Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE SISTEMA DE INVENTARIO Y MONITORIZACIÓN DE RED

Alumno/Alumna: López Urbizu, Eder

Director/Directora (1): Jacob Taquet, Eduardo

Director/Directora (2): Astorga Burgo, Jasone

Curso: 2020-2021

Fecha: Bilbao, 20 de Julio de 2021

RESUMEN TRILINGÜE

CASTELLANO

Este proyecto tiene como objetivo diseñar e implementar un sistema de inventario y monitorización de red. Para empezar, se van a definir los requisitos y analizar las diferentes alternativas. Tras eso, se va a diseñar el sistema, y posteriormente, se va a implementar en el laboratorio I2T. Para finalizar, se van a realizar ciertas pruebas para comprobar su correcto funcionamiento.

EUSKERA

Proiektu honen helburua sare baten inbentario eta monitorizazio sistema bat diseinatzea eta ezartzea da. Hasteko, proiektuaren baldintzak zehaztu eta alternatiba ezberdinak aztertuko dira. Horren ondoren, sistema diseinatu eta I2T laborategian inplementatuko da. Amaitzeko, zenbait proba egingo dira, sistemaren funtzionamendu egokia egiaztatzeko.

INGLÉS

The objective of this project is to design and implement a network inventory and monitoring system. To begin with, the requirements will be defined and the different alternatives will be analyzed. After that, the system will be designed, and then, it will be implemented in the I2T laboratory. Finally, some functional tests will be carried out to verify its correct operation.

Keywords: monitorización, descubrimiento de red, VNF, NFV, OSM, Nagios Core, Grafana, Zenmap.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En los últimos 50 años la tecnología ha evolucionado a pasos agigantados. La sociedad ha pasado de no conocer ni los ordenadores ni Internet a usarlos a diario de forma activa. Esto supone un gran cambio en cuanto a las infraestructuras y a la cantidad de equipos que se usan para que todas las comunicaciones funcionen de forma correcta. Empresas, centros de investigación, entidades públicas y particulares, hoy en día todos usamos los equipos informáticos de forma activa generando una gran red de equipos interconectados mediante Internet.

Habitualmente, en nuestro domicilio o pequeña oficina conocemos como están interconectados estos equipos, ya sea porque son pocos dispositivos o porque los hemos instalado nosotros mismos. También conocemos los servicios que tienen disponibles o los que son capaces de utilizar. Esto, sin embargo, en una gran empresa puede ser todo un reto. Supongamos que tenemos una empresa de tamaño medio con varios departamentos en diferentes localizaciones dentro de un edificio. Cada departamento puede disponer de equipos diferentes los cuales ofrecen servicios distintos. Llevar el control de una red tan grande es una tarea muy costosa y complicada, y ahí es donde entran en juego los sistemas de inventario y monitorización de red, los cuales a día de hoy se usan en infinidad de entornos y cada día están cogiendo más fuerza.

Estos sistemas se encargan de monitorizar de forma automática los equipos que se les determinen. Mediante los protocolos y las configuraciones que se les hagan pueden determinar el estado de una máquina o de sus servicios, como por ejemplo, conocer la carga de la CPU, el uso de disco, uso de memoria RAM, los servicios que están en funcionamiento y los que no en esa máquina, entre otras muchas cosas.

Con esos datos, estos sistemas de monitorización se encargan de mantener la red segura ante los posibles errores que puedan surgir en los equipos que se han configurado, mostrando alertas en tiempo real a la persona encargada de monitorizar la red. De esta forma se consigue ahorrar mucho tiempo, esfuerzo y dinero, puesto que el hecho de que un servicio falle puede generar múltiples pérdidas económicas dentro de una empresa.

Además de mostrar los errores, muchos de estos sistemas también ofrecen la capacidad de ver la topología de la red en tiempo real. Esto tiene la ventaja de saber en todo momento de forma rápida como está estructurada la red sin tener que inspeccionarla físicamente. Esto ayuda a mantener un control de los equipos que hay en la red en todo momento, y así, saber cuáles hay que monitorizar.

Habitualmente, el paso previo a la monitorización suele ser el descubrimiento de equipos. De esta forma, se consigue conocer la red física mediante un mapa completo de dicha red generado automáticamente por el software de descubrimiento.

Además de esto, los sistemas de monitorización también pueden mostrar gráficos del comportamiento de la red y en algunos casos, prever el comportamiento futuro analizando los datos obtenidos en el pasado.

Por todo lo mencionado anteriormente, podemos concluir que los sistemas de monitorización son una herramienta muy potente e indispensable en entornos en los que hay una gran cantidad de equipos interconectados. Es por eso que se ha decidido realizar este

proyecto de puesta en marcha de un sistema de inventario y monitorización de red en el laboratorio de investigación I2T de la Escuela de Ingeniería de Bilbao. De esta forma, se consigue facilitar el monitoreo de la red, mantenerla lo más segura posible ante errores y ahorrar tiempo y esfuerzo en ello, lo cual repercutirá de forma directa en el tiempo y recursos disponibles para realizar otras tareas e investigaciones.

2. CONTEXTO

En este apartado se va a hablar acerca de los sistemas de inventario y descubrimiento de la topología de red de forma más profunda. Se van a explicar los conceptos básicos necesarios para entender el funcionamiento y también los protocolos que se utilizan para conseguir los datos necesarios para hacer el monitoreo y descubrimiento de los equipos. Además de eso, se va a definir el entorno en el que se va a desarrollar este proyecto y los servicios que se van a monitorizar. Por último, se va a detallar como se van a usar los datos obtenidos para conseguir una buena monitorización de red y servicios.

2.1. Protocolos básicos de monitorización de red

Para conocer el funcionamiento básico de los sistemas de monitorización y descubrimiento de red, primero hay que conocer los protocolos de red que utilizan dichos sistemas. Entre ellos, los más conocidos son los siguientes:

- ICMP: Este protocolo forma parte de los protocolos IP y su función es mandar mensajes de control. Por ejemplo, cuando un equipo manda un ICMP Echo Request a otro, se espera que el equipo que lo recibe responda con un ICMP Echo Reply. En el caso en el que no se obtenga respuesta, se entiende que el equipo al que se ha mandado el ICMP Echo Request no está disponible o tiene los mensajes ICMP desactivados. En este último caso, habría que usar otros mecanismos, como por ejemplo, SNMP o NRPE.
- SMTP: Este protocolo se utiliza para realizar transferencias de correos electrónicos.
- HTTP: Este protocolo se usa para transmitir información en entornos web. Es el encargado de realizar consultas y recibir la información solicitada.
- SNMP: Se utiliza para obtener información del estado de los equipos de red, como por ejemplo, los routers y los switches.

Estos protocolos se usan para obtener diversa información de los equipos que se están monitorizando. Los sistemas de monitorización de red y servicios usan el protocolo ICMP para determinar si un equipo está disponible o no. El sistema manda un paquete *ICMP Echo Request* al equipo y si este no responde o se recibe un *Host Unreachable* por parte de algún equipo de enrutamiento, se determina que el equipo no está disponible.

El protocolo SMTP (Simple Mail Transfer Protocol) se usa, como su nombre indica, para realizar transferencias de mensajes de correo electrónico. Se usa encapsulado bajo los protocolos TCP/IP. Los sistemas de monitorización de red lo usan para comprobar el estado del servicio de correo electrónico y para enviar notificaciones a los administradores de red.

Se utiliza HTTP cuando se quiere transmitir información en entornos de páginas web. Cuando un cliente hace una petición para recibir una página web en concreto, se utiliza el protocolo HTTP y se manda un mensaje GET. El servidor recibe ese mensaje, lo interpreta y responde. Si ha habido un error, informará de ello, de lo contrario, mandará la información solicitada con el mensaje *200 OK*. De esta forma se consigue realizar la petición y recepción de la información. Los sistemas de monitorización suelen utilizar este protocolo para saber si un servidor web está funcionando correctamente.

Por último, para monitorizar routers y switches, se usa el protocolo SNMP. Se puede saber si el puerto de un switch o router está activo haciéndole pings (ICMP), pero mediante SNMP se puede obtener mucha más información. Se puede obtener el nombre del equipo, estado de los

puertos, la temperatura del equipo, información sobre la fuente de alimentación y un largo etcétera.

2.2. Funcionamiento básico de los sistemas de monitorización de red

En este apartado se van a tratar los diferentes módulos por los cuales están formados los sistemas de monitorización y descubrimiento de red. Se van a tratar de forma individual y sencilla, intentando simplificar la explicación para conseguir una buena comprensión general de cada módulo. Además, también se va a dar una breve explicación de cada uno de los sistemas.

2.2.1. Sistemas de monitorización de red

Los sistemas de monitorización de red se usan para mantener una supervisión constante de todos los equipos y servicios de una red en concreto. Con esto, se busca que todos los equipos y servicios funcionen de forma adecuada, y en el caso en el que haya algún error, se consigue identificar de forma rápida y mostrar alertas al respecto, haciendo que se puedan tomar medidas para solucionarlo de forma prácticamente inmediata.

Estos sistemas están formados por varios módulos. Entre ellos, están los siguientes:

Módulos de testeo

Los módulos de testeo son los encargados de comprobar los parámetros que se les determina. Por ejemplo, un módulo de testeo es el módulo que realiza el chequeo *check_nrpe*, el cual se encarga de mandar la orden al demonio NRPE instalado en la máquina a monitorizar [1] [2].

El funcionamiento de *check_nrpe* es sencillo, tan solo hay que ejecutarlo y especificarle en los argumentos los parámetros que se quieren obtener de la máquina monitorizada. Al hacer esto, manda una solicitud de dichos datos al demonio NRPE que está corriendo en la máquina monitorizada, y este, le devuelve los datos solicitados. Este funcionamiento se puede observar en la *Figura 1*.

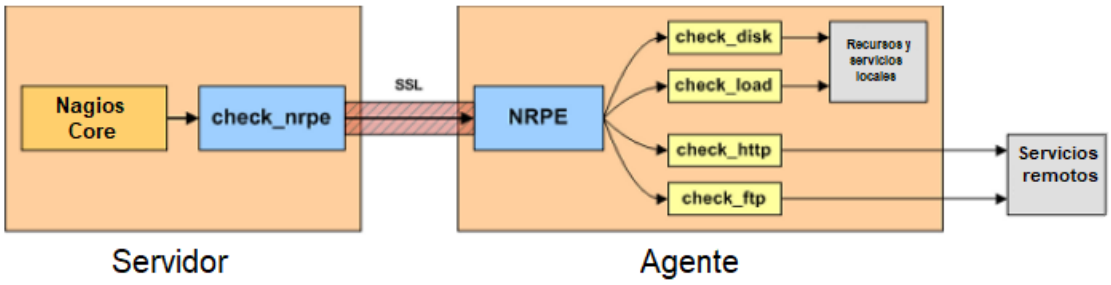

Figura 1 - Funcionamiento del módulo NRPE [2]

Otro ejemplo puede ser el módulo encargado de realizar la monitorización de un router o switch mediante el comando *check_snmp*. Este se encarga de mandar la orden y recibir la respuesta SNMP del equipo a monitorizar [3], tal y como se puede observar en la *Figura 2.*

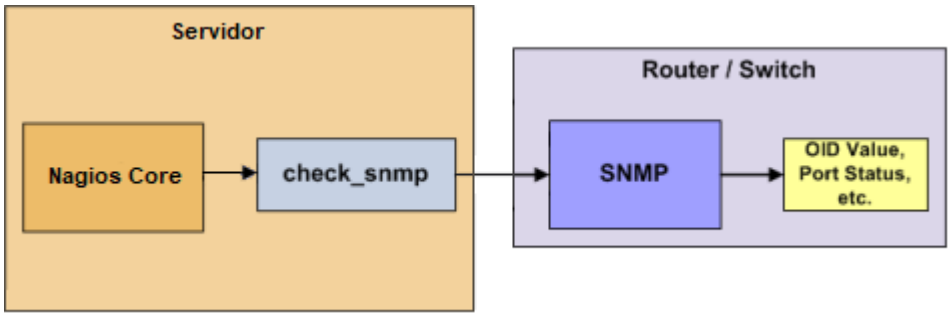

Figura 2 - Funcionamiento de SNMP

Por último, el módulo NSCA también está incluido en los módulos de testeo. Se encarga de realizar chequeos de forma pasiva de las máquinas remotas que se quieran monitorizar y mandar información al Master [1]. La principal diferencia respecto a NRPE, es que los chequeos se realizan de forma pasiva, es decir, cuando NSCA está en uso, se encarga de mandar datos a Nagios Core. De esta forma, se consigue tener datos de monitorización del equipo sin que Nagios Core alerte cuando el equipo se encuentra apagado o fuera de la red. Esto es muy útil en equipos que no se encienden a menudo, pues si se realizara la monitorización de estos equipos mediante NRPE, estaría notificando continuamente que el equipo no está disponible y se obtendrían un montón de notificaciones innecesarias.

Su funcionamiento es simple, se configura en la máquina que se quiere monitorizar para que cuando esté activa mande los datos de monitorización a Nagios Core, tal y como se puede observar en la *Figura 3*. NSCA puede monitorizar prácticamente los mismos parámetros que NRPE.

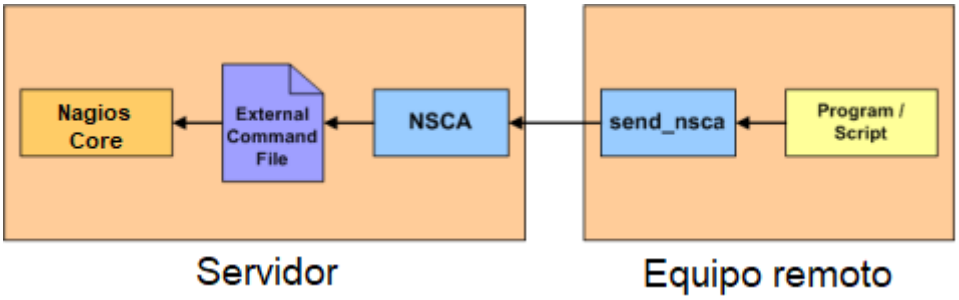

Figura 3 - Funcionamiento de NSCA [1]

Módulos de Almacenamiento de datos

Los módulos de almacenamiento de datos tienen la función de recopilar toda la información que llega a los sistemas de monitorización acerca de los equipos que se están monitorizando, para después, utilizar esos datos para generar gráficas o alertas.

Gracias a ellos, se pueden almacenar los datos de varios días, semanas e incluso meses, y así, obtener una vista general de los acontecimientos que han ido sucediendo y poder buscar patrones o errores recurrentes. De lo contrario, esto sería imposible y solo se podrían analizar los datos a corto plazo.

Estos sistemas se suelen apoyar en una base de datos. Normalmente MySQL o RRD (Round Robin Database), ya que son las más utilizadas.

Módulos de visualización

Los módulos de visualización tienen la función principal de mostrar las alertas y mensajes que generan las aplicaciones de monitorización. También se suelen utilizar para realizar gráficas de los diferentes datos que se recopilan al hacer la monitorización, como por ejemplo, el uso de la CPU, la carga de la memoria RAM o la temperatura del equipo monitorizado.

Un claro ejemplo de un módulo de visualización es PNP4Nagios [4]. En la *Figura 4* podemos ver un ejemplo de lo que puede mostrar, en este caso, el uso de la CPU de un equipo:

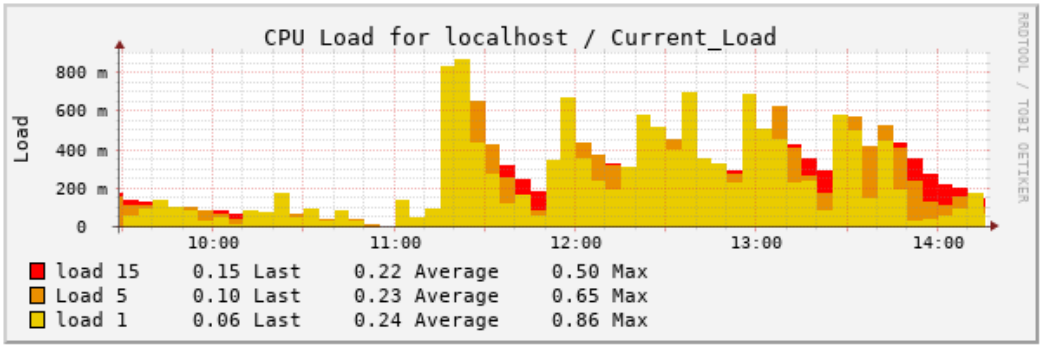

Figura 4 - Gráfica PNP4Nagios carga CPU

Estos módulos de visualización, habitualmente se suelen valer de bases de datos RRD. Estas bases de datos funcionan como almacenamiento temporal de los datos que se muestran en pantalla. Esto quiere decir que se guardan de forma temporal los datos que recopila el sistema de monitorización para generar dichas vistas, y tras pasar un tiempo definido en el sistema, estos datos se eliminan.

2.2.2. Sistemas de descubrimiento de red

Los sistemas de descubrimiento de red se centran en conocer en todo momento los equipos que están conectados a la red que se está supervisando. De forma periódica realizan un análisis de los diferentes equipos de la red que están conectados pudiendo conseguir sus direcciones IP y MAC, así como el nombre del equipo o el fabricante.

Se utilizan en todo tipo de redes, debido a que actualmente resulta muy complicado el hecho de llevar el control de todos los equipos en tiempo real. Antiguamente se conocía dónde estaba instalado cada equipo con conexión cableada, pero teniendo en cuenta el incremento que ha habido en la última década de los dispositivos personales y profesionales que se pueden conectar a Internet de forma inalámbrica, resulta prácticamente imposible llevar el recuento de todos ellos sin un software dedicado para ello.

Hay varios sistemas de descubrimiento de red, entre ellos están PRTG, Advanced IP Scanner, Zenmap y muchos otros. En este proyecto se va a utilizar Zenmap [5], dado que es el que mayor puntuación ha obtenido en el análisis de alternativas que se verá más adelante, concretamente, en el apartado *5.2. Software de descubrimiento*. Su interfaz se puede apreciar en la *Figura 5*:

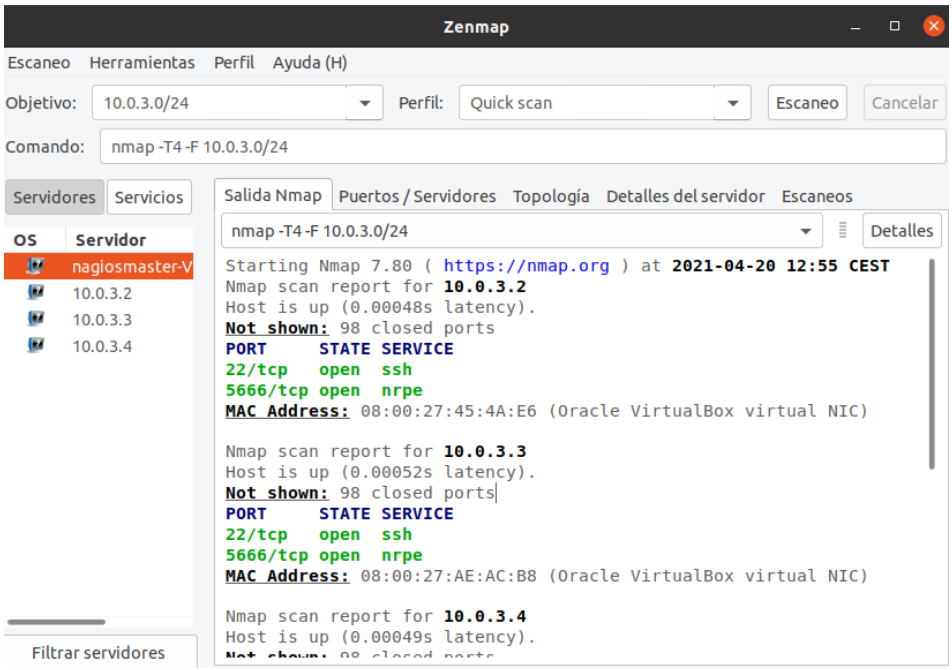

Figura 5 - Interfaz de Zenmap

2.3. Escenario de implementación de la solución

Visto el funcionamiento, se va a proceder a explicar el escenario en el que se va a instalar el sistema de monitorización e inventario de red. Este sistema se va a instalar en el laboratorio de investigación I2T de la Escuela de Ingeniería de Bilbao. Se va a realizar esta instalación, porque en este laboratorio hay multitud de equipos diferentes ofreciendo distintos servicios. Hay unas 50 IPs aproximadamente para monitorizar entre las que se encuentran equipos físicos, equipos virtualizados, routers y switches.

Este proyecto se va a centrar en ofrecer de forma sencilla y visual un resumen del estado de la red en tiempo real, ya que es muy complicado tener una visión general del estado de la red sin estas herramientas.

Las principales tareas de monitorización que se van a realizar son la monitorización de equipos para conocer su estado y la monitorización de cada uno de los servicios disponibles en el laboratorio. Los datos obtenidos se procesarán para generar alertas en caso en el que se encuentren errores y para realizar gráficos que muestren el estado de los equipos y la red.

2.4. Parámetros a monitorizar

Para realizar una buena monitorización y detectar los errores a tiempo y poder solucionarlos de la forma más rápida posible, hay que saber qué parámetros se tienen que monitorizar.

Por ejemplo, momentos antes de que un equipo se sature o deje de responder, habitualmente el uso de la CPU o memoria RAM se dispara. Si no se están monitorizando esos parámetros, es imposible darse cuenta de ello hasta que el equipo deja de responder. Gracias a

la monitorización, podemos prever estas situaciones supervisando estos parámetros, y en el caso de que haya algún comportamiento anómalo de alguno de los parámetros, se generará una alerta que verán los administradores de red.

Para evitar situaciones como las anteriores, en este proyecto se van a monitorizar los siguientes parámetros y servicios:

- **Uso de CPU:** Se va a monitorizar la carga de CPU del equipo en cuestión.
- **Uso de RAM:** Se va a monitorizar la carga de la memoria RAM del equipo en cuestión.
- **Número de procesos:** Se van a monitorizar los procesos del equipo en cuestión.
- **Espacio libre en discos:** Se va a monitorizar el espacio disponible en cada partición.
- **HTTP:** Se va a monitorizar si el servicio HTTP está disponible.
- **FTP:** Se va a monitorizar si el servicio FTP está disponible.
- **SSH:** Se va a monitorizar si el servicio SSH está disponible.
- **Peticiones ICMP:** Se va a monitorizar si el equipo responde a peticiones ICMP para saber si se encuentra activo.

2.5. Despliegue de la solución como VNF

En este proyecto se quieren aprovechar todas las ventajas de la virtualización y las VNF. Por ello, el sistema de monitorización y descubrimiento de red se va a desplegar como una VNF mediante un OpenStack gestionado con OSM, puesto que, la universidad UPV/EHU participa en la iniciativa OSM y se quiere aprovechar los resultados de dicha línea de trabajo. En las siguientes líneas se van a explicar los conceptos de virtualización, VNF y OSM.

2.5.1. Virtualización

La monitorización de los parámetros citados en el apartado *2.4. Parámetros a monitorizar* se va a realizar mediante una máquina virtual, debido a que es la técnica que más se está utilizando actualmente en las nuevas instalaciones de equipos, redes y servicios. La virtualización es una tecnología que aunque lleva muchos años entre nosotros, se empezó a generalizar su uso sobre el año 2000. Estas tecnologías se desarrollaron para acceder a los ordenadores que efectuaban procesamientos por lote hace varias décadas. Desde entonces, han ido avanzando hasta lo que conocemos hoy en día como virtualización. Un ejemplo muy sencillo es el del programa VirtualBox. Tan solo tenemos que instalarlo y crear una máquina virtual con el SO (Sistema Operativo) que queramos y ya tendremos disponible un entorno totalmente diferente al original.

Como se ha podido ver, en los últimos años se ha ido evolucionando notablemente en el ámbito de la virtualización. Hoy en día ya no es habitual ver instalaciones nuevas que no usen algún sistema de virtualización. Atrás han quedado los días en los que para hacer un cambio en una red había que cambiar el hardware de dicha red. Hoy en día se pueden realizar cambios en cuestión de minutos gracias a la virtualización. Un claro ejemplo de ello es la sustitución de los servidores físicos dedicados por máquinas virtuales o la sustitución de elementos de red por equipos virtualizados VNF (Virtualized Network Functions) creando una NFV (Network Function Virtualization) [6].

2.5.2. NFV

La virtualización de las funciones de red NFV es un marco de referencia que se ha desarrollado durante los últimos años debido a la necesidad constante de estar actualizando el hardware de las redes tradicionales para conseguir que ofrezcan más capacidad. El gran problema de las redes tradicionales es que para ser competentes y ofrecer nuevas características y capacidades, el hardware se tiene que sustituir cada poco tiempo, haciendo prácticamente imposible recuperar la inversión inicial. Este sistema no es nada viable, y justo ahí entra en juego la NFV.

La NFV aporta flexibilidad a las redes, ya que no dependen del hardware específico tanto como lo hacen las redes tradicionales. En las NFV, al igual que en las redes tradicionales, hay ciertas funciones que son imprescindibles, tales como el enrutamiento, el control de acceso, etcétera. Estas funciones habitualmente las solían hacer equipos con hardware específico, pero gracias a NFV se pueden sustituir los elementos de red por Funciones Virtualizadas de Red (VNF), que se ejecutarán sobre equipos hardware genérico a través de capas de virtualización.

De esta forma, se consigue una mayor flexibilidad y velocidad a la hora de realizar cambios en la red, ya que no hay que sustituir el hardware y tampoco es necesario desplazarse de forma física al lugar en los que están instalados los equipos, se pueden actualizar de forma telemática. Esto, a su vez, contribuye a disminuir los costes de actualización de una red.

En resumen, se pueden juntar varios elementos de red virtualizados (VNF) para crear una función de red virtualizada (NFV) y así, poder actualizarlas de forma mucho más sencilla, rápida y barata [7]. En la *Figura 6* se muestra de forma resumida todo lo citado anteriormente:

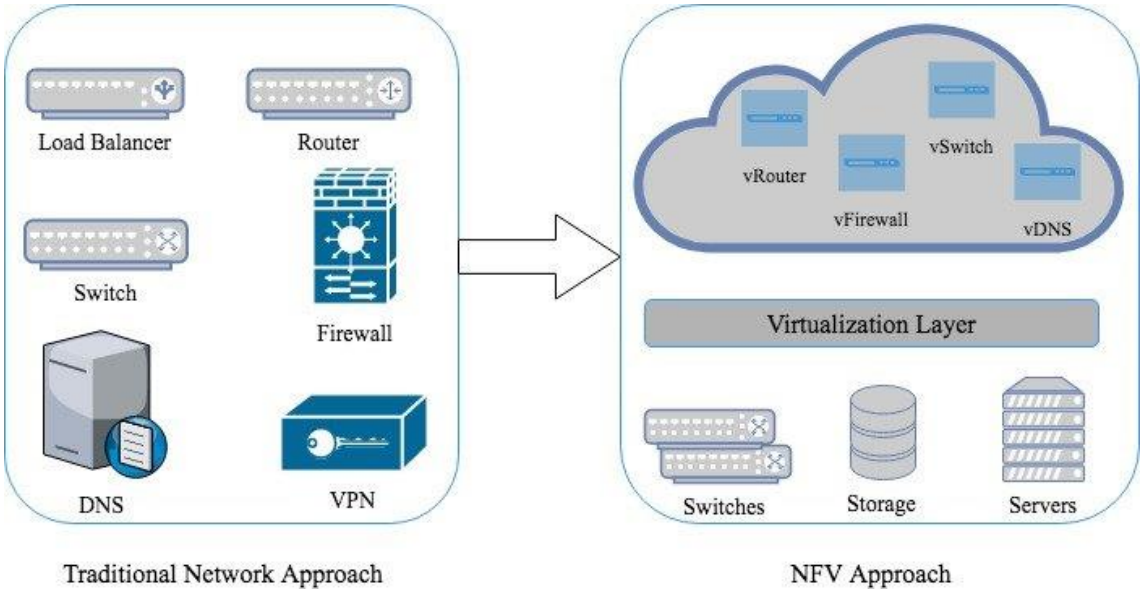

Figura 6 - Redes tradicionales vs redes con NFV [8]

2.5.3. OpenStack

En este proyecto se va a utilizar OSM (Open Source Mano) para desplegar varias VNF. Este OSM va a utilizar OpenStack como VIM (Virtual Infraestructure Manager) y, para que sea más sencillo de comprender, primero se va a explicar de forma resumida ambos conceptos.

OSM es un proyecto de ETSI el cual tiene como objetivo desarrollar software de código abierto para la orquestación y gestión de NFV. OpenStack, por su parte, es un entorno descentralizado, basado en la nube y está formado por varios componentes, cada uno de ellos con una funcionalidad concreta. Por lo tanto, es completamente modular.

Gracias a esto, se pueden ofrecer diversas soluciones basadas en la nube de forma mucho más accesible para los usuarios. Mientras que el usuario tenga una conexión rápida a internet, no tiene la necesidad de tener un hardware potente, ya que es en la nube donde se van a realizar todas las tareas pesadas. De esta forma, se eliminan los altos costes que suele suponer el hardware dedicado.

Cabe destacar que OpenStack y ETSI OSM están publicados bajo la licencia Apache 2.0, por lo cual, su uso es totalmente gratuito, ya que la licencia Apache 2.0 es de software libre, pero tiene algunas limitaciones, ya que no es una licencia copyleft.

Tras conocer estos detalles, se va a proceder a explicar los módulos de OpenStack [9]:

1. **Horizon**: Es el módulo que se encarga de proporcionar una interfaz gráfica al usuario. Gracias a él se puede interactuar con OpenStack de forma sencilla e intuitiva. Es similar a las funciones de web client que proporcionan otros programas pero basado en la nube.
2. **Keystone**: Es el encargado de gestionar todo lo que tiene que ver con la identidad de los usuarios. Se encarga de la autenticación, de los permisos de los usuarios y también de las políticas.
3. **Neutron**: Es el módulo que se encarga de realizar la gestión de la conectividad. Se encarga de gestionar todo lo que tiene que ver con la infraestructura virtual de la red. Gracias a él se pueden gestionar las direcciones IP, VPNs, VLANs etcétera.
4. **Cinder:** Crea dispositivos de almacenamiento en bloque mediante la virtualización. Por ejemplo, puede crear un disco duro de forma virtual, y si en un futuro un usuario necesita más capacidad, puede ampliarla de forma sencilla, favoreciendo la escalabilidad.
5. **Nova**: Este es el módulo que se encarga de la computación. Se puede decir que se trata de la CPU que controla todo. Es uno de los componentes más importantes de OpenStack.
6. **Glance**: Es el encargado de almacenar las imágenes de las máquinas virtuales. Permite guardar, restaurar y recrear las imágenes tantas veces como se desee.
7. **Swift**: Es el encargado de gestionar el almacenamiento de los contenedores para objetos. Permite realizar un almacenamiento distribuido y conseguir redundancia en los datos de forma transparente al usuario.
8. **Ceilometer**: Es el encargado de la telemetría. Proporciona datos que se pueden utilizar para la monitorización, alertas, etcétera. Incluye un demonio de almacenamiento que puede comunicarse mediante mensajes con agentes autenticados.
9. **Heat**: Es el módulo que se encarga de la orquestación. Proporciona plantillas para crear y administrar servicios en la nube, tales como, almacenamiento, servicios de red, aplicaciones etcétera. Este módulo no se utiliza en OSM, ya que se sustituye por

funcionalidades propias. Como en este proyecto se usa OSM, en la solución no se va a utilizar este módulo.

En la *Figura 7* se puede observar el esquema de funcionamiento de los módulos de OpenStack:

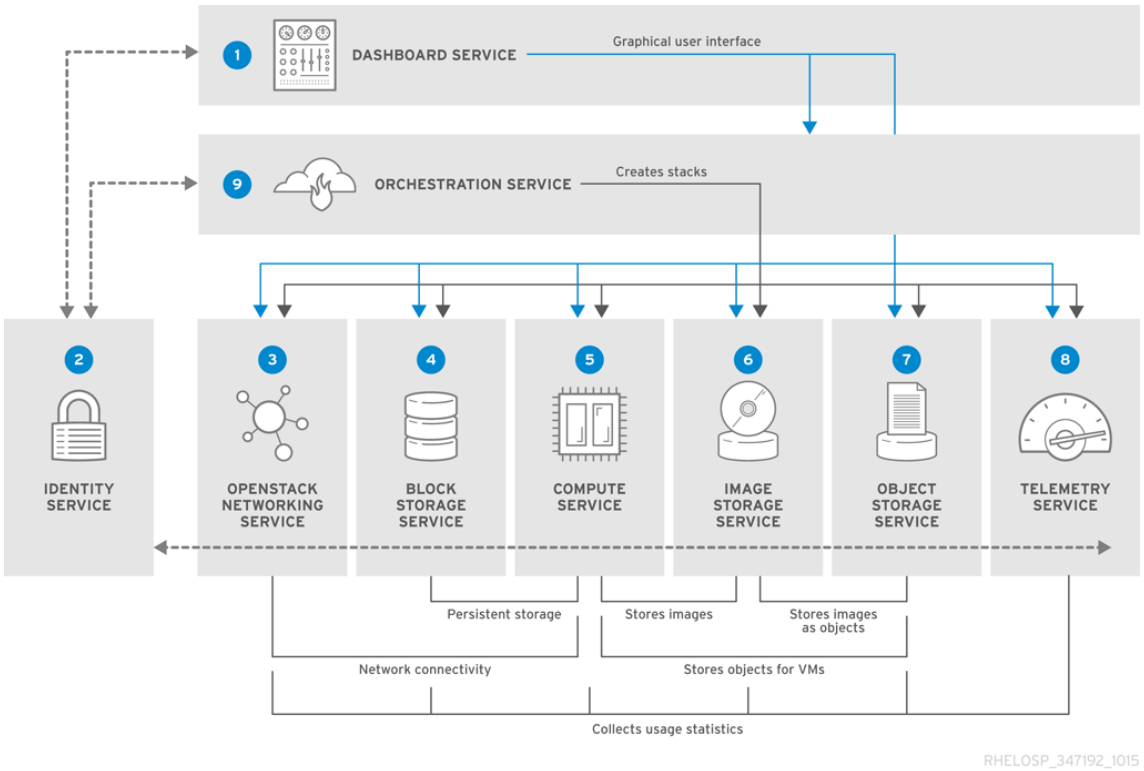

Figura 7 - Descripción de alto nivel de los servicios de OpenStack [9]

Todos estos módulos pueden trabajar de forma descentralizada, localizados cada uno en diferentes equipos físicos, abaratando costes de hardware y favoreciendo la escalabilidad y seguridad de los servicios.

2.5.4. OSM

Se conoce como ETSI Open Source MANO (NFV Management and Orchestration) u OSM a la implementacion de los conceptos de ETSI NFV promovida por ETSI ISG NFV (ETSI Industry Specification Group NFV). Este grupo de ETSI se centra en definir los estándares de NFV [10] [11] y dentro de este grupo hay diversas compañías ayudando a definir la capa de referencia de gestión de NFV. Entre dichas empresas podemos encontrar empresas tecnológicas, del mundo de la telecomunicación o incluso centros educativos. Entre ellas están Amazon, Oracle Corporation, Red Hat, Telefónica, T-Mobile, Verizon UK o incluso la propia UPV/EHU, como se puede apreciar en la página oficial de ETSI [12].

El objetivo principal de OSM es conseguir un software para la orquestación de las NFV que sea de calidad y se presente bajo una licencia Apache 2.0 de software libre. Actualmente, el código de OSM se puede usar para dirigir casos complejos de NFV y permitir a los proveedores de software y hardware dar soluciones de forma económica y mucho más rápida que de forma tradicional.

Desde la creación de este grupo, cada vez se han ido sumando más y más empresas del mundo de la tecnología al ver el gran potencial de este proyecto y todas sus ventajas. Actualmente la lista de miembros y participantes en muy amplia y se prevé que siga aumentando.

2.5.5. Ventajas de la virtualización

Antes de continuar con los pasos para instalar Nagios Core en la máquina virtual del laboratorio I2T, se van a mostrar las ventajas más significativas de la virtualización de los sistemas que se van a instalar [13].

- **Mejora de la agilidad, flexibilidad y escalabilidad:** Al realizar una virtualización y no depender de hardware específico para una solución concreta, se consigue agilizar los procesos de actualización, haciéndolos muchos más flexibles y facilitando la escalabilidad.
- **Reducción de la inversión principal**: En el caso en el que se quisiera realizar este proyecto de forma tradicional, se tendría que instalar en un equipo físico solamente con este propósito, el de monitorizar la red. Mediante la virtualización, se va a instalar Nagios Core sobre un sistema operativo que corre a modo de máquina virtual en un servidor del laboratorio I2T.
- **Reducción de costes:** En el servidor puede haber más de una MV y por lo tanto, se puede hacer un uso más eficiente de los equipos físicos, generando un ahorro considerable a largo plazo, puesto que hay una reducción de costes importante gracias a la utilización eficiente del servidor y el ahorro de energía que supondría tener otro equipo en marcha.

Para que las ventajas anteriores se puedan apreciar de una forma más visual, se muestran las *Figuras Figura* 8 *y Figura* 9, en las que se puede observar claramente la ventaja de la virtualización. Realizando una instalación tradicional, se instalarían 3 servidores físicos, cada uno con una función en concreto. En este caso serian el servidor de correo, web y aplicaciones nativas. En este ejemplo podemos ver cómo solo se usa el 30% de la capacidad de cada servidor físico, por lo que podríamos virtualizar los tres servidores e incluir las 3 funciones en un mismo servidor físico para ahorrar costes. También está la opción de virtualizar únicamente los servidores que se prefieran. En el ejemplo se han virtualizado los servidores de correo y aplicaciones obteniendo una mejor eficiencia y suprimiendo un servidor físico.

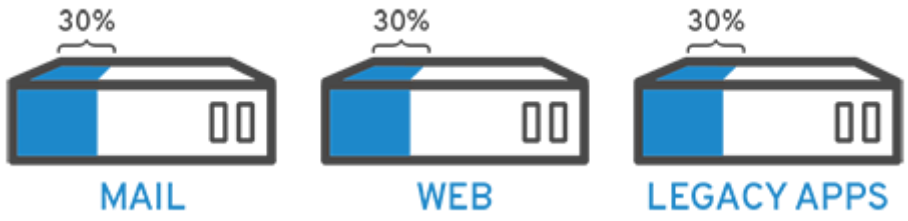

Figura 8 - Servidores físicos sin virtualización [14]

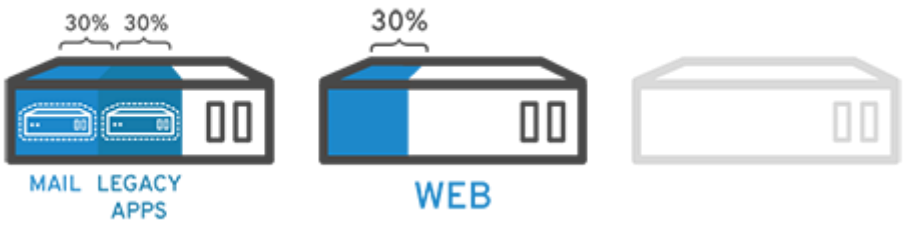

Figura 9 - Servidores físicos con virtualización [14]

3. OBJETIVOS Y ALCANCE DEL TRABAJO

El objetivo principal de este trabajo es el diseño e implementación de un sistema de descubrimiento y monitorización de red y servicios el cual se va a desplegar en el laboratorio de investigación I2T.

Por lo tanto, este TFG se centra principalmente en la instalación de un sistema totalmente funcional de descubrimiento y monitorización de redes y servicios en el laboratorio I2T de la Escuela de Ingeniería de Bilbao. Se quiere conseguir un sistema que monitorice la red en tiempo real, alerte a los administradores de la red cuando haya errores y les muestre los datos de forma muy visual mediante gráficos. A la vez que esto, también se quiere conseguir que el sistema tenga las funciones de descubrimiento de elementos de red. Esta función escaneará la red del laboratorio para hacer un inventario de los diferentes equipos conectados a ella.

Los objetivos secundarios son los siguientes:

- Definición de objetivos y requisitos del proyecto
- Análisis de alternativas y selección
- Diseño del sistema
- Implementación y validación del sistema

Definición de objetivos y requisitos del proyecto

Este es uno de los pasos más importantes a realizar. Es necesario identificar los requisitos para poder hacer la definición de objetivos lo más adecuada posible. Entre los más importantes está conseguir notificaciones en tiempo real en los smartphones de los miembros del laboratorio I2T.

Análisis de alternativas y selección

Se va a realizar un análisis de las diferentes herramientas disponibles para elegir la más adecuada de acuerdo a las necesidades de los investigadores del laboratorio I2T. Se va a realizar un análisis que contemple la clasificación del software, la compatibilidad, el rendimiento, la capacidad de adaptación a lo que se quiere monitorizar, la facilidad de implementación, la frecuencia de actualización y el precio.

Diseño del sistema

Se va a diseñar una solución que cumpla con los requisitos establecidos para este proyecto pensando siempre en intentar realizar una implementación ordenada y de fácil comprensión para los futuros administradores de red.

Implementación y validación del sistema

Se va a realizar una instalación limpia y ordenada, que requiera modificaciones mínimas en los archivos de configuración, para que se asemeje lo máximo posible a la configuración de las guías de instalación que proporcione el fabricante del software a instalar. Además, se va a comprobar el correcto funcionamiento del sistema de monitorización y descubrimiento.

4. BENEFICIOS

En este apartado se van a tratar los diferentes beneficios que va a aportar este trabajo. Se van a dividir en tres categorías: Económicos, técnicos y sociales.

Beneficios técnicos

La realización de este trabajo va a suponer un gran cambio en la forma de monitorizar la red en el laboratorio I2T de la escuela de ingeniería de Bilbao. Al igual que en muchas empresas, en los laboratorios de la universidad también es recomendable tener un sistema de monitorización de red y servicios, dado que habitualmente disponen de muchos equipos informáticos.

Actualmente en el laboratorio I2T no disponen de ningún sistema completo de monitorización de red y servicios, por lo que en caso de que algún equipo o servicio falle, entre en funcionamiento degradado o deje de estar operativo, pueden pasar horas hasta que se detecta dicho fallo.

El mayor beneficio que va a traer este trabajo es la automatización, rápida detección y notificación de la gran mayoría de los errores de la red, proporcionando información en tiempo real de forma muy precisa.

Beneficios sociales

Los beneficios sociales van de la mano de los técnicos, ya que al conseguir una automatización y detección de los errores de la red, se consigue que los miembros del laboratorio I2T no tengan que dedicar recursos a tareas que ya están automatizadas.

Gracias a esto, podrán invertir el tiempo que le deberían de dedicar a la monitorización de la red en otros proyectos o investigaciones que puedan traer beneficios sociales a largo plazo. Por lo tanto, este proyecto ayuda de forma indirecta a conseguir dichos beneficios.

Beneficios económicos

Los beneficios económicos, a su vez, vienen también ligados a los sociales. El hecho de que los investigadores del laboratorio I2T dediquen menos tiempo a monitorizar y solucionar problemas de la red hace que tengan más horas libres para otros trabajos e investigaciones.

Esas horas que no se consumen monitorizando y corrigiendo errores de la red o de los servicios se pueden ver como un ahorro potencial a largo plazo para la universidad, pues son horas de trabajo que no se van a malgastar y se van a usar de forma eficiente, por lo tanto, esas horas son sinónimo de dinero que no se va a gastar en la monitorización de la red.

También cabe destacar que mediante este proyecto se van a proporcionar herramientas para la detección precoz de los problemas de red. Gracias a esto, se van a poder remediar antes de que se conviertan en problemas más graves los cuales puedan generar perdidas de datos, caídas en el sistema o pérdidas económicas derivadas de estos problemas.

5. ANALISIS DE ALTERNATIVAS

En este apartado se va a estudiar el análisis de alternativas realizado para este proyecto. Las problemáticas identificadas para este apartado se han dividido en diferentes grupos, y han sido los siguientes:

- Software necesario para la monitorización de máquinas y servicios.
- Software necesario para el descubrimiento automático de servicios y máquinas.
- Sistema operativo a utilizar para el despliegue de la solución
- Mecanismo de despliegue de la solución
- Almacenamiento y visualización de logs

Con el fin de solucionar las problemáticas, se van a estudiar y analizar cada una de las alternativas. Después de esto, se van a definir los criterios de elección para más tarde, realizar el análisis de las alternativas y los criterios. Después de realizar todo lo anterior, se hará una evaluación para realizar la elección de la solución óptima.

5.1. Software de monitorización

En este apartado se va a tratar la problemática de buscar el software necesario para la función de **monitorización de máquinas y servicios**. Se van a detallar las alternativas disponibles y se procederá a elegir la que más puntuación obtenga en el análisis tras analizarlas mediante los criterios de evaluación.

5.1.1. Descripción de alternativas

Antes de analizar cada una de las alternativas que se van a detallar más abajo, es necesario detallar los subgrupos en los que se van a clasificar dichas alternativas:

- **SaaS:** En inglés, “Software as a Service”. Se refiere a los diferentes tipos de software los cuales se ofrecen como un servicio y se alojan en los servidores de la compañía que ofrece dicho servicio. El cliente accede a los servicios mediante la página web de la empresa que presta el servicio SaaS.
- **Software en propiedad:** Se le denomina software en propiedad a todo el conjunto de programas y software que adquiere una empresa a cambio de una cuota mensual, anual o un único pago y el cual se instala en los servidores de la empresa cliente. Esta opción ofrece un servicio técnico de atención al cliente, actualizaciones y correcciones de errores por parte de la empresa desarrolladora de dicho software.
- **Software de código abierto:** Este tipo de software es de uso público y gratuito. El código fuente del software se puede conseguir de manera sencilla en Internet y cualquier persona o entidad tiene el permiso para modificarlo según sus necesidades sin tener que pagar ningún tipo de licencia o servicio.

Una vez detallados los subgrupos en los que se va a clasificar el software, se van a especificar las diferentes alternativas dentro de cada subgrupo:

5.1.1.1. SaaS (Software as a Service)

Acronis Monitoring Service

Después de que Acronis haya decidido dejar de dar soporte a su servicio *Acronis Monitoring Service* a partir del día 30 de septiembre de 2020, esta empresa ha lanzado *Acronis Cyber Protect*, un nuevo servicio centrado en la seguridad y la simplicidad. Dicho servicio se centra en eliminar la complejidad a la hora de proteger los equipos y en disminuir los incidentes y minimizar los tiempos de respuesta [15]. Para ello utiliza un motor avanzado de detección de comportamiento con inteligencia artificial, realiza copias de seguridad y evaluaciones de vulnerabilidad. Como se puede observar, este software está pensado para ofrecer seguridad y fiabilidad, no es una alternativa real al software de monitorización *Acronis Monitoring Service*, por lo tanto, queda automáticamente descartada y no se va a incluir a la hora de realizar la evaluación.

New Relic

Es un sistema de monitorización de equipos el cual muestra los recursos disponibles en tiempo real. Se utiliza sobre todo para la monitorización de aplicaciones web y móviles, dado que es muy sencillo de implementar. Permite monitorizar los siguientes recursos: tiempo de carga, procesos, discos, uso de red etcétera. Nos puede mostrar estadísticas o monitorizar conexiones HTTP, errores, alertas, usuarios y rendimiento [16] [17].

LogicMonitor

Es un sistema de monitorización híbrido basado en la nube y totalmente automatizado. Ofrece al cliente una monitorización de la red, servidores, discos y bases de datos en la nube, y el cliente puede acceder desde cualquier dispositivo a la monitorización [18].

5.1.1.2. En propiedad

PRTG

Es un software en propiedad de la empresa Paessler que permite supervisar dispositivos, tráfico y aplicaciones. Es de fácil instalación y configuración, y además, escalable a grandes redes. No es necesaria la instalación de software adicional para realizar el monitoreo de los equipos, debido a que usan protocolos estándar como SNMP para preguntar a dichos equipos la información que se necesita. Con la información que devuelven se realizan las gráficas del monitoreo. Permite establecer un límite de ancho de banda por equipo, supervisar datos específicos de bases de datos SQL, administrar y obtener estadísticas mostrando los servicios que se ejecutan en la red, monitorear la disponibilidad, fiabilidad, capacidad y accesibilidad de los servidores en tiempo real y hacer un seguimiento de la red local, entre otros. Es compatible con Linux y con las tecnologías SSH, API REST y SQL [19].

SolarWinds

Este software ofrece una monitorización centrada en el rendimiento de los sistemas y la red. Permite monitorizar las aplicaciones y los servidores con el fin de identificar los problemas de rendimiento antes de que aparezcan. Además, también da la opción de monitorizar la capacidad de los servidores y hacer estadísticas para anticipar cuál va a ser la capacidad necesaria de cada uno de ellos. Es compatible con las tecnologías habituales y también con Ubuntu [20].

ManageEngine

Es un conjunto de software independiente para cada tarea en concreto. *ManageEngine* permite obtener solo el software necesario para cada problemática. De esta forma, se puede obtener un software para monitorizar el acceso, otro para obtener analíticas, otro para controlar aplicaciones y privilegios etcétera; sin tener que adquirir los otros productos. Ofrece soluciones específicas al cliente y aparte de lo mencionado anteriormente, también ofrece servicios de seguridad, bases de datos y nube [21].

5.1.1.3. Código abierto

Nagios Core

Nagios Core es un programa gratuito y de código abierto el cual se encarga de monitorizar los equipos y servicios de la red en la que se configure. Cuando dichos equipos o servicios fallen o no se comporten de manera habitual, creará una alerta para avisar de ello. Para monitorizar los servicios de red hace uso de HTTP, DNS, FTP, SNMP, SSH y SMTP, entre otros muchos. Gracias a las citadas tecnologías, Nagios puede hacer una monitorización en tiempo real de los servicios de red y de los recursos de hardware. Además, también da la opción de realizar la monitorización remotamente, entre otras muchas opciones [22]. No hay que confundir Nagios Core con otros productos de Nagios como Nagios XI o Nagios Fusion, los cuales, son de pago y no son de código abierto.

Icinga

Icinga es un programa que nace a partir de una bifurcación del programa Nagios en el año 2009. Al igual que el software anterior, este también es gratuito y de código abierto. Con esta bifurcación la comunidad de creadores quiso paliar las deficiencias que tenía el programa Nagios tales como una interfaz anticuada, la falta de personal para el desarrollo y la falta de implementación de una API. Además, aparte de todas las funciones que tiene Nagios, Icinga también permite la conexión con diferentes bases de datos y la posibilidad de utilizar una REST API. El desarrollo de este software es relativamente continuo y se publican nuevas actualizaciones regularmente. También cabe mencionar que ofrece un soporte de atención al cliente de pago, pero en este caso no se va a tener en cuenta, pues los posibles contratiempos se solucionarán sin contactar con el servicio de soporte [23] [24].

Zabbix

Zabbix es un programa gratuito y de código abierto que se encarga de realizar la monitorización de servidores, hardware y servicios de red. Dispone de una interfaz sencilla la cual muestra información general y gráficos acerca de la monitorización que está realizando. También dispone de la opción de crear alertas de forma personalizada y se puede utilizar con agentes nativos instalándolos en el equipo que se quiere monitorizar o sin ellos, dependiendo de qué parámetros se quieran monitorizar. El software es compatible con plataformas Linux y Windows y también dispone de una opción de despliegue rápido en la cual se puede descargar las *Zabbix Appliances* en formato .iso, .vmx o .ovf, entre otros [25] [26].

MRTG

MRTG es un software gratuito de monitorización y medición de tráfico en la red. En un principio se desarrolló solo para monitorizar el tráfico de los routers, pero hoy en día puede

medir infinidad de parámetros y con ellos, crear gráficos y estadísticas. Está escrito en Perl y se puede ejecutar en Windows, Linux, Unix, MacOS y NetWare.

MRTG usa el protocolo SNMP para mandar *Requests* con dos identificadores a un dispositivo conectado a la red. El dispositivo devuelve la información que se le solicita y el programa MRTG la recolecta y procesa [27] [28].

5.1.2. Criterios de evaluación

Para hacer la mejor elección de las alternativas descritas, se asignará una nota del 1 al 9 en los criterios de evaluación. Los criterios que se han tomado en cuenta han sido los siguientes:

- **Compatibilidad:** Se busca que el software a utilizar sea compatible con Ubuntu 20.04 LTS, ya que es el que se usa en los equipos y máquinas virtuales del grupo de investigación I2T. De ser compatible con dicho sistema operativo, obtendrá la puntuación más alta, de lo contrario, la más baja. Este criterio de evaluación tendrá un peso del 20%. De esta forma, de tener la puntuación más baja, quedaría prácticamente descartada la alternativa, bajando mucho su puntuación.
- **Rendimiento:** Este criterio evalúa la potencia necesaria para ejecutar el software de la alternativa en cuestión. Cuantos menos recursos consuma la herramienta, más puntuación obtendrá.
    - Nota: Las pruebas de rendimiento y recursos se van a hacer de forma orientativa en el equipo personal utilizando una máquina virtual con el sistema operativo Ubuntu 20.04 LTS instalado. Se probarán los programas compatibles con dicho sistema operativo, y los que no, quedarán descartados y se les asignará la nota más baja en la tabla de evaluación. El equipo personal es considerablemente menos potente que los equipos del grupo. Así, de haber alguna diferencia en cuanto a potencia y rendimiento, sería despreciable para los equipos del grupo de investigación. El peso de este criterio será del 15%.
- **Personalización**: Este criterio va a analizar la capacidad de adaptación y personalización del software a las necesidades del usuario. Cuanto mayor sean las opciones de adaptación, más puntuación obtendrá. El peso de este criterio será del 20%.
- **Implementación**: En este criterio se analizará la facilidad de implementación del software en cuestión. Cuanto más sencillo sea, más puntuación obtendrá. Este criterio tendrá un peso del 7.5%.
- **Actualización**: Este criterio se va a evaluar basándose en el historial de actualizaciones del software en cuestión. Cuanto mayor sea la frecuencia de actualización, mayor será la nota de evaluación. Este criterio tendrá un peso del 12.5%.
- **Precio**: Se busca que el software a utilizar sea lo más económico posible para ahorrar costes. De este modo, cuanto mayor sea el desembolso económico a realizar para la obtención del software, menor será la nota obtenida en este criterio. De esta forma, se favorecerá la utilización de software gratuito. El peso de este criterio será del 25%.

5.1.3. Análisis de las alternativas y los criterios de elección

En el siguiente apartado se va a realizar el análisis de alternativas y los criterios de evaluación. Dado que realizar el criterio de evaluación de rendimiento puede consumir mucho tiempo y esfuerzo, ya que hay que instalar todos los programas en la máquina virtual y probar su funcionamiento, se va a hacer una preselección de software mediante el resto de criterios y se analizará el criterio de rendimiento solo en las alternativas que hayan conseguido mayor puntuación.

New Relic

Para comenzar, se va a analizar el software New Relic. Dicho programa es un SaaS, el cual ofrece una membresía gratuita muy limitada. Ofrece acceso gratuito a un usuario y la monitorización de 100 Gb/mes de datos. A partir de ese umbral, empieza a cobrar al usuario. Además, si se quieren añadir usuarios extra, es necesario abonar 99 €/mes por cada uno. La compatibilidad con Ubuntu está asegurada y en cuanto a la personalización, al ser un software diseñado por una empresa, no da muchas opciones al cliente, aunque sí permite hacer ligeras modificaciones. En la página web hay un foro en el que se discuten las modificaciones y dudas de los usuarios.

En cuanto a la facilidad de instalación, se puede instalar en Ubuntu siguiendo los pasos de la guía que proporcionan en su página web [29] [30] o en YouTube [31]. La instalación es sencilla y no conlleva complicaciones.

En cuanto a la frecuencia de actualizaciones, se puede observar que es bastante continua. Esto lo podemos ver entrando a las release notes del apartado de la nube [32] y de los diagnósticos [33]. Podemos observar que en el apartado de la nube hay actualizaciones cada aproximadamente 15 días y en el de los diagnósticos cada 1,5 meses aproximadamente. El resto de apartados se pueden ver en el apartado de release notes, pero no se va a analizar cada uno, debido a que el fin de este análisis es obtener una idea aproximada de la frecuencia de actualización.

LogicMonitor

La siguiente alternativa de tipo SaaS es LogicMonitor [18]. Para poder utilizar este software hay que abonar una cuota mensual por equipo (entiéndase equipo por IP o entrada DNS la cual se quiera monitorizar) la cual es personalizada para cada empresa. No ha sido posible encontrar un listado de precios en la página oficial, pero indagando en foros se ha podido observar que dicha cuota ronda entre los 7 $/mes y 15 $/mes por equipo, dependiendo del número de equipos, las opciones de monitorización contratadas y la oferta personalizada del plan contratado. Teniendo en cuenta que en el grupo de investigación hay 50 IPs, el total del importe ascendería a 550 $/mes tomando como tarifa aproximada la media de los dos extremos antes mencionados (11 $/mes).

La personalización en este software resulta algo escasa, aunque es mayor que NewRelic, la alternativa anterior. Hay opción de realizar gráficos y widgets personalizados.

Respecto a la implementación, es sencillo de instalar, solo hay que iniciar sesión e instalar un colector en la máquina a monitorear. En cuanto a las actualizaciones de la plataforma y los colectores, se puede ver en las *release notes* [34] que se realizan de forma regular mensualmente, por lo que el software está constantemente actualizado.

PRTG

PRTG es la primera alternativa del software en propiedad. Dispone de una versión gratuita y una de pago. La versión gratuita ofrece la posibilidad de usar 100 sensores. Los sensores son cada uno de los parámetros que se pueden monitorizar en un equipo, así como, un puerto switch, la carga de CPU etc. Las licencias de pago comienzan desde 500 sensores hasta sensores ilimitados y rondan desde los 1300 € + IVA hasta los 12500 € + IVA [35]. Estas licencias son de un único pago y de por vida, y en el caso de necesitar ampliarlas solo habría que pagar la diferencia de precio de la actual a la siguiente. También dan opción a solicitar una licencia personalizada.

Respecto a la compatibilidad, no se especifica bien en la página oficial y se da a entender que es compatible tanto con Windows como con Linux. Tras buscar información en otras webs [36], se ha llegado a la conclusión de que la herramienta principal solo es compatible con sistemas Windows y es necesario tener en la red un equipo Windows. Dado que la herramienta principal se va a tener que instalar en un equipo con Ubuntu 20.04 LTS, la compatibilidad no es favorable.

En cuanto a la personalización, tiene un abanico bastante extenso de sensores, los cuales se pueden elegir de forma individual para hacer la monitorización. La implementación es imposible, dado que el software no es compatible con Ubuntu. En cuanto a las actualizaciones, en las release notes [37] aparece detallada la frecuencia de actualización, la cual es de aproximadamente un mes, por lo que es una buena frecuencia.

SolarWinds

Esta herramienta ofrece al usuario una prueba gratuita de 30 días, tras la cual hay que empezar a pagar los servicios. El precio depende de los servicios que se quieran contratar. El administrador de direcciones IP parte de 915 €, el software para monitorizar el ancho de banda parte de 845 €, el monitor de red parte de 1290 € y el monitor de servidores y aplicaciones de 1275 € [38].

En cuanto a la compatibilidad, en la documentación del software [39] aparece que está probado hasta la versión 18.04 LTS de Ubuntu. No aparece nada al respecto de la versión 20.04 LTS, aunque se entiende que será compatible con esta última versión.

En cuanto a la personalización, el software da bastantes opciones y medidores, entre ellos, la medición de CPU, memoria, discos, tarjeta gráfica, adaptadores de red etc. La implementación es un poco más larga que otras alternativas, pero relativamente sencilla siguiendo los pasos de la documentación.

En cuanto a las actualizaciones, se puede observar en las release notes [40] que se hace una actualización de versión cada aproximadamente 6 meses, respecto a las actualizaciones o parches que puedan recibir esas versiones no hay información al respecto.

ManageEngine

Este software ofrece tanto una prueba gratuita de 30 días como unas herramientas gratuitas. La mayoría de las herramientas gratuitas solo sirven para analizar parámetros básicos de Windows o parámetros de WiFi. Las opciones que interesan para este proyecto son las de *OpManager* y *Network Configuration Manager*, las cuales para 50 equipos cuestan 1095 $ y 2395 $ respectivamente [41].

ManageEngine es compatible con plataformas Linux y especialmente con todas las versiones posteriores a Ubuntu 10.04, por lo que la compatibilidad está asegurada [42].

En cuanto a la personalización, el software tiene opciones muy parecidas a las anteriores, entre ellas, la medición de CPU y memoria, adaptadores de red etcétera. La implementación es realmente sencilla siguiendo los pasos de la guía que se puede encontrar en la documentación [43].

Por último, las actualizaciones son mensuales y muy frecuentes, tal y como se puede ver en las release notes [44]. En algunos meses se han llegado a realizar hasta tres actualizaciones de software.

Nagios Core

Entrando en el último grupo a analizar, en el de código abierto, Nagios Core es la primera alternativa. Al contrario de otros productos de Nagios, este software es completamente gratuito y compatible con plataformas Linux. En cuanto a la personalización, fácilmente se pueden encontrar tutoriales y guías en Internet para personalizar la apariencia y funciones. La implementación es sencilla si se siguen los pasos de la guía en la web oficial y respecto a actualizaciones, en los últimos años son habituales cada 4 meses, siendo 6 meses la vez que más tiempo ha transcurrido entre actualizaciones [45]. Teniendo en cuenta que es un software gratuito, es una frecuencia de actualización comprensible.

Icinga

La siguiente alternativa a analizar es Icinga, la cual se creó a partir de la base de Nagios. Al igual que Nagios, Icinga también es una herramienta gratuita, de código abierto y totalmente compatible con sistemas Linux. Respecto a la personalización, ofrece la posibilidad de editar la información que se muestra en pantalla de una forma muy sencilla y tiene una amplia variedad de opciones de monitorización. La implementación resulta sencilla siguiendo la guía de la página web oficial y, además, se actualiza regularmente, cada 2 meses para corregir errores y cada 5 meses aproximadamente para lanzar una versión con grandes cambios [46].

Zabbix

Para ir finalizando, se va a analizar el software Zabbix. Este software es totalmente gratuito y compatible con plataformas Linux, y concretamente con Ubuntu 20.04 LTS. En cuanto a la personalización, no ofrece posibilidad de realizar muchos cambios a la interfaz, pero presenta muchas opciones de monitorización. La implementación puede ser más tediosa que alguna alternativa anterior, pero tiene la ventaja de que el proceso está bien detallado en la guía de la

página web oficial [47]. Las actualizaciones son constantes en cuanto a parches de seguridad y bugs y se lanza una actualización de versión aproximadamente cada un mes.

MRTG

La última alternativa a analizar es MRTG, la cual también es gratuita y compatible con sistemas Linux. El software no se adapta, tiene una interfaz anticuada y no ofrece tantas posibilidades como las alternativas anteriores, por eso en el criterio de personalización no obtiene buenos resultados. Esto es debido a que no recibe muchas actualizaciones, pues no ha sido posible encontrar información al respecto y la última modificación de la página principal [27] fue en 2017, por lo que se sobreentiende que el programa tampoco ha recibido muchas actualizaciones desde entonces. Por esto mismo, tampoco se puede asegurar una buena compatibilidad con Ubuntu 20.04, puesto que no hay información al respecto. En cuanto a la implementación, es sencillo de implementar siguiendo la guía de la página web [48].

5.1.4. Preselección de las mejores alternativas

Después de evaluar los criterios citados anteriormente se ha creado la Tabla 1 de valoraciones. Se puede observar que las alternativas que más puntos han conseguido han sido Icinga, Nagios Core y Zabbix, respectivamente. Por tanto, son las tres candidatas a realizar las pruebas de rendimiento.

Tabla 1 - Preselección de alternativas de software de monitorización

Alternativas	Precio (25%)	Compatib. (20%)	Personaliz. (20%)	Implement. (7.5%)	Actualiz. (12.5%)	**TOTAL**
NewRelic	1	9	3	8	7	**4.125**
LogicMonitor	1	9	4	8	7	**4.325**
PRTG	1	1	6	1	7	**2.6**
SolarWinds	1	7	6	6	5	**3.925**
ManageEngine	1	9	6	8	7	**4.725**
Nagios Core	9	9	7	7	7	**6.85**
Icinga	9	9	8	7	8	**7.175**
Zabbix	9	9	5	6	8	**6.5**
MRTG	9	5	2	6	2	**4.35**

Pruebas de rendimiento

Icinga

Para instalar Icinga y su vista web, hay que tener instalado LAMP (Linux, Apache, MySQL y PHP). Se ha procedido a la instalación de esos componentes y después se ha seguido la guía de instalación [49] para el programa y su versión web. Tras realizar la instalación se han hecho unas medidas del rendimiento y se ha observado lo siguiente: La diferencia de carga de CPU y RAM cuando el programa Icinga está apagado o monitoreando es muy baja. Aumenta el uso de la CPU de forma notable solo cuando se inicia y después se mantiene corriendo en segundo plano sin apenas consumir recursos. El consumo de CPU es prácticamente nulo y el de la RAM ronda los 20-50 MB.

Donde sí se aprecia un consumo más elevado es a la hora de abrir el navegador para ver la monitorización con interfaz web. Mientras se está abriendo la homepage de Icinga el consumo de CPU se dispara hasta el 80-100% durante unos instantes, para después descender a los valores anteriores. El consumo de RAM se mantiene estable en 453 MB (icinga2 + Mozilla Firefox) todo el tiempo. Se estima que añadiendo más dispositivos a monitorizar (en el momento de las pruebas no hay ninguno) el consumo base de memoria de Icinga suba hasta los 50-80 MB, lo cual, son unas cifras muy buenas.

Nagios Core

Antes de proceder con la instalación de Nagios Core, hay que cumplir ciertos requisitos, entre ellos, tener un servidor web y PHP en la máquina. Se han instalado Apache y phpPgAdmin respectivamente. Después de eso, se procede a instalar la herramienta Nagios Core [50].

Los resultados son muy buenos, ya que con htop se puede ver que el consumo de memoria RAM es de menos de 100Mb y el de CPU es también muy bajo estando en reposo. De momento no se ha añadido ningún equipo para monitorizar, pero se estima que, aun añadiendo equipos, el consumo de recursos se mantenga bajo.

Zabbix

Se ha procedido a la instalación de LAMP antes de instalar la aplicación Zabbix, puesto que es un prerrequisito para asegurar el funcionamiento de la herramienta. Se ha procedido a la instalación y configuración y después de eso, a la puesta en marcha [51]. El servicio corre en segundo plano y es accesible mediante el servidor web Apache. El consumo de esta herramienta es algo más elevado de lo esperado. El uso de memoria RAM ronda los 250Mb y el uso de CPU se mantiene constante entre un 0% y 20% mientras que corre en segundo plano. Se determina que la prueba de rendimiento es favorable, teniendo en cuenta que el uso de recursos se encuentra en una horquilla razonable.

5.1.5. Selección de la mejor alternativa

Después de hacer un análisis en profundidad y teniendo en cuenta los criterios utilizados para la elección de las alternativas, se ha creado la Tabla 2, en la que se pueden ver las valoraciones. En esta tabla se han recalculado las valoraciones de las aplicaciones candidatas a las pruebas de rendimiento tras finalizar dichas pruebas. Como se puede observar, se ha decidido implementar la alternativa Nagios Core, puesto que ha sido la que más puntuación ha obtenido.

Tabla 2 - Selección de alternativas de software de monitorización

Alternativas	Precio (25%)	Compatib. (20%)	Personaliz. (20%)	Implement. (7.5%)	Actualiz. (12.5%)	Rendim. (15%)	**TOTAL**
Nagios Core	9	9	8	7	8	8	**8.375**
Icinga	9	9	7	7	7	8	**8.05**
Zabbix	9	9	5	6	8	7	**7.55**

5.2. Software de descubrimiento

En este apartado se va a tratar la problemática de buscar el software necesario para la función de **descubrimiento automático de servicios y máquinas**. Se van a detallar las alternativas disponibles y tras analizarlas mediante los criterios de evacuación, se procederá a elegir la que más puntuación obtenga en el análisis.

5.2.1. Descripción de alternativas

En este apartado, se van a analizar soluciones en propiedad y gratuitas. Antes de empezar a analizar cada una de las herramientas se van a dividir en dos subgrupos y se va a hacer una breve descripción de cada una de ellas.

5.2.1.1. En propiedad

Paessler PRTG Network Monitor

Esta herramienta se ha analizado anteriormente en las descripciones de alternativas del apartado anterior. Aparte de lo citado anteriormente, también es capaz de realizar un mapeo de la red con descubrimiento automático.

InterMapper

Esta herramienta es muy sencilla y fácil de usar. Se centra en conseguir un mapeo de red fácil de interpretar y proporciona datos de los equipos de la red, tales como la dirección IP, dirección MAC etc. También realiza escaneos periódicos para detectar nuevos dispositivos o dispositivos ya no conectados a la red. Otra opción que tiene es mandar alertas cuando detecta equipos fuera de servicio y monitorizar el tráfico de la red.

5.2.1.2. Gratuitas

Advanced IP Scanner

Esta herramienta [52] ofrece la posibilidad de escanear todos los dispositivos de la red y mostrar sus nombres, direcciones IP, direcciones MAC y fabricante. También ofrece la posibilidad de realizar el control remoto de equipos y dar acceso a carpetas compartidas, entre otros. Sin embargo, esta herramienta no da opción a realizar un mapa de red.

Nmap - Zenmap

Nmap es una herramienta desarrollada para realizar mapeos de redes y funciones de descubrimiento automático. Aparte de estas funciones, también ofrece la posibilidad de visualizar el mapa de red, saber el sistema operativo de los equipos conectados a la red, así como sus IP y direcciones MAC. La versión que se va a analizar va a ser Zenmap, la GUI de Nmap, para que sea más sencilla de utilizar [5].

OpenNMS

Está enfocada a obtener un mapeo y monitorización constante de la red. También da la opción a monitorizar cada uno de los equipos de la red y detectar problemas como altos tiempos de respuesta de servidores y hacer gráficas con dichos datos.

Lantopolog

Lantopolog es una herramienta muy simple y sencilla. Tiene pocas opciones, pero se ajusta a lo que se está buscando para este proyecto. Su función principal es hacer un mapa de red básico y dar la opción a mostrar junto a los equipos sus direcciones IP y MAC.

AngryIpScanner

Esta es una herramienta muy sencilla la cual escanea la red y muestra las IPs de los dispositivos que están conectados a la red, así como sus direcciones MAC, puertos abiertos y nombre de equipo, pero no da la opción a realizar un mapa de red.

5.2.2. Criterios de evaluación

Para asegurar que se realiza la mejor elección de las alternativas posibles, los criterios de evaluación serán los siguientes:

- **Precio**: Se analizarán las diferentes herramientas respecto a su coste económico y al tipo de servicio. La ponderación de este criterio será del 30%.
- **Rendimiento**: Se va a analizar la cantidad de recursos que consume la herramienta. A este criterio se le va a dar una ponderación del 20%.
- **Personalización**: Se van a analizar las opciones de personalización que nos ofrece cada herramienta, para así, saber hasta qué punto se puede adaptar a las necesidades de este proyecto. Aunque la alternativa no tenga muchas opciones de modificación, si se ajusta a las necesidades del proyecto recibirá buena nota. Este criterio va a tener una ponderación del 20%.

- **Dificultad de implementación:** Se analizará la dificultad a la hora de implementar la herramienta y la cantidad de información disponible en Internet para ello. La ponderación de este criterio será del 15%.
- **Frecuencia de actualización:** Se hará un estudio del historial de actualizaciones de cada alternativa buscando las release notes de cada una. La ponderación de este criterio será del 15%.

5.2.3. Análisis de las alternativas y los criterios de elección

La primera herramienta a evaluar es **PRTG**. Como ya hemos podido observar anteriormente, esta herramienta es de pago, y el importe asciende a 1300 € en la opción más sencilla. No es compatible con Ubuntu, por lo tanto, la implementación es imposible y en cuanto a la personalización, ofrece bastantes posibilidades tal como hemos podido ver anteriormente. Por último, cabe destacar que tiene una buena frecuencia de actualizaciones.

La siguiente alternativa es **InterMapper**. Esta herramienta antes era gratuita, pero ha pasado a ser de pago. Es imposible conseguir un listado de precios en la página oficial, pero a través de la información obtenida de diversos foros, se ha conseguido obtener un precio orientativo de 765 $. Está disponible para Windows, Linux y Apple, por lo que la compatibilidad está asegurada. En cuanto a la personalización, da la opción a crear ventanas de monitorización adaptadas al usuario y la implementación se puede realizar siguiendo las guías de la página oficial [53]. En cuanto a actualizaciones, se puede decir que tiene una buena frecuencia porque en el año 2019 la aplicación tuvo 5 actualizaciones y durante el 2020 ha recibido otras 3 actualizaciones [Revisado el 10 de octubre de 2020].

La siguiente alternativa a analizar es **Advanced IP Scanner**. Esta herramienta es gratuita y solo compatible con plataformas Windows, por lo tanto, los apartados de la compatibilidad e implementación no son favorables. Aún y así, al ser un programa muy sencillo y ligero sin una interfaz gráfica muy complicada, se va a valorar instalarlo en **PlayOnLinux**. Esta aplicación permite instalar programas desarrollados para Windows en Ubuntu. Con esto, si la aplicación se coloca entre las mejores a valorar, se procederá a hacer una prueba de rendimiento con ella. En cuanto a la personalización, no ofrece muchas posibilidades, pero se ajusta perfectamente a las funciones que se están buscando. Respecto a las actualizaciones, se han realizado regularmente cada año desde 2013, pero la última ha sido en 2018. Se entiende que después de dos años, el software ha dejado de tener soporte y no va a volver a recibir ninguna actualización.

La siguiente aplicación a analizar es **Zenmap**. Esta es una herramienta gratuita y de código abierto totalmente compatible con plataformas Linux y Windows. En cuanto a las opciones de personalización, no da muchas opciones al usuario, pero se adapta a las necesidades de este proyecto. La implementación resulta un poco larga, pero bastante sencilla siempre y cuando se siga la guía de la página oficial. En cuanto a las actualizaciones, hay mínimo una actualización grande al año la cual soluciona errores, optimiza la aplicación y añade funcionalidades.

OpenNMS es totalmente gratuita y de código abierto. Es compatible con Ubuntu 16.04 o superiores, por lo que la compatibilidad está asegurada. Da bastantes opciones de configuración, aunque no queda muy claro si da opción a conocer las direcciones físicas de los equipos de la red. En cuanto a la implementación, es bastante complicada. La guía es muy extensa y tiene muchos prerrequisitos, tales como tener ciertos programas instalados, ciertas

configuraciones hechas… El trabajo a realizar antes de proceder con la instalación del programa es considerablemente alto. En cuanto a actualizaciones, cabe destacar la buena frecuencia de actualizaciones que tiene, recibiendo mínimo una actualización cada mes.

La siguiente herramienta es **Lantopolog**. Esta aplicación es totalmente gratuita y de código abierto. La compatibilidad se ve comprometida, debido a que solo está disponible para plataformas Windows. Aún y así, en el caso de obtener buena puntuación, se valorará instalarla en PlayOnLinux para realizar una prueba de rendimiento. En cuanto a la personalización, no da muchas opciones, ya que es una herramienta muy sencilla. La implementación es sencilla, solo hay que ejecutar el instalador y realizar la configuración de la aplicación. Respecto a las actualizaciones, suele tener unas dos o tres al año, por lo tanto, tiene una buena frecuencia.

La aplicación **AngryIpScanner** es gratuita y de código abierto. La compatibilidad con Ubuntu es favorable y en cuanto a personalización, no ofrece muchas opciones, aunque cumple con los requisitos para este proyecto. La implementación es muy sencilla, solo hay que descargar el paquete de la página web oficial e instalarlo en el equipo. Las actualizaciones no se pueden determinar, porque no se ha encontrado documentación al respecto en la página web oficial.

5.2.4. Preselección de las mejores alternativas

Tras analizar las alternativas, se han valorado con una puntuación del 1 al 9 y se ha creado la Tabla 3. Las alternativas que más puntos han conseguido han sido Zenmap, AngryIpScanner y OpenNMS, respectivamente. Por tanto, son las tres candidatas a realizar las pruebas de rendimiento.

Tabla 3 - Preselección de alternativas de software de descubrimiento

Alternativas	Precio (25%)	Compatib. (20%)	Personaliz. (20%)	Implement. (7.5%)	Actualiz. (12.5%)	**TOTAL**
PRTG	1	1	7	1	7	2.8
InterMapper	1	8	7	7	7	4.65
Advanced IP Scanner	9	1	8	5	3	4.8
Zenmap	9	8	8	7	6	**6.725**
OpenNMS	9	8	7	2	8	**6.4**
Lantopolog	9	1	7	6	7	5.175
AngryIpScanner	9	8	8	8	5	**6.675**

Pruebas de rendimiento

Zenmap

A la hora de realizar la instalación en Ubuntu 20.04, ha surgido el problema de que no está disponible en los repositorios oficiales de Ubuntu debido a que Zenmap está desarrollada con Python 2, y este ha llegado a su final de vida a principios de 2020. Igualmente, se puede seguir instalando desde el repositorio oficial de Ubuntu 19.10 haciendo lo siguiente:

```
# mkdir -p ~/Downloads/zenmap
# cd ~/Downloads/zenmap
# wget
http://archive.ubuntu.com/ubuntu/pool/universe/p/pygtk/python-
gtk2_2.24.0-6_amd64.deb
# wget
http://archive.ubuntu.com/ubuntu/pool/universe/n/nmap/zenmap_7.80+d
fsg1-1build1_all.deb
# sudo apt install ./*.deb
```

Tras instalar la aplicación y ejecutarla con permisos de usuario y proceder a realizar un escaneo, se puede observar que el rendimiento sigue siendo muy estable y eficiente. El uso de CPU aumenta cuando se está escaneando y baja en periodos en los que no se está haciendo un escaneo. Se ha realizado una prueba de escaneo de puertos y en plena carga el uso de CPU se mantiene en un 30-50% durante unos segundos y después vuelve a bajar al valor estable de 0-5% de uso. Cuanto mayor sea la tarea a realizar o la red a escanear, más tiempo tardará el escaneo, y, por lo tanto, la carga de la CPU será mayor. El consumo de RAM es de aproximadamente 110 MB y el de red se mantiene sin variación (en la prueba de localhost), por lo que la puntuación es favorable. Se estima que en un escaneo en una red grande (el cual no hay medios para realizar) el consumo de red y el envío de paquetes se eleve, pero el consumo de bandwidth sea mínimo, en vista de que se usan paquetes de control de muy poco tamaño.

OpenNMS

Se ha procedido a instalar OpenNMS mediante la guía de Techrepublic.com [54]. En dicha guía aparece un script de GitHub donde se automatiza gran parte de la instalación de OpenNMS. Se ha ejecutado dicho script para facilitar la instalación. Por motivos de seguridad, si se decide implementar esta alternativa en los equipos del grupo de investigación, se procederá a instalar la aplicación mediante la guía y los repositorios oficiales de OpenNMS.

Una vez finalizada la instalación, se procede a realizar las pruebas de rendimiento. Para esta prueba se ha añadido un nodo a analizar a OpenNMS. Dicho nodo ha sido el propio equipo localhost y los resultados obtenidos han sido los siguientes: El uso de RAM es desorbitado comparando con las otras opciones. Con el programa iniciando y monitoreando, pero sin abrir el navegador web, el consumo de RAM del programa es de 1.3 GB y si tenemos en cuenta el navegador asciende a los 1.5 GB de RAM.

El consumo de CPU asciende durante unos segundos al 50%, pero después desciende hasta el 5-10% y se mantiene estable a excepción de un núcleo que se mantiene en un 20% de uso constante.

Tras completar las pruebas y detener OpenNMS (systemctl stop opennms) la cantidad de RAM usada baja de 2.8 GB a 1.3 GB y el procesador se mantiene estable en el 5% de uso. Por lo tanto, queda descartada cualquier alteración provocada por alguna actualización o algún

subproceso del sistema, el consumo excesivo de RAM es provocado por OpenNMS. Las pruebas han dado un resultado no favorable para OpenNMS.

AngryIPScanner

Un requisito de esta aplicación es que estén instalados OpenJDK o Java 8 en la máquina. Para ello, se ha instalado OpenJDK-8 y se ha verificado la instalación:

```
# sudo apt install -y openjdk-8-jre
# java -version
openjdk version "1.8.0_265"
openJDK Runtime Environment (build 1.8.0_265-8u265-b01-
0ubuntu2~20.04-b01)
openJDK 64-Bit Server VM (build 25.265-b01, mixed mode)
```

Después de eso se ha procedido a instalar AngryIpScanner descargando e instalando el paquete de la página oficial [55]. El rendimiento es favorable: Solo está en ejecución cuando se activa el escaneo, no corre en segundo plano con escaneos periódicos. Se ha realizado un escaneo de IPs y puertos de la red 10.0.2.0/24. Se han analizado 254 IPs y sus respectivos puertos durante un escaneo el cual ha durado 11 segundos. El uso de CPU ronda el 20-40%, pero hay un núcleo el cual su uso se dispara al 90% por razones que se desconocen. Se piensa que puede ser por falta de optimización y que la aplicación no esté diseñada para hacer uso de todos los núcleos. El uso de memoria RAM es muy bajo, menos de 150 MB. Esto se ha podido comprobar gracias a la herramienta *htop* filtrando por nombre de aplicación *ipscan*.

5.2.5. Selección de la mejor alternativa

Después de hacer un análisis en profundidad, se ha creado la Tabla 4 y se ha decidido implementar la alternativa Zenmap, pues ha sido la que mayor puntuación ha obtenido.

Tabla 4 - Selección de alternativas de software de descubrimiento

Alternativas	Precio (25%)	Compatib. (20%)	Personaliz. (20%)	Implement. (7.5%)	Actualiz. (12.5%)	Rendim. (15%)	**TOTAL**
PRTG	1	1	7	1	7	1	2.95
InterMapper	1	8	7	7	7	1	4.8
Adv. IP Scanner	9	1	8	5	3	1	4.95
Zenmap	9	8	8	7	6	8	**7.925**
OpenNMS	9	8	7	2	8	4	**7**
Lantopolog	9	1	7	6	7	1	5.325
AngryIpScan.	9	8	8	8	5	8	**7.875**

5.3. Sistema Operativo

En este apartado se va a valorar cuál es el mejor sistema operativo a utilizar en este proyecto. Para ello, se va a proceder de forma similar a lo que se ha visto en los apartados anteriores. Se van a definir unos criterios de evaluación y mediante un sistema de puntos se va a elegir la mejor alternativa.

5.3.1. Descripción de alternativas

En este subapartado se van a analizar los diferentes sistemas operativos que se podrían utilizar en este proyecto. No se va a profundizar mucho en su descripción, debido a que son de sobra conocidos en el ámbito de las telecomunicaciones.

Windows 10

Es el sistema operativo más utilizado en el mundo entero. Se utiliza de forma generalizada en la gran mayoría de hogares y empresas por sus grandes capacidades y sencillez de uso. Se lanzó en 2015 de forma oficial y ha ido añadiendo funciones y mejorando continuamente desde entonces.

Ubuntu 20.04 LTS

Ubuntu es un sistema operativo basado en la arquitectura Linux. Actualmente está disponible la versión 20.10, pero se va a valorar usar la 20.04 LTS, puesto que es la que tiene soporte a largo plazo. Esta versión ofrece soporte y actualizaciones durante 5 años a partir de su lanzamiento, o lo que es lo mismo, hasta abril de 2025.

CentOS

CentOS es una variación de las distribuciones Linux la cual está desarrollada a partir del código fuente de RHEL (Red Hat Enterprise Linux), publicado por *Red Hat*. La principal diferencia es que en CentOS no hay ningún rastro de ninguna marca comercial, se han eliminado todos los logos, marcas y referencias que son propiedad de *Red Hat*. Se basa en ofrecer un sistema gratuito y de muy buena calidad para poder utilizarlo en equipos empresariales.

5.3.2. Criterios de evaluación

- **Adaptabilidad:** Se analizarán los diferentes SO por su capacidad para modificarlos buscando la mayor adaptación al proyecto y la cantidad de información en Internet para ese fin. La ponderación de este criterio será del 30%.
- **Precio:** Se analizarán los diferentes SO por su coste económico. La ponderación de este criterio será del 25%.
- **Uso en el laboratorio**: Se valorarán los SO por la cantidad de instalaciones de dicho SO en el laboratorio I2T. De esta forma, se buscará la total compatibilidad y estandarización entre equipos, evitando añadir sistemas diferentes a los que ya están en funcionamiento. La ponderación de este criterio será del 45%.

5.3.3. Análisis de las alternativas y los criterios de elección

Windows 10 es el sistema operativo más extendido para el uso personal y empresarial. Tiene una gran variedad de funciones y opciones las cuales se adaptan a prácticamente cualquier uso que se le quiera dar. Cuenta con un precio de licencia considerablemente alto y no hay prácticamente ningún equipo con este sistema operativo en el laboratorio I2T.

Ubuntu 20.04 LTS es un sistema operativo basado en Linux, el cual tiene un potencial enorme para realizar cambios en él y adaptarse a todas las necesidades del usuario o proyecto. Su uso es totalmente gratuito, aunque también hay opción a donar el importe que cada usuario vea adecuado. La gran mayoría de los equipos del laboratorio utilizan este SO, por lo que si se quiere conseguir una compatibilidad total, este SO gana mucho peso.

CentOS está basado en Linux, y al igual que Ubuntu, tiene un gran potencial para adaptarse a las necesidades de cada usuario. Su uso es totalmente gratuito, pero se ha notificado que a partir del 31 de diciembre de 2021 se le va a dejar de dar soporte a la última versión que se ha lanzado, la CentOS 8. A partir de ese día se continuará dando soporte a CentOS Stream hasta el 31 de mayo de 2024. Para las empresas que usen CentOS Red Hat recomienda adquirir RHEL (Red Hat Enterprise Linux) [56], una versión con soporte de pago. La versión de CentOS 7 seguirá teniendo soporte hasta 2024, pero como este proyecto se quiere realizar a largo plazo, no se va a tener en cuenta dicha versión.

Por lo tanto, para el análisis de esta alternativa se va a tener en cuenta la actual versión de CentOS 8, su tiempo de vida y su coste de migración a RHEL. El precio de una licencia RHEL Workstation asciende a 179 $, el cual, es un coste considerable para un solo equipo. Además, en el laboratorio I2T no hay ningún equipo que use CentOS, por lo tanto, en el apartado de uso obtiene una puntuación muy baja.

5.3.4. Selección de la mejor alternativa

Tras realizar el análisis anterior, se ha creado la Tabla 5 de evaluación que se puede observar más abajo. La alternativa que más puntuación ha obtenido ha sido **Ubuntu 20.04 LTS**, por lo que se ha decidido utilizar este SO en el proyecto.

Tabla 5 - Selección de alternativas de sistema operativo

Alternativas	Adaptabilidad (30%)	Precio (25%)	Uso en I2T (45%)	TOTAL
Windows 10	7	4	3	4.45
Ubuntu 20.04 LTS	9	9	7	**8.1**
CentOS	9	4	1	4.15

5.4. Mecanismo de despliegue de la solución

En este apartado se va a valorar la mejor alternativa para instalar el sistema operativo. Para ello, se va a proceder al igual que en los apartados anteriores. Se van a definir unos criterios de evaluación y se va a elegir la mejor alternativa mediante un sistema de evaluación por puntos.

5.4.1. Descripción de alternativas

En este subapartado se van a analizar los diferentes tipos de instalación que se pueden realizar para el propósito de este proyecto. Se va a describir cada una de forma breve y a analizarlas de forma objetiva.

Equipo físico

Es la opción que más se ha utilizado durante décadas. Hasta hace pocos años, lo más habitual era realizar las instalaciones nuevas en equipos físicos dedicados. Por ejemplo, si una empresa tenía necesidad de tener un servicio web, de correo y DNS, instalaba 3 servidores físicos. Uno dedicado exclusivamente al servicio web, otro dedicado al correo electrónico y el último al DNS. Esta no es una opción muy eficiente, ya que gran parte del potencial del equipo físico se desaprovecha.

Máquina virtual (VM)

Durante los últimos años es la opción que más fuerza está ganando en las nuevas instalaciones y también en las migraciones de instalaciones antiguas. Esta opción se basa en utilizar varias máquinas virtuales sobre un equipo físico. Esto se consigue gracias a los hipervisores y a los programas de virtualización (se van a analizar más adelante en el apartado 8.1. Virtualización).

5.4.2. Criterios de evaluación

- **Eficiencia:** Se analizará cada alternativa basándose en el aprovechamiento de las capacidades de cada equipo físico. Cuanto más eficiente sea su uso, mayor puntuación se obtendrá.
- **Actualización:** Se va a analizar cada alternativa respecto a la facilidad que ofrece para realizar una actualización. Cuantas más facilidades y velocidad ofrezcan a la hora de realizar la actualización, mayor puntuación obtendrán.
- **Coste**: Se va a analizar el coste de implementación de cada alternativa. Cuanto mayor ahorro supongan, mayor puntuación obtendrán.

5.4.3. Análisis de las alternativas y los criterios de elección

La **instalación en un equipo físico** nos da independencia de cada servicio. Si un equipo físico falla, solo falla un servicio, pero en cuanto a eficiencia no obtiene una puntuación alta, porque se desaprovecha gran parte de las capacidades del equipo físico. En cuanto a actualizaciones, ofrecen una facilidad relativamente grande, al igual que la mayoría de equipos. En cuanto a costes, supone un desembolso inicial elevado, puesto que para cada servicio que se quiera instalar habría que adquirir un servidor. Además, a largo plazo el coste también es alto, debido a que hay que mantener varios servidores y pagar las facturas eléctricas de varios equipos.

Por otra parte, la **instalación en una máquina virtual** nos ofrece una eficiencia muy superior, dado que podemos tener varias máquinas virtuales corriendo al mismo tiempo en el mismo servidor físico. Esto significa que estamos aprovechando de una forma más eficiente los recursos del equipo físico, haciendo uso de todo su potencial. A la hora de hacer actualizaciones esta alternativa es prácticamente igual a la anterior, pues ofrece las mismas facilidades que el equipo físico para el fin de este proyecto en concreto. Para las NFVs ofrece más facilidades y ventajas, pero eso se va a tratar más adelante en el apartado 8.3. NFV.

5.4.4. Selección de la mejor alternativa

Tras realizar el análisis anterior, se ha creado la Tabla 6 de evaluación que se puede observar más abajo. **La alternativa que más puntuación ha obtenido ha sido la VM,** por lo que se ha decidido realizar este tipo de instalación en el proyecto.

Tabla 6 - Selección de alternativas de mecanismo de despliegue

Alternativas	Eficiencia (45%)	Actualización (25%)	Coste (30%)	TOTAL
Equipo físico	5	8	6	6.05
VM	9	8	9	**8.75**

5.5. Almacenamiento y visualización de logs

En este apartado se van a analizar las diferentes opciones para visualizar los datos de rendimiento y logs de las máquinas que se estén monitorizando. Para ello, se van a seguir los mismos pasos que se han visto en apartados anteriores. Se va a empezar describiendo cada una de las alternativas, después se van a definir unos criterios de evaluación y finalmente, se va a elegir la mejor alternativa mediante un sistema de puntos.

5.5.1. Descripción de alternativas

En este subapartado se van a analizar y explicar las diferentes alternativas disponibles para realizar la visualización de los logs. Se van a analizar tanto herramientas individuales como grupos de herramientas que de forma conjunta cumplan el objetivo final.

PNP4Nagios

PNP4Nagios [57] es un plugin gratuito y de código abierto desarrollado para Nagios Core. Se encarga de recolectar los datos de rendimiento de los equipos que se están monitorizando y realizar gráficas con ellos, para que de esta forma, sean fáciles de analizar. Los datos de rendimiento los obtienen los plugins que se encargan de la monitorización, se almacenan en una base de datos temporal RDD (Round Robin Database) durante un periodo de tiempo limitado y PNP4Nagios los procesa para realizar gráficos.

Es muy sencillo de instalar y tiene una interfaz simple y fácil de entender. Puede mostrar gráficos con datos a muy corto plazo o hasta a 90 días.

Nagiosgraph

Nagiosgraph [58] es un plugin muy similar a PNP4Nagios. Se encarga de analizar los datos de rendimiento y con ellos realizar gráficas para el usuario. También es de código abierto y totalmente gratuito. Dispone de una interfaz sencilla y ligera y permite mostrar datos con diferente margen temporal.

Grafana

Grafana [59] es una aplicación que te permite visualizar datos temporales de diferentes maneras. Se pueden visualizar desde los valores numéricos de ciertos parámetros hasta gráficos temporales completos. Tiene una gran cantidad de gráficos disponibles, entre ellos, los temporales, en barra, diagramas… Además, ofrece la posibilidad de tener más de una fuente de datos.

Una de sus funciones más destacables es el modo TV. Este modo se usa sobre todo cuando hay muchos datos que mostrar y no se tiene suficiente espacio en el monitor para ello. Lo que se hace en este caso, es ir mostrando los gráficos de forma periódica de forma repetida, actualizando los datos cada vez que aparecen en pantalla. Además, es totalmente gratuita y tiene una interfaz con infinidad de opciones.

MRTG

MRTG [60] es una aplicación diseñada para monitorizar el tráfico de la red mediante SNMP. Su funcionamiento es sencillo: contabiliza los bytes de entrada y salida de cada interfaz y genera un gráfico con ello. Los datos que recaba los guarda en una base de datos temporal (RRD) y posteriormente los usa para generar gráficos para mostrar al usuario.

PNP4Nagios + Grafana

Otra opción interesante es la de implementar de forma conjunta PNP4Nagios y Grafana, y así, unir las capacidades de ambas herramientas. Mediante este conjunto, se puede aprovechar la potencia gráfica y opciones de personalización de Grafana junto con la base de datos temporal y las capacidades de PNP4Nagios y de esta forma, conseguir gráficos muy elaborados y personalizables con un consumo de recursos muy bajo.

ELK (ElasticSearch Logstash y Kibana)

Kibana [61] es una aplicación que permite representar de manera gráfica los datos de ElasticSearch. Es totalmente gratuita y tiene una interfaz actual. Con Kibana no solo se pueden visualizar los gráficos generados con los datos de ElasticSearch, sino que además, puede funcionar como interfaz de usuario para administrar el Elastic Stack.

5.5.2. Criterios de evaluación

- **Personalización:** Se analizarán las diferentes alternativas por su capacidad de personalización. Cuanto mayor sea la capacidad de adaptación a los datos que se quieren mostrar, mayor será la puntuación obtenida. La ponderación de este criterio será del 35%.
- **Estética**: Se valorará de forma positiva la inclusión de una estética actual y moderna. La ponderación de este criterio será del 15%.

- **Funcionalidades extra**: Se valorará de forma positiva la inclusión de funcionalidades extra aparte de la función principal. La ponderación de este criterio será del 10%.
- **Instalación:** Se analizará cada alternativa para concretar el nivel de dificultad para realizar la instalación y las posteriores modificaciones. La ponderación de este criterio será del 20%.
- **Precio:** Se analizará cada alternativa en función del coste económico que esta suponga. Cuanto mayor coste económico tenga, menor puntuación obtendrá. La ponderación de este criterio será del 20%

5.5.3. Análisis de las alternativas y los criterios de elección

PNP4Nagios es una herramienta ligera pero potente. Obtiene todos los datos de rendimiento de los plugins de Nagios Core, los guarda en una base de datos temporal y posteriormente los utiliza para generar gráficos consumiendo muy pocos recursos. Su capacidad de personalización de gráficos no es muy alta y la estética es anticuada. La instalación es relativamente sencilla siguiendo la guía de la página oficial de Nagios Core y además, es totalmente gratuita.

Nagiosgraph es una herramienta muy parecida a PNP4Nagios, tienen multitud de similitudes. Esta también obtiene los datos de rendimiento, los guarda en una RDD y los usa para generar gráficos. Es ligera, gratuita, consume muy pocos recursos y su instalación es sencilla. La gran pega es que tiene una estética anticuada.

Grafana es una herramienta muy potente y muy personalizable. Ofrece infinidad de opciones a la hora de elegir las fuentes de datos y de generar los gráficos, pudiendo personalizar prácticamente todos los parámetros y ajustes de los gráficos. Esto es a costa de un consumo de recursos más elevado que sus competidores directos. Aún y así, está muy optimizada y la diferencia de consumo de recursos apenas tiene impacto en un equipo actual. Es totalmente gratuita y cuenta con la funcionalidad de TV. En este modo, se van mostrando los gráficos que se elijan de forma periódica en el monitor. Esto es muy útil si se tienen muchos equipos monitorizados y un solo monitor. Existe una guía oficial para realizar la instalación, es un poco más tediosa que la de sus competidores, pero no es complicada. Además, la estética es muy actual y es totalmente gratuita.

MRTG es una aplicación muy simple, ligera y sencilla. Se basa en monitorizar el ancho de banda usado en los equipos que se están monitorizando. Tiene una estética muy anticuada y prácticamente no dispone de opciones a la hora de personalizar los gráficos. Es sencilla de instalar y totalmente gratuita.

El grupo **PNP4Nagios + Grafana** une las mejores características de cada una de las herramientas. PNP4Nagios se encarga de recopilar los datos y almacenarlos en la RRD. Grafana, por otro lado, se encarga de pedir los datos a la RRD y con ellos generar gráficos. La instalación de ambas herramientas es sencilla, hay una guía oficial de nagios con los pasos muy bien detallados y no resulta nada complicada. Aún y así, hay que destacar que la instalación es considerablemente más laboriosa que las alternativas anteriores, ya que hay que instalar dos herramientas, configurarlas para que trabajen juntas y después crear todos los paneles y gráficos en Grafana.

Lo bueno de realizar esta instalación es que las herramientas son totalmente gratuitas, Grafana ofrece un nivel de personalización muy elevado junto con una estética actual y además incluye la función TV para ver de forma periódica varios gráficos en un monitor.

Por último, el grupo **ELK** realiza funciones similares al anterior, pero con ciertas diferencias. Estas aplicaciones recolectan los datos de diferentes servicios, los unifican en un mismo formato y generan reportes con dichos datos. Para ello, se suelen instalar tres herramientas juntas, las que componen el ELK (ElasticSearch Logstash y Kibana), ya que están pensadas para instalarse juntas.

ElasticSearch es una base de datos distribuida, Logstash se encarga de recolectar y transformar los logs a un mismo formato y Kibana se encarga de realizar la visualización de esos logs y eventos.

Ofrecen buen nivel de personalización, una estética actual y son totalmente gratuitas. La pega, es que se recomienda instalarlas con Nagios Log Server, la cual es de pago.

5.5.4. Selección de la mejor alternativa

Después de haber analizado las alternativas anteriores, se ha creado la Tabla 7 de evaluación que se puede observar más abajo. La alternativa que más puntuación ha obtenido ha sido el grupo **PNP4Nagios + Grafana**, por lo que se ha decidido utilizar esta herramienta en el proyecto.

Tabla 7 - Selección de alternativas de almacenamiento y visualización de logs

Alternativas	Personalización (35%)	Estética (15%)	Func. Extra (10%)	Instalación (20%)	Precio (20%)	TOTAL
PNP4Nagios	6	6	4	9	9	**7.00**
Nagiosgraph	6	5	4	9	9	**6.85**
Grafana	8	9	7	7	9	**7.20**
MRTG	4	4	5	8	9	**5.90**
PNP4Nagios + Grafana	9	9	8	7	9	**8.50**
Kibana + ElasticSearch	8	9	5	7	1	**6.25**

6. ANÁLISIS DE RIESGOS

En este apartado se van a analizar los posibles riesgos a la hora de realizar este proyecto. Se van a identificar los problemas y obstáculos que pueden aparecer y también la importancia de las consecuencias que estos puedan ocasionar. Gracias a esto, se sabrá a qué problemas u obstáculos hay que darles más importancia, y por consecuencia, hacer más trabajo de prevención en ellos, ya que son los que más repercusión van a tener en el desarrollo del proyecto.

Para realizar este análisis, se han identificado varias problemáticas, las más importantes han sido las siguientes:

- Funcionamiento incorrecto del sistema (1)
- No cumplir con los plazos establecidos (2)
- Cambios en los requerimientos del proyecto (3)
- Fallos en el diseño (4)
- Responsabilidades y tareas confusas (5)

6.1. Definición de riesgos

En este apartado se van a detallar con una breve descripción los posibles riesgos que se han identificado a la hora de realizar este proyecto.

Funcionamiento incorrecto del sistema

Este es un problema crítico, ya que si al final del proyecto se determina que el sistema tiene un funcionamiento incorrecto, esto implicara que no se han cumplido los objetivos. Aunque este problema sea muy grave, tiene una probabilidad muy baja de que ocurra, ya que durante todo el proyecto se va a ir comprobando que cada paso o funcionalidad implementada trabaja de manera correcta. Es decir, se va a realizar la instalación de forma gradual y modular, comprobando que cada funcionalidad tiene un correcto funcionamiento al terminar de instalarla.

No cumplir con los plazos establecidos

Este también es un problema grave, ya que el proyecto hay que entregarlo dentro de unos plazos concretos y es de suma importancia que esté totalmente finalizado para la fecha de entrega. En este caso, dada la complejidad del proyecto, la probabilidad de que este problema ocurra es considerablemente más alta que el anterior, concretamente, se define una probabilidad media-alta para este problema.

Cambios en los requerimientos del proyecto

Este es un problema que tiene un gran impacto en el desarrollo del proyecto. Es importante definir los requerimientos en la fase inicial del proyecto, antes de hacer el diseño de la solución, ya que cualquier cambio una vez empezada la implementación puede suponer grandes demoras en los plazos. Este es un problema con un impacto medio y una probabilidad media, ya que en todos los proyectos puede ocurrir que surjan nuevas ideas o requerimientos durante el desarrollo de los mismos. Sin duda, esta es una problemática a tener muy en cuenta.

Fallos en el diseño

A la hora de realizar un diseño, es importante verificarlo varias veces, ya que si hay algún error en él y no se detecta hasta la fase de implementación, este error puede causar demoras en el plazo del proyecto, o incluso, que haya que rediseñar completamente la solución. Es una

problemática grave, pero de probabilidad baja, ya que se van a invertir muchas horas en el diseño de la solución y se va a verificar varias veces. Aun y así, es un riesgo a tener en cuenta.

Responsabilidades y tareas confusas

En este proyecto en concreto este es un problema de baja probabilidad y repercusión. Los papeles de cada uno están muy bien definidos y todos los miembros del equipo saben cuáles son sus responsabilidades.

6.2. Conclusiones

Tras analizar todos los problemas anteriores, se ha realizado la Tabla 8, en la que se mide la relación de una problemática entre probabilidad e impacto en el proyecto. Para ello, se han definido unos valores para la probabilidad y el impacto, con los que se van a analizar las problemáticas dependiendo de su probabilidad e impacto. La tabla es la siguiente:

Tabla 8 - Análisis de riesgos

		IMPACTO				
		Muy bajo (0.05)	Bajo (0.1)	Moderado (0.2)	Alto (0.4)	Crítico (0.8)
PROBABILIDAD	Muy baja (0.1)					**1** 0.08
	Baja (0.3)		**5** 0.03		**4** 0.12	
	Media (0.5)			**3** 0.1	**2** 0.2	
	Alta (0.7)					
	Muy alta (0.9)					

Como se puede observar, a la problemática que más atención y tiempo hay que dedicar es *no cumplir con los plazos establecidos*, ya que es la única que está en la zona de máxima atención. Otras problemáticas a tener muy en cuenta son, en orden de prioridad ascendente, el *funcionamiento incorrecto del sistema*, los *cambios en los requerimientos del proyecto* y los *fallos en el diseño*. Por último, la problemática de menos riesgo es la única que está en la zona de baja atención, *responsabilidades y tareas confusas*.

7. DESCRIPCIÓN DE LA SOLUCIÓN

En este trabajo se busca implementar un sistema de monitorización y descubrimiento de equipos de red de forma virtualizada. Se va a instalar una máquina virtual en el laboratorio de investigación I2T la cual se va a encargar de monitorizar la red de dicho laboratorio y también de realizar la tarea de descubrimiento automático de equipos.

Dicha máquina virtual se va a implementar como una VNF dentro de la red del laboratorio I2T, la cual va a aportar la función de monitorización y descubrimiento de la red. Esta VNF se va a añadir a los servicios ya existentes en el laboratorio y va a estar desplegada bajo la implementación OSM.

En la *Figura 10* se puede ver el esquema de alto nivel de la instalación de Nagios que se va a implementar.

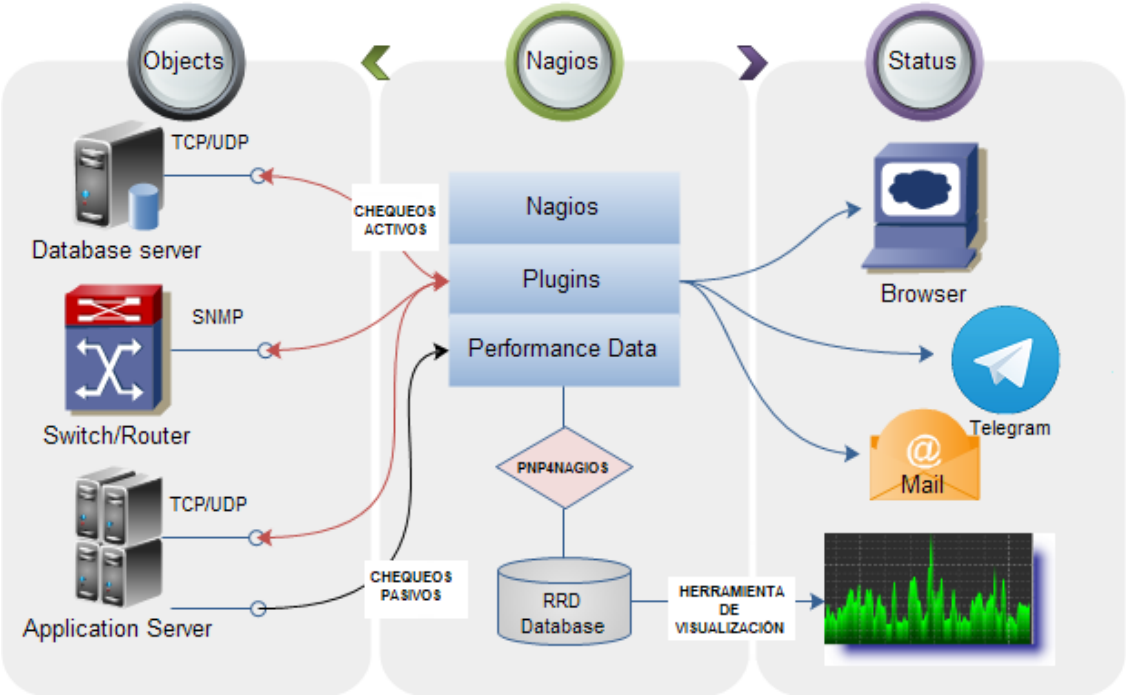

Figura 10 - Funcionamiento de Nagios Core [62]

Tal y como se puede observar en la *Figura 11*, la instalación que se va a realizar tiene diferentes funciones y módulos. Entre las funciones más destacadas se encuentran los chequeos y las notificaciones.

Las alertas se mostrarán en todo momento en la interfaz web de Nagios Core, y además, también se van a enviar notificaciones mediante correo electrónico y Telegram.

En las siguientes líneas se van a describir los módulos utilizados en esta instalación. Se van a detallar los tipos de chequeos y notificaciones que se van a utilizar, así como los módulos de Nagios y los protocolos que participan en la tarea de monitorización.

7.1. Monitorización de red

7.1.1. Chequeos activos

Se les llama chequeos activos a todos los chequeos que inicia el servidor de Nagios Core. El servidor pregunta ciertos parámetros al equipo monitorizado, y este, le responde con los datos que le ha solicitado. Se dice que son activos, ya que si el servidor no pregunta, no recibe ningún dato por parte del equipo monitorizado, es decir, es necesario que el servidor haga la pregunta para poder monitorizar.

Un ejemplo de este tipo de chequeos activos son los que se hacen mediante el protocolo SNMP y mediante el módulo NRPE de Nagios. El protocolo SNMP se usa para preguntar a un equipo de red acerca de unos parámetros en concreto, como pueden ser, el estado de sus puertos, temperatura, carga de CPU etcétera. Al recibir la pregunta, el equipo monitorizado responde con los datos solicitados al servidor, y de esta forma, se da la comunicación entre los dos.

Al igual que con los chequeos con SNMP, los chequeos mediante NRPE también los inicia el servidor de Nagios. Mediante el módulo NRPE el servidor lanza una pregunta al equipo monitorizado, y este, le responde con los datos solicitados. Habitualmente este módulo se usa para monitorizar equipos Linux, y es altamente personalizable y adaptable a los requisitos del usuario.

La *Figura 11* muestra de forma gráfica el funcionamiento de los chequeos activos que se acaban de explicar.

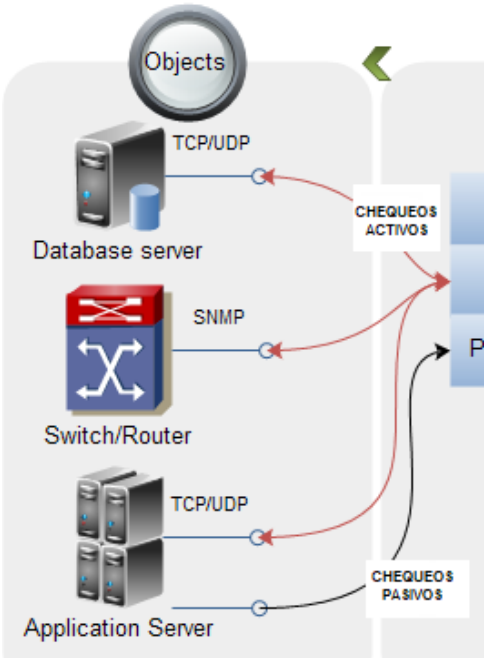

Figura 11 - Chequeos activos Nagios Core [62]

7.1.2. Chequeos pasivos

La principal diferencia de estos chequeos frente a los anteriores es que en este caso el servidor Nagios no tiene que iniciar la comunicación. El envío de información por parte del equipo monitorizado se hace de forma automática de forma periódica mediante Crontab.

Un ejemplo de este tipo de chequeos son los chequeos mediante NSCA. El equipo monitorizado manda la información que se le ha determinado en la configuración de forma periódica al servidor, reduciendo así el tráfico de la red y la carga del servidor. Este tipo de chequeos se va a ver más detalladamente en las siguientes páginas.

En la *Figura 12* se puede observar el funcionamiento de los chequeos pasivos. Se puede apreciar que el servidor manda directamente la información al software Nagios Core sin que este le haya solicitado dicha información.

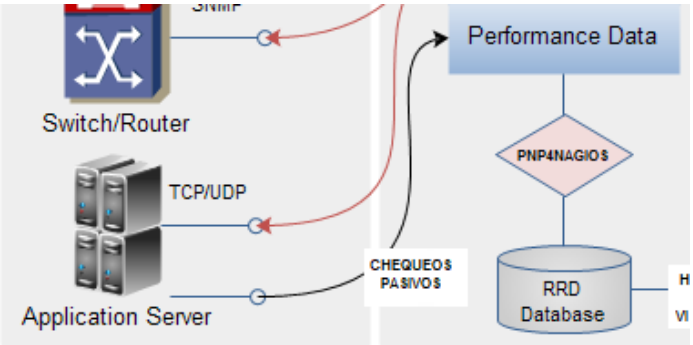

Figura 12 - Chequeos pasivos Nagios Core [62]

7.1.3. Notificaciones

Las notificaciones son un apartado importante a la hora de realizar una monitorización, puesto que cuando en el laboratorio no hay nadie o cuando se está trabajando en algún proyecto importante, normalmente no se suele observar periódicamente la pantalla principal de los sistemas de monitorización. En el caso en el que haya un error, al no haber ninguna notificación, es posible que los integrantes del laboratorio de I2T no se den cuenta del error de forma inmediata, con todos los problemas que ello supone: caídas de servicios prolongadas, errores evitables en los equipos etcétera.

Para evitar esos problemas, se va a implementar un sistema de notificaciones en tiempo real que consta de dos sistemas. El primero es la habitual notificación mediante correo electrónico, la cual se realiza de forma inmediata y es una buena opción. El problema radica en que hoy en día la cantidad de correos electrónicos que recibe una persona promedio a lo largo del día es inmensa, y habitualmente no se les suele prestar atención hasta terminar la jornada laboral o directamente se van a la carpeta de spam o se ignoran. Dada la situación actual, es muy probable que uno de los correos de notificación de Nagios se pase por alto o se ignore, con las consecuencias negativas que ello puede acarrear.

Para evitar estos problemas, se va a implementar un segundo tipo de notificación mediante la aplicación de mensajería instantánea Telegram. Para ello, se va a utilizar un bot de Telegram llamado *BotFather* y también la API de Telegram. En la *Figura 13* se puede ver de forma sencilla el funcionamiento de las notificaciones que se van a implementar en este proyecto. La instalación detallada de estas notificaciones se verá más adelante en el apartado *9.INSTALACIÓN Y CONFIGURACIÓN.*

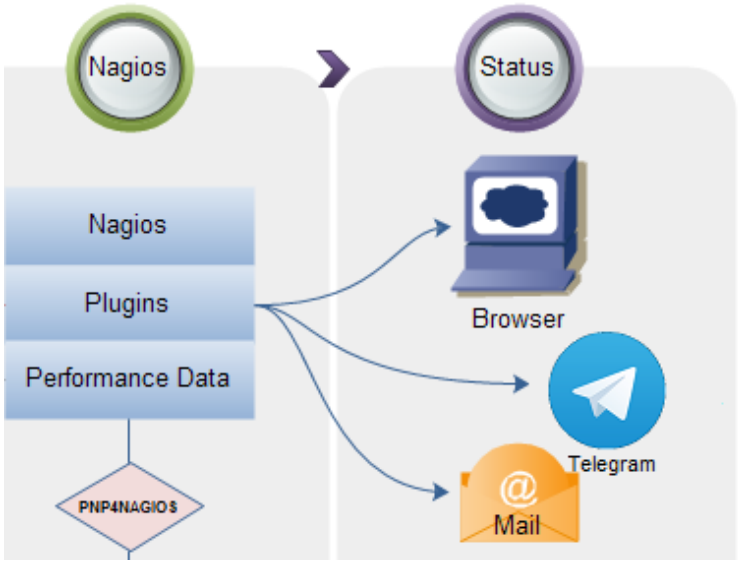

Figura 13 - Notificaciones Nagios Core [62]

7.1.4. Protocolo SNMP

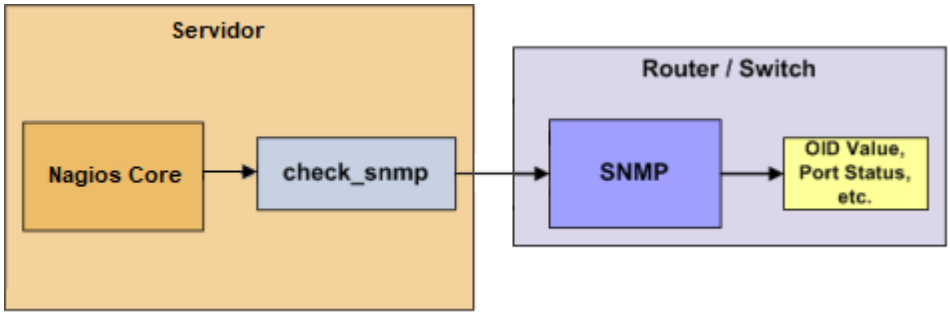

Figura 14 - Funcionamiento de check_snmp

Para conocer el estado de los equipos de red se pueden usar dos técnicas. La primera es hacer ping a los puertos del equipo que se quiera monitorizar para saber su estado. La segunda y más elaborada, es usar el protocolo SNMP (Simple Network Management Protocol). Mediante este protocolo se puede monitorizar multitud de datos de diferentes equipos de red, como por ejemplo, un router, switch o hub. Entre esos datos podemos encontrar la tasa de perdida de paquetes, el estado de los puertos, el número identificador del producto, la tasa de tráfico etcétera.

Teniendo en cuenta que los switches y routers son una parte muy importante de cualquier red, es recomendable tenerlos monitorizados para conocer su estado. La monitorización mediante SNMP se realiza de forma activa, es decir, el servidor Nagios Core es el encargado de realizar el comando *check_snmp* y recibir la información que le devuelva el equipo de red [3], tal y como se puede ver en la *Figura 14*.

7.1.5. Módulos de Nagios

En este subapartado se van a analizar los diferentes módulos que se pueden usar con Nagios para monitorizar los equipos remotos. Los módulos más conocidos de Nagios para este fin son los módulos NSCA y NRPE. Se va a explicar cada uno de ellos de forma detallada en las siguientes líneas.

7.1.5.1. NSCA

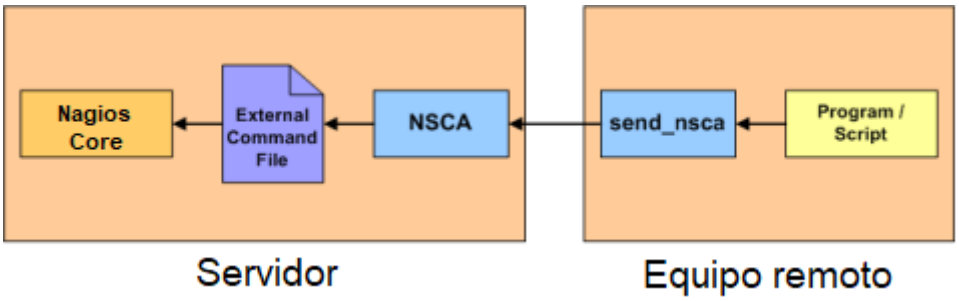

Figura 15 - Funcionamiento de NSCA [1]

El servicio NSCA (Nagios Service Check Acceptor) se utiliza para que los equipos remotos manden información de forma periódica al servidor de Nagios Core. Tal y como se puede apreciar en la *Figura 15,* Estos chequeos se realizan de forma pasiva, es decir, el servidor no manda ninguna señal al equipo que se quiere monitorizar, el propio equipo le manda de forma periódica la información al servidor.

Una ventaja de esta característica es que se reduce considerablemente la carga del servidor principal, ya que tan solo se tiene que centrar en recibir la información y procesarla. Un uso muy habitual de este servicio es monitorizar los equipos que habitualmente no se encuentran en línea o en funcionamiento. De esta forma, Nagios Core no va a emitir ninguna señal de alerta o error porque no está monitorizando esa máquina de forma activa.

Para utilizar este servicio tan solo hay que configurarlo en la máquina remota, no hace falta modificar nada en el servidor. Para realizar la correcta configuración del servicio NSCA, es necesario copiar los archivos de configuración en la máquina remota y modificarlos para que en las configuraciones aparezca la dirección IP del servidor Nagios Core. De esta forma, al ejecutarse los scripts de forma periódica, la máquina remota sabrá a donde tiene que enviar la información.

Para ejecutar los scripts de forma periódica se va a utilizar el archivo *crontab*. Crontab es una herramienta que permite automatizar tareas en Linux. Tan solo hace falta configurar la tarea que se quiera realizar en el periodo que se desee.

El archivo *crontab* tiene en su interior una lista completa de todos los scripts que se tienen que ejecutar de forma periódica. Cron, por su parte, es el demonio encargado de ejecutar dichas tareas. En la *Figura 16* se puede ver el formato de los scripts a configurar:

```
# Ejemplo de definición de tareas:
# .---------------- minuto (0 - 59)
# |  .------------- hora (0 - 23)
# |  |  .---------- dia del mes (1 - 31)
# |  |  |  .------- mes (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- dia de la semana (0 - 6) (Domingo=0 o 7) O sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Figura 16 - Formato de Crontab

7.1.5.2. NRPE

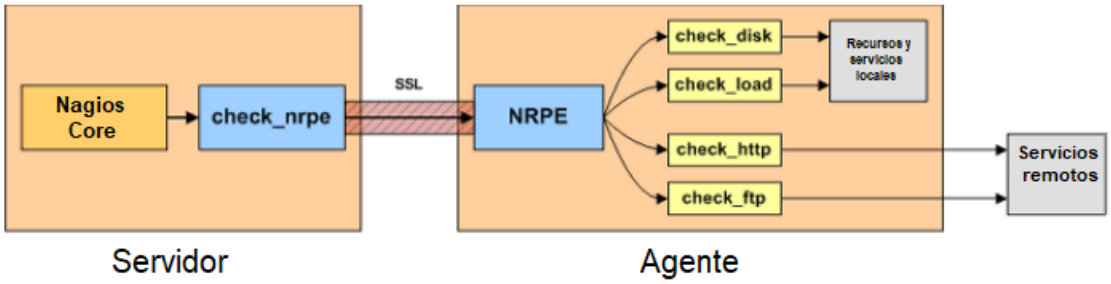

Figura 17 - Funcionamiento de NRPE [2]

El módulo NRPE (Nagios Remote Plugin Executor) [2] se utiliza para que el servidor Nagios se pueda comunicar con los diferentes equipos remotos que se quieren monitorizar. Este módulo hay que instalarlo tanto en el servidor Nagios como en cada uno de los equipos remotos a monitorizar. Esto se verá en las próximas páginas, en el apartado 9. INSTALACIÓN Y CONFIGURACIÓN

Este módulo nos va a permitir monitorizar de forma activa las máquinas remotas, es decir, se les va a poder mandar el comando *check_nrpe* para que respondan al instante con la información que se les pida, en vez de estar esperando a que la información se mande de forma pasiva cada cierto tiempo. Esto lo podemos ver de forma sencilla en la *Figura 17 de arriba.*

7.1.5.2.1. Servidor y agente

Para poder explicar el funcionamiento de forma más extensa, primero hay que definir el servidor y el agente. El servidor es el equipo que tiene instalado Nagios Core y el que va a recibir toda la información de monitorización de los equipos remotos. Los agentes son cada uno de los equipos remotos a monitorizar.

El servidor va a tener instalado Nagios Core y también el plugin NRPE. Él va a ser el encargado de mandar la solicitud *check_nrpe* a cada uno de los equipos remotos que se quieran monitorizar. Los plugins son programas o scripts responsables de realizar el chequeo que se le solicita al equipo a monitorizar. Habitualmente estarán instalados en las rutas */usr/local/nagios/libexec/* o */usr/local/nagios/etc/*, pues son las rutas predeterminadas para ellos.

El agente va a tener instalado el demonio NRPE (NRPE daemon) y los plugins de Nagios, ya que sin ellos, el demonio NRPE no va a poder monitorizar nada. El agente va a estar a la espera

de que el servidor le mande la solicitud *check_nrpe* y en el momento en el que la reciba procederá a recopilar los datos que se le han solicitado y a enviárselos al servidor.

7.1.5.2.2. Procedimiento de la monitorización

Una vez explicados los conceptos básicos de NRPE, se van a detallar los pasos que se realizan a la hora de hacer la monitorización activa mediante NRPE. Dichos pasos son los siguientes:

- Nagios Core (el servidor) ejecuta el comando *check_nrpe*.
- El plugin *check_nrpe* contacta con el demonio NRPE en la máquina remota.
- El demonio NRPE recopila la información que se le ha solicitado.
- La información es enviada al servidor y se procesa en Nagios Core.

Nota: NRPE envía y recibe información mediante el puerto 5666 TCP, por lo que para asegurar su correcto funcionamiento hay que tenerlo abierto en el firewall.

7.1.5.2.3. Tipos de chequeos

El plugin NRPE puede realizar dos tipos de chequeos: Los directos y los indirectos. Los chequeos directos se basan en monitorizar recursos locales o privados en una máquina remota preguntándole directamente a dicha máquina. A modo de ejemplo está el caso de la *Figura 18*, en el que se pregunta el uso de CPU mediante *check_nrpe* a un equipo de la misma red local.

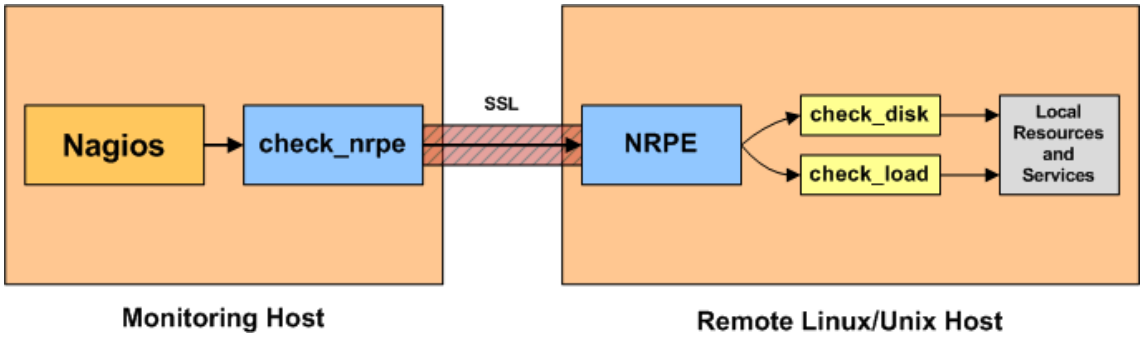

Figura 18 - NRPE chequeos directos [2]

Por otra parte, los chequeos indirectos se utilizan cuando el servidor con NRPE y Nagios Core no tiene conectividad con el equipo que se quiere monitorizar. El servidor no puede hablar directamente con el equipo a monitorizar, pero sí puede hablar con un equipo el cual tiene conectividad o permisos para hablar con el que se quiere monitorizar. En ese caso, el segundo equipo hace de intermediario para hablar con el equipo final, tal como se puede observar en la *Figura 19*.

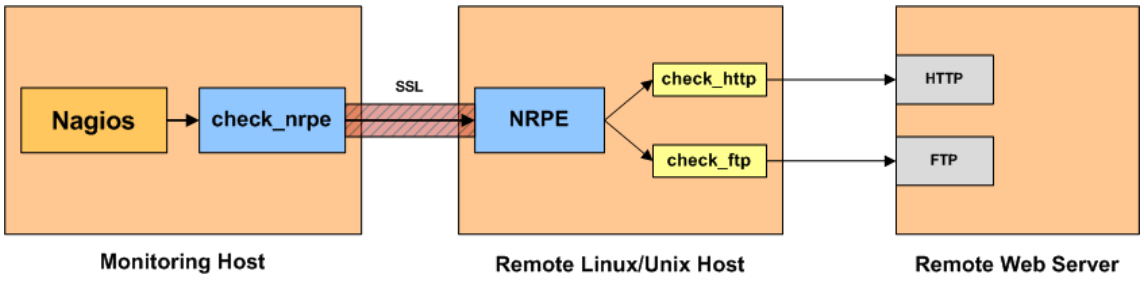

Figura 19 - NRPE chequeos indirectos [2]

7.1.5.2.4. NRPE v4

En este apartado se van a explicar las características principales que se implementan en la versión 4 de NRPE [63]. La mayoría de las características son del ámbito de seguridad, como se puede observar a continuación:

- Añadido soporte para TLSv1.3 y TLSv1.3+
- Añadida la línea por defecto para IPv6 en *allow_from* hosts
- Añadida opción a forzar el uso de NRPE v3
- Añadido refuerzo ante ataques DOS

Cabe destacar que esta última versión de NRPE es retrocompatible con las anteriores, pero para asegurar un correcto funcionamiento es recomendable actualizar todos los equipos a NRPE v4.0.3 (Última versión lanzada a la hora de realizar este proyecto).

7.1.5.2.5. Tamaño de paquete NRPE

Para versiones anteriores a NRPE v3, el tamaño fijo del paquete era 1024 bytes. Esto trae el problema de que cualquier complemento ejecutado que tenga una salida superior a 1024 bytes se ve afectado y se pierde información. Esto puede causar diferentes problemas, tales como, datos erróneos o faltantes o malformaciones en los gráficos realizados con dichos datos.

Antes de la versión 3, esto era posible solucionarlo a mano modificando el código e incrementando la capacidad a un número mayor, pero no era fácil de implementar y dicha modificación no era oficial, por lo tanto no tenía garantía de funcionar bien.

Con la versión NRPE v3 se incrementó el tamaño del paquete hasta los 64 KB. Esta versión es totalmente retrocompatible con versiones anteriores de NRPE. Con NRPE v4 el tamaño se mantiene igual que con NRPE v3. El caso óptimo es que el servidor y el cliente tengan la versión v4 de NRPE, ya que se aprovechan todas sus ventajas. En el caso en el que el servidor o el cliente tengan una versión anterior, también habrá comunicación gracias a la retrocompatibilidad, pero no se aprovecharan las nuevas funciones.

7.2. Descubrimiento de red

En este apartado se va a explicar en qué consiste la solución que se quiere implementar en este proyecto para el descubrimiento de red. Visto que en el apartado 5.2. Software de descubrimiento la alternativa que más puntuación ha obtenido ha sido Zenmap, se va a instalar dicha herramienta. Esta aplicación se va a encargar de realizar el análisis de la red del laboratorio I2T y obtener la información de los equipos conectados, tales como la dirección IP, MAC, nombre del dispositivo etcétera.

La herramienta se va a instalar directamente en la misma máquina virtual en la que se va a instalar Nagios Core y gracias a ella se van a obtener los datos necesarios para configurar el sistema de monitorización de red.

8. INSTALACIÓN Y CONFIGURACIÓN

En este subapartado se van a analizar detalladamente todos los pasos que se han seguido para realizar la implementación de la solución citada anteriormente. Se van a detallar todos los pasos, las configuraciones y las modificaciones que se han tenido que realizar en los ficheros de configuración, explicando cada paso de forma detallada.

8.1. Sistema de descubrimiento de red

En este apartado se van a explicar los pasos a seguir para realizar la implementación del sistema de descubrimiento de red. Para poder tener la aplicación Zenmap en funcionamiento, tan solo hay que instalarla en la máquina virtual que se está usando a modo de servidor. Para ello, se van a seguir los siguientes pasos:

Primero, se va a actualizar la lista de paquetes disponibles desde los repositorios y se va a instalar Python GTK2, ya que Zenmap usa esta herramienta para crear la interfaz gráfica de usuario.

```
# sudo apt update
# wget
http://archive.ubuntu.com/ubuntu/pool/universe/p/pygtk/python-
gtk2_2.24.0-5.1ubuntu2_amd64.deb
# sudo apt install ./python-gtk2_2.24.0-5.1ubuntu2_amd64.deb
```

Tras esto, se procede a instalar Zenmap. Para ello, se descarga el paquete, se instala y se ejecuta con permisos de superusuario para comprobar su correcto funcionamiento.

```
# wget
http://archive.ubuntu.com/ubuntu/pool/universe/n/nmap/zenmap_7.60-
1ubuntu5_all.deb
# sudo apt install ./zenmap_7.60-1ubuntu5_all.deb
# sudo zenmap
```

Tras esto, se hace un análisis de la red del laboratorio y se obtiene la información de los equipos conectados a la red. Por seguridad, no se van a mencionar todos los equipos, solo los más importantes y con unas direcciones IP no reales. Estas direcciones IP se van a utilizar más adelante para definir los hosts que se quieren monitorizar. Los equipos encontrados han sido los siguientes:

- 10.20.30.40: MV en la que se va a instalar Nagios Core
- 10.20.30.50: Impresora HP LaserJet M601
- 10.20.30.60: Switch DELL S3048ON
- 10.20.30.70: Switch HP Procurve 2848
- 10.20.30.80: Router Mikrotik CloudCore CCR1036

Gracias a Zenmap se han obtenido las direcciones de los equipos principales de la red del laboratorio I2T. Con estos datos, se va a proceder a realizar la implementación del sistema de monitorización de red.

8.2. Sistema de monitorización de red

En este apartado se van a explicar los pasos a seguir para realizar la implementación del sistema de monitorización de red.

8.2.1. Agent

En este apartado se van a mostrar los pasos a realizar en la máquina remota a monitorizar. Normalmente, la máquina remota se configura después de hacer las configuraciones en el equipo principal, pero en esta guía se va a configurar antes que la máquina principal, con el fin de poder comprobar la conectividad del módulo NRPE del equipo principal y de tener una máquina preparada para hacer pruebas.

En los equipos remotos se tienen que instalar los plugins de Nagios y el demonio de NRPE. Para realizar una correcta instalación del plugin NRPE en Ubuntu, se van a seguir los pasos de la guía oficial de Nagios [64].

Estas máquinas virtuales pueden disponer de diferente software instalado, por lo que para evitar problemas en la instalación, se van a instalar todas las dependencias.

8.2.1.1. Dependencias

Antes de comenzar la instalación de los plugins de Nagios [65], se va a comprobar que los paquetes necesarios están instalados, y de no ser así, se van a instalar automáticamente.

```
# sudo apt-get update
# sudo apt-get install -y autoconf automake gcc libc6 libmcrypt-dev
make libssl-dev wget openssl
```

8.2.1.2. Instalación de los plugins de Nagios

Después, se proceden a instalar los plugins de Nagios [65], pues para que el demonio NRPE funcione es necesario que los plugins estén instalados.

```
# cd /tmp
# wget http://nagios-plugins.org/download/nagios-plugins-
2.3.3.tar.gz
# tar xzf nagios-plugins-2.3.3.tar.gz
# cd nagios-plugins-2.3.3
# sudo ./configure
# sudo make
# sudo make install
```

8.2.1.3. Añadir IP máquina de monitorización

Tras esto, se actualiza el archivo de configuración */usr/local/nagios/etc/nrpe.cfg* añadiéndole la siguiente línea:

```
allowed_hosts=127.0.0.1,10.20.30.40 (IP servidor Nagios Core)
```

En este comando se configuran los equipos autorizados. Para que NRPE solo escuche las *requests* del equipo en el que está instalado, se pone solo la IP 127.0.0.1 (Localhost). Para que el plugin check_nrpe localizado en la máquina en la que está el servidor de Nagios Core pueda conectarse con la máquina remota, la IP de ese equipo se tiene que poner a continuación de la IP de Localhost diferenciada mediante comas.

8.2.1.4. Uso de comandos NRPE

El siguiente comando determina si el demonio de NRPE permite a los clientes especificar argumentos para los comandos que se ejecuten. En el caso de este proyecto, se va a dejar activado por si en un futuro se quieren utilizar dichas funciones. Para ello, se va a incluir la siguiente línea en el archivo */usr/local/nagios/etc/nrpe.cfg:*

```
dont_blame_nrpe=1
```

8.2.1.5. Testeo de funcionamiento NRPE

Tras esto, ya se puede iniciar y probar el servicio NRPE. En las *Figura 20* y *Figura* 21 se puede observar que el servicio está escuchando en el puerto 5666 TCP y que al hacer una prueba con localhost, este responde correctamente.

```
# sudo systemctl start nrpe.service
# netstat -at | egrep "nrpe|5666"
```

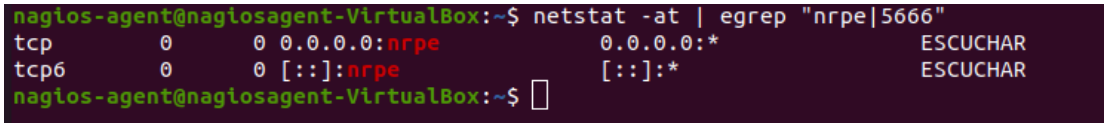

Figura 20 - Comprobación de puertos NRPE

```
/usr/local/nagios/libexec/check_nrpe -H 127.0.0.1
```

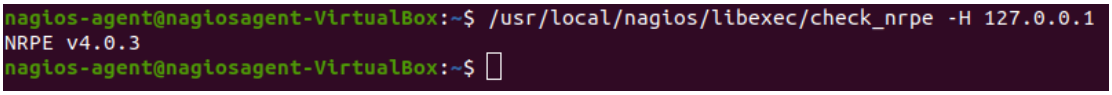

Figura 21 - Comprobación versión NRPE

También se pueden probar los comandos instalados por defecto, como la comprobación de la carga de CPU, los procesos totales o los zombies:

```
# /usr/local/nagios/libexec/check_nrpe -H localhost -c check_load
# /usr/local/nagios/libexec/check_nrpe -H localhost -c
check_total_procs
# /usr/local/nagios/libexec/check_nrpe -H localhost -c
check_zombie_procs
# /usr/local/nagios/libexec/check_swap -w 25% -c 10%
```

Tras todos los pasos anteriores, ya se tiene instalado el demonio NRPE en la máquina a monitorizar. Ahora, solo falta configurar los comandos personalizados.

8.2.1.6. Personalizar los comandos

En este subapartado se van a describir los pasos necesarios para personalizar los comandos que va a utilizar el módulo NRPE. Cabe destacar que si en la máquina remota se define un nuevo comando, para poder usarlo, en la máquina que aloja el servidor de Nagios Core habrá que definir un nuevo servicio, tal y como se va a ver en el apartado 9.2.6. Definición de hosts y servicios.

Lo primero que hay que hacer es entrar al archivo */usr/local/nagios/etc/nrpe.cfg* y buscar los comandos que ya están instalados. Después, tan solo hay que añadir el comando que se quiere configurar con la siguiente estructura:

```
command[<nombre>]=/usr/local/nagios/libexec/<comando instalado>
<argumentos del comando>
```

De esta forma, se han configurado los comandos *check_swap* y *check_sda1* mediante NRPE como aparece en la *Figura 22*:

```
# 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,.25,.20
#command[check_hda1]=/usr/local/nagios/libexec/check_disk -w 20% -c 10% -p /dev/hda1
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
command[check_sda1]=/usr/local/nagios/libexec/check_disk -w 20% -c 10% -p /dev/sda1
command[check_swap]=/usr/local/nagios/libexec/check_swap -w 30% -c 10%
```

Figura 22 - Configuración check_swap y check_sda1

En la *Figura 23* se puede observar cómo se ha reiniciado el servicio de NRPE y se ha comprobado que los comandos funcionan correctamente:

```
nagios-agent@nagiosagent-VirtualBox:/usr/local/nagios/libexec$
nagios-agent@nagiosagent-VirtualBox:/usr/local/nagios/libexec$ sudo systemctl restart nrpe.service
nagios-agent@nagiosagent-VirtualBox:/usr/local/nagios/libexec$ /usr/local/nagios/libexec/check_nrpe -H localhost -c check_sda1
DISK OK - free space: /boot/efi 510 MiB (99.99% inode=-);| /boot/efi=0MiB;408;459;0;510
nagios-agent@nagiosagent-VirtualBox:/usr/local/nagios/libexec$ /usr/local/nagios/libexec/check_nrpe -H localhost -c check_swap
SWAP OK - 100% free (448 MB out of 448 MB) |swap=448MB;134;44;0;448
nagios-agent@nagiosagent-VirtualBox:/usr/local/nagios/libexec$ _
```

Figura 23 - Comprobación check_swap y check_nrpe

8.2.2. Máster

En este apartado se van a detallar todos los pasos que se han realizado en el servidor, el equipo principal. Este equipo se va a encargar de recopilar y almacenar los datos de los equipos y servicios que se están monitorizando y de realizar las gráficas correspondientes para mostrar de forma sencilla y visual todos los datos de rendimiento recopilados a los encargados de supervisar la red.

El equipo principal va a ser una máquina virtual la cual cumple holgadamente con los requisitos mínimos recomendados para el software Nagios Core. Esta máquina virtual funciona sobre dos procesadores, 8 GB de RAM y dispone de 20 GB de almacenamiento. Está disponible en la IP 10.20.30.40 del laboratorio I2T y se ha instalado el sistema operativo Ubuntu 20.04 LTS. Este sistema operativo tiene instalado el firewall UFW, esto es necesario saberlo pues en los siguientes pasos va a ser necesario añadirle ciertas excepciones.

Los pasos que se han seguido han sido los detallados en la página oficial de Nagios [66]. Se va a proceder a detallar la instalación y explicar el porqué de cada paso.

8.2.2.1. Instalación de Nagios Core

Para empezar, se va a actualizar la información disponible en el equipo acerca de los repositorios instalados y también se van a proceder a instalar las diferentes utilidades

imprescindibles para realizar la instalación de Nagios Core siguiendo la guía oficial. Entre ellas, las más importantes son el servidor Apache, PHP y las utilidades para obtener paquetes, compilarlos y descomprimirlos.

```
# sudo apt-get update
# sudo apt-get install -y autoconf gcc libc6 make wget unzip
apache2 php libapache2-mod-php7.4 libgd-dev
```

Mediante los siguientes comandos se va a proceder a realizar la descarga de la última versión disponible del software Nagios Core del repositorio oficial de GitHub. Se va a descargar el paquete en la carpeta temporal /tmp de Ubuntu y se va a proceder a su descompresión.

```
# cd /tmp
# wget -O nagioscore.tar.gz
https://github.com/NagiosEnterprises/nagioscore/archive/nagios-
4.4.6.tar.gz
# tar xzf nagioscore.tar.gz
```

Con los siguientes comandos se va a proceder a compilar el software descargado en el paso anterior y se debería obtener una salida como la que se aprecia en la *Figura 24*.

```
# cd /tmp/nagioscore-nagios-4.4.6/
# sudo ./configure --with-httpd-conf=/etc/apache2/sites-enabled
# sudo make all
```

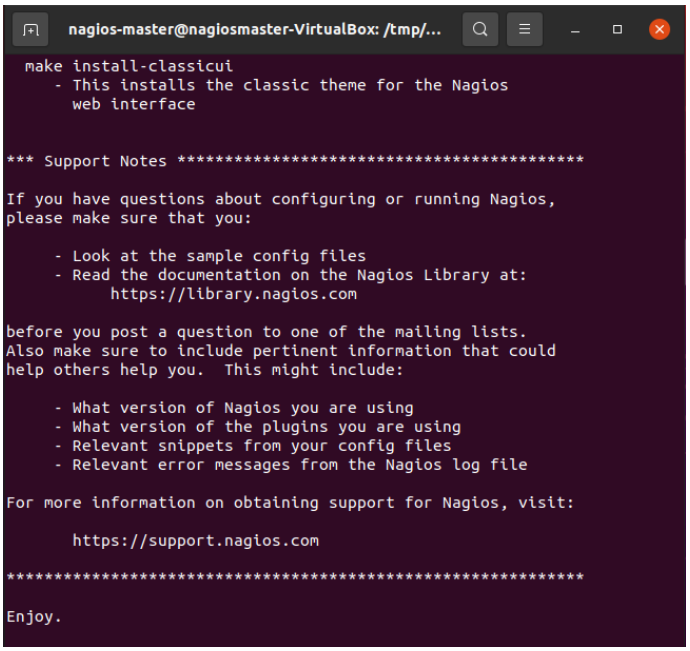

Figura 24 - Salida make Nagios Core

En este paso se va a crear el usuario y el grupo *nagios*. También se va a añadir el usuario *www-data* al grupo *nagios*.

```
# sudo make install-groups-users
# sudo usermod -a -G nagios www-data
```

El comando *sudo make install-groups-users* lo que hace es crear el grupo *nagios* y añadir el usuario *nagios* a dicho grupo. El segundo comando modifica los permisos del usuario *nagios* para darle permisos a acceder a la carpeta *www-data*.

Después de esto, se procede a instalar los *Binaries*, el *Command Mode*, los archivos de configuración y el proceso *demonio,* el cual hace posible que Nagios Core se ejecute en el arranque.

```
# sudo make install
# sudo make install-daemoninit
# sudo make install-commandmode
# sudo make install-config
```

También es necesario instalar los archivos de configuración del servidor web Apache. Esto se realiza con los siguientes comandos:

```
# sudo make install-webconf
# sudo a2enmod rewrite
# sudo a2enmod cgi
# systemctl restart apache2
```

Para poder acceder a la interfaz web de Nagios Core es necesario que el puerto 80 esté habilitado, por lo que se va a autorizar en el firewall. En las siguientes líneas se muestra como añadir la regla en UFW. Tras esto, se va a reiniciar para que surtan efecto los cambios:

```
# sudo ufw allow apache
# sudo ufw reload
```

Ahora hay que crear la contraseña y el usuario para acceder a la interfaz web de Nagios Core. El usuario va a ser *nagiosadmin* y la contraseña se puede encontrar en el *Anexo II,* que por seguridad no se va a incorporar en este documento, pero sí que se va a entregar al personal del laboratorio I2T.

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

El siguiente paso es reiniciar el servidor Apache e iniciar Nagios Core.

```
# sudo systemctl restart apache2.service
# sudo systemctl start nagios.service
```

Con los pasos realizados hasta ahora ya se debería de poder acceder a la interfaz web de Nagios Core. Para ello, desde el navegador que tengamos instalado deberemos de poder acceder a la interfaz web de las siguientes maneras:

10.20.30.40/nagios (IP de la MV del laboratorio I2T)

localhost/nagios

Con ambas deberíamos de lograr un resultado similar al de la *Figura 25*, en la que se puede ver que el proceso de Nagios Core se está ejecutando y que la versión instalada es la v4.4.6:

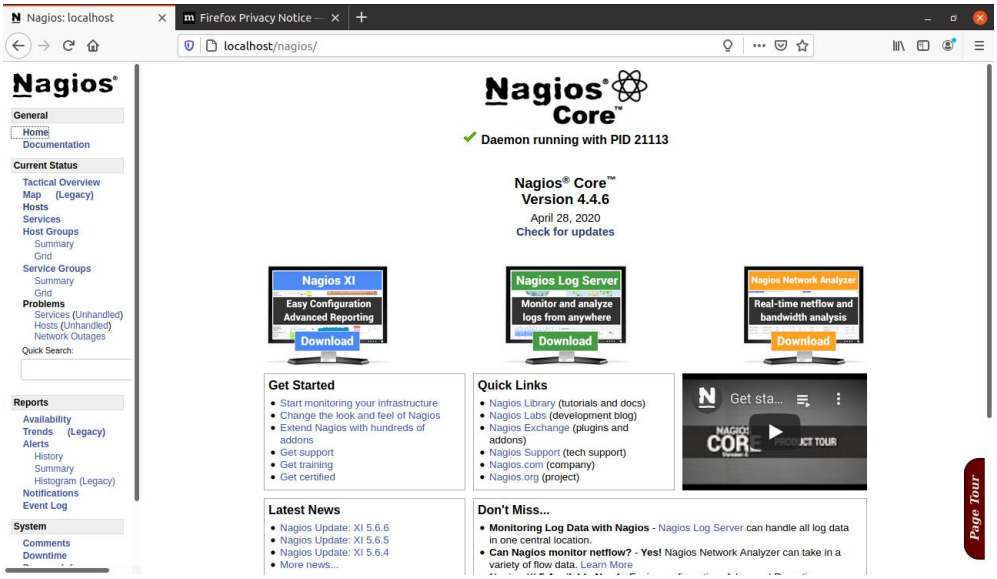

Figura 25 - Interfaz web de Nagios Core

En este punto se puede navegar por la interfaz web de Nagios Core, pero hay varios errores debido a que los plugins de Nagios no están instalados.

8.2.2.2. Instalación de los plugins de Nagios

Antes de instalar los plugins de Nagios, se va a comprobar que se tienen los siguientes paquetes instalados, y si no es así, se van a instalar automáticamente:

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

Se va a descargar y descomprimir la última versión de los plugins de Nagios, para ello, se va a consultar cuál es la última versión disponible en la página oficial [67]. En este caso, la última versión disponible es la 2.3.3, la cual se va a proceder a descargar de la siguiente manera:

```
# cd /tmp
# wget --no-check-certificate -O nagios-plugins.tar.gz
https://github.com/nagios-plugins/nagios-plugins/archive/release-
2.3.3.tar.gz
# tar zxf nagios-plugins.tar.gz
```

Tras la descarga y descompresión, se procede a instalar los plugins:

```
# cd /tmp/nagios-plugins-release-2.3.3/
# sudo ./tools/setup
# sudo ./configure
# sudo make
# sudo make install
```

Nota: Algunos comandos pueden tardar más de lo esperado. El sistema no se ha quedado congelado, pueden tardar más de lo habitual y no muestran ningún mensaje en pantalla.

Tras la instalación de los plugins, se procede a reiniciar el servicio de Nagios Core, y tras esto, se debería de observar que los servicios que antes estaban inaccesibles empiezan a recibir actualizaciones marcando que están funcionando correctamente *(Figura 26).*

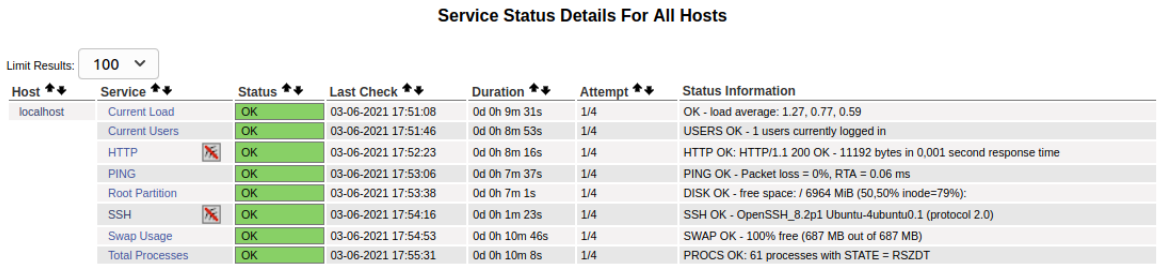

Figura 26 - Servicios Localhost Nagios Core

NOTA: Antes de continuar con la instalación en el Master, es recomendable realizar los pasos del apartado *9.1. Agent* para realizar la instalación del demonio NRPE en mínimo una máquina remota de las que se quiera monitorizar. Así, se podrá comprobar la conectividad y el funcionamiento de dicho módulo.

8.2.2.3. Instalación del módulo NRPE

Se procede a descargar el módulo NRPE de los repositorios oficiales de GitHub:

```
# cd /tmp
# wget --no-check-certificate -O nrpe.tar.gz
https://github.com/NagiosEnterprises/nrpe/archive/nrpe-4.0.3.tar.gz
# tar xzf nrpe.tar.gz
```

Tras la descarga, se procede a compilar e instalar los paquetes.

```
# cd /tmp/nrpe-nrpe-4.0.3/
# sudo ./configure --enable-command-args --with-ssl-
lib=/usr/lib/x86_64-linux-gnu/
# sudo make check_nrpe
# sudo make install-plugin
```

8.2.2.4. Testeo de conectividad

Tras la instalación del plugin NRPE, se comprueba su conectividad con la máquina remota con el siguiente comando:

```
# /usr/local/nagios/libexec/check_nrpe -H <IP remota>

Resultado: NRPE v4.0.3
```

El resultado es positivo, hay conectividad, por lo tanto el plugin en la máquina que aloja el servidor de Nagios Core está bien instalado. También está bien instalado el demonio de la máquina remota, las reglas del firewall y los permisos necesarios para que el módulo NRPE funcione.

8.2.2.5. Comandos plugin check_nrpe

En este punto, es necesario crear una definición de comandos para que Nagios Core pueda utilizar de forma correcta el plugin NRPE. Para ello se va a detener el servicio de Nagios y se va a editar el archivo */usr/local/nagios/etc/objects/commands.cfg* añadiendo lo siguiente:

```
# sudo systemctl stop nagios
# sudo nano /usr/local/nagios/etc/objects/commands.cfg
```

```
define command{
command_name            check_nrpe
command_line            $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}
```

Hay que destacar que los comandos que están definidos en *commands.cfg* se pueden referenciar desde los archivos de definición de *host*, *service* y *contact*, pues se ha programado de esta manera para facilitar la comprensión y la instalación.

Tras este paso, ya es posible empezar a añadir servicios a monitorizar de las máquinas remotas en Nagios Core.

8.2.2.6. Definición de hosts y servicios

Para poder monitorizar las máquinas remotas, hay que crear unas definiciones de servicios y hosts. En este apartado se va a crear una plantilla para todas las máquinas que funcionen con Linux en el laboratorio. Esta plantilla se va a basar en la plantilla existente *generic-host* definida en el archivo */usr/local/nagios/etc/objects/templates.cfg*.

Para definir la plantilla, se va a editar el archivo */usr/local/nagios/etc/objects/templates.cfg* añadiendo la siguiente definición:

```
# sudo nano /usr/local/nagios/etc/objects/templates.cfg
```

```
define host {

name                    equipo-linux        ; Nombre de la plantilla
use                     generic-host        ; Esta plantilla coge los valores de la plantilla generic-
host
check_period            24x7                ; Periodo de chequeo
check_interval          5
retry_interval                  1
```

```
max_check_attempts      10
check_command           check-host-alive        ; Default command to check Linux hosts
notification_period     24x7                    ; Se notifica las 24h del dia
notification_interval   30                      ; Reenviar notificaciones cada 30 minutos
notification_options    d,u,r                   ; Se envian notificaciones para d=down,
u=up, r=recovery
contact_groups          admins                  ; Se mandan las notificaciones a los admins
register                0                       ; NO REGISTRAR, es una plantilla, no un host real.
hostgroups              equipos-linux
}
```

Tras esto, se va a definir un nuevo host aprovechando la plantilla recién creada. Para ello, se va a crear un nuevo archivo de configuración llamado *equipo1.cfg* en el que se va a hacer la definición del host usando la plantilla anteriormente creada. El archivo equipo1.cfg va a estar en la carpeta */usr/local/nagios/etc/servers/.*

Antes de crear dicho archivo, hay que crear la carpeta */servers/* y hay que modificar el archivo */usr/local/nagios/etc/nagios.cfg* para indicarle a Nagios la ruta en la que van a estar los archivos de configuración de las máquinas remotas que se tienen que monitorizar. Para ello, se va a crear la carpeta y descomentar la siguiente línea que aparece en blanco en la *Figura 27,* en el archivo *nagios.cfg*:

```
# sudo mkdir -p /usr/local/nagios/etc/servers
# sudo nano /usr/local/nagios/etc/nagios.cfg
```

```
# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:

cfg_dir=/usr/local/nagios/etc/servers
#cfg_dir=/usr/local/nagios/etc/printers
#cfg_dir=/usr/local/nagios/etc/switches
#cfg_dir=/usr/local/nagios/etc/routers
```

Figura 27 - Ruta máquinas a monitorizar

Tras eso, se crea el archivo *equipo1.cfg* en la carpeta */usr/local/nagios/etc/servers/* y se añade lo siguiente:

```
# sudo nano /usr/local/nagios/etc/servers/equipo1.cfg
```

```
define host{
        use             equipo-linux            ; valores por defecto de la plantilla equipo-linux
        host_name       <nombre_host>           ;Nombre del host
        alias           xxxxxxxxxxxxxx          ; Nombre extenso del host
        address         x.x.x.x                 ; Direccion IP del equipo a monitorizar
}
```

En este caso, se crea de la siguiente manera:

```
define host{
        use             equipo-linux            ; valores por defecto de la plantilla equipo-linux
        host_name       equipo1                 ;Nombre del host
        alias           Ubuntu_equipo1          ; Nombre extenso del host
        address         x.x.x.x                 ; Direccion IP del equipo a monitorizar
}
```

Por último, se definen los servicios que se van a usar para monitorizar la máquina o máquinas remotas. En este caso, se van a definir los servicios de carga de CPU, usuarios actuales, espacio libre en disco, procesos totales y procesos zombies.

Estas definiciones van a usar los mismos comandos definidos en el archivo */usr/local/nagios/etc/nrpe.cfg* de la máquina remota en cuestión.

Para crear el servicio de monitorización de la carga de la CPU en el equipo remoto se usa el comando *check_nrpe* con el argumento *check_load.* Por ello, se va a añadir el siguiente servicio al archivo *equipo1.cfg*:

```
define service{
        use                     local-service
        host_name               equipo1
        service_description     Carga CPU
        check_command           check_nrpe!check_load
}
```

Si se quiere monitorizar el número de usuarios conectados en la máquina remota, se hará de la siguiente manera:

```
define service{
        use                     local-service
        host_name               equipo1
        service_description     Usuarios actuales
        check_command           check_nrpe!check_users
}
```

Para visualizar el espacio libre en la partición sda1 del disco duro, es necesario definir el siguiente servicio:

```
define service{
        use                     local-service
        host_name               equipo1
        service_description     Espacio Libre SDA1
        check_command           check_nrpe!check_sda1
}
```

Para conocer el número total de procesos y procesos zombies, se tienen que definir los siguientes dos servicios:

```
define service{
        use                     local-service
        host_name               equipo1
        service_description     Procesos totales
        check_command           check_nrpe!check_total_procs
}
```

```
define service{
        use                     local-service
        host_name               equipo1
        service_description     Procesos Zombies
        check_command           check_nrpe!check_zombie_procs
}
```

Por último, queda por crear el grupo equipos-linux, al cual se le ha hecho referencia en la plantilla, pero no está creado. Para ello, se crea el archivo *hostgroups.cfg* en la carpeta */objects/* y se define el grupo en el archivo. Tras esto, se añade en la configuración de Nagios en el archivo *nagios.cfg*, tal y como se observa en la *Figura 28.*

```
# sudo nano /usr/local/nagios/etc/objects/hostgroups.cfg
```

```
define hostgroup{
hostgroup_name          equipos-linux       ; The name of the hostgroup
alias                   Equipos Linux       ; Long name of the group
}
```

```
# sudo nano /usr/local/nagios/etc/nagios.cfg
```

```
# 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
cfg_file=/usr/local/nagios/etc/objects/hostgroups.cfg
```

Figura 28 - Hostgroups en nagios.cfg

8.2.2.7. Comprobación y reinicio de Nagios

Antes de reiniciar el servicio se va a comprobar que los cambios se han realizado de forma adecuada y tras eso, se va a proceder a reiniciar el servicio de Nagios.

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

# sudo systemctl restart nagios
```

8.2.2.8. *Enable de Nagios, plugins y Apache*

En este subapartado se van a configurar para que inicien en el arranque los servicios necesarios para garantizar el buen funcionamiento de Nagios Core. Cabe destacar que alguno de estos comandos ya se ha ejecutado con anterioridad, pero para ir sobre seguro se van a volver a ejecutar. También se va a comprobar que el arranque queda bien configurado.

```
# sudo systemctl enable nagios.service
# sudo systemctl enable apache2.service
```

Para comprobar que los servicios han sido configurados en el arranque de forma correcta, ejecutaremos el siguiente comando y comprobaremos que aparece en pantalla lo que se aprecia en la *Figura 29*:

```
# systemctl list-unit-files | grep nagios.service
# systemctl list-unit-files | grep apache2.service
```

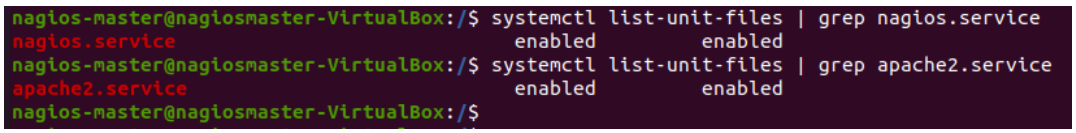

Figura 29 - Comprobación de ejecución de Nagios y Apache en el arranque

8.2.2.9. *Notificaciones por correo*

Para que Nagios Core pueda mandar notificaciones a los miembros del laboratorio I2T es necesario realizar varias configuraciones e instalar el servidor de correo Postfix. Es necesario configurar qué notificaciones se van a enviar, a quien se van a enviar, bajo qué condiciones y por qué vía.

Instalación de Postfix

Para empezar, se va a instalar el servidor de correo electrónico Postfix, el cual se va a encargar de enviar los mensajes de notificación. Para ello, se van a seguir los siguientes pasos:

```
# apt-get install postfix mailutils libsasl2-2 ca-certificates
libsasl2-modules
```

Durante la instalación aparecerá un cuadro en el que preguntará como se quiere realizar la instalación (Figura 30Figura 31Figura 32). Se le dará a “Sitio de Internet” para que realice la configuración predeterminada para enviar correos mediante Gmail.

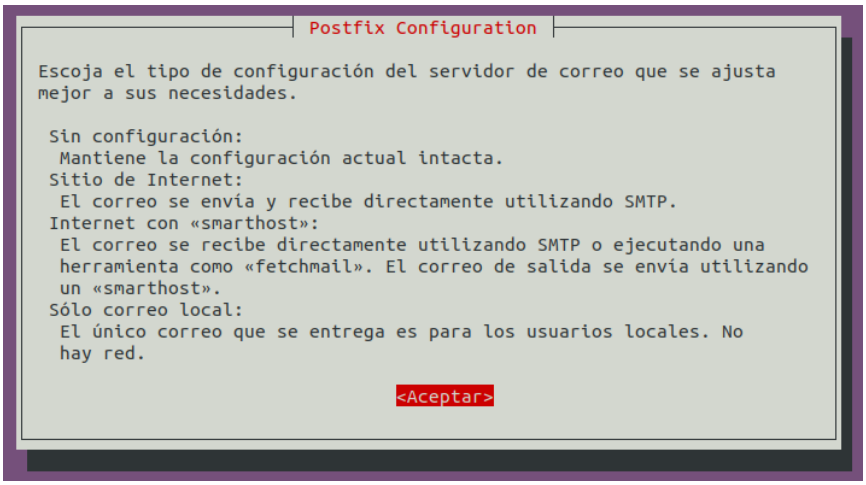

Figura 30 - Ventana de instalación Postfix

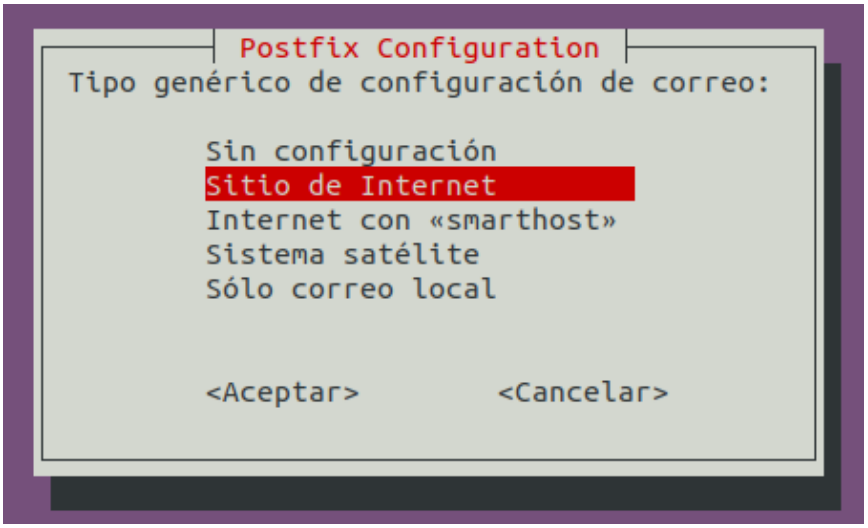

Figura 31 - Sitio de internet Postfix

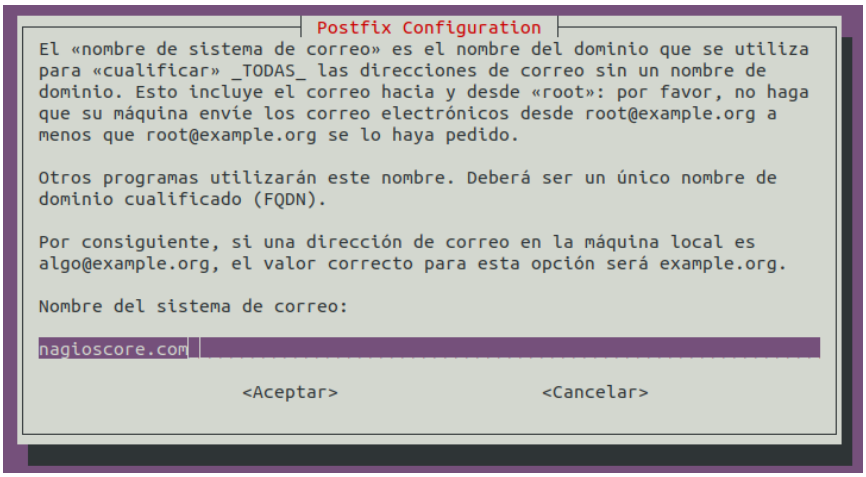

Figura 32 - Nombre del sistema de correo Postfix

Configuración de Postfix

Una vez instalado, procedemos a crear una copia del archivo de configuración */etc/postfix/main.cf* de Postfix y a editarlo de la siguiente manera:

```
# cd /etc/postfix/
# sudo cp main.cf main_copia.cf
# sudo nano main.cf
```

Hay que poner en comentario la línea "*relayhost =*" del final del documento y añadir las siguientes debajo. De esta forma, se configura el servidor de Gmail como servidor principal por el que enviar los mensajes y se especifica la localización de la contraseña y del certificado TLS.

```
relayhost =[smtp.gmail.com]:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/postfix/cacert.pem
smtp_use_tls = yes

smtpd_use_tls=yes
smtpd_tls_CAfile=/etc/ssl/certs/ca-certificates.crt
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtp_tls_CAfile=/etc/ssl/certs/ca-certificates.crt
smtp_tls_key_file=/etc/ssl/ssl-cert-snakeoil.key
smtpd_tls_security_level=encrypt
smtp_tls_security_level=encrypt
```

Tras esto, hay que crear el archivo *sasl_passwd* con las credenciales de la cuenta de correo de Gmail que se va a utilizar para enviar los correos. Para la configuración de este servicio se ha creado la cuenta *nagioscoreI2T@gmail.com* la cual va a pertenecer únicamente a los miembros del departamento I2T de la escuela de ingeniería de Bilbao.

```
# sudo nano /etc/postfix/sasl_passwd
```

```
[smtp.gmail.com]:587 nagiosi2t@gmail.com:contraseñanagios
```

Se crea el hash para Postfix mediante el comando *postmap* y, además, para aumentar la seguridad, se modifican los permisos de los archivos recién creados para que solo root pueda leerlos y escribirlos con los siguientes comandos:

```
# sudo postmap /etc/postfix/sasl_passwd

# sudo chown root:root /etc/postfix/sasl_passwd
/etc/postfix/sasl_passwd.db
# sudo chmod 0600 /etc/postfix/sasl_passwd
/etc/postfix/sasl_passwd.db
```

Ahora, se crea el certificado, y tras esto, debería haber un certificado llamado *cacert.pem* en la carpeta */etc/postfix/:*

```
# cat /etc/ssl/certs/thawte_Primary_Root_CA.pem | sudo tee -a
/etc/postfix/cacert.pem
```

Por último, se reinicia el servicio Postfix para que los cambios surtan efecto:

```
# sudo systemctl restart postfix
```

Aplicaciones poco seguras en Gmail

Antes de enviar un correo de prueba, queda un último paso por hacer. De forma predeterminada el servicio de Gmail no permite que aplicaciones no seguras accedan a él. Esto puede causar un problema con Postfix, ya que la detecta como no segura. Por lo tanto, se va a acceder a la página de Gmail [68] que facilita la configuración de las aplicaciones poco seguras y se va a activar dicha opción (*Figura 33*).

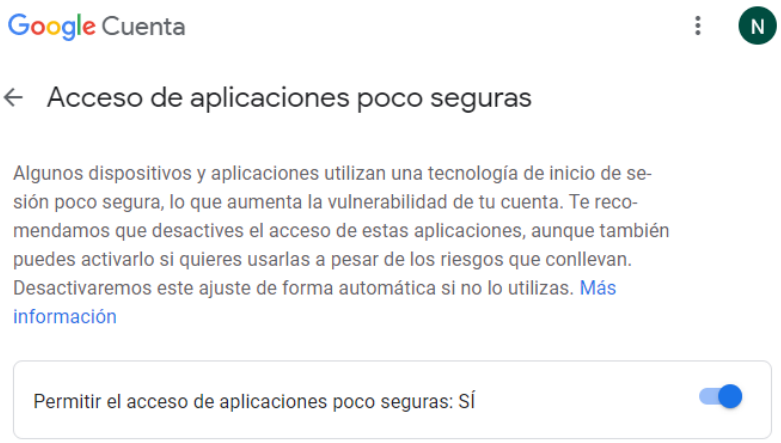

Figura 33 - Acceso de aplicaciones poco seguras Google

Envío de mensaje de prueba

Antes de continuar con los siguientes pasos, se va a probar el servicio de mensajería para comprobar si está bien instalado. Para ello, se va a enviar un mensaje a la cuenta de correo personal *(Figura 34)*:

```
# echo "Prueba Postfix Eder" | mail -s "Mensaje de prueba del
servidor Postfix – Nagios Core" elopez097@ikasle.ehu.eus
```

Figura 34 - Captura de correo de prueba

Configuración en Nagios

Tras comprobar que el servidor Postfix puede mandar mensajes de correo mediante el servidor smtp.gmail.com, se procede a configurar Nagios para que mande notificaciones por correo a los miembros del grupo I2T.

Antes de nada, se va a comprobar que en el archivo *commands.cfg* están definidos los comandos *notify-host-by-email* y *notify-service-by-email*. De no ser así, se definirán de la siguiente manera:

```
# sudo systemctl stop nagios
# sudo nano /usr/local/nagios/etc/objects/commands.cfg
```

```
# Definición de comando 'notify-host-by-email'
define command{
 command_name notify-host-by-email
 command_line /usr/bin/printf "%b" "***** Nagios Core I2T *****\n\nNotification Type: $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" | /usr/bin/mailx -s "** $NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ **" $CONTACTEMAIL$
 }

# Definición de comando 'notify-service-by-email'
define command{
 command_name notify-service-by-email
 command_line /usr/bin/printf "%b" "***** Nagios Core I2T *****\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional Info:\n\n$SERVICEOUTPUT$\n" | /usr/bin/mailx -s "** $NOTIFICATIONTYPE$ Service Alert: $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **" $CONTACTEMAIL$
 }
```

En este paso lo único que se ha hecho ha sido modificar la primera línea del comando para que aparezca “Nagios Core I2T” y cambiar la parte */usr/bin/mail* de cada comando por */usr/bin/mailx*. Este cambio se ha realizado porque *mail* es un programa mucho más antiguo y *mailx* es el que está estandarizado con Postfix [69].

Tras esto, se van a configurar los contactos de los miembros del laboratorio I2T. Para esto, se va a editar el archivo */usr/local/nagios/etc/objects/contacts.cfg* y se van a añadir los miembros a los que se quiere notificar tal y como se puede apreciar en la *Figura 35*.

```
define contact {

    contact_name            ederlopez               ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (defined above)
    alias                   Eder Lopez              ; Full name of user
    email                   elopez097@ikasle.ehu.eus ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}

define contact {

    contact_name            jasoneastorga           ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (defined above)
    alias                   Jasone Astorga          ; Full name of user
    email                   jasone.astorga@ehu.eus  ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}

define contact {

    contact_name            eduardojacob            ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (defined above)
    alias                   Eduardo Jacob           ; Full name of user
    email                   eduardo.jacob@ehu.eus   ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}
```

Figura 35 - Definición de contactos en contacts.cfg

Por último, hay que definir los tipos de notificaciones. Nagios nos informa de forma predeterminada cuando hay algún problema en los hosts o servicios. Nos notifica de todos los *Warning, Critical, Recovery* o *Unknown* que suceden en los servicios, y esto puede ser un problema. Al tener una red de unos 30 equipos como la del laboratorio I2T, toda esta cantidad de notificaciones puede suponer una carga innecesaria para el equipo de monitorización y para el servidor de correo, además de todos los mensajes que se acumularían en las bandejas de entrada de los miembros del laboratorio.

Para evitar esto, se van a configurar las notificaciones de los hosts y los servicios editando el archivo */usr/local/nagios/etc/objects/templates.cfg* para marcar que solo se manden las notificaciones *Down (d)* y *Recovery (r*) en el caso de los hosts, y en el caso de los servicios solo la de *Critical (c)*. En la *Figura 36* se pueden ver los cambios que se han realizado:

```
define host {

    name                            equipo-linux
    use                             generic-host
    check_period                    24x7
    check_interval                  5
    retry_interval                  1
    max_check_attempts              10
    check_command                   check-host-alive
    notification_period             24x7


    notification_interval           30
    notification_options            d,r
    contact_groups                  admins
    register                        0
    hostgroups                      equipos-linux
}
```

```
define service {

    name                            generic-service
    active_checks_enabled           1
    passive_checks_enabled          1
    parallelize_check               1
    obsess_over_service             1
    check_freshness                 0
    notifications_enabled           1
    event_handler_enabled           1
    flap_detection_enabled          1
    process_perf_data               1
    retain_status_information       1
    retain_nonstatus_information    1
    is_volatile                     0
    check_period                    24x7
    max_check_attempts              3
    check_interval                  10
    retry_interval                  2
    contact_groups                  admins
    notification_options            c
    notification_interval           30
    notification_period             24x7
    register                        0
}
```

Figura 36 - Configuración de las notificaciones en templates.cfg

Tras esto, se comprueba que todos los archivos de configuración están correctos y se reinicia el servicio Nagios:

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

8.2.2.10. Notificaciones por Telegram

Las notificaciones por correo son una función muy útil, pero durante la jornada laboral no se suele prestar mucha atención al correo electrónico, y por ello, puede que alguna notificación importante se pase por alto, con los problemas que ello conlleva: averías extendidas a lo largo del tiempo, servicios no operativos durante horas, etcétera.

Para evitar esto, se va a implementar un servicio de notificaciones mediante la aplicación de mensajería instantánea Telegram. Esto se va a hacer como apoyo a las habituales notificaciones por correo electrónico usando la función de los Bots de Telegram.

Lo primero que hay que hacer es crear el Bot que va a realizar las notificaciones. Para ello, se va a utilizar el Bot "BotFather". Este Bot se encarga de crear otros Bots de una manera fácil e intuitiva. El Bot es el que aparece en la *Figura 37*:

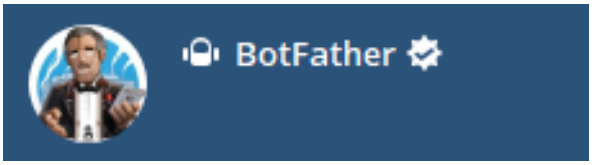

Figura 37 - Bot BotFather

Para ello, vamos a hablar a *@botfather* y mediante el comando */newbot* le vamos a pedir que cree un nuevo Bot. Tras esto, nos pedirá el nombre del Bot que se quiere crear. El Bot que va a realizar las notificaciones de Nagios se va a llamar *Nagios I2T*. Tras esto, nos pedirá que creemos un nombre de usuario para ese Bot. El nombre del Bot va a ser *nagios_i2t_bot*. Estos pasos los podemos ver en la *Figura 38:*

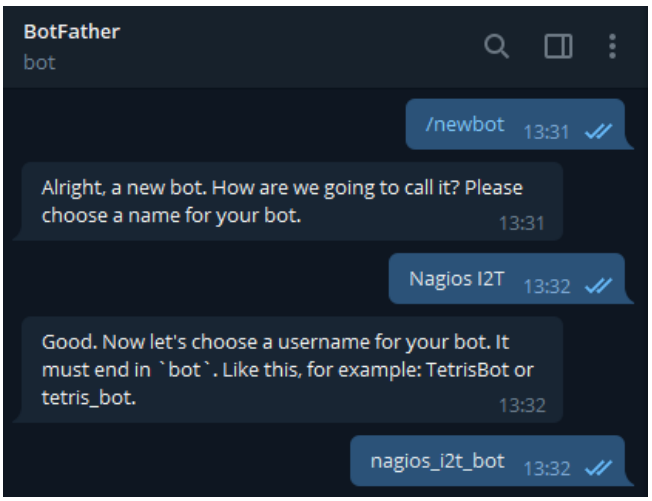

Figura 38 - Creación de nuevo Bot con BotFather

Tras esto, el Bot está creado y se puede encontrar en *t.me/nagios_i2t_bot* y se puede acceder a él mediante la API de Telegram [70]. Es importante guardar el Token y el UserID del Bot para utilizarlo más adelante. En este caso *(Figura 39),* el Token y UserID son los siguientes:

- Token: 1667355960:AAG2aARtFYQ_brHdmtH-La3-C9irLOOpQQw
- UserID: 1667355960

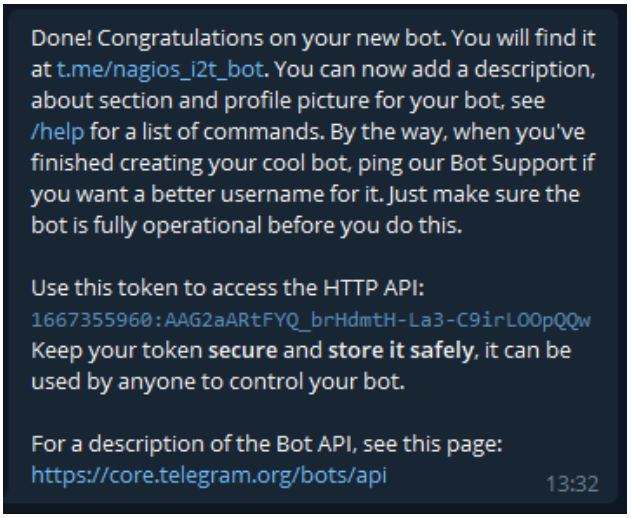

Figura 39 - Datos del bot Nagios_i2t_bot

Antes de empezar a utilizarlo, se han realizado unas configuraciones básicas. Se ha modificado la descripción, la imagen de perfil y la privacidad del Bot. En la descripción se ha especificado que está creado para que lo usen los miembros del grupo I2T, como imagen se ha puesto el logo de Nagios y en privacidad se ha seleccionado ENABLED, pues de esta forma, el Bot solo responde a los mensajes que empiezan por comandos con el símbolo "/". De todas formas, el Bot solo va a enviar notificaciones, no va a aceptar ningún comando, puesto que no tiene ninguno programado. Para esto, se han utilizado los comandos */setdescription*, */setuserpic* y */setprivacy*.

Tras esto, es necesario crear el grupo de Telegram donde van a estar metidos todos los integrantes que vayan a recibir las notificaciones de Nagios. Se añaden los integrantes y el Bot *@GroupIDbot*. Se manda el comando */id* por el grupo y el Bot nos responderá con el ID del grupo. Tras esto, se puede expulsar del grupo al Bot *@GroupIDbot*.

En el ejemplo de la *Figura 40* se puede observar un grupo de prueba que ha sido creado para configurar las notificaciones de Telegram. En este caso el ID ha sido el que se ve en la *Figura 40*, pero a la hora de implementar las notificaciones en los equipos del grupo I2T, habrá que especificar el ID del grupo que corresponda.

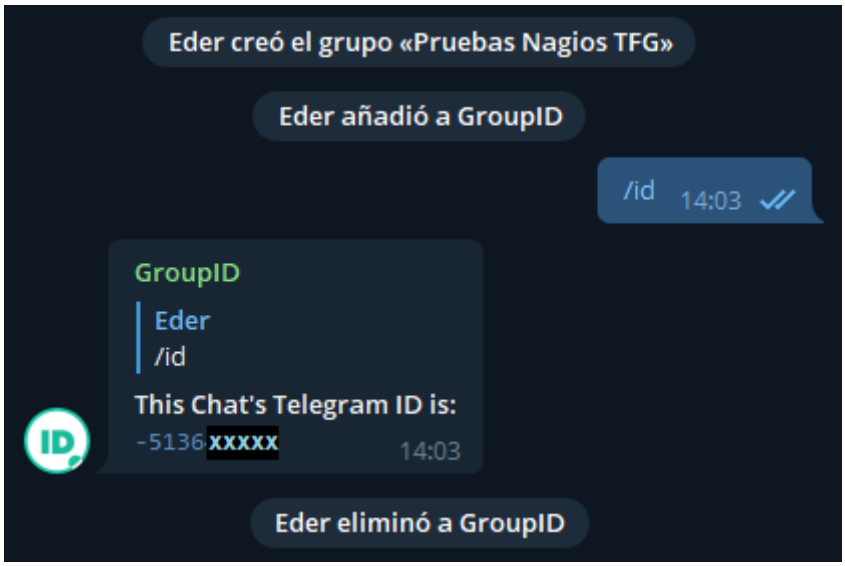

Figura 40 - ID de grupo de prueba de Telegram

- ChatID: -5136XXXXX

Después, hay que instalar *Curl* en el equipo, ya que se va a utilizar para enviar las notificaciones de Telegram. Para ello, se van a ejecutar los siguientes comandos:

```
# sudo apt-get update
# sudo apt install curl
```

Tras esto, tan solo queda modificar los archivos de configuración de Nagios para que mande las notificaciones a la API. Para ello se van a añadir los comandos necesarios en *commands.cfg*, se van a añadir contactos en *contacts.cfg* y se van a configurar las opciones de notificaciones en los hosts y servicios.

Para empezar, se añade el comando *notify-service-by-telegram* poniendo el Token después de *https://api.telegram.org/bot* en el archivo *commands.cfg*.

```
# sudo nano /usr/local/nagios/etc/objects/commands.cfg
```

```
define command{
  command_name    notify-host-by-telegram
  command_line    /usr/bin/curl -X POST --data chat_id=$CONTACTPAGER$ --data
text="%0ATipo de notificación: $NOTIFICATIONTYPE$%0A%0AHost:
$HOSTALIAS$%0ADirección IP: $HOSTADDRESS$%0AEstado: $HOSTSTATE$%0AFecha:
$LONGDATETIME$%0A%0AInformación
adicional:%0A%0A$HOSTOUTPUT$%0A$NOTIFICATIONCOMMENT$%0A"
https://api.telegram.org/bot1667355960:AAG2aARtFYQ_brHdmtH-La3-
C9irLOOpQQw/sendMessage
}
```

```
define command{
  command_name     notify-service-by-telegram
  command_line     /usr/bin/curl -X POST --data chat_id=$CONTACTPAGER$ --data
text="%0ATipo de notificación: $NOTIFICATIONTYPE$%0A%0AServicio:
$SERVICEDESC$%0ADirección IP: $HOSTADDRESS$%0AEstado: $SERVICESTATE$%0AFecha:
$LONGDATETIME$%0A%0AInformación
adicional:%0A$SERVICEOUTPUT$%0A$NOTIFICATIONCOMMENT$%0A"
https://api.telegram.org/bot1667355960:AAG2aARtFYQ_brHdmtH-La3-
C9irLOOpQQw/sendMessage
}
```

Estas definiciones de comandos son muy similares a las que se pueden ver en las notificaciones por correo electrónico, solo que hay que tener en cuenta la variación de los caracteres entre las notificaciones por correo y mediante la API de Telegram. En Telegram, para marcar un salto de línea la combinación de caracteres que hay que utilizar es "%0A" en vez de la tradicional "\n".

Para personalizar las notificaciones, tan solo hay que añadir dentro de las comillas de *text=""* lo que se quiere mostrar en la notificación utilizando los macros que ofrece Nagios Core [71].

Después de esto, se va a crear una nueva definición en la plantilla de comandos *(*Figura 41*)* para usarla con los contactos de Telegram, se va a crear un nuevo contacto *(*Figura 42*)* y se va a añadir al grupo *admins (Figura 43)* para que reciba notificaciones:

```
# sudo nano /usr/local/nagios/etc/objects/templates.cfg
```

```
define contact {

    name                            generic-telegram
    service_notification_period     24x7
    host_notification_period        24x7
    service_notification_options    w,c,r
    host_notification_options       d,r
    service_notification_commands   notify-service-by-telegram
    host_notification_commands      notify-host-by-telegram
    register                        0
}
```

Figura 41 - Definición de plantilla de contacto para Telegram

```
# sudo nano /usr/local/nagios/etc/objects/contacts.cfg
```

```
define contact {
  contact_name                    TelegramI2T
  use                             generic-telegram
  pager                           -5136XXXXX
}
```

Figura 42 - Definición de contacto para Telegram

```
define contactgroup {

    contactgroup_name       admins
    alias                   Nagios Administrators
    members                 nagiosadmin,ederlopez,TelegramI2T
}
```

Figura 43 - Inclusión de contacto TelegramI2T en el grupo de contactos

Tras esto, tan solo queda reiniciar Nagios para que se apliquen los cambios:

```
# sudo systemctl restart nagios
```

Después de esto, se añade el Bot *Nagios I2T* al grupo en el que se quiera recibir las notificaciones y cuando Nagios notifique de alguna incidencia al correo electrónico de los administradores, también lo hará al grupo de Telegram, tal como se puede ver en la *Figura 44*.

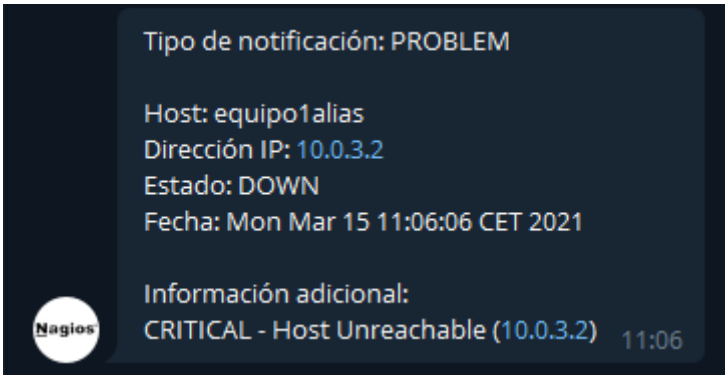

Figura 44 - Ejemplo de notificación en Telegram por Nagios_i2t_bot

Con todo esto, quedan totalmente configuradas las notificaciones en Telegram en castellano y con opción a visualizar los comentarios introducidos al realizar una notificación personalizada desde la interfaz de Nagios Core. Prueba de esto es la *Figura 45*, en la que se puede apreciar que la última línea de la notificación es personalizada.

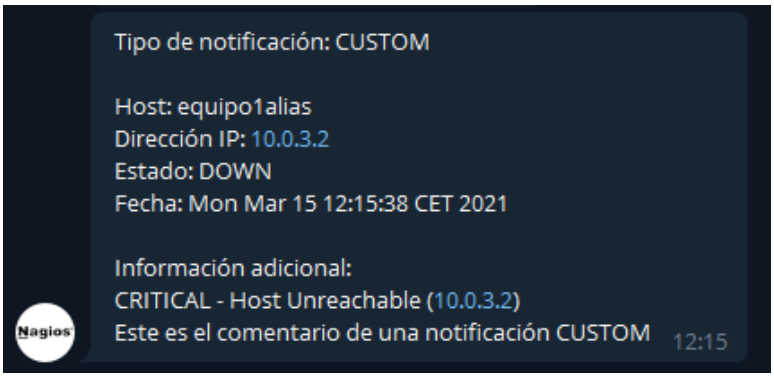

Figura 45 - Ejemplo de notificación personalizada en Telegram por Nagios_i2t_bot

8.2.2.11. Instalación de PNP4Nagios

En este apartado se va a realizar la instalación de PNP4Nagios siguiendo la guía oficial de Nagios [72]. Para empezar, se va a realizar una actualización de los repositorios y después se van a instalar las dependencias de PNP4Nagios.

```
# sudo apt-get update
# sudo apt-get install -y rrdtool librrds-perl php-gd php-xml
```

Tras esto, se va a descargar, descomprimir, compilar e instalar la última versión de PNP4Nagios. En el momento de realizar este trabajo la última versión es la v0.6.26.

```
# cd /tmp
# wget -O pnp4nagios.tar.gz
https://github.com/lingej/pnp4nagios/archive/0.6.26.tar.gz
# tar xzf pnp4nagios.tar.gz
# cd pnp4nagios-0.6.26
# sudo ./configure --with-httpd-conf=/etc/apache2/sites-enabled
# sudo make all
# sudo make install
# sudo make install-webconf
# sudo make install-config
# sudo make install-init
```

Ahora se va a configurar y arrancar el servicio para que arranque de forma automática en el inicio del sistema.

```
# sudo systemctl daemon-reload
# sudo systemctl enable npcd.service
# sudo systemctl start npcd.service
# sudo systemctl restart apache2.service
```

Después, se van a realizar las configuraciones necesarias en los archivos de configuración de Nagios. Se va a editar el valor de la línea *process_performance_data=0* y se va a poner a 1. El resto de líneas se van a copiar al final del documento porque por defecto están comentadas y no van a generar ningún problema.

```
# sudo nano /usr/local/nagios/etc/nagios.cfg
```

```
process_performance_data=1

host_perfdata_file=/usr/local/pnp4nagios/var/host-perfdata
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::$
HOSTNAME$\tHOSTPERFDATA::$HOSTPERFDATA$\tHOSTCHECKCOMMAND::$HOSTCHECKC
OMMAND$\tHOSTSTATE::$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$
host_perfdata_file_mode=a
host_perfdata_file_processing_interval=15
host_perfdata_file_processing_command=process-host-perfdata-file-bulk-npcd

service_perfdata_file=/usr/local/pnp4nagios/var/service-perfdata
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::$TIMET$\tHOSTNA
ME::$HOSTNAME$\tSERVICEDESC::$SERVICEDESC$\tSERVICEPERFDATA::$SERVICEPERFDAT
A$\tSERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$\tHOSTSTATE::$HOSTSTATE$\t
HOSTSTATETYPE::$HOSTSTATETYPE$\tSERVICESTATE::$SERVICESTATE$\tSERVICESTATETYPE
::$SERVICESTATETYPE$
service_perfdata_file_mode=a
service_perfdata_file_processing_interval=15
service_perfdata_file_processing_command=process-service-perfdata-file-bulk-npcd
```

Tras esto, hace falta definir dos comandos en el archivo *commands.cfg*.

```
# sudo nano /usr/local/nagios/etc/objects/commands.cfg
```

```
define command {
  command_name    process-service-perfdata-file-bulk-npcd
  command_line    /bin/mv /usr/local/pnp4nagios/var/service-perfdata
/usr/local/pnp4nagios/var/spool/service-perfdata.$TIMET$
  }

define command {
  command_name    process-host-perfdata-file-bulk-npcd
  command_line    /bin/mv /usr/local/pnp4nagios/var/host-perfdata
/usr/local/pnp4nagios/var/spool/host-perfdata.$TIMET$
  }
```

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

# sudo systemctl restart nagios.service
```

Ahora se puede comprobar la correcta instalación de PNP4Nagios en la siguiente dirección: *http://127.0.0.1/pnp4nagios/.* En esa dirección debe aparecer un listado con todos los elementos en verde, los cuales significan que PNP4Nagios tiene todos los requisitos necesarios para funcionar adecuadamente. Tras verificar esto, se puede eliminar el archivo *install.php* y refrescar el navegador para ver las gráficas que se están generando con PNP4Nagios. Dichas gráficas se pueden observar en la *Figura 46.*

```
# sudo rm -f /usr/local/pnp4nagios/share/install.php
```

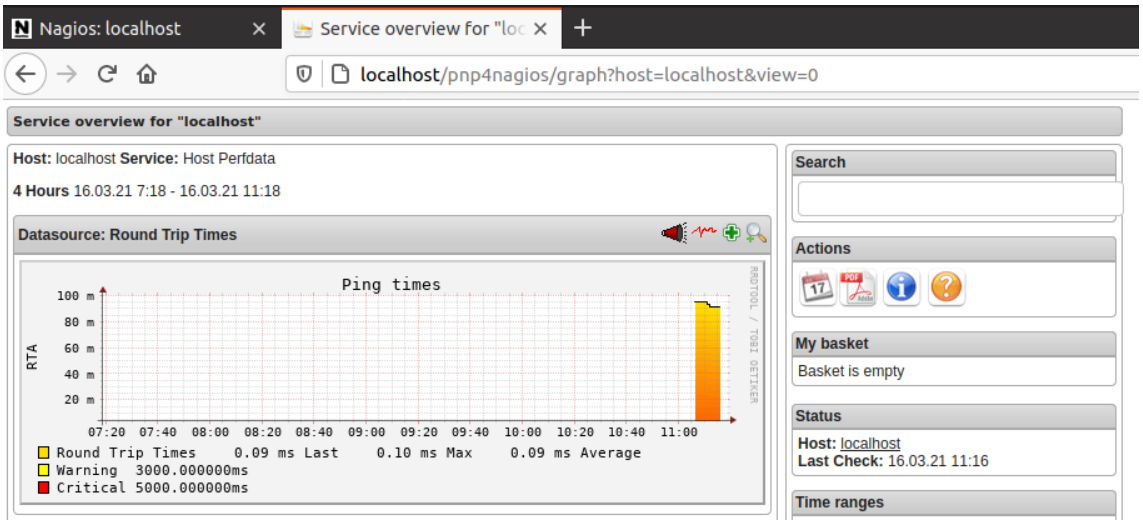

Figura 46 - Captura de PNP4Nagios en Localhost

Ahora, se va a configurar la integración en la interfaz web de Nagios Core. Para ello, se van a definir los siguientes hosts en la plantilla de *templates.cfg*. La razón de esto, es que Nagios utiliza la directiva *action_url* en las definiciones de objetos para proveer de un link o icono en la interfaz web.

```
# sudo nano /usr/local/nagios/etc/objects/templates.cfg
```

```
define host {
  name      host-pnp
  action_url /pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=_HOST_
  register  0
}

define service {
  name      service-pnp
  action_url /pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=$SERVICEDESC$
  register  0
}
```

Ahora, para utilizar estas plantillas, es necesario incluirlas en las definiciones de cada host o servicio en el que queramos que aparezcan los links a las gráficas de PNP4Nagios. Como se quiere que se muestren gráficas en todos los hosts y servicios, se van a editar directamente las plantillas de los hosts y servicios *(Figura 47 yFigura* 48*)* y se van a añadir las líneas “use host-pnp” y “use service-pnp”.

```
# sudo nano /usr/local/nagios/etc/objects/templates.cfg
```

```
define host {

    name                            generic-host
    notifications_enabled           1
    event_handler_enabled           1
    flap_detection_enabled          1
    process_perf_data               1
    retain_status_information       1
    retain_nonstatus_information    1
    notification_period             24x7
    register                        0
    use                             host-pnp
}
```

Figura 47 - Inclusión de host-pnp en la plantilla de hosts

```
define service {

    name                            generic-service
    active_checks_enabled           1
    passive_checks_enabled          1
    parallelize_check               1
    obsess_over_service             1
    check_freshness                 0
    notifications_enabled           1
    event_handler_enabled           1
    flap_detection_enabled          1
    process_perf_data               1
    retain_status_information       1
    retain_nonstatus_information    1
    is_volatile                     0
    check_period                    24x7
    max_check_attempts              3
    check_interval                  10
    retry_interval                  2
    contact_groups                  admins
    notification_options            c
    notification_interval           60
    notification_period             24x7
    register                        0
    use                             service-pnp
}
```

Figura 48 - Inclusión de service-pnp en la plantilla de servicios

```
# sudo /usr/local/nagios/bin/nagios -v
/usr/local/nagios/etc/nagios.cfg
# sudo systemctl restart nagios.service
```

Tras esto, ya se pueden ver las gráficas generadas con PNP4Nagios en la interfaz web de Nagios Core *(Figura 49)* haciendo clic en el icono  de cada host o servicio:

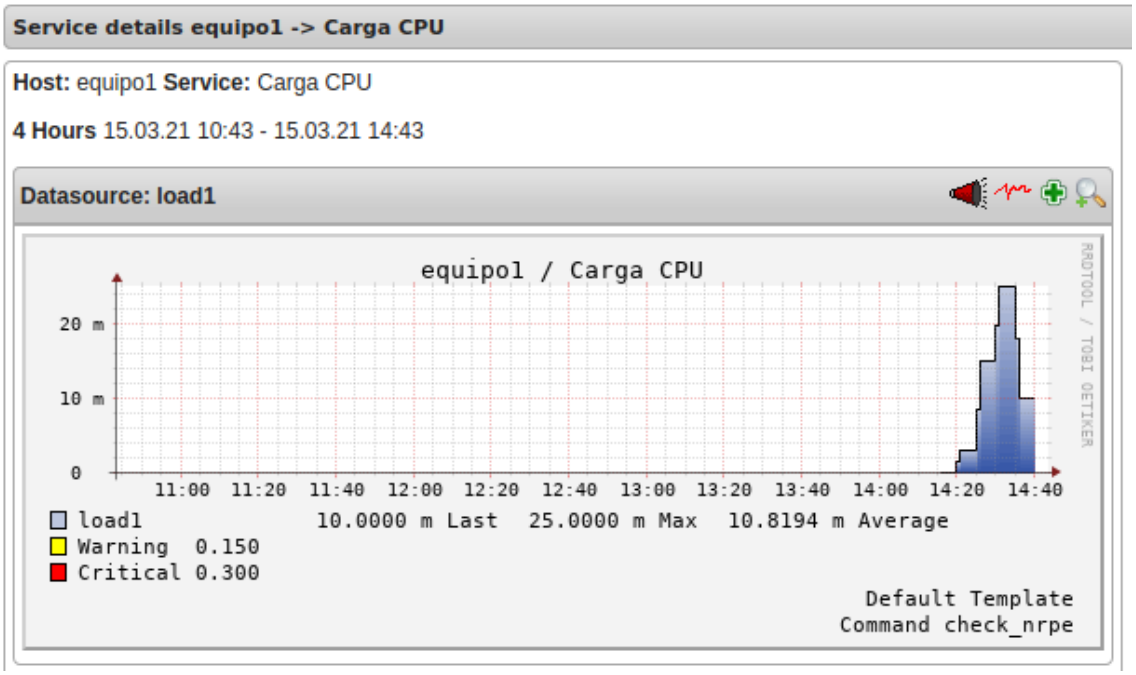

Figura 49 - Carga de CPU en localhost con PNP4Nagios

8.2.2.12. Instalación y configuración de Grafana

En este apartado de se va a realizar la instalación de Grafana en el equipo de monitorización. En la guía de instalación oficial de Grafana [73] hay disponibles varias versiones para instalar. Entre ellas están Grafana OSS, Grafana OSS (Beta), Grafana Enterprise y Grafana Enterprise (Beta). La gran diferencia entre Grafana OSS y Enterprise es que la versión OSS es totalmente Open Source y con la versión Enterprise hay que pagar una suscripción para conseguir los plugins de pago y el mantenimiento que nos ofrece la empresa.

En este caso se va a proceder a instalar la versión estable y Open Source de Grafana, es decir, la versión Grafana OSS. Para ello, primero se van a instalar las dependencias, se va a añadir el repositorio de las versiones Open Source estables y se va a realizar la instalación.

```
# sudo apt-get update
# sudo apt-get install -y apt-transport-https
# sudo apt-get install -y software-properties-common wget
# wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key
add -
# echo "deb https://packages.grafana.com/oss/deb stable main" |
sudo tee -a /etc/apt/sources.list.d/grafana.list
# sudo apt-get update
# sudo apt-get install grafana
```

Ahora, se procede a iniciar el servidor de Grafana:

```
# sudo systemctl daemon-reload
# sudo systemctl start grafana-server
# sudo systemctl status grafana-server
# sudo systemctl enable grafana-server.service
```

Tras estos pasos, Grafana está correctamente instalado en el equipo. Ahora, siguiendo los pasos de la página oficial de soporte de Nagios [74], se va a proceder a configurar Grafana para que trabaje con PNP4Nagios.

Antes de nada, se va a abrir el puerto 3000/TCP. Este es el puerto que utiliza la interfaz web de Grafana.

```
# sudo ufw allow 3000/tcp
# sudo ufw reload
```

Después, se van a instalar los componentes necesarios para que PNP4Nagios sea compatible con Grafana.

```
# sudo grafana-cli plugins install sni-pnp-datasource
# sudo systemctl restart grafana-server.service
# cd /usr/local/pnp4nagios/share/application/controllers/
# sudo wget -O api.php "https://github.com/lingej/pnp-metrics-
api/raw/master/application/controller/api.php"
```

Por último, para terminar con la instalación, se le van a dar los permisos necesarios a PNP4Nagios y se va a reiniciar el servidor Apache para que los cambios surtan efecto.

Se va a editar el archivo */etc/apache2/sites-enabled/pnp4nagios.conf* y se va a añadir la siguiente línea:

```
# sudo sh -c "sed -i '/Require valid-user/a\        Require ip
127.0.0.1 ::1' /etc/apache2/sites-enabled/pnp4nagios.conf"
# sudo systemctl restart apache2.service
```

Tras esto, la instalación de Grafana ha terminado. Ahora es necesario realizar la configuración de forma correcta. Para ello, se va a acceder a la interfaz web de Grafana en la URL *http://localhost:3000* y aparecerá lo que se muestra en la *Figura 50*:

Figura 50 - Log in de Grafana

El usuario por defecto es *admin* y la contraseña *admin*. Tras introducirlo, aparecerá un aviso para cambiar la contraseña. Tras eso, aparece la homepage de Grafana y dentro de ella, hay que hacer clic en el icono de engranaje de la barra de navegación para desplegar los ajustes. Se selecciona *Data Sources* y *Add data source (Figura 51 y Figura 52).*

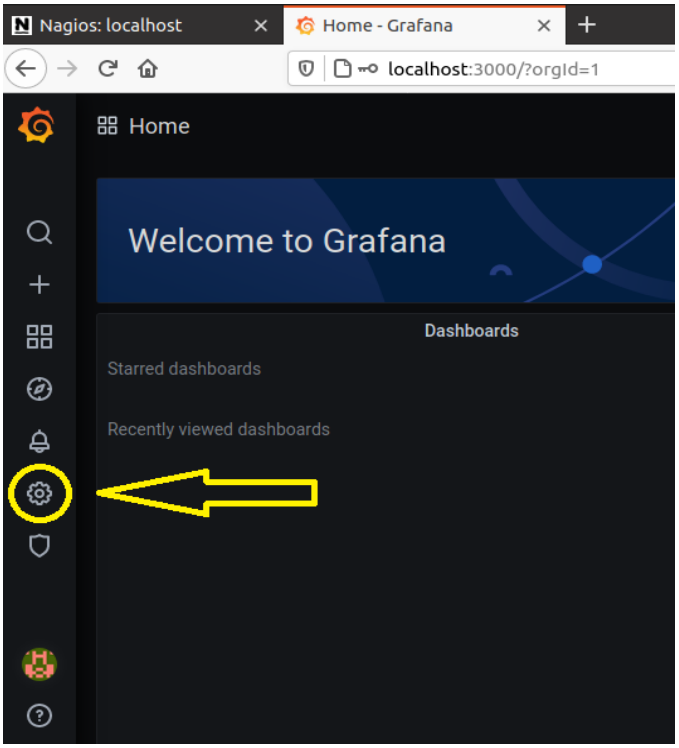

Figura 51 - Página principal de Grafana

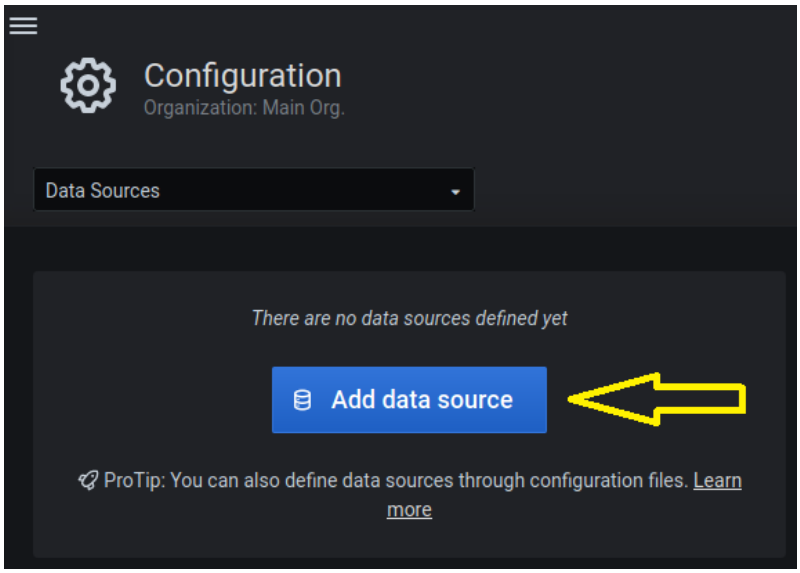

Figura 52 - Grafana Add data source

Haciendo clic en el icono, aparece otra pantalla en la que hay varias opciones *Figura 53*. Hay que seleccionar la opción de PNP y rellenar los campos que aparezcan a continuación con los datos de debajo y hacer clic en el botón para guardar y probar:

- HTTP
    - URL: http://localhost/pnp4nagios
    - Access: Server (default)
- Auth: Dejar los ajustes por defecto, nada seleccionado.

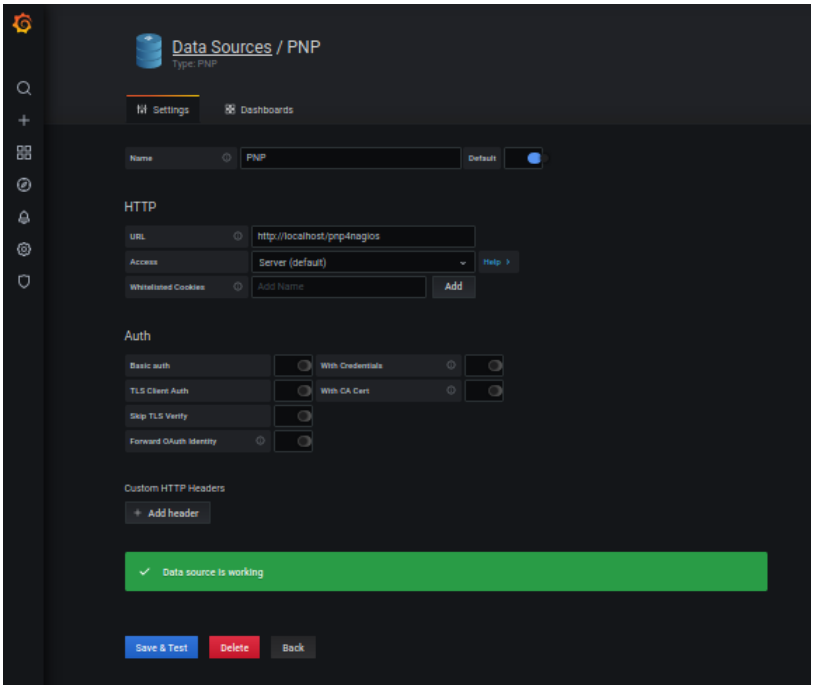

Figura 53 - Configuración correcta de fuente de datos PNP

Si como en el caso de la *Figura 53*, aparece que todo está funcionando correctamente, se puede continuar con los pasos para crear un panel y sus respectivos gráficos.

En el menú de la izquierda hay que hacer clic en el icono de crear (+) y en las opciones que da, hay que seleccionar *Dashboard*. Ahora, hay que escoger la opción *Add new panel*, y tras esto se obtiene un gráfico vacío al que hay que añadirle datos.

Para añadir los datos, tan solo hay que editar los campos que aparecen en pantalla. En la parte de abajo hay una lista de datos añadidos por defecto. Esos datos se pueden editar y añadir los que se desee.

En la *Figura 54* se muestra cómo se crea el gráfico de la carga de CPU del equipo local. Este parámetro necesita cargar tres fuentes de datos para poder crear el gráfico. Para cargar las fuentes de datos se hace lo siguiente:

- Clic en *Select Host* y selecciona *localhost*
- Clic en *Select Service* y selecciona *Current Load*
- Clic en *Select Performance Label* y selecciona *load1*
- Clic en *Type* y selecciona *Average*

- Clic en *Add Query* y repetir los pasos anteriores para *load5* y *load15*
- Poner el título *Carga CPU Localhost*

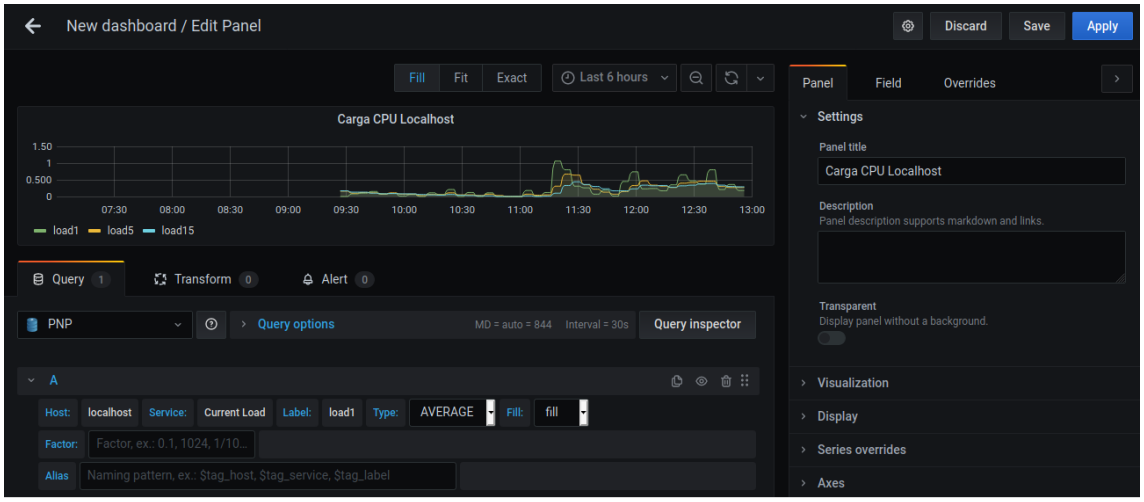

Figura 54 - Edición de panel en Grafana

Ahora, se cierra el editor dándole a *Apply* y el nuevo gráfico deberá aparecer en pantalla. Si se desea editar algo se puede volver a entrar al editor y realizar las modificaciones necesarias; si no, se puede guardar dándole al botón *Save* en la esquina superior derecha de la pantalla.

Por último, se va a modificar la homepage de Grafana para que muestre el tablero que se acaba de crear. Para ello, hay que ir a ajustes, preferencias y en Home Dashboard hay que seleccionar el Dashboard que se prefiera, en este caso, Carga CPU Localhost. Esto se puede ver en la *Figura 55:*

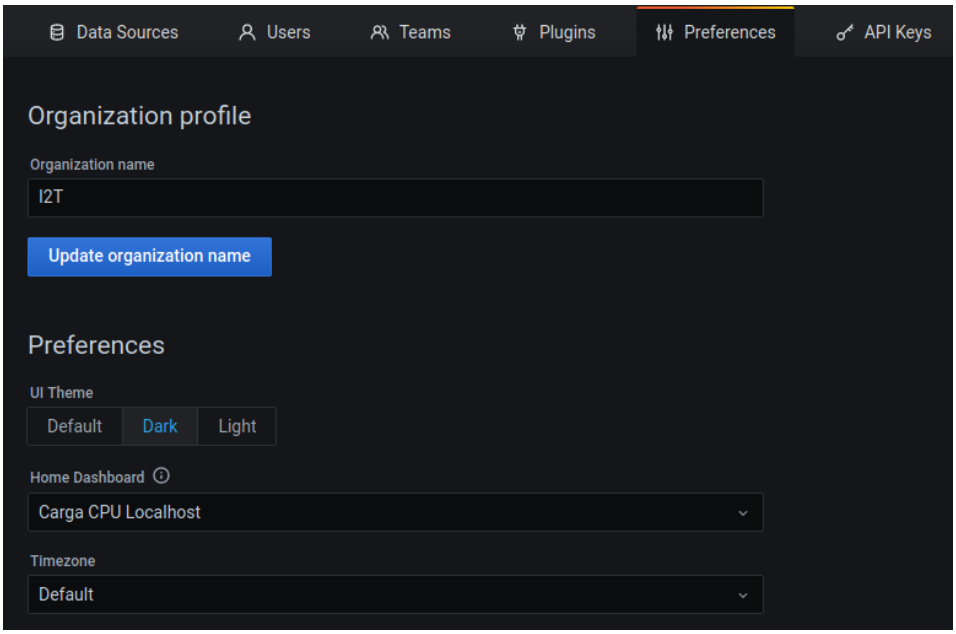

Figura 55 - Configuración de preferencias de Grafana

Tras realizar esa modificación, la homepage de Grafana queda como se puede ver en la *Figura 56*. Si se desean añadir más gráficos, tan solo hay que repetir los pasos citados anteriormente.

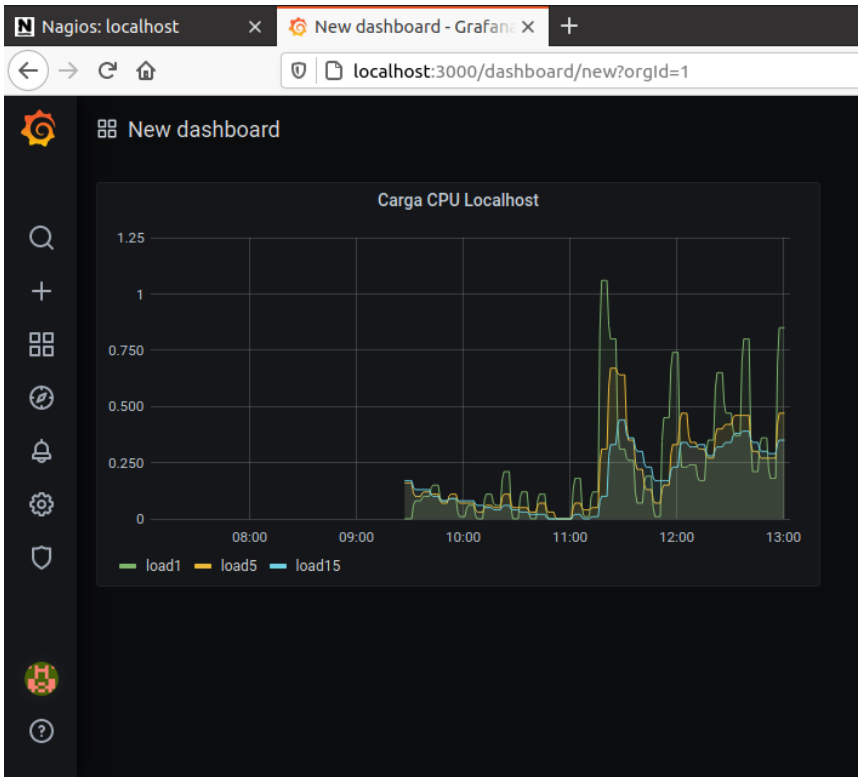

Figura 56 - Panel configurado Grafana

8.2.2.13. Definición de las impresoras

Para poder monitorizar las impresoras de la red, hay que realizar unos pasos muy similares a los ya realizados anteriormente con los hosts y servicios. El plugin encargado de monitorizar la impresora mediante SNMP es *check_hpjd* y *e*n este apartado se va a mostrar el proceso de configuración de Nagios para poder utilizarlo sin errores.

En este proceso, se va a crear una plantilla para las impresoras que se quieran monitorizar. Esta plantilla se va a basar en una plantilla ya existente definida en el archivo /usr/local/nagios/etc/objects/templates.cfg.

Antes de nada, hay que comprobar que el plugin *check_hpjd* está instalado en la ruta */usr/local/nagios/libexec* y que también las dependencias net-snmp y net-snmp-utils están instaladas, ya que sin ellas, será imposible su correcto funcionamiento.

Para comprobar que el plugin está instalado, basta con mirar en la carpeta correspondiente:

```
# ls /usr/local/nagios/libexec
```

En el caso en el que las dependencias no estén instaladas, hay que instalarlas y recompilar los plugins:

```
# sudo apt update
# sudo snap install net-snmp
# sudo snap install net-snmp-utils
# sudo apt install snmpd snmp libsnmp-dev

(+ repetir apartado 9.2.2. Instalacion de los plugins de Nagios)
```

Como esta es la primera vez que se configura una impresora, hay que hacer un par de cambios en la configuración de Nagios, para las siguientes impresoras no será necesario repetir estos pasos, únicamente la definición de cada dispositivo.

Hay que indicarle a Nagios Core que va a existir un directorio con las configuraciones de las impresoras, el directorio */usr/local/nagios/etc/printers/*. Para ello, hay que editar el archivo */usr/local/nagios/etc/nagios.cfg*, descomentar la línea de */printers/* y guardar los cambios:

Nota: Se deja el archivo printer.cfg comentado, ya que es una plantilla para las definiciones de las impresoras y sus servicios. Si se descomenta Nagios Core buscará la impresora definida por defecto en ese archivo y la marcará continuamente como DOWN, ya que no está en la red del laboratorio. Para evitar eso se puede poner "register 0" o dejar la línea comentada para que Nagios Core la ignore, como es el caso.

```
# sudo nano /usr/local/nagios/etc/nagios.cfg
```

```
#cfg_file=/usr/local/nagios/etc/objects/printer.cfg
…
cfg_dir=/usr/local/nagios/etc/printers
…
```

Para definir el host (la impresora) y sus servicios, se va a crear el directorio que se acaba de descomentar y el archivo /usr/local/nagios/etc/printers/hpljm601.cfg para definir la impresora partiendo de la plantilla *generic-printer*:

```
# sudo mkdir -p /usr/local/nagios/etc/printers
# sudo nano /usr/local/nagios/etc/printers/hp_laserjet_m601.cfg
```

```
define host {
   use             generic-printer    ; plantilla
   host_name       impr_hp_m601          ; Nombre corto de la impresora
   alias           HP LaserJet 600 M601  ; Nombre completo de la impresora
   address         10.20.30.50           ; Direccion IP de la impresora
   hostgroups      impresoras         ; Hostgroup de la impresora
}

define service {
   use                     generic-service                ; plantilla
   host_name               impr_hp_m601                   ; nombre corto de la impresora
   service_description     Estado de la impresora         ; descripción del servicio
   check_command           check_hpjd!-C public           ; comando
```

```
    check_interval          10                          ; intervalo en condiciones normales (mins)
    retry_interval          1                           ; intervalo cuando hay error (en minutos)
}

define service {
    use                     generic-service             ; plantilla
    host_name               impr_hp_m601                ; nombre corto de la impresora
    service_description     PING                        ; Descripcion del servicio
    check_command           check_ping!3000.0,80%!5000.0,100%       ; comando
    check_interval          10                          ; intervalo en condiciones normales (mins)
    retry_interval          1                           ; intervalo cuando hay error (en minutos)
}
```

Si se tienen varias impresoras o en un futuro se quieren añadir más, es interesante realizar la definición del hostgroup:

```
# sudo nano /usr/local/nagios/etc/objects/hostgroups.cfg
```

```
define hostgroup{
hostgroup_name          impresoras          ; Nombre del hostgroup
alias                   Impresoras          ; Nombre largo del hostgroup
}
```

Tras esto, tan solo queda verificar la configuración y reiniciar Nagios Core:

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

# sudo systemctl restart nagios
```

8.2.2.14. Instalación y configuración de los plugins de las impresoras

Tras definir las impresoras que se van a monitorizar, hay que instalar los plugins y definir los servicios que se van a utilizar. Para ello, se va a usar el plugin *check_snmp_printer* [75] de Nagios Exchange [76], el cual es compatible con multitud de impresoras de diferentes marcas.

Primero, se descarga el plugin en la máquina virtual local que se está usando para conectarse a la máquina virtual del laboratorio. Después, se copia el plugin a la máquina virtual del laboratorio usando SSH y se mueve a la librería de Nagios.

```
# scp /home/mvi2t/Downloads/check_snmp_printer
ubuntu@10.20.30.40:/home/ubuntu

# ssh ubuntu@10.20.30.40    (se accede a la MV del laboratorio)

# cd /home/ubuntu/

# sudo cp check_snmp_printer /usr/local/nagios/libexec/
```

Ahora, se va a cambiar los permisos para hacer el script ejecutable. Después, se definen los comandos para usar los plugins en el archivo */usr/local/nagios/etc/objects/commands.cfg*.

```
# sudo chmod +x /usr/local/nagios/libexec/check_snmp_printer

# sudo nano /usr/local/nagios/etc/objects/commands.cfg
```

```
#IMPRESORA HP LASERJET M601

define command {

 command_name    check_printer_status
 command_line    $USER1$/check_snmp_printer -H $HOSTADDRESS$ -t STATUS -C $ARG1$
}

define command {

 command_name    check_printer_blackimagingtoner
 command_line     $USER1$/check_snmp_printer -H $HOSTADDRESS$ -t CONSUM Black -C
$ARG1$
}

define command {

 command_name    check_printer_serial
 command_line    $USER1$/check_snmp_printer -H $HOSTADDRESS$ -t MODEL -C $ARG1$
}
```

Ahora, se van a definir los servicios editando el archivo de configuración *printer_hp_m601.cfg* y se van a usar los comandos recién definidos.

```
# sudo nano /usr/local/nagios/etc/printers/printer_hp_m601.cfg
```

```
# ESTADO

define service{
      use                     generic-service
      host_name               impr_hp_m601
      service_description     STATUS
      check_command           check_printer_status!public
      }

# TONER E IMAGING

define service{
      use                     generic-service
      host_name               impr_hp_m601
      service_description     TONER&IMAGING
      check_command           check_printer_blackimagingtoner!public
      }
```

```
# SERIAL NUMER Y MODELO DE IMPRESORA

define service{
      use                     generic-service
      host_name               impr_hp_m601
      service_description     INFO
      check_command           check_printer_serial!public
      }
```

Para finalizar, se comprueban los archivos de configuración y se reinicia el servicio de Nagios.

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

# sudo systemctl restart nagios
```

8.2.2.15. Definición de los Switches

Para definir los switches hay que seguir un proceso similar al apartado anterior. Hay que comprobar las dependencias (las cuales ya están comprobadas) e indicarle a Nagios Core la ruta donde van a estar los archivos de configuración de estos switches. Para ello, se descomenta la línea de configuración de los switches y se define cada switch en su correspondiente archivo de configuración:

```
# sudo nano /usr/local/nagios/etc/nagios.cfg
```

```
…
cfg_dir=/usr/local/nagios/etc/switches
…
```

Para definir cada uno de los switches y sus servicios, se crea el directorio para los switches y un archivo de configuración para cada uno de ellos en la ruta */usr/local/nagios/etc/switches/*. Para definir los switches se va a aprovechar la plantilla *generic-switch.* Primero se va a definir el switch *DELL PowerSwitch S3048-ON* y después el *HP Procurve 2848 J4904A*:

```
# sudo mkdir -p /usr/local/nagios/etc/switches
# sudo nano /usr/local/nagios/etc/switches/switch_dell_s3048on.cfg
```

```
define host {
   use             generic-switch            ; Plantilla
   host_           name   dell_s3048         ; Nombre corto
   alias           SWITCH DELL S3048ON   ; Nombre completo
   address         10.20.30.60                  ; Direccion IP del switch
   hostgroups   switches                  ; Hostgroup del switch
}
```

```
# PING
define service {
   use                     generic-service   ; Plantilla
   host_name               dell_s3048        ; Nombre corto
   service_description     PING              ; Descripcion del comando
   check_command           check_ping!3000.0,80%!5000.0,100% ; Comando
   check_interval          10                ; intervalo en condiciones normales
   retry_interval          1                 ; intervalo cuando hay error (en minutos)
}
# DESCRIPCION
define service{
      use                       generic-service
      host_name                 dell_s3048
      service_description       DESCRIPTION
      check_interval            5
      retry_interval            1
      check_command             check_powerconnect!public!description
      }
```

Ahora se va a definir el switch *HP Procurve 2848 J4904A*:

```
# sudo nano /usr/local/nagios/etc/switches/switch_hp_2848.cfg
```

```
define host {
   use             generic-switch              ; plantilla
   host_name       hp_2848                     ; Nombre corto del switch
   alias           HP Procurve 2848 J4904A     ; Nombre completo del switch
   address         10.20.30.70                 ; Direccion IP del switch
   hostgroups      switches                    ; Hostgroup del switch
}

# PING
define service {
   use                     generic-service     ; plantilla
   host_name               hp_2848             ; nombre corto del switch
   service_description     PING                ; Descripcion del servicio
   check_command           check_ping!3000.0,80%!5000.0,100% ;comando
   check_interval          10                  ; intervalo en condiciones normales
   retry_interval          1                   ; intervalo cuando hay error (en minutos)
}
```

Si se tienen varios switches como es el caso es interesante realizar la definición del hostgroup:

```
# sudo nano /usr/local/nagios/etc/objects/hostgroups.cfg
```

```
define hostgroup{
hostgroup_name          switches        ; Nombre del hostgroup
alias                   Switches        ; Nombre largo del hostgroup
}
```

Tras esto, tan solo queda verificar la configuración y reiniciar Nagios Core:

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

# sudo systemctl restart nagios
```

8.2.2.16. Instalación plugins Dell y HP

Una vez que ya se han definido los switches es necesario instalar los plugins y definir los servicios que se van a usar para monitorizarlos. Para ello, se van a descargar varios plugins de la página de Nagios Exchange [76] y se van a copiar al directorio oportuno de Nagios para su uso. Después de eso se van a definir los comandos y servicios que hacen uso de estos plugins.

Para empezar, se van a descargar los plugins en la máquina virtual local que se está usando para conectarse a la máquina virtual del laboratorio. Tras eso, se van a copiar los plugins a la máquina virtual del laboratorio mediante SSH y después se van a copiar a la librería de Nagios.

```
# scp /home/mvi2t/Downloads/check_powerconnect
ubuntu@10.20.30.40:/home/ubuntu

# scp /home/mvi2t/Downloads/check_snmp_hp-procurve.sh
ubuntu@10.20.30.40:/home/ubuntu


# ssh ubuntu@10.20.30.40   (se accede a la MV del laboratorio)

# cd /home/ubuntu/

# sudo cp check_powerconnect /usr/local/nagios/libexec/

# sudo cp check_snmp_hp-procurve.sh /usr/local/nagios/libexec/
```

Ahora, se van a cambiar los permisos de los scripts para hacerlos ejecutables y, tras eso, se van a definir los comandos para usar los plugins. Dicha definición se va a hacer en el archivo */usr/local/nagios/etc/objects/commands.cfg*.

```
# sudo chmod +x /usr/local/nagios/libexec/check_powerconnect

# sudo chmod +x /usr/local/nagios/libexec/check_snmp_hp-procurve.sh

# sudo nano /usr/local/nagios/etc/objects/commands.cfg
```

```
#HP PROCURVE

define command {

 command_name    check_snmp_hp-procurve
 command_line    $USER1$/check_snmp_hp-procurve.sh public $HOSTADDRESS$
}
```

```
define command{

 command_name            check_hpmemoryfree
 command_line            $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.5.1.1.2.1.1.1.6.1 -t 5 -w $ARG2$ -c $ARG3$ -u bytes -l free
}

define command{

 command_name            check_hp_cpu
 command_line            $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.5.1.9.6.1.0 -t 5  -w $ARG2$ -c $ARG3$ -u % -l "5min cpu"
}

define command{

 command_name            check_hpfan
 command_line            $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.1.2.6.1.4.1 -w $ARG2$ -c $ARG3$ -l 'Fan status'
}

define command{

 command_name            check_hppower
 command_line            $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.1.2.6.1.4.2 -w $ARG2$ -c $ARG3$ -l 'Power Supply status'
}

define command{

 command_name            check_info
 command_line            $USER1$/check_snmp -H 10.20.30.70 -C public -o .1.3.6.1.2.1.1.1.0
}


define command{

 command_name            check_hptemp
 command_line            $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.1.2.6.1.4.4 -w $ARG2$ -c $ARG3$ -l 'Temperature status'
}

# DELL POWERCONNECT

define command{

 command_name            check_powerconnect
 command_line            $USER1$/check_powerconnect -H $HOSTADDRESS$ -C $ARG1$ -t
$ARG2$ -w $ARG3$ -c $ARG4$
}
```

Ahora, se van a definir los servicios en el switch Dell. Para ello, se va a editar el archivo de configuración *switch_dell_s3048on.cfg* y se van a usar los comandos recién definidos.

```
# sudo nano /usr/local/nagios/etc/switches/switch_dell_s3048on.cfg
```

```
# VENTILADOR

define service{
        use                     generic-service
        host_name               dell_s3048
        service_description     FAN-STATUS
        check_interval          5
        retry_interval          1
        check_command           check_powerconnect!public!fans
        }

# FUENTE DE ALIMENTACION

define service{
        use                     generic-service
        host_name               dell_s3048
        service_description     PWR-SUPPLY
        check_interval          5
        retry_interval          1
        check_command           check_powerconnect!public!psus
        }

# PUERTOS

define service{
        use                     generic-service
        host_name               dell_s3048
        service_description     PORTS
        check_interval          5
        retry_interval          1
        check_command           check_powerconnect!public!ports
        }

# ENCENDIDO

define service{
        use                     generic-service
        host_name               dell_s3048
        service_description     UPTIME
        check_interval          5
        retry_interval          1
        check_command           check_powerconnect!public!uptime
        }
```

```
# SALUD

define service{
      use                         generic-service
      host_name                   dell_s3048
      service_description         HEALTH
      check_interval              5
      retry_interval              1
      check_command               check_powerconnect!public!health
      }

# DESCRIPCION

define service{
     use                          generic-service
     host_name                    dell_s3048
     service_description          DESCRIPTION
     check_interval               5
     retry_interval               1
     check_command                check_powerconnect!public!description
}
```

Ahora, se van a definir los servicios del switch HP. Para ello, se va a editar el archivo *switch_hp_2848.cfg*.

```
# sudo nano /usr/local/nagios/etc/switches/switch_hp_2848.cfg
```

```
# ESTADO HARDWARE

define service{
   use                      generic-service      ; plantilla
   host_name                hp_2848              ; nombre corto del switch
   service_description      HW STATUS            ; Descripcion del servicio
   check_command            check_snmp_hp-procurve    ;comando
   check_interval           10                   ; intervalo en condiciones normal>
   retry_interval           1                    ; intervalo cuando hay error (en >
}

# INFORMACION

define service{
   use                      generic-service     ; plantilla
   host_name                hp_2848             ; nombre corto del switch
   service_description      INFORMATION         ; Descripcion del servicio
   check_command            check_info
   check_interval           10                  ; intervalo en condiciones normal>
   retry_interval           1                   ; intervalo cuando hay error (en >
}
```

```
# TEMPERATURA CPU

define service{
        use                     generic-service
        host_name               hp_2848
        service_description     CPU-TEMP
        check_interval          5
        retry_interval          1
        check_command           check_hptemp!public!4!3:5
}

# VENTILADORES

define service{
        use                     generic-service
        host_name               hp_2848
        service_description     FAN-STATUS
        check_interval          5
        retry_interval          1
        check_command           check_hpfan!public!4!3:5
}

# MEMORIA LIBRE

define service{
        use                     generic-service
        host_name               hp_2848
        service_description     MEM-FREE
        check_interval          5
        retry_interval          1
        check_command           check_hpmemoryfree!public!2000:30000000!1000:30000000
}

# CPU

define service{
        use                     generic-service
        host_name               hp_2848
        service_description     CPU
        check_interval          5
        retry_interval          1
        check_command           check_hp_cpu!public!95:90!100:95
}
```

Por último, se comprueban los archivos de configuración y se reinicia el servicio de Nagios.

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

# sudo systemctl restart nagios
```

8.2.2.17. Definición del router Mikrotik

Para definir el router Mikrotik se van a realizar los mismos pasos que en los apartados anteriores. Se va a editar el archivo de configuración *nagios.cfg* y después se va a crear un archivo de configuración para el router. En ese archivo se va a definir el host y también los servicios.

```
# sudo nano /usr/local/nagios/etc/nagios.cfg
```

```
…
cfg_dir=/usr/local/nagios/etc/routers
…
```

Para definir la plantilla se va a editar el archivo */usr/local/nagios/etc/objects/templates.cfg* y se va a añadir lo siguiente:

```
define host {
   name                    generic-router
   use                     generic-host
   check_period            24x7
   check_interval          5
   retry_interval          1
   max_check_attempts      10
   check_command           check-host-alive
   notification_period     24x7
   notification_interval   30
   notification_options    d,r
   contact_groups          admins
   register                0
}
```

Para definir el router, se crea el directorio */usr/local/nagios/etc/routers/* y el archivo de configuración aprovechando la plantilla *generic-router.*

```
# sudo mkdir -p /usr/local/nagios/etc/routers
# sudo nano
/usr/local/nagios/etc/routers/router_mikrotik_CCR1036.cfg
```

```
define host {
   use             generic-router              ; Plantilla
   host_name       mikrotik_ccr1036                    ; Nombre corto del router
   alias           Mikrotik CloudCore CCR1036    ; Nombre completo del router
   address         10.20.30.80                 ; Direccion IP del router
   hostgroups      routers                      ; Hostgroup del router
}

# PING
define service {
   use                     generic-service         ; Plantilla
```

```
    host_name               mikrotik_ccr1036     ; Nombre corto del router
    service_description     PING                 ; Descripcion del comando
    check_command           check_ping!3000.0,80%!5000.0,100% ; Comando
    check_interval          10                   ; intervalo en condiciones normales
    retry_interval          1                    ; intervalo cuando hay error (en minutos)
}
```

En este caso solo se dispone de un router, pero aun y así se va a realizar la definición del hostgroup, ya que es interesante de cara a tener los equipos ordenados por grupos:

```
# sudo nano /usr/local/nagios/etc/objects/hostgroups.cfg
```

```
define hostgroup{
hostgroup_name          routers         ; Nombre del hostgroup
alias                   Routers         ; Nombre largo del hostgroup
}
```

Tras esto, tan solo queda verificar la configuración y reiniciar Nagios Core:

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

# sudo systemctl restart nagios
```

8.2.2.18. Instalación plugins Mikrotik

Una vez definido el router Mikrotik, se van a instalar los plugins y definir los servicios que se van a usar. Para ello, al igual que en los casos anteriores, se van a descargar los plugins de la página de Nagios Exchange [76] y se van a copiar a la biblioteca de plugins de Nagios. Tras eso, eso se van a definir los servicios y los comandos que hacen uso de esos plugins.

Para empezar, se descargan los plugins en la máquina virtual local y se copian a la máquina virtual del laboratorio mediante SSH. Después se copian a la librería de plugins de Nagios.

```
# scp /home/mvi2t/Downloads/check_mikrotik_mem
ubuntu@10.20.30.40:/home/ubuntu

# scp /home/mvi2t/Downloads/check_mikrotik_users
ubuntu@10.20.30.40:/home/ubuntu

# scp /home/mvi2t/Downloads/check_mikrotik_signal
ubuntu@10.20.30.40:/home/ubuntu

# scp /home/mvi2t/Downloads/check_snmp_mikrotik_env.pl
ubuntu@10.20.30.40:/home/ubuntu


# ssh ubuntu@10.20.30.40    (se accede a la MV del laboratorio)

# cd /home/ubuntu/
```

```
# sudo cp check_mikrotik_mem /usr/local/nagios/libexec/

# sudo cp check_mikrotik_users /usr/local/nagios/libexec/

# sudo cp check_mikrotik_signal /usr/local/nagios/libexec/

# sudo cp check_snmp_mikrotik_env.pl /usr/local/nagios/libexec/
```

Tras esto, se cambian los permisos de los plugins para hacerlos ejecutables.

```
# sudo chmod +x /usr/local/nagios/libexec/check_mikrotik_mem

# sudo chmod +x /usr/local/nagios/libexec/check_mikrotik_users

# sudo chmod +x /usr/local/nagios/libexec/check_mikrotik_signal

# sudo chmod +x
/usr/local/nagios/libexec/check_snmp_mikrotik_env.pl
```

Uno de los plugins está escrito en Python 3, por lo tanto, hay que instalar una librería de la que depende:

```
# sudo apt-get update
# sudo apt-get install python3-pysnmp4
```

Ese plugin no se va a descargar de Nagios Exchange sino de GitHub. Se va a crear un nuevo archivo en /usr/local/nagios/libexec/ y se va a hacer copia pega del contenido de GitHub del plugin check_mikrotik_os.py y se va a hacer ejecutable cambiando los permisos:

```
# sudo nano /usr/local/nagios/libexec/check_mikrotik_os.py
# sudo chmod +x /usr/local/nagios/libexec/check_mikrotik_os.py
```

Ahora, hay que definir los comandos que se van a utilizar en los servicios.

```
# sudo nano /usr/local/nagios/etc/objects/commands.cfg
```

```
define command{
   command_name    check_mikrotik_users
   command_line    $USER1$/check_mikrotik_users -H $HOSTADDRESS$
}

define command{
   command_name    check_mikrotik_signal
   command_line    $USER1$/check_mikrotik_signal -H $HOSTADDRESS$
}

define command{
   command_name    check_mikrotik_mem
   command_line    $USER1$/check_mikrotik_mem -H $HOSTADDRESS$
}
```

```
define command{
   command_name    check_mikrotik_temp
   command_line     $USER1$/check_snmp_mikrotik_env.pl -H $HOSTADDRESS$ -C public -t
temp -w $ARG1$ -c $ARG2$
}

define command{
   command_name    check_mikrotik_power
   command_line     $USER1$/check_snmp_mikrotik_env.pl -H $HOSTADDRESS$ -C public -t
power -w $ARG1$ -c $ARG2$
}

define command{
   command_name    check_mikrotik_update
   command_line      $USER1$/check_mikrotik_os.py -H $HOSTADDRESS$ -v 2c -C public -c
Current
}
```

Tras definir los comandos, se puede hacer la definición de servicios en el archivo de configuración del router Mikrotik:

```
# sudo nano
/usr/local/nagios/etc/routers/router_mikrotik_CCR1036.cfg
```

```
# USUARIOS

define service {
  use                     generic-service          ; Plantilla
  host_name               mikrotik_ccr1036         ; Nombre corto
  service_description     USERS                    ; Descripcion
  check_command           check_mikrotik_users     ; Comando
  check_interval          10                       ; Intervalo normal
  retry_interval          1                        ; Intervalo critico
}

# SEÑAL

define service {
  use                     generic-service          ; Plantilla
  host_name               mikrotik_ccr1036         ; Nombre corto
  service_description     SIGNAL                   ; Descripcion
  check_command           check_mikrotik_signal    ; Comando
  check_interval          10                       ; Intervalo normal
  retry_interval          1                        ; Intervalo critico
}

# MEMORIA

define service {
  use                     generic-service
```

```
  host_name               mikrotik_ccr1036
  service_description     MEMORY
  check_command           check_mikrotik_mem
  check_interval          10
  retry_interval          1
}

# TEMPERATURA

define service {
  use                     generic-service
  host_name               mikrotik_ccr1036
  service_description     TEMPERATURE
  check_command           check_mikrotik_temp!70!90
  check_interval          10
  retry_interval          1
}

# CONSUMO

define service {
  use                     generic-service
  host_name               mikrotik_ccr1036
  service_description     POWER
  check_command           check_mikrotik_power!350!400
  check_interval          10
  retry_interval          1
}

# ACTUALIZACIONES

define service {
   use                    generic-service
   host_name              mikrotik_ccr1036
   service_description    UPDATE
   check_command          check_mikrotik_update
   check_interval         1440        ; Se comprueba cada 24h
   retry_interval         1440        ; Si falla, se vuelve a comprobar a las 24h
   notification_interval  1440        ; Notificaciones cada 24h
}
```

Tras esto, se comprueban los archivos de configuración y se reinicia el servicio de Nagios.

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

# sudo systemctl restart nagios
```

9. PLAN DE PRUEBAS

En este apartado se van a tratar las diferentes pruebas que se han realizado para comprobar el correcto funcionamiento del sistema. Se van a definir diferentes pruebas y se van a analizar los resultados de ellas.

Hay que mencionar que la implementación del sistema se ha realizado paso a paso, comprobando que lo que se ha implementado funciona adecuadamente. Aún y así, se va a realizar este plan de pruebas para intentar buscar y corregir los posibles fallos que no hayan sido detectados durante el proceso de implementación.

Las pruebas se van a dividir en dos apartados, son los siguientes:

- Observaciones
- Pruebas reales desactivando servicios

9.1. Observaciones

Este apartado va a consistir en analizar los resultados obtenidos del sistema después de que haya estado completamente operativo durante un mes en el laboratorio. Se van a analizar los datos obtenidos y sacar conclusiones. Así pues, lo observado ha sido lo siguiente:

Accediendo al panel de control de Nagios Core mediante la IP 10.20.30.40 se puede observar que hay varias alertas WARNING y CRITICAL, tal y como se muestra en la *Figura 57*:

Figura 57 - Notificaciones de Nagios Core

De la *figura 57* podemos intuir que el router Mikrotik tiene una actualización dispone y que los servidores *web*, *DHCP*, *DNS* y *VPN* tienen demasiados procesos activos. Tras comprobar que la función UPDATE del router Mikrotik es correcta, se procede a comprobar el histórico de procesos de los servidores mencionados. Para ello, se accede a Grafana y se hacen varias peticiones para que muestre el gráfico de los últimos 30 días de los respectivos servidores. El resultado se puede observar en las Figura 58Figura 59Figura 60:

Figura 58 - Procesos totales servidor web

Figura 59 - Procesos totales servidor DHCP DNS

Figura 60 - Procesos totales servidor VPN

Como se puede observar, todos los servidores funcionan de manera correcta aun con los valores actuales de *warning* y *critical*. Estos valores se definieron antes de la implementación, sin conocer cuál iba a ser el resultado final. En este caso, se puede observar que están definidos a la baja, por lo que se recomienda subir el *warning* a 210 procesos y el *critical* a 240 procesos. Así, se evitan las notificaciones constantes por parte de Nagios Core cuando el sistema tiene un funcionamiento correcto. Estas correcciones se van a realizar en el apartado 10.FUTURAS MEJORAS.

También se ha analizado el apartado de los logs de Nagios Core y como se puede observar en la *Figura 61*, la gran mayoría de las notificaciones son causadas por los parámetros *warning* y *critical* de los servidores anteriormente citados. Con esto, queda reafirmada la idea de modificar los parámetros de *warning* y *critical* a un valor más adecuado.

Host	Service	Type	Time	Contact	Notification Command	Information
servidor_vpn	Procesos totales	CRITICAL	07-05-2021 23:53:30	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 206 processes
servidor_vpn	Procesos totales	CRITICAL	07-05-2021 22:53:30	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 205 processes
servidor_dhcp_dns	Procesos totales	WARNING	07-05-2021 22:14:46	nagiosadmin	notify-service-by-email	PROCS WARNING: 200 processes
servidor_dhcp_dns	Procesos totales	CRITICAL	07-05-2021 22:04:46	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 201 processes
servidor_dhcp_dns	Procesos totales	WARNING	07-05-2021 21:54:46	nagiosadmin	notify-service-by-email	PROCS WARNING: 200 processes
servidor_vpn	Procesos totales	CRITICAL	07-05-2021 21:53:30	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 205 processes
servidor_dhcp_dns	Procesos totales	CRITICAL	07-05-2021 20:59:46	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 202 processes
servidor_apache	Procesos totales	WARNING	07-05-2021 20:54:12	nagiosadmin	notify-service-by-email	PROCS WARNING: 200 processes
servidor_vpn	Procesos totales	CRITICAL	07-05-2021 20:53:30	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 205 processes
servidor_apache	Procesos totales	CRITICAL	07-05-2021 20:34:12	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 201 processes
servidor_apache	Procesos totales	WARNING	07-05-2021 20:29:12	nagiosadmin	notify-service-by-email	PROCS WARNING: 200 processes
servidor_apache	Procesos totales	CRITICAL	07-05-2021 20:19:12	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 202 processes
mikrotik_ccr1036	UPDATE	WARNING	07-05-2021 20:17:49	nagiosadmin	notify-service-by-email	Warning : Router upgrade from 6.46.6 to 6.48.3 is required Firmware Requires Upgrade (6.42.12 -> 6.46.6).
servidor_vpn	Procesos totales	CRITICAL	07-05-2021 19:53:30	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 205 processes
servidor_dhcp_dns	Procesos totales	WARNING	07-05-2021 19:19:46	nagiosadmin	notify-service-by-email	PROCS WARNING: 200 processes
servidor_apache	Procesos totales	CRITICAL	07-05-2021 19:14:13	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 202 processes
servidor_dhcp_dns	Procesos totales	CRITICAL	07-05-2021 19:09:46	nagiosadmin	notify-service-by-email	PROCS CRITICAL: 201 processes

Figura 61 - Resumen notificaciones Nagios Core

Se han realizado los mismos pasos para comprobar los diferentes hosts y servicios y se ha determinado que tienen una correcta configuración. Los hosts están activos, se ven variaciones en las gráficas durante los 30 días de actividad y los resultados son coherentes. Por lo tanto, se da por finalizado este apartado con resultado satisfactorio y una mejora sustancial pendiente para realizar.

9.2. Pruebas reales desactivando servicios

En este apartado se van a desactivar ciertos servicios para comprobar que Nagios Core los detecta y notifica adecuadamente. Para ello, se ha decidido detener durante unos minutos cada uno de los servidores que se están monitorizando, para así, poder observar el funcionamiento de Nagios Core. Estas paradas se han realizado de forma individual, nunca más de un equipo parado a la vez. Los routers y switches se ha decidido no detenerlos por razones evidentes, ya que, sino la red del laboratorio empezaría a fallar y se quedaría incomunicada.

Para ello, ha hecho falta la ayuda por parte de los miembros del laboratorio, ya que las contraseñas de administración de los servidores solo las conocen ellos. Gracias a ellos, se ha podido observar cómo al detener un equipo, Nagios Core lo detecta en un margen muy pequeño de tiempo y lo notifica a los administradores en tiempo real.

Tras estas pruebas, se da por finalizado el apartado del plan de pruebas. Como resultado se ha obtenido la necesidad de cambiar los parámetros de *warning* y *critical* de los servidores y la confirmación de que el sistema implementado funciona de manera correcta.

10. FUTURAS MEJORAS

En este apartado se van a tratar las posibles mejoras que se podrían hacer en un futuro, y se va a realizar una mejora para solventar un inconveniente encontrado en el apartado *9.PLAN DE PRUEBAS.*

En el plan de pruebas se ha detectado que los valores warning y critical de los servidores están configurados de forma inadecuada y hay que darles un valor adaptado a su uso real. Para ello, se va a editar el archivo */usr/local/nagios/etc/nrpe.cfg* de cada uno de los servidores, ya que en ese archivo es donde se definen los valores de warning y critical, tal y como se puede observar en la *figura 22* Del apartado *8.2.1.6.Personalizar los comandos.* Estas ediciones de archivos las tiene que realizar un miembro del laboratorio I2T, ya que para editar esos archivos es necesario tener la contraseña de cada equipo.

Tras esto, se va a reiniciar el proceso de Nagios Core y se va a comprobar que funciona adecuadamente y solo manda las notificaciones tras superar el nuevo umbral.

Respecto a futuras mejoras, se van a explicar varias, pero dado el potencial que tiene Nagios Core, se podría incluir casi cualquier equipo que se conecte a la red y admita el envío de paquetes mediante SNMP. Además, Nagios Core se ha probado en instalaciones de miles de equipos, por lo que en un futuro no habría problemas si los miembros del laboratorio añaden más equipos a monitorizar, esto no supondría una pega para el sistema. Las mejoras que se podrían implementar son las siguientes:

- Añadir más equipos: Se pueden añadir a la red equipos de oficina, nuevos servidores, routers, switches y hasta impresoras. Cualquier equipo que se conecte a la red podrá ser monitorizado. Dependiendo del equipo conectado, tendrá más opciones de monitorización o menos. Se puede tomar este proyecto como guía de instalación, ya que se han detallado con mucha exactitud todos los pasos que se han seguido.
- Añadir una nueva red: Se puede añadir una nueva red, ya sea del mismo centro o de un centro en otra localización, eso no supone ningún problema para la instalación. Evidentemente, antes de añadir la red al sistema de Nagios Core, habrá que comprobar que ambas redes tienen conectividad total entre sí.
- Cambiar los parámetros definidos: En este proyecto se ha hecho una estimación y se han elegido los parámetros que mejor se ha pensado que se van a adaptar a la instalación y a los miembros del laboratorio I2T. Entre estos parámetros están los valores de warning y critical, el margen de tiempo de las notificaciones, el tiempo de chequeo de los equipos… Según se vaya usando el sistema, los miembros del laboratorio podrán ajustar esos valores como mejor se adapte a sus necesidades.

Esto son solo unas cuantas propuestas de mejora, las posibilidades son muy amplias y dependerá de los miembros del laboratorio implementar una mejora u otra.

11. PLAN DE TRABAJO

En este apartado se van a detallar las diferentes tareas que se han realizado durante el transcurso del proyecto. Se van a detallar de forma precisa y agrupar en paquetes de trabajo para facilitar la comprensión de dichas tareas. Los **paquetes de trabajo (PT)** y las **tareas (T)** son las siguientes:

1. **Definición del proyecto (PT1):** En este paquete de trabajo se han incluido las tareas que hay que realizar al comienzo del proyecto. En estas tareas, se analizan las especificaciones que va a tener el proyecto y se realiza un análisis de alternativas para cubrir esas especificaciones.
    1. **Definición de especificaciones (T11):** En esta tarea se definen las funcionalidades que va a cumplir este proyecto.
    2. **Análisis de alternativas (T12):** En esta tarea se realiza un análisis exhaustivo de las diferentes alternativas disponibles para las problemáticas que se presentan en el desarrollo de este proyecto y se realiza un análisis de rendimiento de las más importantes.

2. **Diseño de la solución (PT2):** En este paquete de trabajo se procede a diseñar de forma teórica la solución que se va a desarrollar durante el proyecto teniendo en cuenta los resultados del análisis de alternativas.
    1. **Definición de equipos (T21):** En esta tarea se realiza el diseño y la definición de equipos que se van a usar para monitorizar y cuáles van a ser monitorizados.
    2. **Definición de la instalación (T22):** En esta tarea se definen los pasos a seguir para realizar la instalación, tales como, los comandos a seguir, las dependencias a instalar etcétera.

3. **Instalación a nivel local (PT3):** En este paquete de trabajo se instala una versión de la solución en varias máquinas virtuales a nivel local, fuera de la red del laboratorio, con el fin de realizar una prueba de instalación antes de instalarlo de forma definitiva en el laboratorio.
    1. **Instalación de la red local (T31):** En esta tarea se instala una red local de varias máquinas virtuales en un mismo equipo y se comprueba la conectividad entre ellas.
    2. **Instalación de Zenmap (T32):** En esta tarea se instala la herramienta de descubrimiento de red de forma local en la máquina virtual que va a hacer de veces de servidor de monitorización.
    3. **Instalación Nagios Core (T33):** En esta tarea se instala de forma local Nagios Core en una máquina virtual que va a hacer las veces de servidor.
    4. **Instalación de NRPE (T34):** En esta tarea se instala el plugin NRPE en varias máquinas virtuales, las cuales, van a hacer las veces de máquinas monitorizadas.
    5. **Prueba de rendimiento (T35):** En esta tarea se realiza una prueba de rendimiento y funcionamiento para detectar posibles fallos. Si la prueba es favorable, se continúa con las siguientes tareas del siguiente paquete de trabajo. En el caso contrario, se corrigen los errores y se repite la prueba.

4. **Instalación en el laboratorio I2T (PT4):** En este paquete de trabajo se procede a instalar la solución en el laboratorio I2T. Se instala la herramienta de descubrimiento de red y el servidor de monitorización, y también se configuran los agentes que van a ser monitorizados.
    1. **Instalación de Zenmap (T41):** En esta tarea se instala la herramienta de descubrimiento de red Zenmap en la máquina virtual que va a actuar como servidor de monitorización en el laboratorio I2T.
    2. **Instalación de Nagios Core (T42):** En esta tarea se instala y configura el servidor de monitorización Nagios Core en la misma máquina virtual en la que se ha instalado la herramienta de descubrimiento de red.
    3. **Instalación del NRPE (T43):** En esta tarea se instala el plugin NRPE en los equipos que se quieren monitorizar.
    4. **Prueba de funcionamiento (T44):** En esta tarea se realizan diferentes tipos de pruebas para verificar el correcto funcionamiento del conjunto de herramientas instaladas.

5. **Gestión del proyecto (PT5):** Este paquete de trabajo se va a realizar desde el comienzo hasta el final del proyecto. Durante la duración del mismo, se realizará un seguimiento de los avances y se redactará la documentación correspondiente.
    1. **Seguimiento del proyecto (T51):** En esta tarea se realiza un seguimiento para comprobar que se cumple con los plazos y tareas establecidas.
    2. **Documentación (T52):** En esta tarea se redacta la memoria del TFG de forma conjunta con el desarrollo del proyecto.

Aparte de las tareas y paquetes de trabajo, también se han definido varios hitos a lo largo del proyecto:

- **Comienzo del proyecto:** Se establecen las características y objetivos que va a tener el proyecto.
- **Demostración de la instalación a nivel local:** Se muestra el correcto funcionamiento de la solución planteada en una red ajena al laboratorio I2T.
- **Demostración de la instalación en el laboratorio I2T:** Se muestra el correcto funcionamiento de la solución instalada en el laboratorio I2T.

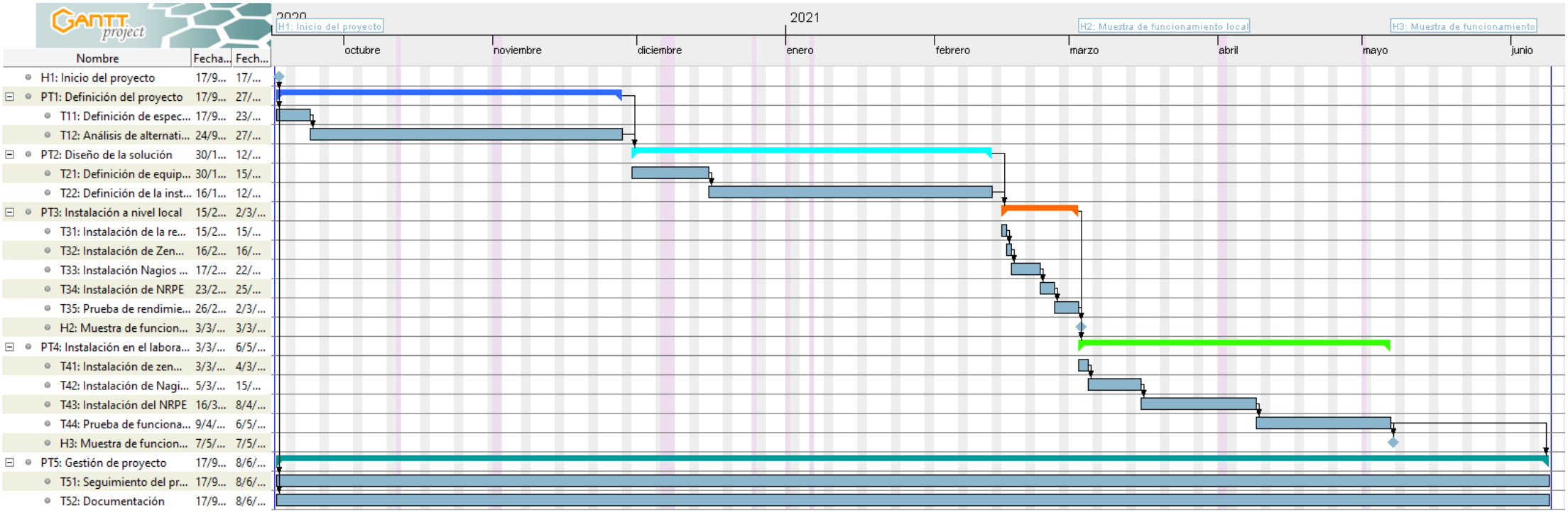

Figura 62 - GANTT del proyecto

12. ASPECTOS ECONÓMICOS

En este apartado se va a tratar todo lo correspondiente a los gastos que va a suponer este proyecto. Se van a tratar los costes económicos derivados de los equipos informáticos y también los costes del personal necesario para la realización de este proyecto. Se van a dividir los gastos en horas internas, amortizaciones, costes fijos y coste total del proyecto.

12.1. Horas internas

En este apartado se va a detallar el coste total de las horas de trabajo de los integrantes de este proyecto. Se dividirán por tipo de trabajador y coste de cada hora de dicho trabajador. Se hará un cálculo de las horas totales y del costo total de dichas horas. De esta forma se podrá conocer el gasto económico que supondrá la mano de obra.

En este proyecto han trabajado 2 ingenieros senior y 1 ingeniero junior, siendo el puesto y coste horario de cada uno el siguiente:

Nombre	Puesto	Tasa horaria
Eduardo Jacob	Ingeniero Senior (Director del proyecto)	50,00 €/h
Jasone Astorga	Ingeniera Senior (Codirectora de proyecto)	50,00 €/h
Eder López	Ingeniero Junior	20,00 €/h

En la siguiente tabla se muestra el desglose horario para cada integrante del equipo de trabajo en cada paquete de trabajo.

Paquete de trabajo	Responsable	N.º de horas
Análisis de alternativas	Ingeniero Junior	50 h
Diseño de la solución	Ingeniero Junior	90 h
Implementación	Ingeniero Junior	100 h
Pruebas de funcionamiento	Ingeniero Junior	30 h
Redacción de documentación	Ingeniero Junior	50 h
Supervisión	Ingeniera/o Senior	60 h

En la siguiente tabla se puede observar el cálculo del coste de las horas internas:

Horas internas	Coste por hora	N.º de horas	Total
Ingeniero Senior	50,00 €/h	60	3.000,00 €
Ingeniero Junior	20,00 €/h	320	6.400,00 €
		SUBTOTAL	**9.400,00 €**

12.2. Amortizaciones

En este apartado se va a proceder a detallar el coste de los equipos informáticos necesarios para realizar este proyecto. Cabe destacar que estos equipos ya están instalados en el laboratorio de investigación I2T, por lo tanto, no se van a adquirir nuevos y el desembolso no va a ser significativo, ya que solo se van a tener en cuenta las amortizaciones, es decir, la pérdida de valor o distribución del gasto de los equipos por el uso que se les va a dar durante el periodo que dure este proyecto. No se va a tener en cuenta el valor residual de los equipos a la hora de calcular el presupuesto, porque se desconoce cuándo se van a sustituir.

Para ello, se va a tener en cuenta el coste de los 4 servidores que forman el nodo completo OpenStack en el que se va a instalar la máquina virtual con el sistema de monitorización e inventario de red. Se va a definir como coste individual el precio de un servidor de gama media actual [77], con 4 años de vida útil y un periodo de amortización de 9 meses. Se va a hacer la suposición de que no se va a usar más del 20% de la capacidad de cada servidor físico, por lo que se va a tener únicamente en cuenta el 20% del precio de cada unidad de servidor. Además, no se va a tener en cuenta el tiempo que el software vaya a estar instalado, eso entrará en los costes de mantenimiento, los cuales no se van a tratar en este proyecto.

Equipo	**Valor inicial**	**Valor residual**	**Vida útil**	**Horas de uso**	**Coste**
Servidor gama media	3373,00 €	750,00 €	35040 h	6570 h	491,81 €
				COSTE POR UNIDAD	**491,81 €**

El coste por unidad es de 491,81 €. Ahora se va a tener en cuenta el 20% del coste de cada unidad y las 4 unidades que forman el nodo completo de OpenStack.

COSTE POR UNIDAD	% DE USO	UNIDADES	COSTE
491,81 €	20	4	**393,45 €**
		SUBTOTAL	**393,45 €**

12.3. Gastos

En este apartado se van a ver reflejados los gastos fijos para la realización de este proyecto. Se tendrán en cuenta los gastos no amortizables como por ejemplo, las facturas de electricidad e Internet.

Nota: El concepto de electricidad se ha calculado suponiendo que el servidor va a estar las 24 h del día en funcionamiento con un consumo medio de 300 W durante 9 meses y con un precio de 0.1187 €/kWh mediante la web calculadoraconversor.com [78].

Concepto	**Coste**
Electricidad	230,76 €
Internet	150,00 €
Materiales	45,00 €
SUBTOTAL	**425,76 €**

12.4. Coste total del proyecto

En este apartado se va a hacer la suma de todos apartados anteriores con la finalidad de conocer la suma total de los costes económicos que van a derivar de la realización de este proyecto.

Apartado	Subtotal
Horas internas	9.400,00 €
Amortizaciones	393,45 €
Gastos	425,76 €
TOTAL	**10.219,21 €**

El coste total del proyecto asciende a 10.219,21 €, IVA no incluido.

13. CONCLUSIONES

En este apartado se va a hacer una reflexión final acerca de este proyecto. Se va a analizar el cumplimiento de los objetivos y la complejidad del proyecto.

Para empezar, cabe destacar que se han cumplido tanto el objetivo principal, que es diseñar e implementar un sistema de descubrimiento de red y servicios, como todos los objetivos secundarios.

También hay que destacar que durante el proyecto ha habido diferentes problemas a la hora de realizar la instalación, los cuales se han podido solventar para tener una instalación totalmente funcional. Entre ellos, los más importantes han sido:

- **Complejidad del sistema Nagios:** Tras haber estado trabajando durante meses con este sistema, el nivel actual de conocimiento es alto, pero de entrada hay una barrera muy importante. No se comprende de manera fácil el funcionamiento de los archivos de configuración y se tardan varias semanas en ir adecuándose al entorno.
- **Complejidad del entorno de la implementación:** La implementación se ha realizado en el laboratorio I2T usando OSM para desplegar varias VNFs. Conceptos como OSM, VNFs y VIM han resultado un poco complicados de comprender al comienzo y durante el proyecto, pero gracias a las explicaciones y apoyo de los directores de este trabajo se han conseguido interiorizar de forma correcta.
- **Programación:** En este proyecto se han aprovechado varios plugins realizados por la comunidad de Nagios Exchange. Varios de esos plugins eran perfectamente funcionales con versiones anteriores de Nagios Core, pero no con la versión actual que se ha instalado en el laboratorio. Factores como la versión de PHP o las políticas de los fabricantes de hardware por intentar ocultar los datos internos de sus equipos, como los OIDs, han hecho que haya sido necesario modificar el código de muchos de los plugins, y en algunos casos, buscar una alternativa diferente. Además, alguno de estos plugins estaba escrito en lenguajes de programación que no se han estudiado durante el *Grado en Ingeniería en Tecnología de Telecomunicación*, por lo que ha habido que hacer una investigación individual para poder corregir el código de estos plugins.

Por último, he de decir que he aprendido mucho acerca de los sistemas de monitorización y descubrimiento de red y servicios, los cuales, no conocía antes de embarcarme en este proyecto. Personalmente, el software de Nagios Core me ha parecido muy potente, versátil y eficiente, y creo que es una solución muy adecuada para entornos de investigación, empresas, centros educativos o cualquier entidad que tenga una cantidad de equipos considerable. El software de Zenmap también me ha parecido muy interesante de cara a tener controlados los equipos conectados a la red, y creo que en muchos entornos puede ser de gran utilidad.

14. BIBLIOGRAFÍA

En este apartado se van a documentar todos los documentos y páginas webs que se han visitado para la realización de este proyecto.

[1] "Nagios Core Addons." Assets.nagios.com. https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/addons.html (Último acceso: 26 Marzo 2021).

[2] E. Galstad. *NRPE Documentation.* (2017). Último acceso: 26 Marzo 2021. [En línea]. Disponible en: https://assets.nagios.com/downloads/nagioscore/docs/nrpe/NRPE.pdf

[3] "Monitoring Routers and Switches." Assets.nagios.com. https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/monitoring-routers.html (Último acceso: 2 Marzo 2021).

[4] Wolfgang. "Documentation." PNP4Nagios.org. https://docs.pnp4nagios.org/pnp-0.6/start (Último acceso: 12 Marzo 2021).

[5] "Pagina Principal Zenmap." NMAP.com. https://nmap.org/zenmap/ (Último acceso: 06 Febrero 2021).

[6] "¿Qué es la virtualización?." RedHat.com. https://www.redhat.com/es/topics/virtualization/what-is-virtualization (Último acceso: 16 02 2021).

[7] "¿Qué es la NFV?." https://www.redhat.com/es/topics/virtualization/what-is-nfv (Último acceso: 2 Marzo 2021).

[8] A.M. Alwakeel, A. Alnaim, E.B. Fernández. “Figure 1 - Difference between traditional network approach and NFV approach.” Researchgate.net. https://www.researchgate.net/figure/Difference-between-traditional-network-approach-and-NFV-approach_fig1_334699708 (Último acceso: 3 Marzo 2021).

[9] "Página de arquitectura Openstack." RedHat.com. https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/11/html/architecture_guide/components (Último acceso: 4 Marzo 2021).

[10] "23 operadoras y proveedores de equipos se unen para crear una comunidad global de Open Source Mano (OSM)." Catedratelefonica.ulpgc.es. https://catedratelefonica.ulpgc.es/open-source-mano (Último acceso: 19 Febrero 2021).

[11] "Página principal OSM." ETSI.org. https://osm.etsi.org/ (Último acceso: 19 Febrero 2021).

[12] "OSM members and participants." ETSI.org. https://portal.etsi.org/TB-SiteMap/OSM/List-of-OSM-Members (Último acceso: 19 Febrero 2021).

[13] "¿En qué consiste la virtualización?." VMWare.com. https://www.vmware.com/es/solutions/virtualization.html (Último acceso: 19 Febrero 2021).

[14] "¿Qué es la virtualización?." RedHat.com. https://www.redhat.com/es/topics/virtualization/what-is-virtualization (Último acceso: 19 Febrero 2021).

[15] "Acronis Cyber Protect." Acronis.com. https://www.acronis.com/es-es/business/cyber-protect/ (Último acceso: 05 12 2020).

[16] "Acerca de NewRelic." NewRelic.com. https://newrelic.com/about (Último acceso: 12 Diciembre 2020).

[17] K. Elisea, " ¿Qué es New Relic?." Inbest.cloud. https://www.inbest.cloud/comunidad/qu%C3%A9-es-new-relic-mexico (Último acceso: 13 Diciembre 2020).

[18] "Página principal LogicMonitor." Logicmonitor.com. https://www.logicmonitor.com (Último acceso: 13 Diciembre 2020).

[19] "PRTG Network Monitor." Paessler.com. https://www.es.paessler.com/prtg (Último acceso: 14 Diciembre 2020).

[20] "Página principal SolarWinds." SolarWinds.com. https://www.solarwinds.com/ (Último acceso: 14 Diciembre 2020).

[21] "Página principal ManageEngine." ManageEngine.com. https://www.manageengine.com/ (Último acceso: 14 Diciembre 2020).

[22] "Página principal Nagios Core." Nagios.com. https://www.nagios.com/products/nagios-core/ (Último acceso: 14 Diciembre 2020).

[23] "Monitor Your Entire Infrastructure with Icinga." Icinga.com. https://icinga.com/ (Último acceso: 14 Diciembre 2020).

[24] "Icinga." Wikipedia.org. https://en.wikipedia.org/wiki/Icinga (Último acceso: 14 Diciembre 2020).

[25] "Página principal Zabbix." Zabbix.com. https://www.zabbix.com/ (Último acceso: 14 Diciembre 2020).

[26] "Zabbix." Wikipedia.org. https://en.wikipedia.org/wiki/Zabbix (Último acceso: 14 Diciembre 2020).

[27] T. Oetiker. "Tobi Oetiker's MRTG - The Multi Router Traffic Grapher." Oetiker.ch. https://oss.oetiker.ch/mrtg/ (Último acceso: 14 Diciembre 2020).

[28] "Multi Router Traffic Grapher." Wikipedia.org. https://en.wikipedia.org/wiki/Multi_Router_Traffic_Grapher (Último acceso: 14 Diciembre 2020).

[29] ManageEngine. *User Guide. (2012).* Último acceso: 14 Diciembre 2020. [En línea]. Disponible en: https://download.manageengine.com/es/applications_manager/AppManager_Help.pdf

[30] "Install New Relic." NewRelic.com. https://docs.newrelic.com/docs/agents/manage-apm-agents/installation/install-agent (Último acceso: 14 Diciembre 2020).

[31] New Relic. "Install New Relic for Linux Servers Tutorial." YouTube.com. https://www.youtube.com/watch?v=ZKUvYHc8P84&ab_channel=NewRelic (Último acceso: 14 Diciembre 2020).

[32] "Cloud integrations release notes." NewRelic.com. https://docs.newrelic.com/docs/release-notes/platform-release-notes/cloud-integrations-release-notes (Último acceso: 14 Diciembre 2020).

[33] "Diagnostics CLI (nrdiag) release notes." NewRelic.com. https://docs.newrelic.com/docs/release-notes/platform-release-notes/diagnostics-release-notes (Último acceso: 14 Diciembre 2020).

[34] "Release notes." Logicmonitor.com. https://www.logicmonitor.com/release-notes (Último acceso: 16 Diciembre 2020).

[35] "Precios - PRTG Network Monitor." Paessler.com. https://www.es.paessler.com/prtg/pricing (Último acceso: 16 Diciembre 2020).

[36] "PRTG Network Monitor." Wikipedia.org. https://en.wikipedia.org/wiki/PRTG_Network_Monitor (Último acceso: 16 Diciembre 2020).

[37] "Release notes for the 'stable' release channel." Paessler.com. https://www.paessler.com/prtg/history/stable (Último acceso: 16 Diciembre 2020).

[38] "Opciones de licencias." Solarwinds.com. https://www.solarwinds.com/es/licensing-options (Último acceso: 16 Diciembre 2020).

[39] "Install a Linux agent." Solarwindsmsp.com. http://documentation.solarwindsmsp.com/N-central/documentation/Content/Deploying/Agent/Agents_InstallRHELinuxAgents.html (Último acceso: 16 Diciembre 2020).

[40] "Orion Platform 2020.2.1 Release Notes." Solarwinds.com. https://documentation.solarwinds.com/en/Success_Center/orionplatform/Content/Release_Notes/Orion_Platform_2020-2-1_release_notes.htm (Último acceso: 16 Diciembre 2020).

[41] "Tienda Manage Engine." ManageEngine.com. https://store.manageengine.com/ (Último acceso: 16 Diciembre 2020).

[42] "Linux Management." ManageEngine.com. https://www.manageengine.com/products/desktop-central/linux-management.html (Último acceso: 16 Diciembre 2020).

[43] ManageEngine. *Manual de usuario.* https://download.manageengine.com/network-monitoring/opmanager_userguide.pdf (Último acceso: 16 Diciembre 2020).

[44] "OpManager v12.5." ManageEngine.com. https://www.manageengine.com/network-monitoring/help/read-me.html?releaseNotes (Último acceso: 16 Diciembre 2020).

[45] "Nagios Core 4.x Version History." Nagios.org. https://www.nagios.org/projects/nagios-core/history/4x/ (Último acceso: 16 Diciembre 2020).

[46] "Icinga 2 CHANGELOG." Icinga.com. https://icinga.com/docs/icinga2/latest/CHANGELOG/ (Último acceso: 16 Diciembre 2020).

[47] "Zabbix Documentation 4.0." Zabbix.com. https://www.zabbix.com/documentation/4.0/manual/installation/getting_zabbix (Último acceso: 17 Diciembre 2020).

[48] T. Oetiker. "The MRTG 2.17.4 Linux/Unix Installation Guide." Oetiker.ch. https://oss.oetiker.ch/mrtg/doc/mrtg-unix-guide.en.html (Último acceso: 17 Diciembre 2020).

[49] "Installation." Icinga.com. https://icinga.com/docs/icinga-2/latest/doc/02-installation/ (Último acceso: 26 Diciembre 2020).

[50] "Nagios Core - Installing Nagios Core From Source." Nagios.com. https://support.nagios.com/kb/article/nagios-core-installing-nagios-core-from-source-96.html (Último acceso: 29 Diciembre 2020).

[51] "Zabbix Documentation 5.2 - Installation." Zabbix.com. https://www.zabbix.com/documentation/current/manual/installation (Último acceso: 29 Diciembre 2020).

[52] "Página principal." Advanced-IP-Scanner.com. https://www.advanced-ip-scanner.com/es/ (Último acceso: 3 Enero 2021).

[53] "Intermapper - Training." Helpsystems.com. https://www.helpsystems.com/products/network-monitoring-software/training (Último acceso: 19 Octubre 2020).

[54] "How to install OpenNMS network manager on Ubuntu 18.04." Techrepublic.com. https://www.techrepublic.com/article/how-to-install-opennms-network-manager-on-ubuntu-18-04/ (Último acceso: 30 Octubre 2020).

[55] "Download for Windows, Mac or Linux." AngryIP.com. https://angryip.org/download/ (Último acceso: 30 Octubre 2020).

[56] "Linux platforms." RedHat.com. https://www.redhat.com/en/store/linux-platforms (Último acceso: 12 Febrero 2021).

[57] J. Linge. "Documentación" PNP4Nagios.org. http://docs.pnp4nagios.org/es/pnp-0.6/start (Último acceso: 26 Marzo 2021).

[58] A. Brenner, M. Wall. "Nagiosgraph" SourceForge.net. http://nagiosgraph.sourceforge.net/ (Último acceso: 27 Marzo 2021).

[59] "Homepage de Grafana" Grafana.com. https://grafana.com/ (Último acceso: 2 Abril 2021).

[60] T. Oetiker. "Tobi Oetiker's MRTG - The Multi Router Traffic Grapher" Oetiker.ch. https://oss.oetiker.ch/mrtg/ (Último acceso: 15 Marzo 2021).

[61] "¿Qué es Kibana?" Elastic.co. https://www.elastic.co/es/what-is/kibana (Último acceso: 28 Marzo 2021).

[62] "Nagios" Wikipedia.org. https://es.wikipedia.org/wiki/Nagios (Último acceso: 29 Marzo 2021).

[63] "Comparing changes." GitHub.com. https://github.com/NagiosEnterprises/nrpe/compare/nrpe-3.2.1...nrpe-4.0.3 (Último acceso: 6 Marzo 2021).

[64] "NRPE - How To Install NRPE v4 From Source." Nagios.com. https://support.nagios.com/kb/article.php?id=515#Ubuntu (Último acceso: 12 Febrero 2021).

[65] "Nagios Plugins - Installing Nagios Plugins From Source." Nagios.com. https://support.nagios.com/kb/article.php?id=569 (Último acceso: 26 Febrero 2021).

[66] "Nagios Core - Installing Nagios Core From Source." Nagios.com. https://support.nagios.com/kb/article/nagios-core-installing-nagios-core-from-source-96.html#Ubuntu (Último acceso: 21 Febrero 2021).

[67] "CATEGORY ARCHIVES: RELEASES." Nagios-plugins.org. https://nagios-plugins.org/category/releases/ (Último acceso: 21 Febrero 2021).

[68] "Aplicaciones poco seguras y la cuenta de Google." Support.google.com. https://support.google.com/accounts/answer/6010255 (Último acceso: 22 Febrero 2021).

[69] "Difference between mail and mailx?." StackExchange.com. https://unix.stackexchange.com/questions/89530/difference-between-mail-and-mailx (Último acceso: 22 Febrero 2021).

[70] "Telegram Bot API." Telegram.org. https://core.telegram.org/bots/api (Último acceso: 4 Marzo 2021).

[71] "Standard Macros in Nagios." Nagios.com. https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/macrolist.html (Último acceso: 9 Marzo 2021).

[72] "Nagios Core - Performance Graphs Using PNP4Nagios." Nagios.com. https://support.nagios.com/kb/article.php?id=801#Ubuntu (Último acceso: 12 Marzo 2021).

[73] "Install on Debian or Ubuntu." Grafana.com. https://grafana.com/docs/grafana/latest/installation/debian/ (Último acceso: 14 Marzo 2021).

[74] "Nagios Core - Using Grafana With PNP4Nagios." Nagios.com. https://support.nagios.com/kb/article/nagios-core-using-grafana-with-pnp4nagios-803.html (Último acceso: 12 Marzo 2021).

[75] "SNMP Printer Check" Exchange.nagios.org. https://exchange.nagios.org/directory/Plugins/Hardware/Printers/SNMP-Printer-Check/details (Último acceso: 02 Julio 2021).

[76] "Nagios Exchange" Exchange.nagios.org. https://exchange.nagios.org/ (Último acceso: 02 Julio 2021).

[77] "Servidor 4U BlackRock X-Server – Intel Xeon Scalable 2ª GEN – Configurable a medida." Sntservicios.es. https://www.sntservicios.es/tienda/servidores-a-medida/servidor-4u-blackrock-x-server-intel-xeon-scalable-2-gen-configurable-a-medida/ (Último acceso: 26 Febrero 2021).

[78] "Calculadora de consumo eléctrico." Calculadoraconversor.com. https://www.calculadoraconversor.com/calculadora-consumo-electrico/ (Último acceso: 21 Febrero 2021).

15. ANEXOS

15.1. ANEXO I

En este anexo se van a describir todas las siglas, acrónimos y conceptos técnicos utilizados durante la redacción de este documento.

- **API**: Application Programming Interfaces
- **CPU**: Central Processing Unit
- **DNS**: Domain Name System
- **EHU**: Euskal Herriko Unibertsitatea / Universidad del País Vasco
- **ETSI**: European Telecommunications Standards Institute
- **FTP**: File Transfer Protocol
- **GUI**: Graphical User Interface
- **HTTP**: Hypertext Transfer Protocol
- **ICMP**: Internet Control Message Protocol
- **IP**: Internet Protocol
- **LAMP**: Linux Apache MySQL PHP
- **LTS**: Long-Term Support
- **MAC**: Media Access Control
- **MANO**: Management and Orchestration
- **MV**: Máquina Virtual
- **MySQL**: Base de datos basada en SQL
- **NFV**: Network Function Virtualization
- **NRPE**: Nagios Remote Plugin Executor
- **NSCA**: Nagios Service Check Acceptor
- **OSM**: Open Source MANO
- **PHP**: Lenguaje de programación que originalmente significaba Personal Home Page
- **RAM**: Random Access Memory
- **REST**: Representational State Transfer
- **RHEL**: Red Hat Enterprise Linux
- **RRD**: Round Robin Database
- **SaaS**: Software as a Service
- **SMTP**: Simple Mail Transfer Protocol
- **SNMP**: Simple Network Management Protocol
- **SO**: Sistema Operativo
- **SQL**: Structured Query Language
- **SSH**: Secure Shell
- **TCP**: Transmission Control Protocol
- **TLS**: Transport Layer Security
- **UFW**: Uncomplicated Firewall
- **VLAN**: Virtual Local Area Network
- **VM**: Virtual Machine
- **VNF**: Virtual Network Function
- **VPN**: Virtual Private Network