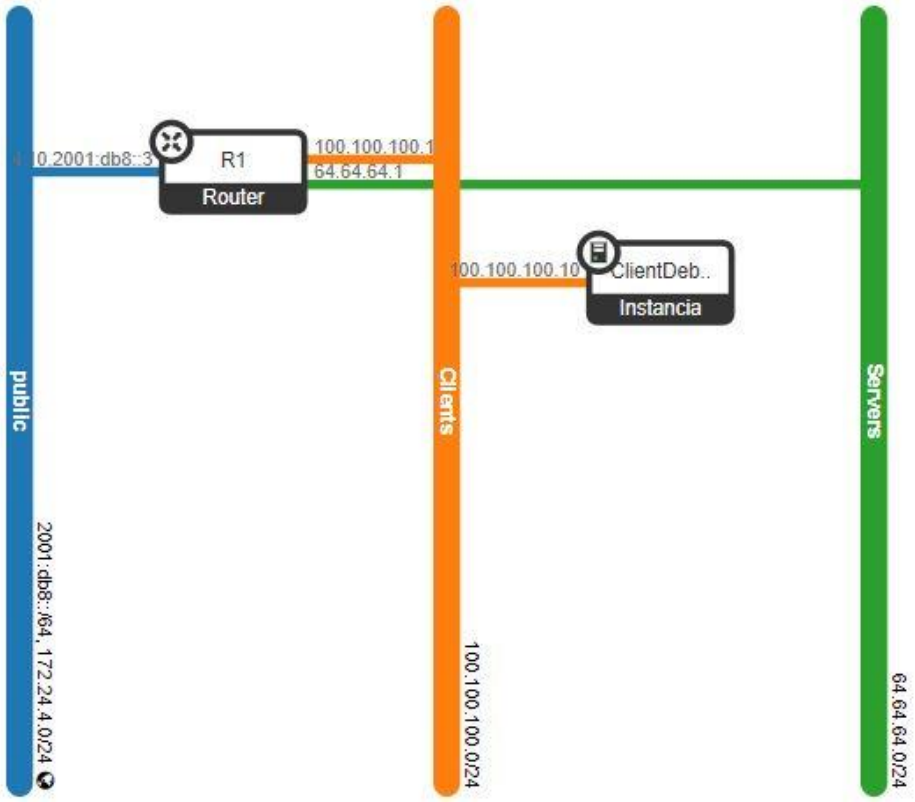

Figura 33: Topología de red

Consta de las dos redes, de la instancia o máquina virtual *ClientDebian1*, del Router R1 y de la red que por defecto utiliza la MV de devstack para acceder a este entorno cloud que es la red "publica" (172.24.4.0/24). De manera más gráfica a través del *dashboard* es posible obtener la topología de la red creada como se puede observar en la Figura 34.

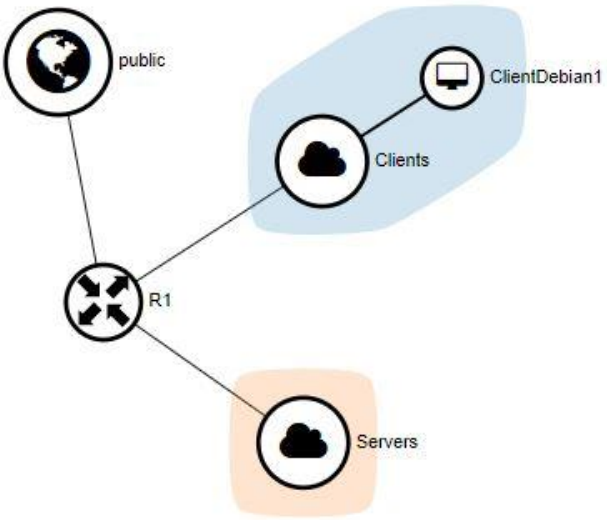

Figura 34: Gráfica de la topología de red

Capítulo 6

Creación de un entorno de gestión basado en Nagios sobre devstack

A continuación, se instala en la misma máquina virtual en la que se ha instalado el devstack la herramienta de monitorización Nagios, con el objetivo de monitorizar desde la propia MV todo el entorno cloud.

6.1 Topología del entorno a gestionar

La topología de red del entorno a gestionar es la creada con anterioridad sobre devstack y que puede observarse en la Figura 35, incluidas las direcciones IP de cada instancia y cada interfaz.

Nagios se instalará sobre la misma máquina virtual en la previamente se ha instalado devstack que posee la dirección IP 192.168.56.102, y es de especial importancia destacar la IP 172.24.4.1 ya que es por la cual la MV accede al entorno creado por devstack en el que se incluyen las instancias y routers a gestionar, mientras que es por la interfaz de red NAT (10.0.2.0/24) por la que accede la MV a Internet, dentro de la nube Openstack cabe señalar el router con dirección IP 172.24.4.10 y las dos redes creadas (100.100.100.0/24 y 64.64.64.0/24) encontrándose la instancia monitorizar en la primera red "*Clients*" con IP: 100.100.100.10

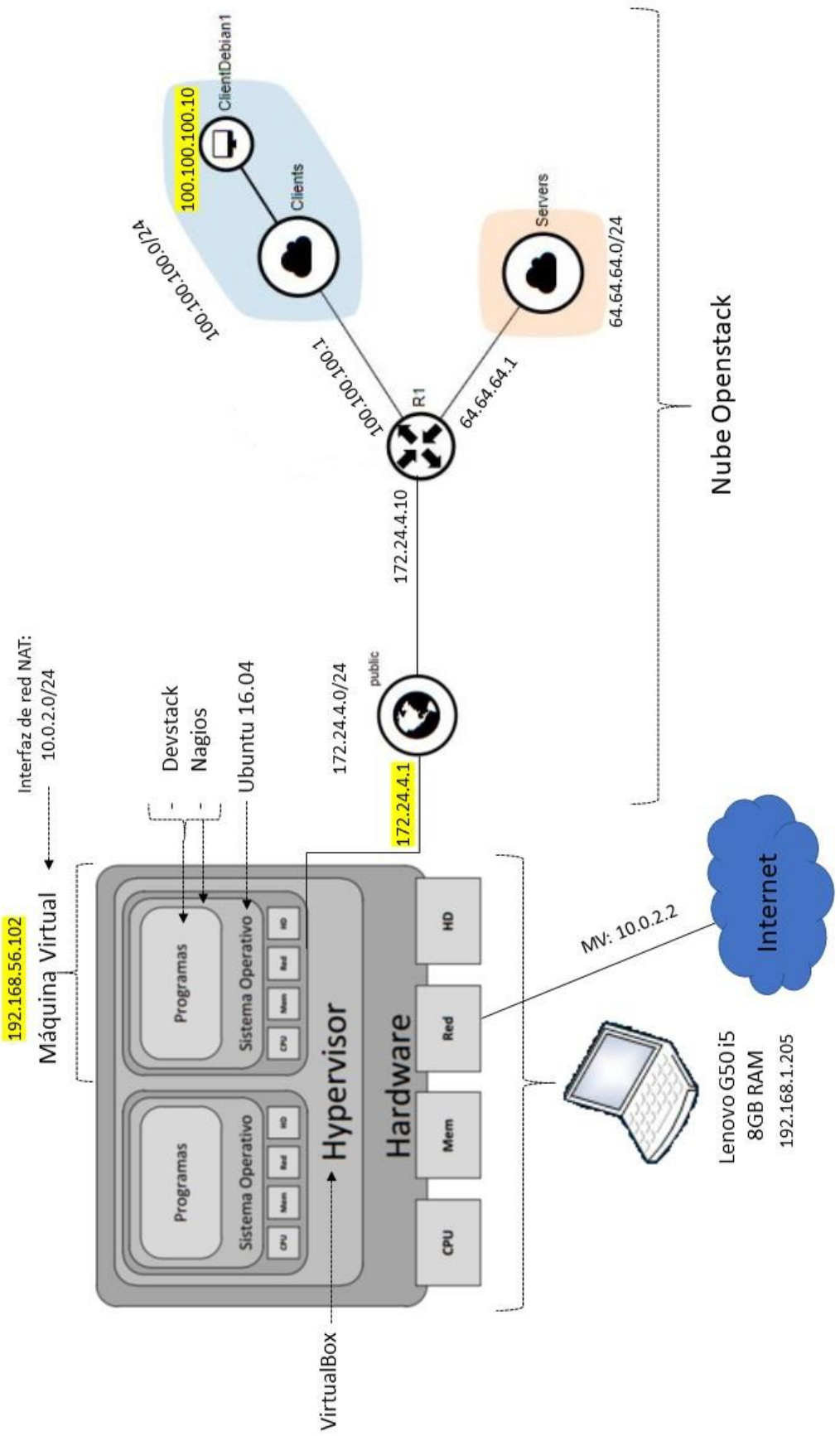

Figura 35: Topología de red a gestionar

6.2 Instalación y configuración de Nagios

Para instalar Nagios [38], primero es necesario descargar los paquetes necesarios a través del comando wget y la dirección de descarga como se puede observar en la Figura 36, y una vez descargados se compilan a través de los siguientes comandos:

```
# cd /tmp/nagioscore-nagios-4.3.2/
# sudo ./configure --with-httpd-conf=/etc/apache2/sites-enabled
# sudo make all
```

Figura 36: Descarga de Nagios

Mediante las siguientes instrucciones se crean un usuario y un grupo que se utilizarán durante todo el proceso de instalación.

```
# sudo useradd nagios
# sudo usermod -a -G nagios www-data
```

Se instalan los archivos binarios, CGIs y los archivos HTML necesarios:

```
# sudo make install
```

Se instala el servicio y se configura para encenderse.

```
# sudo make install-init
```

```
# sudo update-rc.d nagios defaults
```

También se instala y configura el archivo de comandos externos, así como los archivos de configuración de ejemplos necesarios para comenzar.

```
# sudo make install-commandmode
```

```
# sudo make install-config
```

Para instalar el servidor web Apache[17] y su configuración se emplean los siguientes comandos:

`# sudo make install-webconf` (En la Figura 37 se indica la correcta instalación)

```
#sudo a2enmod rewrite
```

```
#  sudo a2enmod cgi
```

```
stack@ubuntu:/tmp/nagioscore-nagios-4.3.2$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/apache2/sites-enabled/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/apache2/sites-enabled/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***
```

Figura 37: Instalación de la configuración de Nagios y Apache

Para poder acceder a la interfaz web de Nagios es necesario permitir el puerto 80 en el firewall, como se observa el resultado de los siguientes comandos en la Figura 38.

```
# sudo ufw allow Apache
```

```
# sudo ufw reload
```

```
stack@ubuntu:/tmp/nagioscore-nagios-4.3.2$ sudo ufw allow Apache
Reglas actualizadas
Reglas actualizadas (v6)
```

Figura 38: Permitir Apache en las reglas del Firewall

Para crear el usuario con el que accede a la consola web Nagios se emplea la instrucción:

```
# sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

Y finalmente se reinicia el servicio del servidor web Apache mediante el comando indicado en la Figura 39 y se inicia el servicio de Nagios a través de la instrucción indicada en la Figura 40.

```
stack@ubuntu:/tmp/nagioscore-nagios-4.3.2$ sudo service apache2 restart
```

Figura 39: Reinicio del servicio Nagios

```
stack@ubuntu:/tmp/nagioscore-nagios-4.3.2$ sudo service nagios start
```

Figura 40: Iniciar el servicio Nagios

[17] Apache es un servidor web *open source* que puede instalarse tanto en Linux como en Windows y en este caso es usado para proveer la interfaz web de Nagios Core.

Una vez reiniciado el servicio ya se puede acceder vía web a la herramienta de monitorización Nagios, a través en este caso de la dirección *http://192.168.56.102/nagios* y mostrará tal y como se ve en la Figura 41.

Figura 41: Acceso vía web a Nagios

En la pestaña de "*Hosts*" donde aparecen todos los dispositivos que se estén monitorizando, se observa que se producen unos errores que impiden su correcta monitorización, este error se puede ver en la Figura 42.

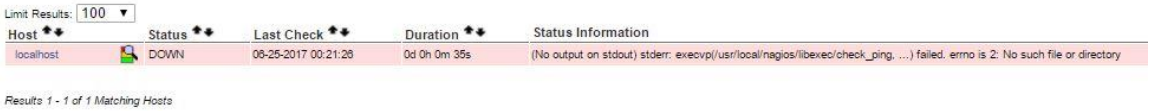

Figura 42: Errores Hosts Nagios

Esto se soluciona instalando una serie de plugins necesarios para el correcto funcionamiento de la herramienta, la descarga de estos plugins se realiza tal como aparece en la Figura 43.

Figura 43: Instalación de plugins necesarios en Nagios

Previamente, hay que descargarse una serie de paquetes que son necesarios para la instalación, como son los siguientes:

```
# sudo apt-get install -y autoconf gcc libc6 libmcrypt-dev make libssl-dev wget bc gawk dc build-essential snmp libnet-snmp-perl gettext
```

Una vez que los plugins han sido descargados correctamente, se procede a su compilación e instalación haciendo uso de los siguientes comandos:

```
# cd /tmp/nagios-plugins-release-2.2.1/

# sudo ./tools/setup

# sudo ./configure

# sudo make

# sudo make install
```

Tras su instalación, es necesario reiniciar los servicios empleando el comando: `sudo "service nagios restart"` y pedirá credenciales de autenticación como se ve en la Figura 44.

Figura 44: Reinicio del servicio de Nagios

Una vez reiniciado el servicio se observa ya en la interfaz web, en la Figura 45, que no presenta el anterior error mostrado y se empieza a monitorizar sin errores.

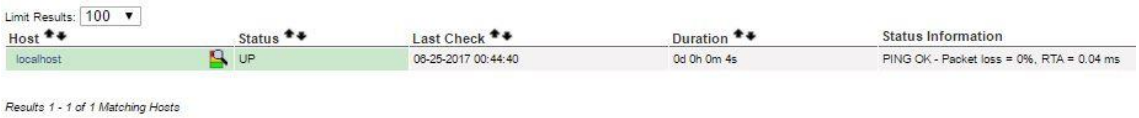

Figura 45: Monitorización correcta en Nagios

Una vez que tenemos instalada completamente la herramienta de Nagios se busca monitorizar los elementos de red que tenemos en la nube, Nagios distingue entre 5 clases o formas distintas de monitorización, dependiendo de si se trata de una maquina Linux, Windows, si son elementos de red como routers o switches, si se trata de monitorizar servicios públicamente disponibles (HTTP, FTP, SSH) o si se tratan de impresoras en red.

Centrándolo en la topología de red que se tenía anteriormente se monitorizará las instancias Linux y el Router R1 que enlaza la red pública con la red virtual privada.

6.2.1 Monitorización Router:

Con el objetivo de monitorizar todos los elementos de red, se debe tener en cuenta que puede ser necesario controlar los routers que forman parte de la red en la nube, con ello se pueden obtener datos como la pérdida de paquetes, información de estado SNMP, la velocidad del ancho de banda y del tráfico, etc.

Los routers se pueden monitorizar fácilmente enviando un ping, si el *switch* soporta SNMP se podría comprobar por ejemplo el estado de los puertos a través del plugin "*check_snmp*" y determinar el ancho de banda a través de "*check_mrtgtraf*", el esquema de monitorización se observa en la Figura 46.

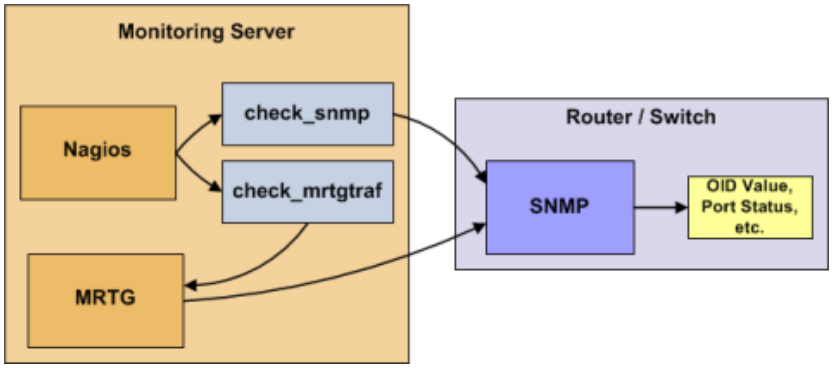

Figura 46: Monitorización router en Nagios

Para configurar el Router R1 y que pueda ser monitorizado por Nagios es necesario editar el archivo de configuración router que se encuentra en: `/usr/local/nagios/etc/nagios.cfg` tal como se muestra en la Figura 47, la línea que define la monitorización del router o switch.

```
# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg

# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg

# Definitions for monitoring a router/switch
cfg_file=/usr/local/nagios/etc/objects/switch.cfg

# Definitions for monitoring a network printer
#cfg_file=/usr/local/nagios/etc/objects/printer.cfg
```

Figura 47: Archivo de configuración de Nagios

Con esto se le añade las definiciones de equipos y servicios para los routers y switches. Para monitorizar en este caso el router R1 es necesario crear las definiciones de los objetos, esto se realiza en el archivo "*switch.cfg*" que se muestra parcialmente en la Figura 48 y se encuentra en la ruta: `/usr/local/nagios/etc/switch.cfg`

```
###############################################################################
###############################################################################
#
# HOST DEFINITIONS
#
###############################################################################
###############################################################################

# Define the switch that we'll be monitoring

define host{
        use             generic-switch          ; Inherit default values from a template
        host_name       R1              ; The name we're giving to this switch
        alias           Router 1        ; A longer name associated with the switch
        address         172.24.4.10             ; IP address of the switch
        hostgroups      switches                ; Host groups this switch is associated with
        }
```

Figura 48: Archivo de configuración del Router R1

En este archivo de configuración es necesario definir un nuevo host indicando principalmente su nombre (R1) y su dirección IP (172.24.4.10) en este caso.

En el mismo ficho también hay que definir los servicios que se pretenden monitorizar. Por ejemplo, si se quiere monitorizar la perdida de paquetes cada 5 minutos (indicando el intervalo en el campo *check_interval*), este servicio que consiste en un ping al router se puede clasificar como "Critico" (si la media de tiempo entre el host de Nagios y el router es mayor a 600 milisegundos o si la perdida de paquetes es superior al 60%), como "Alerta" (si dicho tiempo es mayor de 200 ms o la perdida de paquetes es mayor al 20%) o como estado "OK" (si ese tiempo es menor a 200 ms o la perdida de paquetes es inferior al 20%). Esto es totalmente configurable en el campo *check_command*. Este servicio se muestra en la Figura 49.

```
# Create a service to PING to switch

define service{
        use                     generic-service ; Inherit values from a template
        host_name               R1      ; The name of the host the service is associated with
        service_description     PING            ; The service description
        check_command           check_ping!200.0,20%!600.0,60%  ; The command used to monitor the s$
        check_interval  5               ; Check the service every 5 minutes under normal conditions
        retry_interval  1               ; Re-check the service every minute until its final/hard st$
        }
```

Figura 49: Servicios de monitorización para el router

Si el switch o router soportara SNMP se puede monitorizar definiéndolo de la siguiente forma como en la Figura 50.

```
# Monitor Port 1 status via SNMP

define service{
        use                     generic-service ; Inherit values from a template
        host_name               R1
        service_description     Port 1 Link Status
        check_command           check_snmp!-C public -o ifOperStatus.1 -r 1 -m RFC1213-MIB
        }
```

Figura 50: Monitorización SNMP Router

Otro servicio capaz de monitorizar es el ancho de banda o la tasa de tráfico. Usando la aplicación MRTG es posible obtener alertas si la tasa de tráfico supera un umbral específico. Este servicio se define de la siguiente manera como se observa en la Figura 51.

```
# Monitor bandwidth via MRTG logs

define service{
        use                     generic-service ; Inherit values from a template
        host_name               R1
        service_description     Port 1 Bandwidth Usage
        check_command           check_local_mrtgtraf!/var/lib/mrtg/192.168.1.253_1.log!AVG!1000000,$
```

Figura 51: Servicio de monitorización de ancho de banda en el Router

Finalmente, tras realizar todas estas modificaciones conviene verificar la configuración de Nagios para descartar posibles fallos de configuración, esto se realiza a través de la instrucción:

```
# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

El resultado se observa en la Figura 52 que indica que no se han encontrado errores de configuración.

```
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2017-05-09
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
        Checked 12 services.
        Checked 2 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

Figura 52: Comprobación de la configuración de Nagios

Ahora si, visto que no se ha encontrado ningún error se procede a reiniciar el servicio de Nagios que se puede realizar desde la propia interfaz web o por comandos: `/etc/rc.d/init.d/nagios reload`

6.2.2 Monitorización máquinas Linux:

A través de Nagios se puede monitorizar los servicios privados de las máquinas virtuales como la carga de la CPU, el uso de la memoria y del disco, los usuarios logueados o lo procesos que se están ejecutando, por ejemplo.

Hay muchas maneras de monitorizarlo, una es usando un par de claves SSH y el plugin *check_by_ssh* para ejecutar plugins en las maquinas remotas, pero este modo no es eficiente si se contemplan cientos de máquinas ya que requiere una gran carga por parte de la CPU en la máquina de monitorización.

Otro método de monitorizar es a través del uso de NRPEs, lo que se probará a continuación, un NRPE (*Nagios Remote Plugin Executor*) es un *addon* (un conjunto de funcionalidades agregadas al ERP) que permite ejecutar plugins en las maquinas remotas Linux. En la Figura 53 se observa el esquema de funcionamiento de este método.

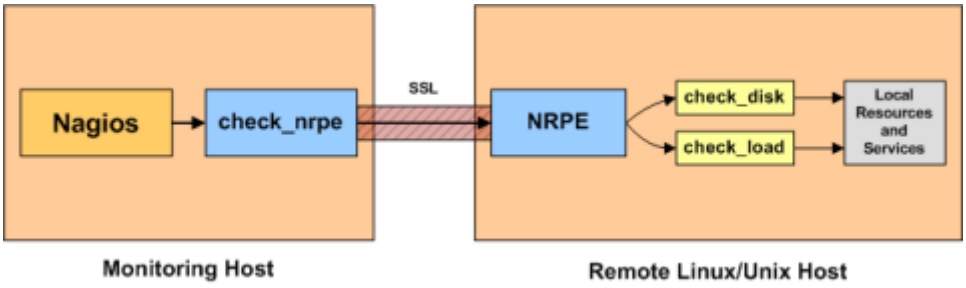

Figura 53: Esquema Monitorización Host Linux

Para acceder a la maquina remota de Linux o lo que es lo mismo a la instancia creada en Openstack con la imagen de Debian, es necesario acceder mediante SSH con el comando que aparece en la Figura 54, indicando la ruta en la que se encuentran las claves generadas anteriormente, el nombre de usuario "*debian*" y la dirección IP de la instancia.

```
stack@ubuntu:/$ ssh -i /home/stack/ssh-key debian@100.100.100.10

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun 26 17:30:34 2017 from 172.24.4.1
debian@clientdebian1:~$ su stack
```

Figura 54: Acceder mediante SSH a la instancia ClientDebian1

Una vez que se ha accedido a la instancia mediante SSH y con el usuario por defecto de la imagen "*debian*" se crea un nuevo usuario "*stack*" con privilegios de administrador para poder acceder desde la herramienta de Putty y gestionar la instancia más cómodamente, introduciendo tan solo la dirección IP de la instancia a la que se desea acceder, como se observa en la Figura 55.

Así se consigue en una nueva ventana de Putty acceder por comandos a la instancia, como se ve en la Figura 56.

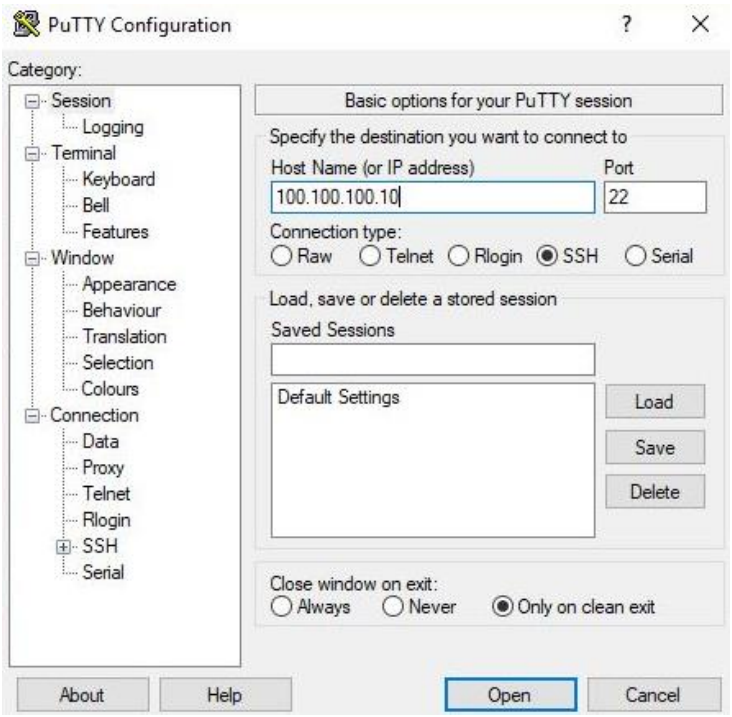

Figura 55: Acceso a la instancia mediante Putty

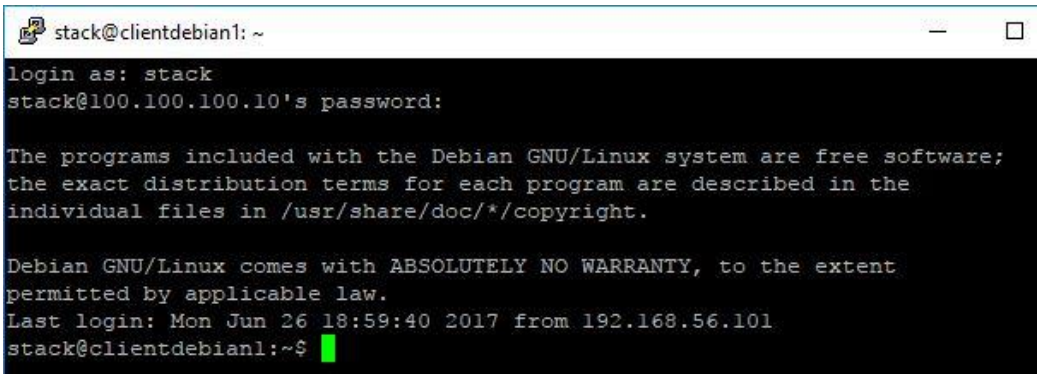

Figura 56: Acceso por consola a la instancia a través de Putty

Una vez que se accede a la instancia o maquina remota, es necesario instalar los plugins de Nagios (en este caso se ha instalado la última versión disponible,

versión 2.2.1), para descargarlos se emplea el comando wget y a continuación el enlace de descarga, el resultado se observa en la Figura 57.

```
wget https://nagios-plugins.org/download/nagios-plugins-2.2.1.tar.gz
```

Figura 57: Descarga de los plugins de Nagios

Ya descargados, se extraen los plugins y se compilan a través de estos comandos empleados en la Figura 58.

Figura 58: Compilación de los plugins

Una vez compilados se configuran y se instalan mediante los siguientes comandos:

```
# cd nagios-plugins-2.2.1
# ./configure
# make
# make install
```

Es necesario fijar los permisos del directorio de los plugins mediante el empleo de los comandos:

```
# chown nagios.nagios /usr/local/nagios
# chown -R nagios.nagios /usr/local/nagios/libexec
```

También, es necesario descargar e instalar *xinetd* puesto que será el servicio en el que se ejecuten los conexiones del NRPE, se emplea:

```
# sudo apt-get install xinetd
```

Aparte de los plugins, en la máquina remota a monitorizar se descarga el demonio NRPE (en este caso la última versión 3.2) a través del comando *wget*:

```
wget https://github.com/NagiosEnterprises/nrpe/archive/nrpe-3.2.tar.gz
```

Se procede a extraerlo, compilarlo e instalarlo de la misma manera que con los plugins.

```
# tar xzf nrpe-3.2.tar.gz
# cd nrpe-nrpe-3.2
# ./configure
# make all
# make install
# make install-config
# make install-inetd
```

Se puede observar en la Figura 59 el resultado tras la configuración del plugin de NRPE 3.2.

```
*** Configuration summary for nrpe 3.2.0 2017-06-27 ***:

 General Options:
 -------------------------
 NRPE port:    5666
 NRPE user:    nagios
 NRPE group:   nagios
 Nagios user:  nagios
 Nagios group: nagios


Review the options above for accuracy.  If they look okay,
type 'make all' to compile the NRPE daemon and client
or type 'make' to get a list of make options.
```

Figura 59: Correcta configuración de NRPE

Para la comunicación entre la maquina remota y el servidor Nagios es necesario habilitar el puerto 5666 de TCP, editando el fichero */etc/services* como se observa en la Figura 60.

```
  GNU nano 2.2.6              File: etc/services                      Modified

mdns            5353/tcp                        # Multicast DNS
mdns            5353/udp
postgresql      5432/tcp        postgres        # PostgreSQL Database
postgresql      5432/udp        postgres
freeciv         5556/tcp        rptp            # Freeciv gameplay
freeciv         5556/udp
nrpe            5666/tcp
```

Figura 60: Puertos habilitados en el cliente

Se puede comprobar que está abierto y en modo de escucha este puerto bien realizando desde el propio cliente el comando *netstat* como se observa en la Figura 61 o bien desde el propio servidor de Nagios mediante *nmap.*

```
stack@clientdebian1:/$ netstat -at | egrep "nrpe|5666"
tcp        0      0 *:nrpe                  *:*                     LISTEN
tcp6       0      0 [::]:nrpe               [::]:*                  LISTEN
stack@clientdebian1:/$
```

Figura 61: Puerto 5666 en escucha

Para comprobar el correcto funcionamiento del plugin *check_nrpe* se ejecuta como en la Figura 62 y devuelve la versión instalada de NRPE (3.2.0)

```
stack@clientdebian1:/$ /usr/local/nagios/libexec/check_nrpe -H localhost
NRPE v3.2.0
```

Figura 62: Comprobación check_nrpe

Una vez que se comprueba que está instalado correctamente solo quedan realizar dos modificaciones en los archivos de configuración.

En el archivo "*/etc/xinetd.d/nrpe*" se debe cambiar las direcciones de IP de host permitidos incluyendo tanto la IP de la MV en la que está instalado Nagios (192.168.56.102) como la IP de la interfaz de red por la que la MV se dirige a la red en la nube (172.24.4.1). Este fichero se observa completamente en la Figura 63, en el que se ha modificado los parámetros de "*disable*" a "*no*" y el parámetro "*only_from*" con las IPs necesarias.

```
stack@clientdebian1: ~
  GNU nano 2.2.6           File: /etc/xinetd.d/nrpe

# default: off
# description: NRPE (Nagios Remote Plugin Executor)
service nrpe
{
    disable         = no
    socket_type     = stream
    port            = 5666
    wait            = no
    user            = nagios
    group           = nagios
    server          = /usr/local/nagios/bin/nrpe
    server_args     = -c /usr/local/nagios/etc/nrpe.cfg --inetd
    only_from       =172.24.4.1 127.0.0.1 192.168.56.102
   log_on_success  =
}
```

Figura 63: Configuración IPs servicio xinetd

En el archivo de configuración del NRPE *(/usr/local/nagios/etc/nrpe.cfg)* también se edita para que permita los hosts desde lo que se accede, es decir las mismas direcciones IP mencionadas anteriormente, este extracto se puede ver en la Figura 64, el archivo completo se encuentra en el Anexo 2.

```
# ALLOWED HOST ADDRESSES
# This is an optional comma-delimited list of IP address or hostnames
# that are allowed to talk to the NRPE daemon. Network addresses with a bit mask
# (i.e. 192.168.1.0/24) are also supported. Hostname wildcards are not currently
# supported.
#
# Note: The daemon only does rudimentary checking of the client's IP
# address.  I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

#allowed_hosts=127.0.0.1,::1
allowed_host=172.24.4.1
allowed_host=192.168.56.102
```

Figura 64: Archivo configuración NRPE

También en este mismo archivo se editan los comandos que se emplearan, como son los siguientes y que se observan en la Figura 65.

Check_users: para contabilizar los usuarios logueados.

Check_load: para medir la carga de la CPU.

Check_hda1: Para medir la utilización del espacio del disco.

Check_zombie_prcs: Para contabilizar los procesos que se encuentran en modo de reposo.

Check_total_prcs: Para contabilizar los procesos totales que se están ejecutando.

Figura 65: Comandos en NRPE

Una vez instalado el agente NRPE en la instancia *ClientDebian1* que está dentro de la red cloud de Openstack, hay que instalar el plugin *check_nrpe* en la propia MV que hace de servidor en la que se ha instalado Nagios. Para ello se descarga, es extrae el plugin, se compila y se instala con los siguientes comandos:

```
# wget https://github.com/NagiosEnterprises/nrpe/archive/nrpe-
3.2.tar.gz

# tar xzf nrpe-3.2.tar.gz # cd nrpe-nrpe-3.2

# ./configure

# make check_nrpe

# make install-plugin
```

Una vez debidamente configurado e instalado se puede comprobar que desde la MV de devstack y Nagios se puede conectar a través del *check_nrpe* a la instancia, en este caso ClientDebian1 con la IP 100.100.100.10, ejecutando *check_nrpe* a esa dirección IP, devuelve la versión instalada de NRPE como se ve en la Figura 66.

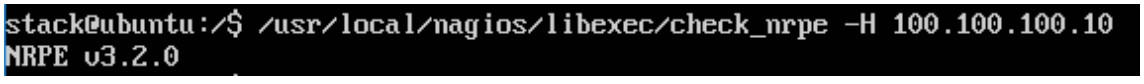

Figura 66: Comprobación de conexión

Ahora solo queda añadir los comandos que se van a utilizar en la monitorización editando el archivo */usr/local/nagios/etc/objects/commands.cfg.* Aquí se definen los comandos como se indica en la Figura 67.

```
# 'check_local_disk' command definition
define command{
        command_name    check_local_disk
        command_line    $USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$
        }


# 'check_local_load' command definition
define command{
        command_name    check_local_load
        command_line    $USER1$/check_load -w $ARG1$ -c $ARG2$
        }


# 'check_local_procs' command definition
define command{
        command_name    check_local_procs
        command_line    $USER1$/check_procs -w $ARG1$ -c $ARG2$ -s $ARG3$
        }


# 'check_local_users' command definition
define command{
        command_name    check_local_users
        command_line    $USER1$/check_users -w $ARG1$ -c $ARG2$
        }
```

Figura 67: Comandos en NRPE

Para definir los servicios se crea un nuevo archivo de configuración */usr/local/nagios/etc/objects/debian.cfg* en el cual se debe detallar en el nombre de la instancia a monitorizar, así como su dirección IP y un alias como en la Figura 68.

```
define host{
        use     generic-switch;
        host_name       clientdebian1;
        alias           Client Debian 1;
        address         100.100.100.10;
        }
```

Figura 68: Definición del host o instancia

En este mismo archivo se definen los servicios, como se puede ver en la Figura 69, que se quieren monitorizar de esta instancia en concreto, basándose en la definición de los comandos vista anteriormente.

Figura 69: Definición de los servicios a monitorizar

Y es en el archivo */usr/local/nagios/etc/nagios.cfg* donde se definen los archivos de configuración que debe utilizar como se observa en la Figura 70.

Finalmente, tras todos los cambios se comprueba que no se encuentren ningún fallo en los archivos de configuración y se reinicia el servicio *xinetd* en la instancia mediante la instrucción: `sudo service xinetd restart`

Y se reinicia también el servicio de Nagios en la MV en la que está instalado:

```
# Sudo service nagios restart
```

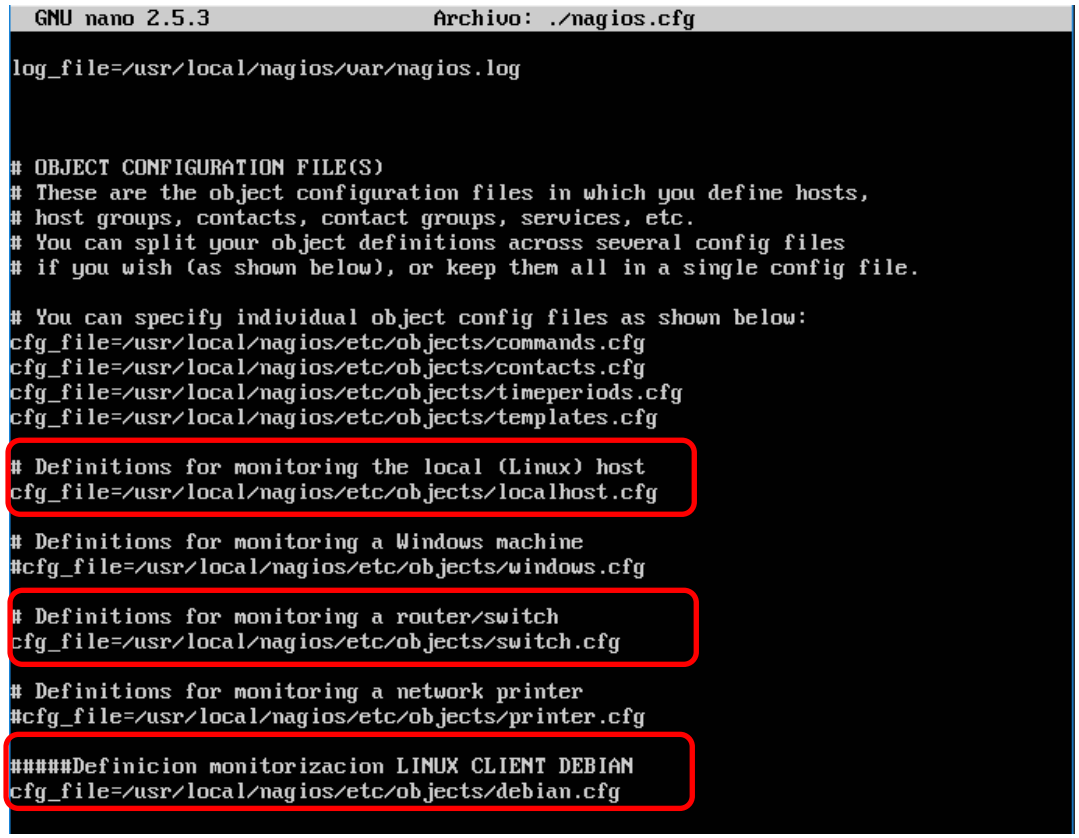

Figura 70: Archivo nagios.cfg

También sería posible monitorizar algunos de estos servicios mediante SNMP, porque aunque principalmente lo que se busca a través de Nagios no son valores exactos, si no que sirva de un sistema de alertas en el que indique si un servicio específico se encuentra por encima de umbral, para poder atender esa incidencia, pero en el caso de que por ejemplo se busquen los valores exactos de la memoria RAM que esté utilizando, se necesitaría emplear SNMP, para ello sería necesario instalar los paquetes SNMP en el host, configurar las direcciones IP del servidor Nagios en su archivo de configuración (*snmpd.conf*) y arrancar el servicio (*service snmpd start*).

Primero de todo como se vio anteriormente habría que configurar el archivo de comandos (*commands.cfg*) como en la Figura 67, pero con las siguientes instrucciones de ejemplo de tamaño de la memoria RAM:

```
define command{
        command_name    snmp_RamSize
        command_line    $USER1$/check_snmp -H $HOSTADDRESS$ $ARG1$
}
define command{
```

```
        command_name    snmp_RamFree
        command_line    $USER1$/check_snmp -H $HOSTADDRESS$ $ARG1$
}
define command{
        command_name    snmp_RamUsed
        command_line    $USER1$/check_snmp -H $HOSTADDRESS$ $ARG1$
}
```

También, al igual que se ha visto en la Figura 69 se añadirían los servicios a monitorizar en el archivo de configuración del host específico, añadiendo las siguientes instrucciones:

```
define service{
        use                     generic-service
        host_name               node1.example.com
        service_description     RAM Size
        check_command           snmp_RamSize!-C public
}
define service{
        use                     generic-service
        host_name               node1.example.com
        service_description     RAM Free
        check_command           snmp_RamFree!-C public
}
define service{
        use                     generic-service
        host_name               node1.example.com
        service_description     RAM Used
        check_command           snmp_RamUsed!-C public
}
```

Así se comprobaría a través de SNMP la utilización de la memoria RAM y se podría visualizar en la consola web de Nagios.

6.2.3 Monitorización máquinas Windows:

También sería posible monitorizar una maquina Windows, instalando en ella el programa "*NSClient++*" e instalando en el servidor de monitorización el plugin "*check_nt*". El esquema se observa en la Figura 71.

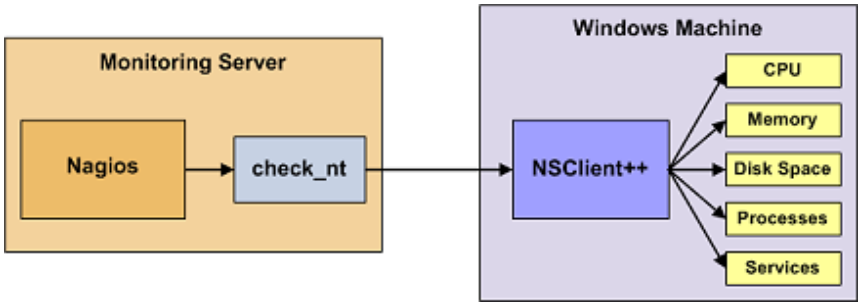

Figura 71: Monitorización maquinas Windows

Por lo que por ejemplo, en este caso, se podría monitorizar también la maquina Windows sobre la que se han montado la máquina virtual a través de VirtualBox, para su monitorización haría falta instalar en el propio Windows el agente descargándolo de la siguiente dirección: https://www.nsclient.org/ y en la MV de Nagios instalar el plugin necesario, en este caso sería "*check_nt*", tras esto solo haría falta configurarlo creando un nuevo archivo de configuración como el creado para la instancia de Debian en el que se incluya su dirección IP, y los servicios que se desea monitorizar, así como en el propio archivo de configuración de "*nagios.cfg*" hacer una referencia a este archivo, y también introducir la dirección IP del propio Windows en los archivos de configuración del nuevo plugin permitiendo poder acceder a ellos para conocer su información, como se hizo con NRPE.

6.3 Monitorización de la red en la nube desde la consola web de Nagios:

Una vez comprobado que se comunican correctamente Nagios y el agente instalado en el cliente se puede analizar el comportamiento de las instancias, es decir de las máquinas virtuales que haya en el entorno en la nube accediendo a la consola web previamente instalada de Nagios y en la que aparecerán las instancias que se estén monitorizando.

Desde la consola web se puede visualizar los dispositivos que se están monitorizando, desde la pantalla principal indica si están o no activos o si por el

contrario se han caído, esto lo resuelve a través de un ping, como se ve en la Figura 72 los dispositivos a monitorizar son: localhost (la propia MV donde está instalado tanto devstack, como Nagios), el Router R1 y la instancia ClientDebian1 creadas con anterioridad a través de Openstack.

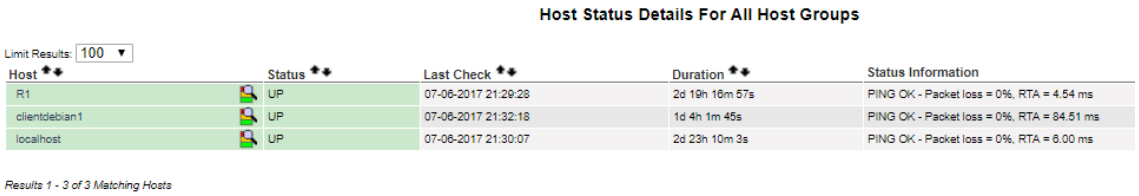

Figura 72: Dispositivos monitorizados

Si se va al detalle de los servicios que se están monitorizando, en la pestaña "*Services*" se observa la situación de cada servicio como en la Figura 73.

Dentro de los servicios de cada dispositivo se pueden observar los usuarios logueados en un determinado instante, así como sacar una gráfica temporal de los usuarios que acceden al mismo tiempo a la instancia, la carga de la CPU, la capacidad que está libre en el disco duro, lo que por ejemplo se observa es que en la instancia ClientDebian1 la carga de la CPU está casi al límite, ya que estas instancias creadas en Openstack suelen estarlo, hay que tener en cuenta que se creó con pocos recursos.

También se pueden ver el número de procesos que se están ejecutando en dicha máquina, en el caso de la instancia de Openstack se observa que marca que se están ejecutando 62 procesos concretamente.

Current Network Status
Last Updated: Thu Jul 6 22:53:41 CEST 2017
Updated every 90 seconds
Nagios® Core™ 4.3.2 - www.nagios.org
Logged in as *nagiosadmin*

View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts

Host Status Totals			
Up	Down	Unreachable	Pending
3	0	0	0

All Problems	All Types
0	3

Service Status Totals				
Ok	Warning	Unknown	Critical	Pending
11	1	2	2	0

All Problems	All Types
5	16

Service Status Details For All Hosts

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
R1	PING	OK	07-04-2017 13:56:06	4d 2h 52m 37s	1/3	PING OK - Packet loss = 0%, RTA = 0.13 ms
	Port 1 Bandwidth Usage	UNKNOWN	07-04-2017 13:56:07	4d 2h 51m 34s	3/3	check_mrtgtraf: Unable to open MRTG log file
	Port 1 Link Status	UNKNOWN	07-04-2017 13:57:15	3d 8h 36m 32s	3/3	External command error: MIB search path: $HOME/.snmp/mibs:/usr/share/snmp/mibs:/usr/share/snmp/mibs/iana:/usr/share/sn
	Uptime	CRITICAL	07-04-2017 13:58:14	4d 2h 49m 29s	3/3	CRITICAL - Plugin timed out while executing system call
clientdebian1	/dev/hda1 Free Space	CRITICAL	07-04-2017 13:54:46	2d 10h 42m 55s	3/3	DISK CRITICAL - /dev/hda1 is not accessible: No such file or directory
	CPU Load	WARNING	07-04-2017 13:50:16	2d 9h 3m 25s	3/3	WARNING - load average: 0.00, 0.09, 0.12
	Current Users	OK	07-04-2017 13:51:43	2d 9h 31m 58s	1/3	USERS OK - 1 users currently logged in
	Total Processes	OK	07-04-2017 13:56:07	2d 9h 37m 34s	1/3	PROCS OK: 62 processes
localhost	Current Load	OK	07-04-2017 13:58:21	2d 9h 0m 20s	1/4	OK - load average: 0.44, 1.00, 2.06
	Current Users	OK	07-04-2017 13:58:02	4d 6h 45m 47s	1/4	USERS OK - 1 users currently logged in
	HTTP	OK	07-04-2017 13:58:38	4d 6h 45m 10s	1/4	HTTP OK: HTTP/1.1 302 Found - 501 bytes in 0,001 second response time
	PING	OK	07-04-2017 13:59:11	4d 6h 44m 32s	1/4	PING OK - Packet loss = 0%, RTA = 0.06 ms
	Root Partition	OK	07-04-2017 13:54:53	4d 6h 43m 55s	1/4	DISK OK - free space: / 59567 MB (81,98% inode=96%):
	SSH	OK	07-04-2017 13:56:33	2d 13h 12m 12s	1/4	SSH OK - OpenSSH_7.2p2 Ubuntu-4ubuntu2.2 (protocol 2.0)
	Swap Usage	OK	07-04-2017 13:59:05	2d 11h 59m 36s	1/4	SWAP OK - 16% free (627 MB out of 3996 MB)
	Total Processes	OK	07-04-2017 13:56:50	4d 6h 42m 2s	1/4	PROCS OK: 81 processes with STATE = RSZDT

Figura 73: Servicios monitorizados

También es posible sacar reportes, en la pestaña de "*Reports*" y poder obtener uno totalmente configurable, respecto de que se quiere obtener el report, si de un determinado Host o de un servicio, el tiempo en el que se desea ver el reporte, etc. Un ejemplo de esto es la Figura 74 que se observan los dispositivos monitorizados y el porcentaje de tiempo que durante 24 horas se han encontrado activos, caídos, inalcanzables o sin determinar, siendo en este caso el que más bajo porcentaje tiene corresponde a la instancia "ClientDebian1"

Host State Breakdowns:

Host	% Time Up	% Time Down	% Time Unreachable	% Time Undetermined
R1	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
clientdebian1	88.803% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	11.197%
localhost	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
Average	96.268% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	3.732%

Figura 74: Reporte del estado de los dispositivos

Así como también es posible sacar graficas de las tendencias que han sufrido ciertos hosts, como en la Figura 75 que se observa más detalladamente el estado en el que se ha encontrado "Up", en este caso el 88.59% del tiempo.

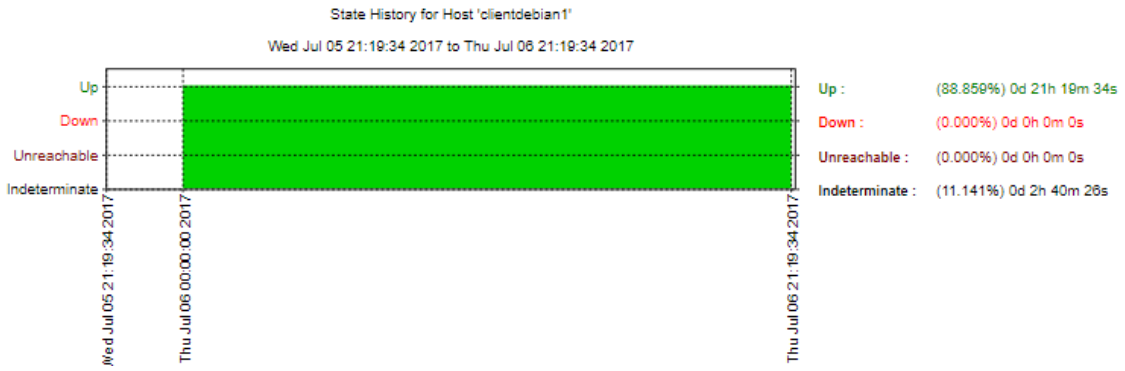

Figura 75: Grafico host clientdebian1

Incluso se pueden obtener tablas del rendimiento como la que se ofrece en la Figura 76 que ofrece el servicio de monitorización de Nagios, como el tiempo de ejecución del servicio *check_nrpe*, así como su latencia.

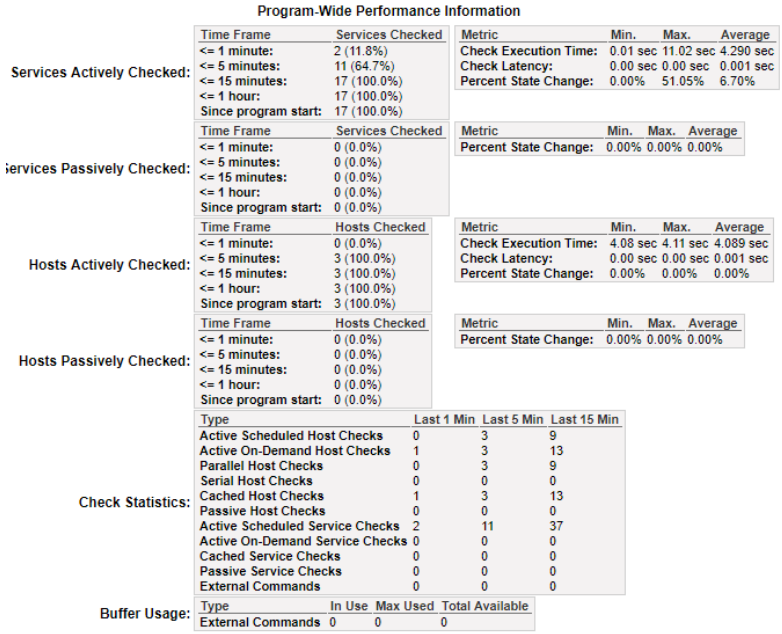

Figura 76: Rendimiento de los dispositivos

Capítulo 7

Conclusiones y líneas futuras

El modelo cloud como se ha visto durante este trabajo conlleva a los procesos de industrialización y automatización del modelo empresarial e incluso a nivel personal en cuanto al acceso, gestión y contratación de servicios y recursos, los cuales se ubican en la nube.

En la actualidad la oferta de servicios en la nube, así como su demanda, se encuentra en pleno crecimiento y se prevé que siga creciendo a un mayor ritmo en los próximos años. Como se ha visto el modelo cloud se considera como una plataforma en continuo desarrollo, en la que se producen continuamente cambios y para la cual la movilidad, y la flexibilidad son dos requerimientos indispensables. Estas características hacen que la gestión sea de vital importancia, y esta gestión debe ser específica para cada servicio puesto que no todos los servicios pueden gestionarse de la misma manera.

Una de las líneas de trabajo a mejorar en el entorno cloud es la interoperabilidad entre los servicios ofrecidos y a su vez entre los diferentes proveedores puesto que estos puntos hoy en día se encuentran poco avanzados y es bastante difícil encontrar herramientas que trabajen con distintos entornos en la nube al mismo tiempo.

Hay que tener en cuenta que el modelo en la nube no puede garantizar un rendimiento de la red al 100%, ya que en realidad existen diferentes problemas que continuarán afectando, como problemas en la WAN, pérdida de paquetes, problemas de la nube del proveedor... pero pese a ello la gestión de la Infraestructura, el Software y las Plataformas en un entorno cloud es mucho

más sencilla y eficiente que con los entornos convencionales, lo que permite también una reducción de costes.

Esto es debido a que gracias al modelo cloud se hace innecesario la gestión de los elementos que anteriormente consumían un gran número de recursos, por un lado, desaparece la gestión de los elementos de la infraestructura, y por el otro también desaparece la gestión del coste del software.

De cara a futuro de esta gestión y a medida del crecimiento de la nube, así como su aumento en complejidad será necesario implementar un modelo de gestión que se adapte las nuevas necesidades. Una de estas propuestas de futuro es el Cloud-to-Cloud (C2C), el cual plantea la idea de enfocar la gestión desde la propia nube, es decir, disponer de una nube de gestión para administrar la nube, y que se explica a continuación.

En definitiva, el entorno cloud requiere una planificación de la gestión especialmente a medida del crecimiento de la nube. En el futuro se invertirá cada vez más en su gestión y seguridad, mejorando los modelos existentes y en el momento en que se consideren obsoletos, planteando nuevas alternativas que siempre se adapten a la flexibilidad de una tecnología tan cambiante como la nube.

En lo que respecta al modelo de prueba utilizado en este trabajo mediante las herramientas más usadas como son Openstack y Nagios se puede decir que las dos son herramientas bastante útiles, que ofrecen bastante soporte en cuanto son herramientas *open source*, pero cabe mencionar que adaptarse a ellas y a su utilización requiere un gran aprendizaje.

Además si es cierto que la creación de diversas instancias y una mayor topología de red requiere de mayores recursos donde se implementa, es por ello que aunque en este trabajo se ha creado una instancia, dos redes y un router, disponiendo de los recursos necesarios la herramienta de Openstack es bastante flexible a la hora de crear nuevas redes, interconectarlas como se desee mediante routers, así como crear instancias con imágenes tanto de distintas distribuciones Linux como de Windows, incluso se pueden crear snapshots de estas instancias o imágenes a partir de ellas para una implementación más rápida si se trata del mismo tipo del cual ya existe al menos una.

Una de las líneas futuras de investigación respecto a este trabajo, por ejemplo, puede ser la integración de "*Ceilometer*" con Nagios. Como se vio durante la

introducción a Openstack, *Ceilometer* es utilizado para recopilar estadísticas de uso del clúster con el objetivo de su monitorización y medición del rendimiento, aunque *Ceilometer* es una herramienta de medición bastante potente y flexible para OpenStack, carece de capacidades para visualizar los datos recopilados, es por lo que se debe complementar con herramientas como Nagios.

Si es cierto que, aunque no hay plugins de Nagios para integrar *Ceilometer,* algunas empresas como *Enovance* [39] han desarrollado plugins para comprobar que los componentes de OpenStack se encuentren activos, visualizándolo desde el entorno de monitorización de Nagios y por lo tanto permiten a Nagios monitorear no sólo los componentes de OpenStack, sino también todas las máquinas virtuales y otros servicios.

En lo que respecta más concretamente a la monitorización de Nagios se podría incluir SNMP para ver los valores específicos de un servicio con el objetivo de ir al detalle, como se ha indicado en el capítulo 6 como se podría realizar, o también es posible monitorizar servicios públicos como HTTP, FTP, SMTP, IMAP, etc. En este caso, por cada servicio que se quiera gestionar se requiere definir un servicio en Nagios, como los de la Figura 69, que esté asociado con la definición del host especifico, como en la Figura 68, además de definir el comando en concreto por ejemplo para HTTP se definiría primero el comando para usar el complemento "*check_http*" y luego el servicio. Esto resulta interesante si lo que se quiere es aparte de gestionar cada instancia o cada host, gestionar distintos protocolos que se estén empleando en la red.

También, de cara a futuros desarrollos, se pueden utilizar otras herramientas que complementen esta monitorización con Nagios y que ofrezcan nuevas mejoras a la gestión, a través por ejemplo, de herramientas como "*Weathermap*" [40] o "*Nagvis"* [41] que son capaces de crear entornos gráficos más cómodos o intuitivos visiblemente hablando, la primera es una herramienta de visualización de red de código abierto, para tomar datos que ya tiene y mostrarle una visión general de su actividad de red en forma de mapa, y la segunda es una aplicación que permite crear de forma muy sencilla páginas web en el servidor Nagios, con imágenes de fondo e iconos de servicios o hosts de Nagios.

Bibliografía

[1] Sanz, Juan Ignacio, «Ibérmatica,» Abril 2011. [En línea]. Available: http://www.ibermatica.com/sala-de-prensa/opinion/gestion-de-servicios-de-cloud-computing. [Último acceso: Abril 2017].

[2] CloudTweaks, «Cloud Computing and its effects on the network management,» Enero 2013. [En línea]. Available: https://cloudtweaks.com/2013/01/cloud-computing-and-its-effects-on-network-management/. [Último acceso: Abril 2017].

[3] «Cloud Computing Foro,» [En línea]. Available: www.cloudforum.org.

[4] «Consorcio Open Cloud,» [En línea]. Available: www.opencloudconsortium.org.

[5] Zhou, Liang-Jie Zhang and Qun, «CCOA: Cloud Computing Open Architecture,» New York, USA, 2009.

[6] «Amazon Web Services,» [En línea]. Available: https://aws.amazon.com/es/.

[7] «Google Cloud Platform,» [En línea]. Available: https://cloud.google.com/?hl=es.

[8] «Microsoft Azure,» [En línea]. Available: https://azure.microsoft.com/.

[9] «VMWare,» [En línea]. Available: https://www.vmware.com/.

[10] Wilkins, Sean, «Pros And Cons Of Cloud-Based Network Monitoring Tools,» Febrero 2014. [En línea]. Available: http://www.tomsitpro.com/articles/cloud-based-network-monitoring,2-693.html. [Último acceso: Abril 2017].

[11] Alex Amies, Harm Sluiman, Qiang Guo Tong, Guo Ning Liu, Developing and hosting Applications in the cloud, USA: IBM, 2012.

[12] Peng Sun, Jennifer Rexford, «Integrating network management for cloud computing services,» Princeton University, 2015.

[13] D. Clark, S. Bauer, K. Claffy, A. Dhamdhere, B. Huffaker, W. Lehr., «Measurement and Analysis of Internet Interconnection and Congestion in Telecommunications Policy Research Conference (TPRC),» 2014.

[14] «Amazon Web Services Elastic Load Balancing.,» [En línea]. Available: http://aws.amazon.com/elasticloadbalancing/.

[15] Chompu Nuangjamnong, Stanislaw P.Maj, David Veal, «The OSI Network Management Model,» Bangkok, Thailand. IEEE, 2008.

[16] John Panneerselvam, Lu Liu, Richard Hill, Yongzhao Zhan, Weining Liu, «An Investigation of the Effect of Cloud Computing on Network Management,» 2012.

[17] M. Head, A. Sailer, H. Shaikh, and M. Viswanathan, «Taking IT Management Services to a Cloud,» 2009.

[18] Rodriguez Ismael P., Pettoruti José E., Chichizola Franco, De Giusti Armando E., «Despliegue de un Cloud Privado,» XVII Congreso Argentino de Ciencias de la computación, Facultad de Informática, Universidad Nacional de La Plata, Argentina, 2011.

[19] «Iana,» [En línea]. Available: https://www.iana.org/.

[20] «Kaseya,» [En línea]. Available: http://www.kaseya.com/.

[21] «LogicMonitor,» [En línea]. Available: https://www.logicmonitor.com/.

[22] «Monitis,» [En línea]. Available: http://www.monitis.com/es.

[23] [En línea]. Available: http://www.traverse-monitoring.com/solutions/cloud-monitoring-and-management.

[24] Reese, G., «Cloud Application Architectures,» 2009.

[25] T. Choi, N. Kodirov, T. Lee, D. Kim, and J. Lee, «Autonomic Management Framework for Cloud-based Virtual Networks,» 2011.

[26] Sabahi, F., «Cloud Computing Security Threats and Responses,» 2011.

[27] Gorod, R. Gove, B. Sauser, and J. Boardman, «System of Systems Management: A Network Management Approach,» 2007.

[28] «Chef,» [En línea]. Available: https://www.chef.io/solutions/cloud-management/.

[29] «Puppet,» [En línea]. Available: https://puppet.com/.

[30] «Herramientas de monitorizacion propias de la nube,» [En línea]. Available: https://blog.pandorafms.org/es/que-debes-saber-sobre-monitorizacion-en-la-nube/.

[31] «Nagios,» [En línea]. Available: http://www.nagios-es.org/.

[32] «Cacti,» [En línea]. Available: http://www.cacti.net/.

[33] «Zabbix,» [En línea]. Available: http://www.zabbix.com/product.

[34] «Openstack,» [En línea]. Available: https://www.openstack.org/.

[35] «Funcionamiento de Openstack,» [En línea]. Available: http://vmartinezdelacruz.com/en-pocas-palabras-como-funciona-openstack/.

[36] «Devstack,» [En línea]. Available: https://docs.openstack.org/developer/devstack/.

[37] Garcia, Alberto Eloy, «Lab 1: Understanding a Cloud Based Environment,» DECAMP: Open Distributed European Virtual Campus on ICT Security, Santander, 2017.

[38] «Nagios Core,» [En línea]. Available: https://support.nagios.com/kb/article/nagios-core-installing-nagios-core-from-source.html#Ubuntu.

[39] Blog de la ZHAW Zurich University of Applied Sciences, «ICCLAB SPLAB,» Abril 2014. [En línea].

[40] «Weathermap,» [En línea]. Available: https://network-weathermap.com/.

[41] «NagVis,» [En línea]. Available: http://www.nagvis.org/.

Anexo 1: Configuración devstack

El archivo de configuración de devstack (*local.conf*) es el siguiente:

```
# Sample ``local.conf`` for user-configurable variables in
``stack.sh``

# NOTE: Copy this file to the root DevStack directory for it to work
properly. ``local.conf`` is a user-maintained settings file that is
sourced from ``stackrc``. This gives it the ability to override any
variables set in ``stackrc``. Also, most of the settings in
``stack.sh`` are written to only be set if no value has already been
set; this lets ``local.conf`` effectively override the default values.

# This is a collection of some of the settings we have found to be
useful in our DevStack development environments. Additional settings
are described in
http://docs.openstack.org/developer/devstack/configuration.html#local-
conf

# These should be considered as samples and are unsupported DevStack
code.

# The ``localrc`` section replaces the old ``localrc`` configuration
file.

# Note that if ``localrc`` is present it will be used in favor of this
section.

[[local|localrc]]

# Minimal Contents

# ----------------


# While ``stack.sh`` is happy to run without ``localrc``, devlife is
better when there are a few minimal variables set:


# If the ``*_PASSWORD`` variables are not set here you will be
prompted to enter values for them by ``stack.sh``and they will be
added to ``local.conf``.

ADMIN_PASSWORD=stack

DATABASE_PASSWORD=stack

RABBIT_PASSWORD=stack

SERVICE_PASSWORD=stack


# ``HOST_IP`` and ``HOST_IPV6`` should be set manually for best
results if the NIC configuration of the host is unusual, i.e. ``eth1``
has the default route but ``eth0`` is the public interface.  They are
auto-detected in ``stack.sh`` but often is indeterminate on later runs
```

```
due to the IP moving from an Ethernet interface to a bridge on the
host. Setting it here also makes it available for ``openrc`` to
include when setting ``OS_AUTH_URL``.

# Neither is set by default.

HOST_IP=192.168.56.102

#HOST_IPV6=2001:db8::7


# Logging

# -------


# By default ``stack.sh`` output only goes to the terminal where it
runs.  It can be configured to additionally log to a file by setting
``LOGFILE`` to the full path of the destination log file.  A timestamp
will be appended to the given name.

LOGFILE=$DEST/logs/stack.sh.log


# Old log files are automatically removed after 7 days to keep things
neat.  Change the number of days by setting ``LOGDAYS``.

LOGDAYS=2


# Nova logs will be colorized if ``SYSLOG`` is not set; turn this off
by setting

# ``LOG_COLOR`` false.

#LOG_COLOR=False


# Using milestone-proposed branches

# --------------------------------


# Uncomment these to grab the milestone-proposed branches from the

# repos:

#CINDER_BRANCH=milestone-proposed

#GLANCE_BRANCH=milestone-proposed

#HORIZON_BRANCH=milestone-proposed

#KEYSTONE_BRANCH=milestone-proposed

#KEYSTONECLIENT_BRANCH=milestone-proposed

#NOVA_BRANCH=milestone-proposed
```

```
#NOVACLIENT_BRANCH=milestone-proposed

#NEUTRON_BRANCH=milestone-proposed

#SWIFT_BRANCH=milestone-proposed


# Using git versions of clients

# ------------------------------

# By default clients are installed from pip.  See LIBS_FROM_GIT in

# stackrc for details on getting clients from specific branches or

# revisions.  e.g.

# LIBS_FROM_GIT="python-ironicclient"

# IRONICCLIENT_BRANCH=refs/changes/44/2.../1


# Swift

# -----


# Swift is now used as the back-end for the S3-like object store.
Setting the hash value is required and you will be prompted for it if
Swift is enabled so just set it to something already:

SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5


# For development purposes the default of 3 replicas is usually not
required. Set this to 1 to save some resources:

SWIFT_REPLICAS=1


# The data for Swift is stored by default in (``$DEST/data/swift``),
or (``$DATA_DIR/swift``) if ``DATA_DIR`` has been set, and can be
moved by setting ``SWIFT_DATA_DIR``. The directory will be created if
it does not exist.

SWIFT_DATA_DIR=$DEST/data
```

Anexo 2: Configuración nrpe nagios

```
#############################################################################
########

#  Sample NRPE Config File

#  Notes:

#  This is a sample configuration file for the NRPE daemon.  It needs
to be located on the remote host that is running the NRPE daemon, not
the host from which the check_nrpe client is being executed

#############################################################################
#######

# LOG FACILITY

# The syslog facility that should be used for logging purposes.

log_facility=daemon

# LOG FILE

# If a log file is specified in this option, nrpe will write to that
file instead of using syslog.

#log_file=/usr/local/nagios/var/nrpe.log


# DEBUGGING OPTION

# This option determines whether or not debugging messages are logged
to the syslog facility.

# Values: 0=debugging off, 1=debugging on

debug=0


# PID FILE

# The name of the file in which the NRPE daemon should write it's
process ID number.  The file is only written if the NRPE daemon is
started by the root user and is running in standalone mode.

pid_file=/usr/local/nagios/var/nrpe.pid


# PORT NUMBER

# Port number we should wait for connections on.

# NOTE: This must be a non-privileged port (i.e. > 1024).

# NOTE: This option is ignored if NRPE is running under either inetd
or xinetd
```

```
server_port=5666


# SERVER ADDRESS

# Address that nrpe should bind to in case there are more than one
interface and you do not want nrpe to bind on all interfaces.

# NOTE: This option is ignored if NRPE is running under either inetd
or xinetd

#server_address=127.0.0.1


# LISTEN QUEUE SIZE

# Listen queue size (backlog) for serving incoming connections.

# You may want to inc

# LISTEN QUEUE SIZE

# Listen queue size (backlog) for serving incoming connections.

# You may want to increase this value under high load.

#listen_queue_size=5


# NRPE USER

# This determines the effective user that the NRPE daemon should run
as.

# You can either supply a username or a UID.

# NOTE: This option is ignored if NRPE is running under either inetd
or xinetd

nrpe_user=nagios


# NRPE GROUP

# This determines the effective group that the NRPE daemon should run
as.

# You can either supply a group name or a GID.

# NOTE: This option is ignored if NRPE is running under either inetd
or xinetd

nrpe_group=nagios
```

```
# ALLOWED HOST ADDRESSES

# This is an optional comma-delimited list of IP address or hostnames
that are allowed to talk to the NRPE daemon. Network addresses with a
bit mask (i.e. 192.168.1.0/24) are also supported. Hostname wildcards
are not currently supported.

# Note: The daemon only does rudimentary checking of the client's IP
address.  I would highly recommend adding entries in your
/etc/hosts.allow file to allow only the specified host to connect to
the port you are running this daemon on.

# NOTE: This option is ignored if NRPE is running under either inetd
or xinetd

#allowed_hosts=127.0.0.1,::1

allowed_host=172.24.4.1

allowed_host=192.168.56.102



# COMMAND ARGUMENT PROCESSING

# This option determines whether or not the NRPE daemon will allow
clients to specify arguments to commands that are executed.  This
option only works if the daemon was configured with the --enable-
command-args configure script option.

# *** ENABLING THIS OPTION IS A SECURITY RISK! ***

# Read the SECURITY file for information on some of the security
implications of enabling this variable.

# Values: 0=do not allow arguments, 1=allow command arguments

dont_blame_nrpe=0



# BASH COMMAND SUBSTITUTION

# This option determines whether or not the NRPE daemon will allow
clients to specify arguments that contain bash command substitutions
of the form $(...).  This option only works if the daemon was
configured with both the --enable-command-args and --enable-bash-
command-substitution configure script options.

# *** ENABLING THIS OPTION IS A HIGH SECURITY RISK! ***

# Read the SECURITY file for information on some of the security
implications of enabling this variable.

# Values: 0=do not allow bash command substitutions,

#         1=allow bash command substitutions
```

```
allow_bash_command_substitution=0


# COMMAND PREFIX

# This option allows you to prefix all commands with a user-defined
string.

# A space is automatically added between the specified prefix string
and the command line from the command definition.

# *** THIS EXAMPLE MAY POSE A POTENTIAL SECURITY RISK, SO USE WITH
CAUTION! ***

# Usage scenario:

# Execute restricted commmands using sudo.  For this to work, you need
to add the nagios user to your /etc/sudoers.  An example entry for
allowing execution of the plugins from might be:

# nagios          ALL=(ALL) NOPASSWD: /usr/lib/nagios/plugins/

# This lets the nagios user run all commands in that directory (and
only them without asking for a password.  If you do this, make sure
you don't give random users write access to that directory or its
contents!

# command_prefix=/usr/bin/sudo


# MAX COMMANDS

# This specifies how many children processes may be spawned at any one
time, essentially limiting the fork()s that occur.

# Default (0) is set to unlimited

# max_commands=0


# COMMAND TIMEOUT

# This specifies the maximum number of seconds that the NRPE daemon
will allow plugins to finish executing before killing them off.

command_timeout=60


# CONNECTION TIMEOUT

# This specifies the maximum number of seconds that the NRPE daemon
will wait for a connection to be established before exiting. This is
sometimes seen where a network problem stops the SSL being established
even though all network sessions are connected. This causes the nrpe
daemons to accumulate, eating system resources. Do not set this too
low.

connection_timeout=300
```

```
# WEAK RANDOM SEED OPTION

# This directive allows you to use SSL even if your system does not
have a /dev/random or /dev/urandom (on purpose or because the
necessary patches were not applied). The random number generator will
be seeded from a file which is either a file pointed to by the
environment valiable $RANDFILE or $HOME/.rnd. If neither exists, the
pseudo random number generator will be initialized and a warning will
be issued.

# Values: 0=only seed from /dev/[u]random, 1=also seed from weak
randomness

#allow_weak_random_seed=1


# These directives allow you to specify how to use SSL/TLS.

# SSL VERSION

# This can be any of: SSLv2 (only use SSLv2), SSLv2+ (use any
version), SSLv3 (only use SSLv3), SSLv3+ (use SSLv3 or above), TLSv1
(only use TLSv1), TLSv1+ (use TLSv1 or above), TLSv1.1 (only use
TLSv1.1), TLSv1.1+ (use TLSv1.1 or above), TLSv1.2 (only use TLSv1.2),
TLSv1.2+ (use TLSv1.2 or above)

# If an "or above" version is used, the best will be negotiated. So if
both ends are able to do TLSv1.2 and use specify SSLv2, you will get
TLSv1.2.

# If you are using openssl 1.1.0 or above, the SSLv2 options are not
available.

#ssl_version=SSLv2+

# SSL USE ADH

# This is for backward compatibility and is DEPRECATED. Set to 1 to
enable ADH or 2 to require ADH. 1 is currently the default but will be
changed in a later version.

#ssl_use_adh=1


# SSL CIPHER LIST

# This lists which ciphers can be used. For backward compatibility,
this defaults to 'ssl_cipher_list=ALL:!MD5:@STRENGTH' for < OpenSSL
1.1.0, and 'ssl_cipher_list=ALL:!MD5:@STRENGTH:@SECLEVEL=0' for
OpenSSL 1.1.0 and greater.

#ssl_cipher_list=ALL:!MD5:@STRENGTH

#ssl_cipher_list=ALL:!MD5:@STRENGTH:@SECLEVEL=0

#ssl_cipher_list=ALL:!aNULL:!eNULL:!SSLv2:!LOW:!EXP:!RC4:!MD5:@STRENGT
H
```

```
# SSL Certificate and Private Key Files
#ssl_cacert_file=/etc/ssl/servercerts/ca-cert.pem
#ssl_cert_file=/etc/ssl/servercerts/nagios-cert.pem
#ssl_privatekey_file=/etc/ssl/servercerts/nagios-key.pem

# SSL USE CLIENT CERTS
# This options determines client certificate usage.
# Values: 0 = Don't ask for or require client certificates (default)
#         1 = Ask for client certificates
#         2 = Require client certificates
#ssl_client_certs=0

# SSL LOGGING
# This option determines which SSL messages are send to syslog. OR
values together to specify multiple options.
# Values: 0x00 (0)  = No additional logging (default)
#         0x01 (1)  = Log startup SSL/TLS parameters
#         0x02 (2)  = Log remote IP address
#         0x04 (4)  = Log SSL/TLS version of connections
#         0x08 (8)  = Log which cipher is being used for the
connection
#         0x10 (16) = Log if client has a certificate
#         0x20 (32) = Log details of client's certificate if it has
one
#         -1 or 0xff or 0x2f = All of the above
#ssl_logging=0x00

# NASTY METACHARACTERS
# This option allows you to override the list of characters that
cannot be passed to the NRPE daemon.
# nasty_metachars="|`&><'\\[]{};\r\n"

# INCLUDE CONFIG FILE
# This directive allows you to include definitions from an external
config file.
#include=<somefile.cfg>
```

```
# INCLUDE CONFIG DIRECTORY

# This directive allows you to include definitions from config files
(with a .cfg extension) in one or more directories (with recursion).

#include_dir=<somedirectory>

#include_dir=<someotherdirectory>


# COMMAND DEFINITIONS

# Command definitions that this daemon will run.  Definitions are in
the following format:

# command[<command_name>]=<command_line>

# When the daemon receives a request to return the results of
<command_name> it will execute the command specified by the
<command_line> argument.

# Unlike Nagios, the command line cannot contain macros - it must be
typed exactly as it should be executed.

# Note: Any plugins that are used in the command lines must reside on
the machine that this daemon is running on!  The examples below assume
that you have plugins installed in a /usr/local/nagios/libexec
directory.  Also note that you will have to modify the definitions
below to match the argument format the plugins expect.  Remember,
these are examples only!

# The following examples use hardcoded command arguments...

# This is by far the most secure method of using NRPE

command[check_users]=/usr/local/nagios/libexec/check_users -w 5 -c 10

command[check_load]=/usr/local/nagios/libexec/check_load -r -w
.15,.10,.05 -c .30$

command[check_hda1]=/usr/local/nagios/libexec/check_disk -w 20% -c 10%
-p /dev/hd$

command[check_zombie_procs]=/usr/local/nagios/libexec/check_procs -w 5
-c 10 -s Z

command[check_total_procs]=/usr/local/nagios/libexec/check_procs -w
150 -c 200


# The following examples allow user-supplied arguments and can only be
used if the NRPE daemon was compiled with support for command
arguments *AND* the dont_blame_nrpe directive in this config file is
set to '1'.  This poses a potential security risk, so make sure you
read the SECURITY file before doing this.


### MISC SYSTEM METRICS ###

#command[check_users]=/usr/local/nagios/libexec/check_users $ARG1$
```

```
#command[check_load]=/usr/local/nagios/libexec/check_load $ARG1$

#command[check_disk]=/usr/local/nagios/libexec/check_disk $ARG1$

#command[check_swap]=/usr/local/nagios/libexec/check_swap $ARG1$

#command[check_cpu_stats]=/usr/local/nagios/libexec/check_cpu_stats.sh
$ARG1$

#command[check_mem]=/usr/local/nagios/libexec/custom_check_mem -n
$ARG1$


### GENERIC SERVICES ###

#command[check_init_service]=sudo
/usr/local/nagios/libexec/check_init_service $A$

#command[check_services]=/usr/local/nagios/libexec/check_services -p
$ARG1$


### SYSTEM UPDATES ###

#command[check_yum]=/usr/local/nagios/libexec/check_yum

#command[check_apt]=/usr/local/nagios/libexec/check_apt


### PROCESSES ###

#command[check_all_procs]=/usr/local/nagios/libexec/custom_check_procs

#command[check_procs]=/usr/local/nagios/libexec/check_procs $ARG1$


### OPEN FILES ###

#command[check_open_files]=/usr/local/nagios/libexec/check_open_files.
pl $ARG1$


### NETWORK CONNECTIONS ###

#command[check_netstat]=/usr/local/nagios/libexec/check_netstat.pl -p
$ARG1$ $ARG$


### ASTERISK ###

#command[check_asterisk]=/usr/local/nagios/libexec/check_asterisk.pl
$ARG1$

#command[check_sip]=/usr/local/nagios/libexec/check_sip $ARG1$

#command[check_asterisk_sip_peers]=sudo
/usr/local/nagios/libexec/check_asterisk_$
```

```
#command[check_asterisk_version]=/usr/local/nagios/libexec/nagisk.pl -
c version

#command[check_asterisk_peers]=/usr/local/nagios/libexec/nagisk.pl -c
peers

#command[check_asterisk_channels]=/usr/local/nagios/libexec/nagisk.pl
-c channels

#command[check_asterisk_zaptel]=/usr/local/nagios/libexec/nagisk.pl -c
zaptel

#command[check_asterisk_span]=/usr/local/nagios/libexec/nagisk.pl -c
span -s 1
```