



TRABALLO FIN DE GRAO (ANTEPROYECTO)
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DA INFORMACIÓN



Diseño e implementación de una red empresarial orientada a la monitorización y acceso remoto de un entorno industrial

Estudiante: Ibrahim Méndez Fernández

Dirección: Francisco Javier Nóvoa Manuel

A Coruña, junio de 2025.

A mi familia, amigos y pareja por apoyarme y darme la oportunidad de ser quién soy.

Agradecimientos

Quiero agradecer a mis padres por el apoyo emocional a lo largo de mi vida y el sustento económico que me ha permitido llegar aquí. A mi hermana por ser mi referencia y pilar en los momentos más importantes. A mis amigos por todos los momentos que hemos vivido y los instantes en los que he podido ser mi verdadero yo. A mi pareja por aguantarme y estar ahí en los momentos más difíciles cuando más lo necesitaba. Y un agradecimiento y mención especial a mi tutor Francisco Javier Nóvoa Manuel por hacer crecer mi interés por las redes en mi etapa universitaria y por el gran esfuerzo realizado ayudándome durante la realización de este trabajo.

Resumen

La creciente convergencia entre las redes tradicionales ([IT](#)) y las redes [Operational Technology \(OT\)](#) plantean el reto del intercambio y monitorización de datos de forma remota y segura, al que se le añade el reto de automatizar la instalación de nuevas redes para agilizar el proceso. En el escenario propuesto existen una red industrial ya desplegada y una nueva sede administrativa situada a decenas de kilómetros de distancia, sin infraestructura de red propia. El objetivo es crear una red de empresa que enlace ambos dominios mediante una solución segura, segmentada, escalable y redundante que ofrezca los componentes y mecanismos para extraer, almacenar y monitorizar en tiempo real los datos de la red industrial, acceder remotamente a dispositivos situados la zona industrial y crear las restricciones de acceso pertinentes dentro de la red.

Para el desarrollo se adoptará la metodología [PPDIOO](#) de Cisco. Durante sus fases se diseñará una topología adecuada, se definirán sus tecnologías ([VLAN](#), [VRRP](#), [OSPF](#), [ACLs](#)), se configurará la base de datos InfluxDB acompañada de la aplicación Grafana, se establecerá una [VPN](#) que conectará ambas redes y un servidor en la nube de Oracle que facilite el intercambio de datos mediante un servidor [MQTT](#) como Mosquitto. También se automatizará la configuración de dispositivos de red y el despliegue del servidor (haciendo uso de Python, Ansible y Terraform). Para todo esto se hará uso principalmente de un laboratorio virtual (GNS3), pero también de un montaje físico reducido. Finalmente se comprueban los resultados accediendo a los servicios o dispositivos necesarios.

Abstract

The increasing convergence between traditional ([IT](#)) networks and [Operational Technology \(OT\)](#) networks poses the challenge of remote and secure data exchange and monitoring, coupled with the challenge of automating the installation of new networks to speed up the process. In the proposed scenario, there is an industrial network already deployed and a new administrative headquarters located tens of kilometres away, with no network infrastructure of its own. The objective is to create an enterprise network linking both domains through a secure, segmented, scalable and redundant solution that provides the components and mechanisms to extract, store and monitor real time data from the industrial network, access remote devices located in the industrial area and create the relevant access restrictions within the network.

For the development, Cisco's [PPDIOO](#) methodology will be adopted. During its phases, a suitable topology will be designed, its technologies ([VLAN](#), [VRRP](#), [OSPF](#), [ACLs](#)) will be

defined, the InfluxDB database will be configured together with the Grafana application, a [VPN](#) will be set up to connect both networks and an Oracle cloud server that facilitates data exchange via a [MQTT](#) server such as Mosquitto. The configuration of network devices and the deployment of the server will also be automated (using Python, Ansible and Terraform). For all this we will mainly make use of a virtual lab (GNS3), but also of a reduced physical setup. Finally, the results are tested by accessing the necessary services or devices.

Palabras clave:

- Grafana
- InfluxDB
- VLAN
- VPN
- Switch Capa 3/Multicapa
- Seguridad de redes
- OPNsense
- Redundancia de Red
- Virtualización
- GNS3
- MQTT
- Modbus

Keywords:

- Grafana
- InfluxDB
- VLAN
- VPN
- Layer 3/Multilayer Switch
- Network security
- OPNsense
- Network Redundancy
- Virtualization
- GNS3
- MQTT
- Modbus

Índice general

| | | |
|----------|--|-----------|
| 1 | Introducción | 1 |
| 1.1 | Motivación | 1 |
| 1.2 | Objetivos | 2 |
| 1.3 | Estructura | 3 |
| 2 | Fundamentos tecnológicos | 4 |
| 2.1 | Herramientas de automatización | 4 |
| 2.1.1 | Ansible | 4 |
| 2.1.2 | Terraform | 4 |
| 2.2 | Transmisión y recepción de datos | 5 |
| 2.2.1 | Protocolo Modbus | 5 |
| 2.2.2 | MQTT Broker | 5 |
| 2.3 | Fundamentos de redes | 6 |
| 2.3.1 | ACL | 6 |
| 2.3.2 | ZeroTier | 7 |
| 2.3.3 | VLAN | 7 |
| 2.3.4 | STP | 8 |
| 2.3.5 | OSPF | 8 |
| 3 | Metodología | 10 |
| 3.1 | PPDIOO | 10 |
| 3.2 | Fases principales | 10 |
| 4 | Preparación | 11 |
| 4.1 | Introducción | 11 |
| 4.2 | Topología | 12 |
| 4.3 | Entorno | 17 |

| | |
|--|-----------|
| 5 Planificación | 20 |
| 5.1 Etapas | 20 |
| 5.2 Costes y recursos | 22 |
| 5.2.1 Hardware | 22 |
| 5.2.2 Software | 22 |
| 5.2.3 Recursos humanos | 23 |
| 5.2.4 Coste total | 23 |
| 6 Diseño | 24 |
| 6.1 Herramienta de automatización | 24 |
| 6.2 Detalle de tecnologías en la capa 2 | 25 |
| 6.3 Detalle de tecnologías en la capa 3 | 28 |
| 7 Implementación | 30 |
| 7.1 Automatización de la red | 30 |
| 7.1.1 Inicio con Python y Netmiko | 30 |
| 7.1.2 Continuación con Ansible | 34 |
| 7.2 Aprovisionamiento del servidor en la nube | 49 |
| 7.3 Pasos necesarios en la red industrial | 56 |
| 7.4 Configuración Teltonika RUT901 | 57 |
| 7.5 Configuración del servidor local | 60 |
| 7.6 Configuración de OPNsense | 63 |
| 7.7 Configuración de Zerotier | 68 |
| 7.8 Limitaciones en el entorno físico | 69 |
| 8 Operación | 70 |
| 8.1 Generación de Datos y MQTT | 70 |
| 8.2 Recepción de datos y monitorización | 71 |
| 8.3 Comprobación de Control de Acceso y Firewall | 71 |
| 8.3.1 VLAN 10 | 72 |
| 8.3.2 VLAN 20 | 73 |
| 8.3.3 VLAN 30 | 74 |
| 8.3.4 VLAN 99 | 74 |
| 8.3.5 Otros | 75 |
| 8.4 Comprobación de Protocolos | 75 |
| 8.4.1 OSPF | 75 |
| 8.4.2 VRRP | 77 |

| | |
|--|-----------|
| 9 Conclusiones | 78 |
| 9.1 Resumen | 78 |
| 9.2 Lecciones Aprendidas | 78 |
| 9.3 Comparativa de entornos | 79 |
| 9.4 Trabajo futuro | 80 |
| A Material adicional | 82 |
| A.1 Ansible | 82 |
| A.2 Grafana e InfluxDB | 84 |
| A.3 Oracle Cloud | 89 |
| B Imágenes dispositivos físicos | 91 |
| Lista de acrónimos | 94 |
| Bibliografía | 96 |

Índice de figuras

| | | |
|------|---|----|
| 4.1 | Escalabilidad de una topología colapsada a una de tres niveles. <i>Origen:[1]</i> | 13 |
| 4.2 | Diagrama de red del proyecto | 13 |
| 4.3 | Captura de pantalla del presupuesto añadido en Oracle Cloud | 15 |
| 6.1 | Diagrama de red detallado de capa 2 (sin mostrar STP) | 25 |
| 6.2 | Diseño tradicional en bucle con VLAN. <i>Origen:[1]</i> | 27 |
| 6.3 | Diagrama de red detallado de capa 3 con STP | 28 |
| 7.1 | Ejecución del programa setup.py para conexiones Telnet | 34 |
| 7.2 | Ejecución del programa setup.py para conexiones serial | 34 |
| 7.3 | Ejecución del playbook con el rol "base" en Ansible 1 | 39 |
| 7.4 | Ejecución del playbook con el rol "base" en Ansible 3 | 40 |
| 7.5 | Ejecución del playbook con el rol "base" en Ansible 4 | 40 |
| 7.6 | Ejecución del playbook con el rol "base" en Ansible 5 | 40 |
| 7.7 | Ejecución del playbook con el rol "base" en Ansible 6 | 41 |
| 7.8 | Ejecución del playbook con el rol "vlan" en Ansible 1 | 42 |
| 7.9 | Ejecución del playbook con el rol "vlan" en Ansible 2 | 43 |
| 7.10 | Ejecución del playbook con el rol "acl" en Ansible 1 | 44 |
| 7.11 | Ejecución del playbook con el rol "dist_switch" en Ansible 1 | 46 |
| 7.12 | Ejecución del playbook con el rol "dist_switch" en Ansible 2 | 46 |
| 7.13 | Ejecución de Terraform-Ansible para el servidor en la nube 1 | 54 |
| 7.14 | Ejecución de Terraform-Ansible para el servidor en la nube 2 | 55 |
| 7.15 | Teltonika ZeroTier | 57 |
| 7.16 | Teltonika Modbus Client TCP | 58 |
| 7.17 | Teltonika Modbus Client Request | 58 |
| 7.18 | Teltonika Modbus Request Test | 59 |
| 7.19 | Teltonika Data to Server MQTT | 59 |
| 7.20 | Teltonika Dato to Server MQTT 2 | 60 |

| | |
|--|----|
| 7.21 Script Python como servicio Linux | 62 |
| 7.22 OPNsense configuración de interfaz LAN IPv4 | 63 |
| 7.23 OPNsense creación de un bridge LAN | 63 |
| 7.24 OPNsense configuración de la red VPN ZeroTier | 64 |
| 7.25 OPNsense configuración de las interfaces | 64 |
| 7.26 OPNsense creación de alias para facilitar su gestión | 65 |
| 7.27 OPNsense creación de grupos para facilitar su gestión | 65 |
| 7.28 OPNsense reglas firewall de la interfaz LAN | 66 |
| 7.29 OPNsense reglas firewall de la interfaz de ZeroTier | 66 |
| 7.30 OPNsense configuración de la traducción NAT hacia el exterior | 67 |
| 7.31 Selección de la interfaz OSPF | 67 |
| 7.32 Configuración de las redes a compartir con OSPF | 67 |
| 7.33 Zerotier servidor sin autorización | 68 |
| 7.34 ZeroTier servidor autorizado | 69 |
| 8.1 Datos generados por el simulado modbus en Python | 70 |
| 8.2 Datos sin escribir en el bróker MQTT | 70 |
| 8.3 Datos escritos en el bróker MQTT | 71 |
| 8.4 Grafana Vista de Tablero | 71 |
| 8.5 Pruebas de conectividad de la VLAN 10 (1) | 72 |
| 8.6 Pruebas de conectividad de la VLAN 10 (2) | 73 |
| 8.7 Pruebas de conectividad de la VLAN 20 | 73 |
| 8.8 Pruebas de conectividad de la VLAN 30 | 74 |
| 8.9 Pruebas de conectividad de la VLAN 99 | 74 |
| 8.10 Pruebas de conectividad de otros dispositivos | 75 |
| 8.11 Prueba de vecinos OSPF en OPNsense | 75 |
| 8.12 Prueba de rutas OSPF en OPNsense | 76 |
| 8.13 Prueba de funcionamiento del protocolo OSPF en switch | 76 |
| 8.14 Prueba de funcionamiento del protocolo VRRP | 77 |
| A.1 Creación de un panel simple en Grafana | 85 |
| A.2 Añadiendo InfluxDB como fuente de datos de Grafana (1) | 86 |
| A.3 Añadiendo InfluxDB como fuente de datos de Grafana (2) | 87 |
| A.4 Interfaz general de InfluxDB | 88 |
| A.5 Visualización de datos desde la interfaz InfluxDB | 89 |
| A.6 Creación de un token api en InfluxDB | 89 |
| A.7 Instancia corriendo de Oracle Cloud | 90 |

| | | |
|-----|--|----|
| B.1 | Switches Cisco y router OPNsense reales | 92 |
| B.2 | Router Teltonika RUT901 de la red industrial | 93 |

Índice de tablas

| | | |
|-----|---|----|
| 5.1 | Listado de equipamiento y estimación de antigüedad/coste. | 22 |
| 5.2 | Listado de Software con método de pago necesario | 23 |
| 5.3 | Listado de recursos humanos | 23 |
| 5.4 | Resumen de costes del proyecto | 23 |

Capítulo 1

Introducción

1.1 Motivación

ESTE trabajo surge de la propuesta de una empresa ourensana, especializada en el depósito y automatización del ciclo integral del agua, que busca rediseñar sus infraestructuras de comunicaciones para obtener nuevas funcionalidades. En primer lugar, desea conectar diferentes dispositivos de su red industrial con la red de su sede central de manera que sean accesibles y sea posible la transmisión de datos a la sede central en tiempo real. Los dispositivos se encuentran repartidos a lo largo de toda la planta industrial e interconectados entre sí en una misma red gracias a distintos *switches* y *routers* que la componen. Están compuestos por ordenadores, cámaras, Programmable Logic Controller (PLC), Human-Machine Interface (HMI) y otros sensores que permiten monitorizar y realizar de manera automática las tareas laborales. En nuestro trabajo simularemos un único dispositivo Internet of Things (IoT) conectado al *router* extremo de la planta industrial, obviando el resto de su infraestructura de red para centrarnos en hacer posible la conectividad con la red de la sede central, separadas entre sí por varias decenas de kilómetros.

En segundo lugar, es necesario diseñar la red de datos de la sede u oficina desde cero, de manera que se adecúe a las buenas prácticas de diseño y que permita organizar la red en distintas zonas o departamentos. Además se deben incorporar a esta red servicios y soluciones para el acceso remoto a los dispositivos de la red industrial y para la transmisión, almacenamiento y monitorización de los datos provenientes de los dispositivos de la zona industrial, albergados localmente o en la nube.

Finalmente, se desea aplicar una política de seguridad que permita establecer mecanismos de control de acceso, de modo que la navegación a través de los recursos de información de la red corporativa y el acceso remoto entre redes estén restringidas a trabajadores o usuarios no deseados.

El resultado final de este trabajo será una prueba de concepto que se corresponderá con

una implementación a menor escala de una infraestructura funcional que cumpla con los requisitos requeridos por la empresa y los objetivos propuestos en este trabajo. El producto final permitirá visualizar el proyecto en un entorno de prueba y validar los componentes, procedimientos y tecnologías necesarios para comprobar la viabilidad del proyecto en un entorno de producción y planificar su futura implementación en escala real.

Por último, durante la realización de este trabajo no sólo se buscará cumplir las exigencias generales de la propuesta inicial, sino que además se combinarán conocimientos de redes, ciberseguridad, bases de datos y otras ramas de la Informática para orientar el prototipo de red de la empresa hacia una solución segura, funcional, escalable y sobretodo ajustable a los recursos disponibles.

1.2 Objetivos

El objetivo principal del proyecto es diseñar e implementar una prueba de concepto de una red de empresa segura, escalable, redundante y segmentada que permita la conexión remota entre la red industrial y la red de datos corporativa, y garantice la monitorización remota de datos industriales, así como su transmisión, almacenamiento y el acceso restringido a distintos servicios y zonas de la red dependiendo del tipo de usuario. Para ello hay que cumplir los siguientes objetivos concretos:

- Analizar y seleccionar una herramienta de virtualización de entornos de red y aprovechar su uso junto con un entorno físico
- Analizar soluciones y tecnologías que permitan una conexión segura entre redes y seleccionar las que mejor se ajusten
- Proporcionar almacenamiento, servicio de monitorización de datos y acceso remoto a dispositivos
- Segmentar la red y administrar el control de acceso de distintos departamentos
- Diseñar la red y su implementación teniendo en cuenta la redundancia, escalabilidad y seguridad en nuestro contexto
- Implementar, configurar, automatizar y administrar la red empleando distintas tecnologías o herramientas
- Probar y analizar el despliegue y funcionamiento de la red, física y virtualmente, y contrastar ambas experiencias

1.3 Estructura

La memoria del proyecto se estructurará de la siguiente manera

- Introducción: introducimos como surgió y el contexto de nuestro caso de uso, sus objetivos y la estructura de la memoria.
- Fundamentos tecnológicos: describimos varias tecnologías en profundidad que ayuden a comprender mejor el trabajo realizado.
- Metodología: escogemos la metodología y explicamos sus ciclos.
- Preparación: Estudio de requisitos y objetivos a partir de la situación inicial del entorno de trabajo y selección y análisis a alto nivel de tecnologías necesarias para cumplirlos
- Planificación: exponemos y analizamos los recursos y tiempo necesarios para la realización del trabajo.
- Diseño: diseñamos a un nivel bajo y más específico las tecnologías que componen la topología del proyecto.
- Implementación: mostramos los programas, archivos y configuraciones empleadas en la red.
- Operación: con la red operativa realizamos pruebas que nos muestren el el correcto funcionamiento de acuerdo a los requisitos y tecnologías escogidas
- Conclusiones: Recogemos los resultados del trabajo realizado y comentamos conclusiones y sensaciones personales.

Capítulo 2

Fundamentos tecnológicos

EXPLICAREMOS previamente uno conceptos y tecnologías iniciales que se presentarán varias veces a lo largo de la memoria. Entenderlos con anterioridad facilita la interpretación y comprensión del trabajo realizado.

2.1 Herramientas de automatización

2.1.1 Ansible

Ansible^[2] es una herramienta software de código abierto dirigida a la automatización del despliegue, aprovisionamiento y configuración de sistemas de la información. Ansible es una herramienta de automatización de las [Tecnología de las Informaciones \(IT\)](#) que se basa en descripciones declarativas de las configuraciones deseadas en formato YAML. En el ámbito de redes, Ansible cuenta con colecciones de módulos especializados proporcionados por la comunidad y fabricantes que encapsulan tareas comunes de configuración. La estructura típica de Ansible incluye: un inventario de *hosts* que define qué dispositivos hay y sus datos de conexión (direcciones IP, credenciales, grupos, variables); uno o varios *playbooks* que son conjuntos de *plays* (secuencias de tareas a realizar en grupos de *hosts*) escritas en YAML; y los módulos, que son unidades de acción reutilizables que Ansible ejecuta en los dispositivos para llevar a cabo cada tarea (por ejemplo, un módulo `ios_vlan`^[3] para asegurar que una [Virtual Local Area Network \(VLAN\)](#) existe con determinado nombre en un *switch* Cisco). Ansible se conecta por Secure Shell (SSH) a los equipos y aplica los cambios de forma idempotente, es decir, solo realiza modificaciones si el estado actual difiere del deseado.

2.1.2 Terraform

Terraform^[4] es una herramienta de código abierto desarrollada por HashiCorp que emplea un lenguaje declarativo (HCL) para describir el estado deseado de recursos (máquinas

virtuales, redes informáticas, Domain Name System (DNS), etc.) sin detallar sus pasos de provisión. Con "terraform plan" se genera un plan de cambios perceptible antes de aplicar "terraform apply", garantizando idempotencia y orden de dependencias. Su fichero de estado, local o en *backend* remoto (Amazon S3, Consul, Terraform Cloud), sincroniza equipos y mantiene historial. Mediante módulos promueve la reutilización, y su amplio catálogo de proveedores (AWS, Azure, Google Cloud, Kubernetes...) unifica el aprovisionamiento en entornos heterogéneos.

2.2 Transmisión y recepción de datos

2.2.1 Protocolo Modbus

En entornos industriales es común el uso de protocolos *legacy* para control y adquisición de datos. Uno de los más difundidos es Modbus[5], en sus variantes serial (RTU/ASCII) y sobre TCP/IP. Modbus es un protocolo sencillo y abierto, desarrollado para la comunicación entre controladores lógicos programables conocidos como PLC. Sigue una arquitectura maestro-esclavo: el maestro envía peticiones de lectura/escritura, y los esclavos responden con los datos solicitados o ejecutan la acción indicada. Modbus define un espacio de direcciones de registros y *coils* (bobinas) que representan entradas, salidas y variables internas del dispositivo esclavo; por ejemplo, mediante códigos de función estándar, un maestro puede leer registros de entrada (lecturas analógicas) o escribir *coils* (salidas digitales) en un esclavo. Modbus TCP no es más que la adaptación de ese protocolo al mundo de Internet: encapsula las tramas Modbus en paquetes TCP (puerto estándar 502) aprovechando la infraestructura Ethernet/TCP-IP. En este modelo, el maestro Modbus actúa como cliente TCP y cada esclavo es un servidor TCP que escucha en el puerto 502. A nivel funcional, el maestro establece una conexión persistente con cada dispositivo esclavo y luego intercambia mensajes Modbus según el mismo formato que en serie (Modbus TCP replica casi 1:1 la especificación Modbus RTU en el payload de TCP). La simplicidad de Modbus es una ventaja para entornos industriales: es fácil de implementar, con bajo *overhead* y ampliamente soportado por dispositivos de distintos fabricantes (es un estándar de facto en la automatización industrial). Al no tener ningún tipo de cifrado en los datos transmitidos, normalmente Modbus se usa dentro de segmentos aislados o túneles Virtual Private Network (VPN) seguros cuando se requiere salir a redes corporativas o públicas.

2.2.2 MQTT Broker

En la solución propuesta, se emplea el protocolo Message Queueing Telemetry Transport (MQTT)[6] para la monitorización de los datos de dispositivos del entorno industrial. MQTT

es un protocolo de mensajería publicación/suscripción ligero, diseñado específicamente para entornos de **Internet of Things (IoT)** y **Machine-to-Machine (M2M)** donde los dispositivos pueden tener conectividad limitada o recursos de *hardware* reducidos. Opera sobre TCP/IP y sigue un modelo desacoplado en el que los clientes se conectan a un bróker central: algunos clientes publican mensajes en ciertos tópicos (temas) y otros se suscriben a esos tópicos para recibir los mensajes correspondientes. Esta arquitectura *hub-and-spoke* facilita comunicaciones uno-a-muchos y muchos-a-uno de forma asíncrona, sin que los emisores y receptores conozcan directamente la dirección unos de otros, es el bróker quien recibe todos los mensajes y los distribuye a los suscriptores interesados. **MQTT** se ha convertido en un estándar de facto para **IoT** por varios motivos técnicos. Es extremadamente ligero y eficiente: el encabezado de un mensaje **MQTT** puede ser de apenas 2 bytes, y el protocolo minimiza la sobrecarga en cada transmisión, lo cual es ideal para enlaces de baja velocidad (por ejemplo, redes celulares). Además, **MQTT** define tres niveles de **Quality of Service (QoS)** para la entrega de mensajes: “como máximo una vez”, “al menos una vez” y “exactamente una vez” (**QoS** 0, 1 y 2 respectivamente)

2.3 Fundamentos de redes

2.3.1 ACL

Las listas de control de acceso (**Access Control List (ACL)**)^[7] son un mecanismo fundamental de filtrado de tráfico en redes IP que se utiliza tanto en *routers* como en *firewalls*. Una **ACL** es básicamente una serie ordenada de reglas que definen si se permite o deniega el paso de paquetes según criterios de las cabeceras de red y transporte. Estos criterios típicamente incluyen la dirección IP de origen, la dirección IP de destino, el protocolo de nivel de transporte (TCP, UDP, ICMP, etc.) y, en caso de TCP/UDP, los puertos de origen o destino. Al procesar el tráfico, el dispositivo evalúa cada paquete contra las reglas en secuencia y aplica la primera que coincide; de no coincidir ninguna, suele haber una regla implícita final que bloquea el tráfico (*implicit deny*). Existen varios tipos de **ACL**: las **ACL** estándar filtran solo por la IP origen, mientras que las **ACL** extendidas permiten especificar origen, destino, protocolo y puertos, ofreciendo un control más granular del tráfico. Por ejemplo, es posible autorizar en una **ACL** extendida que cierto *host* o red origen pueda acceder a un puerto específico en un servidor y denegar el resto de accesos no necesarios. Las **ACL** pueden aplicarse tanto en sentido de entrada a una interfaz (*inbound*) como de salida (*outbound*), y una buena práctica es ubicarlas lo más cerca posible del origen del tráfico no deseado para evitar que paquetes indeseados recorran innecesariamente la red. Cisco, por ejemplo, recomienda aplicar **ACL** extendidas cerca de la fuente y **ACL** estándar (si se usan) cerca del destino, para optimizar el filtrado^[8].

2.3.2 ZeroTier

Para dotar de acceso remoto a la red, se ha optado por ZeroTier[9], una solución de redes definidas por software ([Software Defined Wide Area Network \(SD-WAN\)](#)[10]) que permite crear redes virtuales privadas sobre Internet de forma sencilla y segura. ZeroTier combina la facilidad de uso de una [VPN](#) punto a punto con capacidades avanzadas de virtualización de red, funcionando esencialmente como una [Local Area Network \(LAN\)](#) virtual distribuida a través de [WAN](#). A diferencia de las [VPN](#) tradicionales, donde suele haber un servidor central al que se conectan los clientes, ZeroTier crea una red malla (*mesh*) en la que todos los nodos pueden comunicarse directamente entre sí mediante técnicas de *peer-to-peer*. Emula una conectividad de capa 2: soporta tráfico *multipath*, *multicast* y *bridging*, lo que implica que los dispositivos remotos aparecen como si estuvieran en la misma red local, facilitando protocolos que requieren difusión o multidifusión en [LAN](#). Entre las ventajas principales de ZeroTier está su mínima configuración y rápida implementación: en pocos minutos es posible crear una red virtual, unir dispositivos (clientes) a ella y autorizarlos, asignándoles direcciones virtuales automáticas. La latencia añadida es muy baja, ya que siempre que es posible los nodos establecen enlaces directos (usando *UDP hole punching* para atravesar [Network Address Translation \(NAT\)](#) y [firewalls](#)) en lugar de reenviar todo el tráfico por servidores centrales. ZeroTier cifra todas las comunicaciones de extremo a extremo con claves de 256 bits, garantizando confidencialidad en los datos transmitidos. Además, la arquitectura incluye un controlador central (puede usarse el servicio en nube de ZeroTier o desplegar uno propio) que gestiona las pertenencias a la red y facilita la orquestación, pero el tráfico de datos en sí viaja de forma directa entre los clientes siempre que sea posible (no pasa por la nube de ZeroTier una vez establecida la conexión).

2.3.3 VLAN

Una [Virtual Local Area Network \(VLAN\)](#)[11] permite segmentar lógicamente una red local para separar el tráfico en distintos dominios de difusión. El estándar IEEE 802.1Q etiqueta las tramas Ethernet para identificar a qué [VLAN](#) pertenece cada paquete, manteniendo aislado el tráfico de diferentes redes lógicas incluso cuando se comparten los mismos enlaces físicos. Este mecanismo de etiquetado [VLAN](#) tiene un valor incalculable para reducir el alcance de la difusión y mejorar la seguridad aislando los segmentos sensibles de la red. Cada [VLAN](#) se identifica mediante un número (ID) comprendido entre 1 y 4095, aunque en la práctica suele limitarse a un rango menor (por ejemplo, 1-1000 en entornos corporativos). Cuando se implementan las [VLAN](#), es una buena práctica utilizar una única subred IP por [VLAN](#) para simplificar el enruteamiento y el control de acceso. Los puertos de *switch* configurados como troncal (etiquetados) transportan tráfico de varias [VLAN](#), mientras que los puertos no

etiquetados o de acceso pertenecen a una única **VLAN** específica. El mismo conjunto de **VLAN** permitidas y la misma **VLAN** nativa (sin etiquetar) deben configurarse de forma coherente en ambos extremos de cada enlace troncal, ya que las discrepancias en la **VLAN** nativa pueden causar problemas de seguridad como el *vlan hopping*[12]. También es aconsejable no utilizar ni cambiar la **VLAN** 1 por defecto de, así como asignar a los puertos no utilizados y a las **VLAN** nativas de los puertos troncales una **VLAN** residual que no se vaya a utilizar.

2.3.4 STP

El protocolo **Spanning Tree Protocol (STP)**[11] es esencial para evitar bucles de comunicación en las redes en las que existen enlaces redundantes (varios *switches* interconectados para proporcionar redundancia). **STP** (originalmente estandarizado como IEEE 802.1D) detecta los caminos redundantes en la topología y bloquea los enlaces sobrantes, construyendo una topología de árbol libre de bucles. De este modo, sólo habrá una única ruta activa entre dos *switches* cualesquiera, lo que evita las tormentas de difusión, la duplicación infinita de tramas y la saturación y consecuente pérdida de rendimiento de la red en caso de bucles. **STP** elige automáticamente a un *switch* como raíz **STP**, que actúa como referencia; los demás dispositivos de red bloquean algunos de sus puertos redundantes en función de las prioridades y los costes de ruta para garantizar un único camino activo. Aunque **STP** introduce un tiempo de convergencia (tradicionalmente 30-50 segundos con IEEE 802.1D), sus evoluciones como **Rapid Spanning Tree Protocol (RSTP)** (IEEE 802.1w) mejoran significativamente estos tiempos de reconvergencia cuando ocurren cambios de topología. En la práctica, se recomienda habilitar **RSTP** o variantes más modernas (**Multiple Spanning Tree Protocol (MSTP)**, **Per-Vlan Spanning Tree (PVST+)** en entornos Cisco, etc.) por las grandes mejoras en velocidad que ofrecen, aunque si combinamos varias versiones en una misma red se utilizará siempre la versión más antigua compatible. Gracias a **STP** y sus variantes, la red puede disponer de enlaces duplicados para resiliencia pero sin incurrir en bucles, lo que es crítico en entornos de alta disponibilidad

2.3.5 OSPF

Open Shortest Path First (OSPF)[13][14] es un protocolo de estado de enlace y de estándar abierto (**Interior Gateway Protocol (IGP)** de tipo *link-state*), lo que permite interoperabilidad entre equipos de distintos fabricantes. Cada *router* **OSPF** mantiene una base de datos topológica idéntica que describe la red completa (las áreas y sus enlaces). A partir de dicha base, cada *router* calcula de manera independiente el árbol de rutas de menor costo hasta el resto de destinos, usando el algoritmo de Dijkstra, y construye su tabla de enrutamiento en consecuencia. Una de las ventajas de **OSPF** es su rápida convergencia ante cambios: los routers OSPF envían actualizaciones incrementalmente solo cuando ocurre una modificación en la topología (por

ejemplo, caída de un enlace), en lugar de periódicamente, optimizando el uso de ancho de banda. Además, admite *equal-cost multi-path routing*, pudiendo balancear carga entre hasta 4–6 rutas de igual coste por defecto. OSPF soporta jerarquía mediante el concepto de áreas para escalar a redes grandes: se puede dividir la red en áreas OSPF, con un área troncal (*backbone area 0*) que conecta con las demás, reduciendo el tamaño de la base de datos y contenido el flooding de [Link State Advertisements \(LSA\)](#) dentro de cada área. En este proyecto OSPF es el encargado de propagar rutas IP entre el *router* extremo de la sede central y sus *switches* de distribución de forma automática y escalable, garantizando que nuevas subredes (por ejemplo, [VLANs](#) añadidas) se anuncien automáticamente sin necesidad de configurar rutas estáticas manuales. En definitiva, OSPF contribuye a la escalabilidad y resiliencia de la red en la fase de diseño e implementación del ciclo PPDIOO, asegurando una topología siempre disponible incluso ante fallos.

Capítulo 3

Metodología

3.1 PPDIOO

La metodología seleccionada es la [Preparar Planear Diseñar Identifica Operar Optimizar](#) (PPDIOO) de Cisco. Esta se centra en el diseño, implementación y mantenimiento de la infraestructura de redes por lo que se ajusta completamente a nuestro proyecto.

En nuestro caso la última de las fases de ésta metodología no se llegará a aplicar, ya que se basa en la optimización de una red operativa a lo largo de su vida útil y eso no entra dentro de nuestro rango de trabajo.

3.2 Fases principales

El transcurso del proyecto se dividirá en las siguientes fases que van de acuerdo a la metodología escogida:

- Preparación: Estudio de requisitos y recopilación de información de las tecnologías necesarias para crear nuestro caso de uso
- Planificación: Se identifican las necesidades de la red y el entorno de operaciones basado en los objetivos, medios, necesidades ...
- Diseño: se diseña toda la topología de la red con los requisitos técnicos necesarios que cumplan las metas propuestas mientras se aplican buenas prácticas de diseño de redes.
- Implementación: se instala y se configura toda la red y sus servicios de la manera prevista
- Operación: se despliega y se gestiona la red de manera que esté operativa y se pueda probar su funcionamiento

Capítulo 4

Preparación

4.1 Introducción

En esta fase identificamos principalmente los requisitos funcionales y de negocio de nuestro proyecto, así como sus limitaciones. También es el periodo donde se estudian la mayor parte de las tecnologías, estándares, implementaciones reales y buenas prácticas para poder plantear un buen diseño inicial de la topología. Finalmente se definen al completo los componentes que van a ser necesarios para el desarrollo e implementación de la infraestructura de red y de sistemas.

El objetivo general es interconectar las redes informáticas de dos lugares remotos pertenecientes a una misma entidad. Uno de estos lugares será una planta industrial, con una red ya diseñada y operativa, en la cual pueden coexistir decenas de dispositivos como sensores térmicos, PLCs, cámaras IP, robots industriales, HMI y ordenadores, que son capaces de generar, recoger y/o mostrar datos de interés para la monitorización de la actividad y estado de la planta. Aquí se deberán hacer los cambios y configuraciones mínimas necesarias sobre la infraestructura ya existente, conformada por un *router* y un dispositivo IoT en nuestro proyecto.

La otra zona será una sede u oficina central, en la que tendremos que diseñar e implementar desde cero una red con los sistemas necesarios para que se satisfagan ciertos requisitos mínimos. Antes de enumerar los requisitos cabe mencionar que en la oficina se diferencian dos grupos con distintos niveles de privilegios, un primer grupo privilegiado compuesto por los directivos y encargados del departamento IT y un segundo grupo no privilegiado compuesto por el resto de trabajadores. Se estima que hay no más de 60 trabajadores, entre ellos 20 directivos o encargados de IT. Teniendo todo esto en cuenta, los requisitos mínimos son:

- El tráfico que circule a través de esta conexión entre redes debe de estar cifrado para proteger la privacidad de la empresa.

- Los datos generados en la planta industrial deben almacenarse en la oficina con un formato estructurado.
- Los datos almacenados tienen que ser tratados de tal manera que se muestren tablas y gráficas en tiempo real o en períodos de tiempo pasados.
- La red industrial no debería tener ningún tipo de acceso directo dentro de la red local de la oficina.
- Los usuarios privilegiados deberían poder acceder a los dispositivos o sistemas remotos en la otra red desde la red de la oficina.
- Los usuarios privilegiados deberían poder acceder al almacenamiento y otros servicios locales.
- Los usuarios no privilegiados deben de tener una zona de la red aislada, donde únicamente puedan comunicarse con Internet y otros dispositivos de la misma red aislada. Nada proveniente de otras partes de la red local de la oficina debería poder acceder aquí.
- La red debería facilitar la escalabilidad en un futuro y proporcionar redundancia manteniendo unos gastos justificables dentro del contexto financiero de una Pequeña y Media Empresa (PYME).
- La infraestructura de la red debe ser orquestada y configurada mediante herramientas de automatización siempre que sea posible.

4.2 Topología

A través del estudio de tecnologías, topologías y documentación oficial de diseño de redes, hemos decidido que lo que más se ajusta a nuestro caso de uso es una topología de dos niveles de núcleo colapsado o *collapsed core network*[15][16], ya que reducimos el coste total a la vez que mantenemos la mayor parte de las ventajas de una topología de tres niveles, como la disponibilidad, rendimiento y facilidad para escalar el diseño de la red. La topología de núcleo colapsado surge de fusionar los niveles de núcleo y distribución en un único nivel de núcleo/distribución colapsado, que se encarga de agrupar las conexiones del nivel de acceso, proporcionar redundancia y realizar funciones inteligentes de enrutamiento y control de acceso al resto de la red.

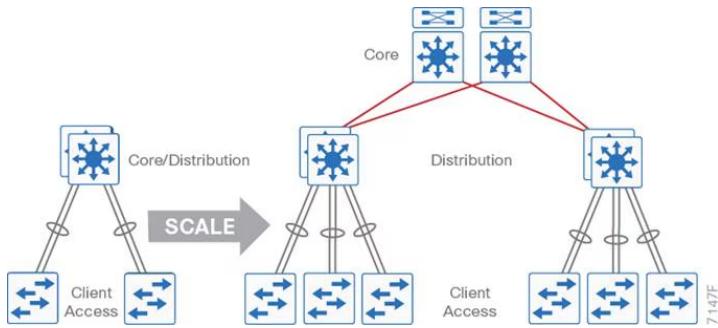


Figura 4.1: Escalabilidad de una topología colapsada a una de tres niveles. *Origen:[1]*

Además esta topología colapsada siempre se puede convertir en una de nivel tres si fuera necesario tal y como muestra la Figura 4.1. Por ejemplo, cuando la empresa crece y se expande a varios edificios o a un edificio entero.

En la Figura 4.2 se puede observar el diagrama de la red que vamos a diseñar:

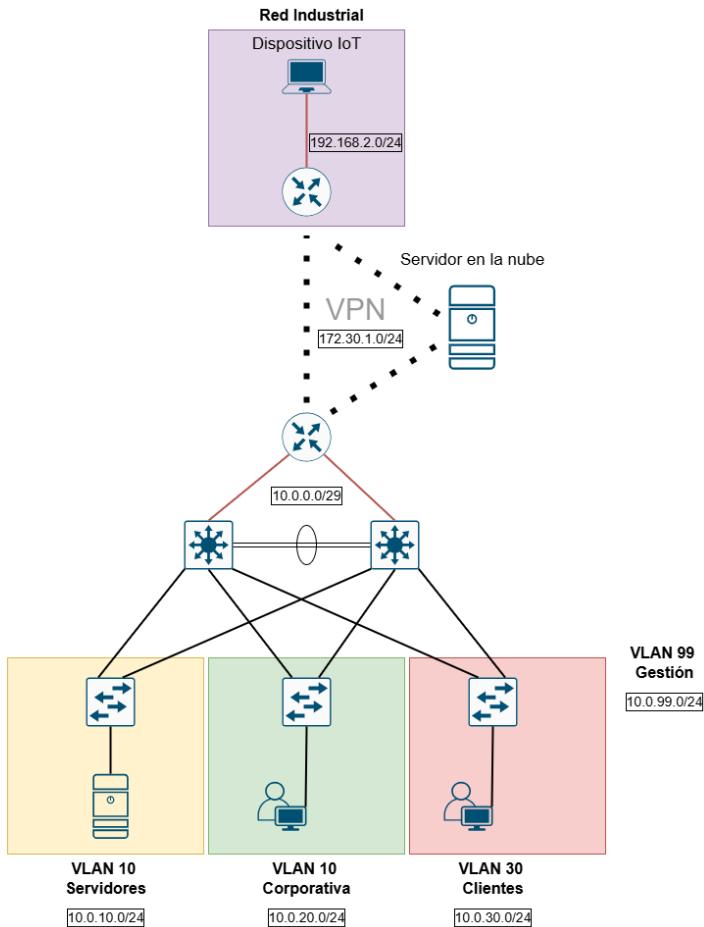


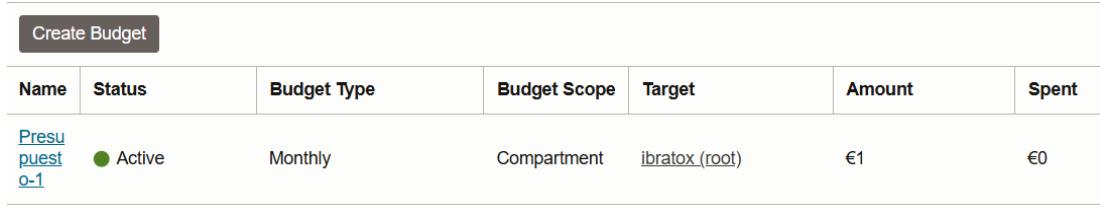
Figura 4.2: Diagrama de red del proyecto

Para la conexión remota entre redes vamos a utilizar una [VPN](#) de la que formarán parte los *routers* de ambas redes y un servidor en la nube. El servidor en la nube es nuestra solución a ciertos problemas que podríamos encontrarnos en la realidad:

1º: Uno o ambos *routers* pueden estar dentro de una [Carrier-Grade NAT \(CGNAT\)](#), que es el nombre que reciben las [NAT](#) de las proveedoras de Internet, gracias a la cual las direcciones de varios clientes pasan por un proceso de traducción que permite que todos ellos comparten la misma dirección IP pública. Si bien esto reduce el gasto de los proveedores y combate la escasez de direcciones del protocolo IPv4, tiene la desventaja de no se puede llegar a nuestro *router* directamente desde Internet. Así que cualquier servicio que necesite redirección de puertos como ciertas [VPN](#), por ejemplo Wireguard o OpenVPN, no se pueden utilizar si se alojan en o detrás de nuestro *router*. Por eso tener un servidor en la nube con una dirección IPv4 pública nos permite usarlo de intermediario en un túnel [VPN](#) en caso de que fuera necesario.

2º: Necesitamos un servidor que también haga de intermediario en nuestro caso de uso para la recogida o envío de datos. Esto no es solo también debido al problema anterior, sino que aún si no estuviésemos dentro de una [CGNAT](#), también careceríamos de un centro de Datos o un servidor local situado en una [demilitarized zone \(DMZ\)](#), una zona intermedia aislada donde se ubican servidores expuestos a Internet. En estas circunstancias queremos evitar posibles problemas de seguridad que pudieran surgir por permitir el acceso externo directo a nuestro servidor de la red local, por ello el uso de un servidor en la nube se adecúa a la situación.

Tras un breve periodo de investigación y comparar todas las posibilidades, se ha decidido utilizar el servicio de [Oracle Cloud Infrastructure \(OCI\) \(OCI\)](#)[17] para el despliegue de un servidor en la nube con una dirección IPv4 pública estática. La decisión se justifica en que Oracle ofrece un nivel de "acceso siempre gratis" que nos permite tener un servidor corriendo cada mes las 24 horas del día si lo aprovisionamos con un máximo de 4 OCPU y 24 GB de memoria, o también se podría, por ejemplo, tener dos instancias corriendo a la vez con 2 OCPU y 12 GB de memoria. Un OCPU equivale a un núcleo físico de un procesador Intel Xeon con *hyperthreading* habilitado. Para que el servidor sea permanente necesitamos registrar una cuenta, añadir un método de pago y confirmar el pago de un euro que luego reembolsan como verificación. Una vez hecho esto la cuenta se convertirá al modo *Pay as you go* en el que pagas por lo que consumes solamente si se exceden los límites de uso del acceso siempre gratis, por lo que si nos mantenemos dentro del límite antes mencionado y utilizamos un sistema operativo e infraestructura compatible con el acceso siempre gratis (viene indicado visualmente en la interfaz web), no hay peligro de tener que pagar por el servicio y para nuestro caso de uso esos límites son más que suficientes. En todo caso por precaución hemos declarado un presupuesto o *budget* que limite el gasto total de la cuenta a un euro como máximo, como se observa en la Figura 4.3.



The screenshot shows a table for creating a budget. The columns are: Name, Status, Budget Type, Budget Scope, Target, Amount, and Spent. There is one row with the following data:

| Name | Status | Budget Type | Budget Scope | Target | Amount | Spent |
|----------------|--------|-------------|--------------|----------------|--------|-------|
| Presupuest o-1 | Active | Monthly | Compartment | ibratox (root) | €1 | €0 |

Figura 4.3: Captura de pantalla del presupuesto añadido en Oracle Cloud

Para enviar los datos de la planta industrial y luego recogerlos en la oficina, con la finalidad de almacenarlos y visualizarlos, se va a utilizar un bróker MQTT que centralizará el envío y recogida de datos en un mismo servicio del servidor. Se utiliza MQTT por su flexibilidad y facilidad de uso frente a otras opciones. El bróker que hemos decidido utilizar es el proporcionado por Eclipse Mosquitto[18], una solución de código abierto muy ligera, fiable y fácil de utilizar.

La red de la oficina se divide en cuatro VLAN distintas:

- **VLAN 10:** contendrá los servidores locales de la empresa, en nuestro caso será un único servidor local. Esta VLAN tendrá asignada la subred 10.0.10.0/24, lo que quiere decir que el número de hosts está en el rango de 10.0.10.1-254, ya que el 0 y el 255 están reservados para la dirección de la red y la dirección broadcast. Según los requisitos mínimos, puede acceder a Internet, a la VLAN 20, a sí misma y a la red VPN. Sin embargo, como el acceso a la VPN por parte de los servidores no aparece explícita en los requisitos, consideramos que la mejor opción es que accedan al túnel VPN a través del router pero sin que los dispositivos exteriores puedan acceder directamente a los servicios del servidor si no es utilizando la redirección de puertos del router hacia la dirección local del servidor. Así reducimos el riesgo de ser objeto de un ataque o de provocar por accidente algún error en la red local.
- **VLAN 20:** en este dominio de difusión se encuentra lo que llamamos VLAN corporativa, es decir, donde se encuentran los usuarios privilegiados. Desde aquí los usuarios pueden acceder a la VLAN de los servidores, a dispositivos en la misma VLAN 20, a Internet y al túnel VPN para tanto acceder al servidor en la nube o router remoto como para acceder directamente a los dispositivos de la red local del router remoto.
- **VLAN 30:** esta es la VLAN que utilizarán usuarios no privilegiados, a la que llamamos red cliente, para realizar sus labores diarias. Podrán conectarse únicamente a Internet y a otros dispositivos de la misma zona.
- **VLAN 99:** la última VLAN de todas es en realidad utilizada para crear un dominio de difusión entre todos los switches para facilitar la gestión y automatización de los dispo-

sitivos de la red y no generar tráfico en las otras **VLAN**. Comprende únicamente a los *switches* de la red.

Los *switches* serán de tipo capa 2 en el nivel de acceso, a los cuales van conectados los equipos finales, y de capa tres o multicapa en el nivel de distribución (le llamaremos así a partir de ahora al nivel de distribución/núcleo colapsado). Los *switches* de capa 3 se diferencian en que presentan funcionalidades que les permiten operar a nivel IP, es decir, enrutar paquetes entre distintas **VLAN** sin que tengan que pasar por el *router*, aplicar listas de control de acceso a nivel de IP, utilizar protocolos de enrutamiento dinámico (pues también son capaces de tener su propia tabla de enrutamiento) y muchas más funcionalidades propias de un *router*. En la actualidad ya muchas marcas comercializan *switches* totalmente compatibles con la capa 3, hasta el punto de añadir nuevas variedades de topologías en el diseño de redes donde se utilizan enlaces de capa 3 en todos los niveles de la red, incluido el nivel de acceso. Las funcionalidades de capa 3 conllevan también un incremento del coste del producto y, para minimizar la inversión como en nuestro caso, decidimos colocar *switches* de acceso de tipo capa 2.

Todos los *switches* de acceso tienen dos conexiones a la capa de distribución (una por cada *switch* de distribución) que nos ayudan a proporcionar redundancia, lo cual se traduce en resistencia a fallos y alta disponibilidad de la red con las tecnologías necesarias. También hay redundancia en las conexiones entre los dos *switches* de distribución, cuya conexión suele utilizarse para el funcionamiento de tecnologías o protocolos de redundancia. Las conexiones finales de los *switches* de distribución terminan los dos en el mismo *router*, aunque no es algo que esté documentado como buena práctica en la documentación oficial, se mantuvo ésta decisión de diseño no sólo por estar limitados en la realidad a una única conexión de fibra óptica proveniente del proveedor de internet, sino porque contratar dos tarifas de fibra óptica no es una prioridad en una **PYME** mientras no crezca la red lo suficiente o se traten con datos tan críticos e importantes como que merezca la pena el gasto a mayores. También se limitó por decisiones financieras la utilización a un único servidor y un único *switch* de acceso en la **VLAN** de servidores, si bien se podría haber planteado un sistema de copias de seguridad o simular virtualmente varios servidores para conseguir una mayor disponibilidad de los servicios, no es suficientemente relevante al proyecto general como para añadir más funcionalidades de ese estilo.

Nuestro servidor local en la red de la oficina albergará una base de datos InfluxDB[19], el servicio web de Grafana[20] y por último otro servicio o herramienta para extraer los datos del bróker **MQTT** del servidor en la nube y guardarlos en la base de datos. La base de datos InfluxDB tiene un consumo mínimo de procesador y es ideal como sistema de almacenamiento en nuestro caso de uso ya que no es una base de datos relacional, sino una base de datos de series temporales, lo que significa que en vez de tablas, claves primarias, claves foráneas...

ahora los datos se almacenan en "buckets" por nombre de métrica, con cero o más etiquetas que los diferencien de otros datos del mismo tipo de métrica y, por último, tienen una marca de tiempo que puede generarse automáticamente al insertar datos o manualmente dentro de los parámetros de los datos que se quieran almacenar. Grafana es una aplicación que proporciona la posibilidad de crear paneles, gráficas y similares en una interfaz web interactiva en tiempo real, teniendo como finalidad la visualización gráfica de datos para monitorizar y analizar cualquier dato que forme parte de una fuente de datos compatible con Grafana[21].

Por último, debido a nuestras necesidades de conectar la red de la oficina a una [VPN](#), sería necesario utilizar un *router* que tenga compatibilidad con el tipo de [VPN](#) que vayamos a utilizar, para no dificultar más la lógica del diseño y también facilitar el control del tráfico entrante y saliente del túnel [VPN](#). Después de una larga búsqueda comparativa entre todas las opciones hemos decidido utilizar ZeroTier, una solución [VPN](#) muy sencilla de configurar y que, gracias a su implementación, hace que no sea necesario tener un intermediario, que generalmente se sitúa en la nube, para crear un túnel [VPN](#) entre dos dispositivos que se encuentra dentro de una [CGNAT](#). Así, el problema número uno mencionado como justificación del despliegue de un servidor en la nube, queda completamente resuelto aún si no dispusiésemos de un servidor en la nube.

4.3 Entorno

Nuestro entorno de trabajo durante este proyecto será, principalmente, virtual con una pequeña parte física (el *router* y dispositivo [IoT](#) de la planta industrial). Luego durante un breve periodo de tiempo se trasladará a un entorno totalmente físico para trabajar con equipos reales y comparar las diferencias entre lo virtual y lo real y comentar nuestras sensaciones y conclusiones al trabajar con ambos. Para la parte virtual se ha decidido usar GNS3[22], una aplicación gratuita de código abierto que permite emular redes informáticas con una gran variedad de dispositivos de distintas marcas como equipos Cisco, equipos Juniper, contenedores Docker que ofrecen distintos servicios, sistemas operativos como Windows, Debian... Se ha decidido escoger GNS3 frente a otras herramientas debido a su flexibilidad, documentación, gran comunidad y por disponer de una interfaz bastante intuitiva. La instalación se realizó en una máquina virtual de VMware Workstation, tal y como se recomienda en la documentación oficial. Una vez listo, posteriormente, se instalaron los sistemas y equipos virtualizados que vamos a utilizar en durante el desarrollo del proyecto. Estos son una máquina virtual Debian 12 para el servidor y como máquina principal para automatizar la red, un contenedor de Docker con Mozilla Firefox para acceder a interfaces web cuando sea necesario, un *router* OPNsense[23], cinco instancias de la imagen de *switch* Cisco IOSvL2[24], algunos ordenadores virtuales cuya imagen viene instalada por defecto con GNS3 para probar la conectividad

y otras funciones básicas y, por último, una instancia de **NAT**. Todo esto se ejecuta en un ordenador de sobremesa con 64GB de memoria **RAM** y un procesador Ryzen 5 3600 con 12 núcleos lógicos, ya que es el proceso de virtualización es costoso y consume muchos recursos en proyectos medianos y grandes.

La instancia de **NAT** en GNS3, como su nombre indica, hace uso de la tecnología **NAT** para traducir cualquier interfaz GNS3 que esté conectada a ella a la dirección de una tarjeta de red del equipo en el que se está ejecutando GNS3. Utiliza esta tarjeta o adaptador de red para enviar paquetes provenientes de la interfaz de GNS3 a Internet con la dirección del adaptador y los paquetes de respuesta se traducen en su vuelta al adaptador para llegar de nuevo a la interfaz virtual de GNS3. En resumen, nos permite tener acceso a Internet dentro de una máquina virtual de GNS3 sin mayor dificultad para el usuario que arrastrar la instancia y conectar su interfaz al equipo deseado.

OPNsense^[23] es una plataforma de código abierto basada en FreeBSD que combina un *firewall* de estado con potentes funciones de enrutamiento, **VPN**, balanceo de carga y **Quality of Service (QoS)**, todo gestionado desde su interfaz web. Aunque su propósito principal es el filtrado de paquetes, OPNsense incluye un completo plano de control de enrutamiento (con rutas estáticas, **OSPF** y **Policy Based Routing (PBR)**) de forma nativa. Esto nos permite emplearlo como *router* principal y nos ahorraremos tener que añadir un *firewall* adicional en un futuro para funcionalidades de seguridad avanzadas como, por ejemplo, un **Intrusion Detection System (IDS)**.

Para la capa de acceso y distribución en el laboratorio virtual utilizamos imágenes Cisco IOSvL2^[24], que son instancias emuladas del software IOS de los *switches* de Cisco. Estas imágenes reproducen las funcionalidades de un *switch* de capa 2 real (**VLAN**, **STP**, agregación de enlaces) y disponen de un plano de control limitado de capa 3, por lo que se podría considerar como un *switch* multicapa. A pesar de sus funcionalidades de capa 3 nos limitaremos a las funcionalidades de capa 2 en el nivel de acceso exceptuando la creación de **Switched Virtual Interface (SVI)**, de la que son capaces la gran mayoría de *switches* Cisco tanto antiguos como modernos, principalmente para permitir añadir una dirección IP a una interfaz virtual para facilitar tareas de gestión o mantenimiento remoto. GNS3 emula el *hardware* necesario para ejecutar estas imágenes tal y como lo haría un equipo físico, de modo que los mismos comandos y configuraciones son transferibles sin cambios entre los entornos virtual y físico. Así garantizamos que lo desarrollado con IOSvL2 en el laboratorio refleja con fidelidad el comportamiento de los *switches* reales Cisco. Cabe mencionar que la descarga de las imágenes Cisco no está directamente disponible en GNS3, es necesario una licencia de **Cisco Modelings Labs (CML)**, la cual tiene un precio fijo de 200€ anuales, nos permite descargar y utilizar imágenes de Cisco y también utilizar su propio laboratorio virtual para el diseño de redes, similar a GNS3, pero con equipos exclusivamente de la marca Cisco.

Para nuestro entorno físico se tendrán a disposición los siguientes equipos y componentes:

- 2 *Switches* Cisco Catalyst 3750 para la capa de distribución de la red, ya que tienen funcionalidades de capa 3
- 1 *Switch* Cisco Catalyst 3560 y 2 *switches* Cisco Catalyst 2960 para la capa de acceso. Son modelos de capa 2 a excepción del Catalyst 3560 que ofrece gran cantidad de funcionalidades de capa 3 que no serán utilizadas para adecuarse a la situación.
- 1 *Router* Teltonika RUT901[25] que hará de *router* de extremo en nuestra hipotética planta industrial. Es un *router* compatible con la tecnología celular 4G que se utiliza sobretodo en empresas que trabajan con elementos industriales, ya que además de enrutar ofrece funcionalidades como, por ejemplo, ser servidor o cliente Modbus a través de TCP entre otros,
- 1 Portátil HP Envy 15, que será el empleado para configurar los equipos y para participar en las pruebas de conectividad.
- 1 PC Lenovo ThinkCentre M73 8GB RAM / 120 GB SSD / 3.40GHz. Este ordenador será donde irá instalado el software OPNsense y por lo tanto lo que usaremos como *router*.
- PC Ryzen 5 3600 / 64GB RAM. El mismo equipo que se utilizó para ejecutar nuestro laboratorio virtualizado hará la función de servidor local en nuestro laboratorio físico.
- 1 Portátil Lenovo Ideapad Slim 3i. Será utilizado como dispositivo IoT en la red industrial, desde el cual se generarán los datos que tendrán que ser enviado al bróker MQTT y recibidos para almacenar en la red de oficina.
- Varios cables Ethernet de 50 centímetros para las conexiones de los *switches*, de 3 metros para conectar el ordenador de sobremesa al equipo OPNsense y de 15 metros para conectar el OPNsense al módem del proveedor de Internet.
- Cables de Consola USB a RJ45 específico para poder interactuar con las consolas de comandos de los *switches* Cisco.

Capítulo 5

Planificación

5.1 Etapas

La fase de planificación traduce los requisitos definidos en la fase de preparación en en tareas concretas, estimando tiempos y recursos. Para nuestro caso de uso hemos identificado las siguientes diez etapas principales:

1. Recopilación de información técnica

En esta fase se recopila la información ya obtenida hasta ahora, se investiga cómo se utilizan, implementan y funcionan las tecnologías y herramientas definidas en la fase de preparación como, por ejemplo el protocolo [MQTT](#) y el bróker Mosquitto, OPNsense, Cisco IOSvL2, GNS3, ZeroTier, entre otros. La duración de esta etapa fue de 35 horas a lo largo de dos semanas.

2. Especificación de bajo nivel de las tecnologías a emplear en la red

Consiste en diseñar la red del la topología especificando con claridad que tecnologías se van a emplear y en dónde. Esto se consigue tras comparar tecnologías, valorarlas en nuestro caso de uso e investigar alternativas, partiendo de nuestra topología de la fase de preparación. La duración de esta etapa fue de aproximadamente 20 horas a lo largo de cinco días.

3. Despliegue y configuración del servidor en la nube

Se busca aprovisionar una instancia Oracle Cloud dentro del límite de "acesso siempre gratis" y una vez activa se realizará, dentro de ella, la instalación de un bróker [MQTT](#) y la configuración para conectarse a la [VPN](#). Además, se buscará automatizar en la medida de lo posible el despliegue esta fase. La duración de esta etapa fue de 15 horas a lo

largo de dos días.

4. Configuración de los *switches* de acceso capa 2

Se configurarán los *switches* de acceso de manera que cumplan los requisitos y topología de la fase de preparación, automatizando todo lo posible este proceso. La duración de esta etapa fue de 18 horas a lo largo de cinco días.

5. Configuración de los *switches* de distribución

Se configurarán los *switches* de distribución capa 3 de manera que cumpla los requisitos y topología de la fase de preparación y se automatizará todo este proceso lo que sea posible. La duración de esta etapa fue de 44 horas a lo largo de 2 semanas.

6. Configuración de OPNsense.

Se configurará la [VPN](#), reglas de *firewall*, control de acceso, tecnologías de enrutamiento y se comprobará la viabilidad de intentar la automatización. La duración de esta etapa fue de 16 horas repartidos en 6-7 días dispersos.

7. Configuración del dispositivo IoT Y Router Industrial

Se utilizará un portátil como dispositivo IoT y en la medida de lo posible simulará ser un dispositivo industrial que genera los datos y se comunica con el router para luego enviarlos todos en una estructura personalizada a nuestro bróker MQTT. La duración de esta etapa fue de 12 horas a lo largo de cinco días.

8. Configuración del servidor en la sede corporativa

Instalar en el servidor las herramientas necesarias para recoger los datos de nuestro bróker MQTT, procesar su estructura y almacenarlos en la base de datos InfluxDB, a la cual se conectará la aplicación de Grafana para mostrar estos datos recogidos de manera visual. La duración de esta etapa fue de aproximadamente 18 horas a lo largo de una semana..

9. Realización de las pruebas finales en ambos entornos

Se verifican las pruebas realizadas en ambos entornos de prueba para detectar posibles errores y formar una conclusión enfrentando el laboratorio virtual con el laboratorio físico. La duración de esta etapa fue de aproximadamente 5 horas a lo largo de un día.

10. Documentación.

Esta fase es recurrente y se realiza durante y después la realización de las otras fases y tareas. La duración de esta etapa fue de aproximadamente 100 horas a lo largo de un mes y medio.

En total, las etapas declaradas en la fase de planificación abarcaron alrededor de en un espacio temporal de **7 semanas** de calendario y, aproximadamente, **289 horas** de trabajo.

Para saber la duración total del proyecto habría que añadirle el tiempo que se tardó en la fase de preparación (aproximadamente 20 horas en 1 semana) más el tiempo perdido en fallos externos no relacionados como incompatibilidad de versiones, problemas de rendimiento...(no más de 15 horas). Por lo que la duración total aproximada de este proyecto ronda las **319 horas**

5.2 Costes y recursos

5.2.1 Hardware

Para el *hardware* utilizaremos como referencia el precio de mercado del producto si es nuevo, el precio medio de segunda mano si es usado o el precio por el que se compró en caso de saberse y ser menor que las otras referencias.

| Hardware | Antigüedad (años) | Coste aprox. (€) |
|---------------------------------|-------------------|------------------|
| <i>HP Envy 15</i> | 10 | 600 |
| <i>Lenovo ThinkCentre M73</i> | 10 | 40 |
| <i>PC Personal de Sobremesa</i> | 7 | 650 |
| <i>Lenovo Ideapad Slim 3i</i> | 2 | 500 |
| <i>Teltonika RUT901</i> | 1 | 220 |
| <i>2 × Cisco Catalyst 3750</i> | 12 | 120 |
| <i>1 × Cisco Catalyst 3560</i> | 12 | 60 |
| <i>2 × Cisco Catalyst 2960</i> | 10 | 90 |
| TOTAL | - | 2180 |

Tabla 5.1: Listado de equipamiento y estimación de antigüedad/coste.

5.2.2 Software

Solo se contarán aquellos por lo que haya que pagar o insertar un método de pago.

| Software | Coste (€) |
|----------------------------|------------|
| Licencia Cisco CML | 200 |
| Oracle Cloud Pay as you Go | 0 |
| Total | 200 |

Tabla 5.2: Listado de Software con método de pago necesario

5.2.3 Recursos humanos

Para calcular el coste humano, en este caso nosotros, conocemos las horas trabajadas, el rol equivaldría a un contrato del grupo B2 del área 3 (Software, programación y explotación de sistemas) cuyo salario base definido en el BOE[26] es de 25.974,36€ o 14,76€/hora.

| Rol | Coste / hora (€) | Horas trabajadas | Coste total (€) |
|-------------|------------------|------------------|-----------------|
| Técnico BII | 14,76 | 312 | 4 605,12 |

Tabla 5.3: Listado de recursos humanos

5.2.4 Coste total

| Concepto | Coste total (€) |
|----------------------------------|-----------------|
| Hardware (Tabla 5.1) | 2 180,00 |
| Software (Tabla 5.2) | 200,00 |
| Recursos humanos (Tabla 5.3) | 4 605,12 |
| COSTE GLOBAL DEL PROYECTO | 6 985,12 |

Tabla 5.4: Resumen de costes del proyecto

Capítulo 6

Diseño

DURANTE la fase de Diseño desarrollamos una solución técnica completa y coherente a partir de los requisitos, restricciones y diseños de las fases anteriores. Definimos también la topología de la red a un nivel mucho más bajo que en las fases anteriores, es decir, detallando la configuración de puertos, asignación de [VLANs](#), tecnologías de redundancia y de enrutamiento, las herramientas de automatización, control de acceso, etc. El objetivo es poder entender e implementar la infraestructura de red sin ambigüedades, siendo ésta fase el último eslabón que separa la planificación teórica de la ejecución práctica que la continúa en las siguientes fases. El resultado son planos detallados que servirán de plantilla para realizar la próxima fase de implementación.

6.1 Herramienta de automatización

Al final de esta fase, teniendo ya toda la información de alto y bajo nivel de la infraestructura de red, es cuando podemos escoger la herramientas que nos permiten automatizar lo máximo posible la configuración de los dispositivos de red. Tras investigar con profundidad y comparar alternativas hemos decidido utilizar lo siguiente:

- **Terraform**[[4](#)]: Utilizamos el proveedor de Oracle para aprovisionar la instancia Oracle Cloud y gestionar su ciclo de vida con los ficheros de configuración. Gracias a su estado remoto, cualquier cambio queda versionado y la infraestructura puede reproducirse idénticamente en otra cuenta.
- **Ansible**[[2](#)]: En nuestro caso emplearemos Ansible para configurar todos los *switches* de la red de la oficina y para automatizar en el servidor de Oracle la instalación del bróker [MQTT Mosquitto](#)[[18](#)] y la conexión a la [VPN](#) de Zerotier[[9](#)] de manera consecutiva a su aprovisionamiento por parte de Terraform. También preveíamos utilizar Ansible para automatizar toda la gestión y configuración del *router* OPNsense gracias a la colección

disponible "ansibleguy.opnsense"[27], pero tras una exhausta revisión preferimos usar la interfaz web que proporciona OPNsense. Esto se debe a una documentación confusa, escasez o existencia de módulos obsoletos y a la falta de funcionalidades como actualizar el sistema, instalar *plugins*, crear un *bridge LAN* con varios puertos... Por todo ello reducimos el uso de Ansible con OPNsense a la posibilidad de hacer un *backup* y restauración automática de la configuración.

- **Python con librería netmiko:** Netmiko[28] es una librería multiplataforma de Python que simplifica la conexión a la línea de comandos de red y el envío de comandos. La usamos especialmente para el primer paso antes de emplear Ansible para la ejecución automática de las tareas, ya que permite automatizar la ejecución de comandos en el laboratorio virtual mediante el protocolo Telnet[29] o en el laboratorio físico mediante un puerto serial. Es necesario generar un nombre de dominio, un par de llaves Rivest, Shamir y Adleman (RSA), habilitar SSH, añadir un usuario, contraseñas y configurar una interfaz virtual con dirección IP alcanzable para que Ansible pueda correctamente conectarse mediante SSH y automatizar el resto.

6.2 Detalle de tecnologías en la capa 2

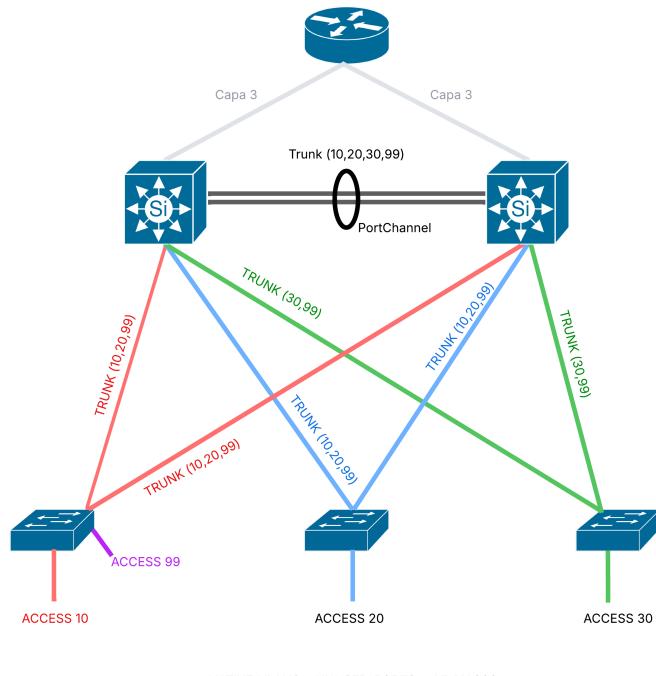


Figura 6.1: Diagrama de red detallado de capa 2 (sin mostrar STP)

El diagrama de red de la Figura 6.1 nos muestra en detalle las tecnologías y configuraciones que se le tienen que realizar a nuestra topología en la capa 2, ignorando de momento el STP.

En esta capa todo va a girar en torno a las **VLAN**, que son las encargadas de separar nuestra red local en cuatro dominios de difusión aislados que corresponden a cada uno de los departamentos y a la **VLAN** 99 de gestión. Hemos escogido una manera de acceder a la **VLAN** de gestión que brilla por su simplicidad a la vez que sacrificamos en parte algo de seguridad: dejar uno de los puertos del *switch* de acceso de la zona de servidores como puerto de acceso a la **VLAN** 99, normalmente el último de ellos para no olvidarse en el futuro. El acceso a ésta **VLAN** podría haberse planeado de una manera más segura como un *Jump Host*, que es un servidor, normalmente con medidas robustas de seguridad, que se utiliza para acceder y administrar dispositivos que se encuentran en una "zona" de seguridad distinta a la nuestra. Pero lo cierto es que para nuestro caso de uso y circunstancias actuales puede resultar una buena y muy simple solución temporal añadir un puerto de acceso la zona menos frecuentada de todas. Respecto a las otras **VLAN**, cada una tendrá en el *switch* de su departamento puertos de acceso que estarán conectados a usuarios finales y dos puertos o enlaces troncales conectados cada uno a un *switch* de distribución lo que proporciona redundancia a nuestra red que nos ayuda mantener una alta disponibilidad del tráfico con resistencia a fallos en las zonas donde se encuentran estas soluciones y/o tecnologías. Los enlaces troncales de cada departamento permitirán la transmisión de paquetes etiquetados que coincidan con alguna de las **VLAN** que hayamos permitido en la conexión. Las **VLAN** permitidas para cada enlace troncal, como se puede ver en la imagen, siguen la lógica de requisitos proveniente de la etapa de preparación, es decir, como las **VLAN** 10 y 20 tienen que tener conectividad entre ellas a través del enrutamiento que realizarán los *switches* de distribución, por eso en ese caso necesitamos que en enlace hacia los *switches* de distribución puedan viajar paquetes de estas dos **VLAN**, además de la de gestión, que aunque no necesite acceder a los departamentos, sí que tiene que poder llegar a todos los *switches* tanto de acceso como de distribución. Siguiendo las buenas prácticas de diseño de redes [30] se deben todos los puertos que no estén siendo utilizados y asignarlos a una **VLAN** residual distinta de la **VLAN** 1, que al ser la viene por defecto en todos los puertos de los equipos, debería evitarse al completo su uso, para reducir así la posibilidad de ataques como **VLAN hopping**[12]. Otra buena práctica es asignar otra **VLAN** residual como **VLAN** nativas de los enlaces troncales, así los paquetes sin etiquetar reciben la etiqueta de la **VLAN** residual que al no estar configurada en ningún dispositivo, simplemente, se propaga por el dominio de difusión de la **VLAN** residual hasta que encuentre un camino sin salida.

Por último, hemos creado un *PortChannel* [31][16] a partir de dos enlaces troncales entre los *switches* de distribución. El *PortChannel* agrupa como mínimo a dos interfaces hasta un máximo de ocho o dieciséis en un grupo que se comporta como si fuera una única interfaz lógica, proporciona balanceo de carga entre las interfaces del grupo, incrementa el ancho de banda

de la conexión y ofrece una alta disponibilidad y resistencia a fallos ya que el grupo permanece operativo siempre que haya al menos una de las interfaces funcionando correctamente. Los miembros del grupo *PortChannel* deben de ser compatibles, transmitiendo a la misma velocidad y operando en el modo *full-duplex*, es decir, que pueden enviar y recibir mensajes simultáneamente. Un *PortChannel* puede ser estático, donde todo se configura manualmente y el único fallo detectable es el fallo físico de una interfaz, o puede utilizar el protocolo [Link Aggregation Control Protocol \(LACP\)](#)[32][16]. Este protocolo detecta y desconecta del grupo automáticamente a interfaces mal configuradas, detecta fallos no sólo a nivel físico sino también cuando no recibe paquetes de respuesta tras un tiempo de alguna de las interfaces y cumple el estándar IEEE 802.3ad por lo que es compatible con gran parte de dispositivos de otras compañías.

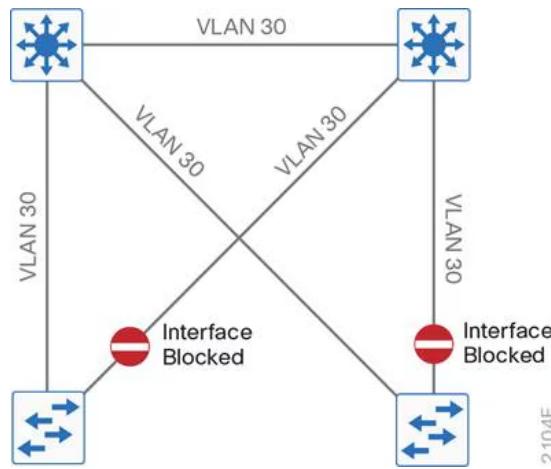


Figura 6.2: Diseño tradicional en bucle con VLAN. *Origen:[1]*

En nuestro caso el grupo *Portchannel* va a ser un enlace troncal con todas las [VLANs](#) de la red permitidas y vamos a configurarlo con el protocolo [LACP](#) por las ventajas que ofrece. El tipo de enlace del *Portchannel* es troncal y permite todas las [VLANs](#), en lugar de, por ejemplo, ser un enlace de capa 3. Esto se justifica por lo mostrado en la imagen superior: al tener dominios de difusión con bucles en nuestra topología cabe la posibilidad que el [STP](#) bloquee los enlaces de un *switch* de distribución con los *switches* de acceso. Si no se tuviera un enlace troncal entre los dos *switches* de distribución el *switch* podría quedar incomunicado. Se podría prescindir de un enlace de capa 2 por uno de capa 3 en contextos donde no hay bucles posibles en la topología.

6.3 Detalle de tecnologías en la capa 3

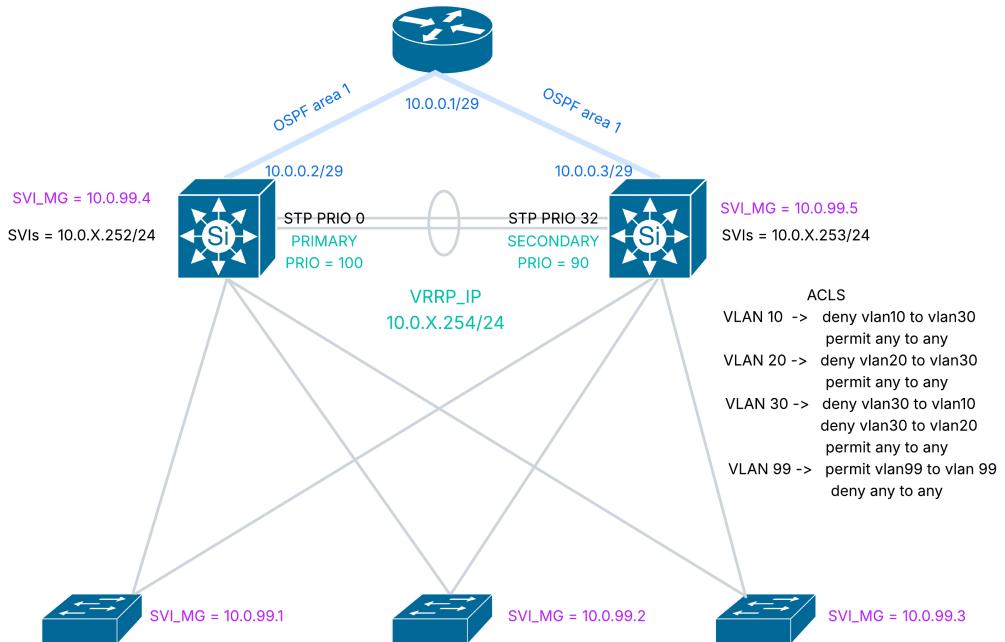


Figura 6.3: Diagrama de red detallado de capa 3 con STP

Ahora vamos a pasar a concretar las tecnologías en la capa 3 más el protocolo STP. Lo primordial es crear las **SVI** de la **VLAN** de gestión con sus IP correspondientes en cada uno de los **switches**, necesario para poder en el futuro acceder y gestionar de manera automática los dispositivos. Hemos decidido utilizar **Virtual Router Redundancy Protocol (VRRP)** [33][16] para proporcionar redundancia y tolerancia a fallos entre los **switches** de distribución. El protocolo funciona de tal manera que agrupa ambos **switches** y forma un único **switch lógico virtual**, el cuál tiene una dirección IP y MAC distintas de los **switches** del grupo y que se emplean para comunicarse con el resto de dispositivos. Los **switches** tienen un número de prioridad del protocolo que podemos cambiar y el que tenga mayor prioridad se convierte en el maestro del grupo, lo que significa que todos los paquetes que lleguen a las interfaces del **switch lógico virtual** serán redireccionados a través del **switch** maestro. El maestro es el encargado de enviar paquetes al grupo de **switches** para anunciar su estado. El resto de **switches** simplemente escuchan los avisos del maestro hasta el momento en que estos dejen de llegar por cualquier circunstancia y se vuelva elegir un nuevo maestro del grupo. Es recomendable asignar prioridades del **STP** de tal manera que se alinee el maestro del grupo **VRRP** con el **root STP**, para

que así se evite la posibilidad de tomar caminos subóptimos[34][16]. El protocolo **VRRP** se tendrá que aplicar sobre las **SVIs** de cada **VLAN**, usando en nuestro caso como IP compartida la terminada en 254 de la red de cada **SVI**. Se prefirió utilizar **VRRP** sobre otras similares por ser un protocolo no exclusivo genérico para todos los fabricantes.

A la derecha de la imagen vemos detalladas las **Access Control List (ACL)**[7] y sus reglas, que están basadas en la información recopilada en la fase de preparación. Solamente habría que añadirle un número de secuencia para que es ejecuten en el orden correcto e intercambiar el nombre de la **VLAN** por la dirección más los bits de *host*.

La última tecnología que veremos en ésta sección es el protocolo **OSPF**[13][14][16], presente en los enlaces de capa 3 que hay desde ambos *switches* de distribución hacia el *router* OPNsense. La función de éste protocolo es la propagación de rutas entre dispositivos y la elección de la ruta más rápida para el envío de los paquetes, lo cual nos asegura que, si uno de los enlaces cae, se reenvié el tráfico por el otro. Así el *router* aprende las rutas que le envíen los *switches* de manera dinámica y viceversa, ahorrándonos tener que declarar rutas estáticas que muy probablemente tengan que revisarse ante un cambio futuro en nuestra topología de red.

Capítulo 7

Implementación

EN la fase de implementación convertimos todas las decisiones de diseño recogidas hasta ahora en un entorno real o un laboratorio de pruebas. Vamos a explicar con detalle los archivos de configuración, el código y las acciones necesarias a seguir para materializar nuestros objetivos.

7.1 Automatización de la red

7.1.1 Inicio con Python y Netmiko

Para que Ansible pueda tener conectividad con los *switches* se tienen que conseguir que cada uno de ellos siga el proceso necesario para tener un par de llaves RSA locales, nombrar un dominio y asegurar la entrada indeseada a estos equipos mediante medidas de autenticación por contraseña o usuario/contraseña. Por ello hemos creado un *script* en Python que utiliza la librería Netmiko[28] para realizar una conexión Telnet[29] o *serial port* según escoja el usuario en los parámetros que acepta el programa.

setup.py

```
1  from netmiko import ConnectHandler
2  import argparse
3  import re
4
5  DEVICE_TEMPLATE = {
6      'device_type': 'cisco_ios_telnet',
7      'ip': '192.168.117.128',
8      'port': 5000,
9      'timeout': 60
10 }
11
12 DEVICE_TEMPLATE_SERIAL = {
13     "device_type": "cisco_ios_serial",
```

```

14     "serial_settings": {
15         "port": "X",
16         "baudrate": 9600,
17     },
18 }
19
20 COMMANDS = [
21     "ip domain-name sede.local",
22     "enable secret level 15 Admin1234.",
23     "username admin privilege 15 secret Admin1234.",
24     "ip ssh version 2",
25     "ip ssh time-out 60",
26     "crypto key generate rsa modulus 2048",
27     "interface vlan 99",
28     "ip address 10.0.99.0 255.255.255.0",
29     "no shutdown",
30     "line console 0",
31     "password cisco",
32     "login",
33     "line vty 0 4",
34     "transport input ssh",
35     "login local",
36     "exec-timeout 10 0"
37 ]
38
39
40 parser = argparse.ArgumentParser(description='Initial device setup')
41 parser.add_argument('--p', type=str, help='List of telnet ports to
   which connect', required=False)
42 parser.add_argument('--s', type=str, help='List of serial ports to
   which connect', required=False)
43 parser.add_argument('--ip', type=str, help='List of ending number
   of ip for each device in same order as ports', required=True)
44
45 args = parser.parse_args()
46 ip_list = [f'ip address 10.0.99.{ip} 255.255.255.0' for ip in
47     args.ip.split(',')]
48 port_list = []
49 serial_list = []
50 if args.p: port_list = [port for port in args.p.split(',')]
51 if args.s: serial_list = [serial for serial in args.s.split(',')]
52
53 if (len(port_list) != 0 != len(serial_list)):
54     raise Exception('Choose only one type between serial and telnet
      port')

```

```

55 if not (max(len(port_list),len(serial_list)) == len(ip_list)):
56     raise Exception('Number of arguments do not match each other')
57
58 for i in range(0,len(port_list)+len(serial_list)):
59     if serial_list:
60         device = DEVICE_TEMPLATE_SERIAL
61         device['serial_settings']['port'] = serial_list[i]
62     if port_list:
63         device = DEVICE_TEMPLATE
64         device['port'] = port_list[i]
65     print(device)
66     commands = COMMANDS
67     commands[7] = ip_list[i]
68     try:
69         net_connect = ConnectHandler(**device)
70         net_connect.enable()
71         print("Connected")
72         net_connect.send_command("terminal length 0")
73         int_output = net_connect.send_command('show ip int brief')
74         interfaces = re.findall(r'\b\S*ethernet\S*\b', int_output,
75                                re.IGNORECASE)
76         print("Looping...")
77         for match in interfaces:
78             commands.append(f"interface {match}")
79             commands.append(f"switchport access vlan 99")
80
81         output = net_connect.send_config_set(commands,
82                                             cmd_verify=False)
83         print(output)
84         net_connect.disconnect()
85     except Exception as e:
86         raise Exception(e)

```

El script "setup.py" en Python utiliza la librería Netmiko[28], una librería especializada en la automatización de dispositivos de red. El objetivo es estandarizar la configuración básica inicial de uno o más switches Cisco, dotándolos de parámetros comunes como el nombre de dominio, usuarios administrativos, accesos seguros, direcciones IP de gestión, VLAN de gestión y bloqueo de interfaces no utilizadas. De este modo, se minimizan errores humanos, se reduce considerablemente el tiempo necesario para configurar una gran cantidad de dispositivos y se garantiza la uniformidad en toda la infraestructura.

Al comienzo del código se definen tres variables globales: DEVICE_TEMPLATE y DEVICE_TEMPLATE_SERIAL se utilizan ambas para especificar el tipo y características de las conexiones, Telnet y *serial port* respectivamente. Luego se declara la variable COMMANDS que es una plantilla de todos los comandos que queremos que se ejecuten. En-

tre esos comandos se establece un nombre de dominio con `ip domain-name`, que es un requisito para poder generar el par de llaves RSA y habilitar SSH, se crea un usuario y contraseña para usarse más adelante como autenticación tanto en local como por SSH, se crea la SVI VLAN 99 y le añadimos la dirección IP que utilizaremos luego para conectarnos a los equipos, establece una contraseña para el *login* más básico del *switch* para que nadie pueda entrar sin medidas de protección aunque sea a través de la consola del *switch*.

Continuando ahora con el código funcional del *script* se puede observar como utilizamos la librería argparse[35] para permitir la entrada de parámetros al ejecutar el código, con el formato `--argumento x, y, z`. Hay un parámetro siempre obligatorio, `--ip`, que espera un solo número y será utilizado como el número del último octeto en la IP de la VLAN de gestión. Luego entre los otros dos parámetros solo es necesario uno de ellos dependiendo de la circunstancia, `--p` para puerto Telnet o `--s` para *serial port*.

Después de la entrada de los parámetros suministrados por el usuario, se llevan a cabo una serie de comprobaciones para asegurar que el *script* recibe información coherente: se verifica que no se mezclen argumentos de puertos Telnet y *serial port* al mismo tiempo y que la cantidad de puertos se alinee con el número de direcciones IP determinadas. Esta simple verificación inicial evita fallos de configuración futuros.

Finalmente, se recorre un bucle la misma cantidad de veces que número de argumentos se hayan introducido en donde:

- Primero se selecciona la plantilla de conexión de las variables globales según que parámetro de conexión se haya introducido. Aquí es donde actualiza de la lista de comandos el apartado de la dirección IP con la terminación introducida por el usuario y luego se intenta iniciar una conexión con Netmiko.
- Para el propósito que nos ocupa se deshabilita el límite de caracteres impresos de manera continua, ya que queremos enviar un comando que muestra por pantalla un resumen de todas las interfaces del dispositivo y, utilizando una expresión regular o *regex* mediante la librería re[36], cogemos únicamente el nombre de las interfaces y las inser-tamos en una lista.
- El último paso en el bucle es recorrer la lista de interfaces para usar los nombres es una plantilla de comando individual que se añade a la variable de comandos que habíamos copiado de la plantilla global. Ésta plantilla de comando lo que hace es poner todas las interfaces en la VLAN 99 (suponiendo que es la configuración inicial y por lo tanto todas las interfaces están en modo acceso) para simplificar futuras configuraciones y perdidas de conexión que podrían ocurrir al cambiar la VLAN de una conexión existente por SSH.

Podemos ver el comienzo de la ejecución del programa cuando hace uso de la conexión Telnet en la Figura 7.1 y serial en la Figura 7.2.

```
PS C:\Users\ibra4\Downloads\initialsetuptools> python .\setup.py --p 5006,5004,5010,5000,5002 --ip 1,2,3,4,5
{'device_type': 'cisco_ios_telnet', 'ip': '192.168.117.128', 'port': '5006', 'timeout': 60}
Connected
Looping...

Switch#
Switch#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)
Switch(config)#
Switch(config)#
Switch(config)#ip domain-name sede.local
Switch(config)#
Switch(config)#enable secret level 15 Admin1234.
Switch(config)#
Switch(config)#username admin privilege 15 secret Admin1234.
Switch(config)#
Switch(config)#ip ssh version 2
Please create RSA keys to enable SSH (and of atleast 768 bits for SSH v2).
```

Figura 7.1: Ejecución del programa setup.py para conexiones Telnet

```
(env) PS C:\Users\ibra4\Downloads\testserialport> python .\main.py --ip 1 --s COM9
{'device_type': 'cisco_ios_serial', 'serial_settings': {'port': 'COM9', 'baudrate': 9600}}
Connected
Looping...
ip domain-name sede.local
Switch(config)#enable secret level 15 Admin1234.
Switch(config)#username admin privilege 15 secret Admin1234.
Switch(config)#ip ssh version 2
Please create RSA keys to enable SSH (of atleast 768 bits size) to enable SSH v2.
Please create RSA keys to enable SSH (of atleast 768 bits size) to enable SSH v2.
```

Figura 7.2: Ejecución del programa setup.py para conexiones serial

7.1.2 Continuación con Ansible

Lo siguiente sería aprovechar el hecho de que los *switches* ya permiten las sesiones SSH y tienen una IP para conectarse con Ansible[2] y ejecutar todas las configuraciones de manera automática. Pero antes de cualquier configuración vamos a explicar rápidamente qué estructura de archivos y carpetas se sigue y cuál es la finalidad de cada una.

- **./ansible.cfg** - Es el archivo de configuración por defecto de Ansible donde se pueden especificar tipos de conexión, tiempos de espera, silenciar advertencias, rutas a las carpetas de inventario, colecciones, etc.
- **./collections/requirements.yml** - Aquí van declarados las colecciones de Ansible que se quieran o tengan que descargar para su posterior uso.
- **./inventory/hosts.ini** - Se establecen los nombres de los *hosts* y los grupos a los que pertenecen si es el caso para poder referenciar al ejecutar una tarea a exactamente qué grupo de *hosts* queremos que le afecte.

- **./inventory/group_vars/*** - En este directorio group_vars pueden opcionalmente existir archivos .yml donde se guardan todas las variables que pertenecen a un grupo concreto, el cual se indica con el nombre del archivo en sí.
- **./inventory/host_vars/*** - Sigue la misma lógica que el anterior pero ahora enfocado a las variables de un único host en concreto.
- **./playbooks/*** - Es el lugar donde se guardan las plays de Ansible que asocian un grupo de hosts con una serie de tareas a ejecutar. También se le puede indicar privilegios, variables, roles...
- **./collections/roles/*** - Son funcionalidades que agrupan tareas con variables en un formato de carpetas específico. Por ejemplo el rol /test/ puede tener /test/tasks/ para guardar dentro las tareas de este rol y /test/vars/ para almacenar variables relacionadas con las tareas del rol.

collections/requirements.yml

```

1 #ansible-galaxy collection install -r collections/requirements.yml
2 collections:
3     - name: cisco.ios
4         version: 8.0.0
5     - name: ansibleguy.opnsense
6     - name: ansible.utils
7         version: 4.1.0
8     - name: ansible.netcommon
9         version: 6.1.3

```

Se nombran las 4 colecciones que se utilizan para ejecutar correctamente nuestras tareas de automatización de la red. La colección ansible.utils[37] ofrece herramientas para gestionar, cambiar y visualizar los datos. Ansible.opnsense[27] es una colección para automatizar configuraciones en el router OPNsense, aunque le faltan muchas funcionalidades y no es tan intuitivo como otras colecciones. Ansible.netcommon[38] y Ansible.ios[39] se utilizan ambos para tareas de configuración de redes, siendo netcommon más general e ios enfocado a dispositivos de Cisco.

group_varsall.yml

```

1 # CISCO SWITCHES CONNECTION
2 ansible_user: admin
3 ansible_password: Admin1234.
4 ansible_connection: network_cli
5 ansible_network_os: cisco.ios.ios

```

Estas son las variables que van a compartir todos los *hosts* existentes. Entre ellas el tipo de conexión que se va a realizar con los otros dispositivos, el *firmware* o sistema operativo que utilizan y el nombre y contraseña necesarias para la conexión a los equipos a través de **SSH**.

ansible.cfg

```

1 [defaults]
2 inventory = inventory/hosts.ini
3 collections_path = ./collections
4 roles_path = ./roles
5 host_key_checking = False
6 retry_files_enabled = False
7 ansible_python_interpreter=/home/debian/.venv
8 persistent_command_timeout = 180

```

Es el archivo de configuración por defecto que ya comentamos con anterioridad.

playbooks/netsetup

```

1 - name: Base configuration
2   hosts: 12switches:13switches
3   gather_facts: no
4   roles:
5     - base
6
7 - name: Configuring Vlans and SVIs
8   hosts: 12switches:13switches
9   gather_facts: no
10  roles:
11    - vlan
12
13 - name: Setting Ip Routing and Acls
14   hosts: 13switches
15   gather_facts: no
16   roles:
17     - acl
18
19 - name: Distribution Switch Configuration
20   hosts: 13switches
21   gather_facts: no
22   roles:
23     - dist_switch

```

Este *playbook* de Ansible ejecuta los roles consecutivamente en el orden y a los *hosts* declarados en el archivo.

role/base/...main.yml

```

1 - name: gather facts
2   cisco.ios.ios_facts:

```

```

3   gather_subset: interfaces
4
5 - name: debug
6   debug:
7     msg: "{{ansible_net_interfaces}}"
8
9 - name: Set hostname
10  cisco.ios.ios_hostname:
11    config:
12      hostname: "{{ inventory_hostname }}"
13
14 - name: Create UNUSED_VLAN
15  cisco.ios.ios_config:
16    lines:
17      - "vlan 999"
18      - "name UNUSED_VLAN"
19
20 - name: Shutdown all unused interfaces and assign UNUSED_VLAN
21  cisco.ios.ios_config:
22    lines:
23      - "switchport access vlan 999"
24      - "shutdown"
25    parents: "interface {{ item.key }}"
26  loop: "{{ ansible_net_interfaces | dict2items }}"
27  when: item.value.operstatus == "down"
28
29 - name: Setting access ports
30  cisco.ios.ios_12_interfaces:
31    config:
32      - name: "{{ item.interface }}"
33        mode: "{{ item.mode }}"
34        access:
35          vlan: "{{ item.vlan }}"
36  loop: "{{ access_ports }}"
37  when: access_ports is defined
38
39 - name: Set encapsulation for trunk ports
40  cisco.ios.ios_12_interfaces:
41    config:
42      - name: "{{ item.interface }}"
43        trunk:
44          encapsulation: "{{ item.encapsulation }}"
45  loop: "{{ trunk_ports }}"
46
47 - name: Cambiar los trunks y, si se pierde la sesión, reintentarlo
48  block:

```

```

49      - name: Configurar trunks
50        cisco.ios.ios_12_interfaces:
51          config:
52            - name: "{{ item.interface }}"
53              trunk:
54                allowed_vlans: "{{ item.allowed_vlans }}"
55                encapsulation: "{{ item.encapsulation }}"
56                native_vlan: "{{ item.native_vlan }}"
57                mode: "{{ item.mode }}"
58                state: merged
59              loop: "{{ trunk_ports }}"
60
61    rescue:
62      - name: Esperar hasta 2 min a que vuelva la conexión
63        wait_for_connection:
64          delay: 10
65          sleep: 5
66          timeout: 120
67        register: wait_result
68
69
70      - meta: reset_connection
71
72    - name: Reintentar la configuración de trunks
73      cisco.ios.ios_12_interfaces:
74        config:
75          - name: "{{ item.interface }}"
76            trunk:
77              allowed_vlans: "{{ item.allowed_vlans }}"
78              encapsulation: "{{ item.encapsulation }}"
79              native_vlan: "{{ item.native_vlan }}"
80              mode: "{{ item.mode }}"
81            state: merged
82          loop: "{{ trunk_ports }}"

```

El rol "base" es el primero que se ejecuta para configurar los puertos necesarios en el equipo. Primero recolecta información sobre todas las interfaces del *host*, crea la [VLAN](#) residual 999 para luego asignársela y a continuación apagar todos los puertos que tengan el estatus de "down" (en un *switch* "down" son los puertos sin ningún cable o conexión activa) a partir de la información recolectada.

Ahora podemos observar una técnica recursiva durante todo el proyecto en las tareas de Ansible. Se trata de utilizar una variable del *host* como lista para recorrer en un bucle y utilizar las variables de cada iteración por ejemplo para configurar puertos de acceso distintos en la misma tarea y centralizar los cambios que tiene que realizar el usuario si quiere modificar

la topología, que será dentro de los inventario de los *hosts*. Más adelante se mostrará una plantilla con las variables que pueden llegar a tener y se utilicen en algún rol. En este caso recorremos la lista de variables con puertos de acceso y como son diccionarios referenciamos a las llaves para utilizar sus valores, configurando así con la estructura proporcionada por el modulo `ios_l2_interfaces` que se puede encontrar en la documentación.

Lo siguiente es establecer en el tipo de encapsulado o etiquetado en los puertos troncales antes de su configuración.

Ahora un paso muy importante es que aprovechamos el uso del bloque de Ansible, que crea grupos lógicos de tareas, y eso nos permite establecer un apartado `"rescue"` cuyo contenido se ejecuta cuando hay un error en el bloque. En el bloque lo que se hace es establecer oficialmente los modos de los puertos troncales a troncales, pero esto puede hacer que la conexión con otros *hosts* se pierda si no estás conectado directamente a ellos y coincide justamente el puerto troncal recién establecido con el que se utilizaba en la ruta para acceder al *host*. Lo que hace aquí en caso de fallo es entrar en el apartado `"rescue"`, esperar a que vuelva la conexión un máximo de 2 minutos (si la negociación de los puertos de acceso está en auto no debería haber problema más que el tiempo que tarde el proceso de negociación), luego reinicia la conexión SSH con otra sesión nueva y vuelve a intentar ejecutar la misma tarea que la que se encuentra dentro del bloque.

Podemos observar fragmentos de la ejecución de las tareas de este rol en las Figuras 7.3, 7.4, 7.5, 7.6 y 7.7.

```

TASK [base : Set hostname] ****
ok: [swl2_corporate]
ok: [swl2_clients]
ok: [swl3_dist2]
ok: [swl3_dist1]
ok: [swl2_server1]

TASK [base : Create UNUSED_VLAN] ****
[WARNING]: To ensure idempotency and correct diff the input configuration
similar to how they appear if present in the running configuration on d
changed: [swl2_clients]
changed: [swl2_corporate]
changed: [swl3_dist2]
changed: [swl3_dist1]
changed: [swl2_server1]

TASK [base : Shutdown all unused interfaces and assign UNUSED_VLAN] ***
skipping: [swl2_server1] => (item={'key': 'GigabitEthernet0/0', 'value'
', 'mtu': 1500, 'bandwidth': 1000000, 'mediatype': 'unknown media type'
': 'up', 'type': 'iGbE', 'ipv4': []})
skipping: [swl2_server1] => (item={'key': 'GigabitEthernet0/1', 'value'
', 'mtu': 1500, 'bandwidth': 1000000, 'mediatype': 'unknown media type'
': 'up', 'type': 'iGbE', 'ipv4': []})
skipping: [swl2_clients] => (item={'key': 'GigabitEthernet0/0', 'value'
', 'mtu': 1500, 'bandwidth': 1000000, 'mediatype': 'unknown media type'
': 'up', 'type': 'iGbE', 'ipv4': []})

```

Figura 7.3: Ejecución del playbook con el rol `"base"` en Ansible 1

```
TASK [base : Setting access ports] *****
changed: [swl2_corporate] => (item={'interface': 'g0/2', 'mode': 'access', 'vlan': 20})
changed: [swl2_clients] => (item={'interface': 'g0/2', 'mode': 'access', 'vlan': 30})
changed: [swl3_dist2] => (item={'interface': 'g3/3', 'mode': 'access', 'vlan': 99})
changed: [swl3_dist1] => (item={'interface': 'g3/3', 'mode': 'access', 'vlan': 99})
changed: [swl2_server1] => (item={'interface': 'g0/2', 'mode': 'access', 'vlan': 10})
changed: [swl2_corporate] => (item={'interface': 'g3/3', 'mode': 'access', 'vlan': 99})
changed: [swl2_clients] => (item={'interface': 'g3/3', 'mode': 'access', 'vlan': 99})
changed: [swl2_server1] => (item={'interface': 'g3/3', 'mode': 'access', 'vlan': 99})

TASK [base : Set encapsulation for trunk ports] *****
changed: [swl2_corporate] => (item={'interface': 'gi0/0', 'mode': 'trunk', 'native_vlan': 1, 'encapsulation': 'dot1q'})
changed: [swl3_dist2] => (item={'interface': 'gi0/0', 'mode': 'trunk', 'native_vlan': 1, 'encapsulation': 'dot1q'})
changed: [swl2_clients] => (item={'interface': 'gi0/0', 'mode': 'trunk', 'native_vlan': 1, 'encapsulation': 'dot1q'})
changed: [swl3_dist1] => (item={'interface': 'gi0/0', 'mode': 'trunk', 'native_vlan': 1, 'encapsulation': 'dot1q'})
changed: [swl2_server1] => (item={'interface': 'gi0/0', 'mode': 'trunk', 'native_vlan': 1, 'encapsulation': 'dot1q'})
```

Figura 7.4: Ejecución del playbook con el rol "base" en Ansible 3

```
TASK [base : Configurar trunks] *****
changed: [swl2_server1] => (item={'interface': 'gi0/0', 'mode': 'trunk', 'natncapsulation': 'dot1q'})
changed: [swl2_server1] => (item={'interface': 'gi0/1', 'mode': 'trunk', 'natncapsulation': 'dot1q'})
failed: [swl2_corporate] (item={'interface': 'gi0/0', 'mode': 'trunk', 'natncapsulation': 'dot1q'}) => {"ansible_loop_var": "item", "changed": false, "item": "dot1q", "interface": "gi0/0", "mode": "trunk", "native_vlan": 999}, "msg": "Module failed - No start of json char found\nSee stdout/stderr for the full output", "module_stderr": "Module failed - No start of json char found\nSee stdout/stderr for the full output", "module_stdout": "", "rc": 1}
failed: [swl2_clients] (item={'interface': 'gi0/0', 'mode': 'trunk', 'natncapsulation': 'dot1q'}) => {"ansible_loop_var": "item", "changed": false, "item": "dot1q"}
```

Figura 7.5: Ejecución del playbook con el rol "base" en Ansible 4

```
TASK [base : Esperar hasta 2 min a que vuelva la conexión] *
[DEPRECATION WARNING]: The connection's stdin object is depr-
display.prompt_until(msg) instead. This feature will be remo-
warnings can be disabled by setting deprecation_warnings=Fal-
ok: [swl3_dist2]
ok: [swl2_clients]
ok: [swl2_corporate]
ok: [swl3_dist1]

TASK [base : meta] ****
```

Figura 7.6: Ejecución del playbook con el rol "base" en Ansible 5

```

TASK [base : Reintentar la configuración de trunks] **
changed: [swl2_clients] => (item={'interface': 'gi0/0',
psulation': 'dot1q'})
changed: [swl2_corporate] => (item={'interface': 'gi0/
'encapsulation': 'dot1q'})
changed: [swl3_dist2] => (item={'interface': 'gi0/0',
apsulation': 'dot1q'})
changed: [swl3_dist1] => (item={'interface': 'gi0/0',
apsulation': 'dot1q'})
changed: [swl2_clients] => (item={'interface': 'gi0/1'
psulation': 'dot1q'})
changed: [swl2_corporate] => (item={'interface': 'gi0/
'encapsulation': 'dot1q'})
changed: [swl3_dist2] => (item={'interface': 'gi0/2',
apsulation': 'dot1q'})
changed: [swl3_dist1] => (item={'interface': 'gi0/2',
apsulation': 'dot1q'})
changed: [swl3_dist2] => (item={'interface': 'gi0/3',
ulation': 'dot1q'})
changed: [swl3_dist2] => (item={'interface': 'Port-cha
30,99', 'encapsulation': 'dot1q'})
changed: [swl3_dist1] => (item={'interface': 'gi0/3',
ulation': 'dot1q'})
changed: [swl3_dist1] => (item={'interface': 'Port-cha
30,99', 'encapsulation': 'dot1q'})

PLAY RECAP ****
swl2_clients : ok=9    changed=6    unre
swl2_corporate : ok=9    changed=6    unre
swl2_server1   : ok=8    changed=6    unre
swl3_dist1     : ok=9    changed=6    unre
swl3_dist2     : ok=9    changed=6    unre

```

Figura 7.7: Ejecución del playbook con el rol "base" en Ansible 6

role/vlan/...main.yml

```

1 - name: Create VLANs Layer 2
2   cisco.ios.ios_vlans:
3     config:
4       - vlan_id: "{{ item.id }}"
5         name: "{{ item.name }}"
6         shutdown: disabled
7     loop: "{{ vlans }}"
8     when: vlans is defined
9
10 - name: Crear SVIs for VLAN Layer 3
11   cisco.ios.ios_13_interfaces:
12     config:
13       - name: "{{ item.name }}"
14         ipv4:
15           - address: "{{ item.address }}/{{ item.mask }}"
16     loop: "{{ svls }}"

```

```

17    when: svls is defined
18
19 - name: No shutdown
20   cisco.ios.ios_config:
21     lines:
22       - "interface {{ item.name }}"
23       - "no shutdown"
24   loop: "{{ svls }}"
25   when: svls is defined

```

Este es un rol básico y sencillo que crea las **VLANs** y los **SVIs** especificados en las variables y los enciende como último paso del rol para evitar que queden algunas interfaces apagadas.

Se pueden observar fragmentos de la ejecución de las tareas de este rol en las Figuras 7.8 y 7.9.

```

PLAY [Configuring Vlans and SVIs] ****
TASK [vlan : Create VLANs Layer 2] ****
ok: [swl2_clients] => (item={'id': 30, 'name': 'Clients'})
ok: [swl3_dist2] => (item={'id': 10, 'name': 'Servers'})
ok: [swl2_corporate] => (item={'id': 10, 'name': 'Servers'})
ok: [swl3_dist1] => (item={'id': 10, 'name': 'Servers'})
ok: [swl2_server1] => (item={'id': 10, 'name': 'Servers'})
ok: [swl3_dist2] => (item={'id': 20, 'name': 'Corporate'})
ok: [swl2_clients] => (item={'id': 99, 'name': 'Management'})
ok: [swl2_corporate] => (item={'id': 20, 'name': 'Corporate'})
ok: [swl3_dist1] => (item={'id': 20, 'name': 'Corporate'})
ok: [swl2_server1] => (item={'id': 20, 'name': 'Corporate'})
ok: [swl3_dist2] => (item={'id': 30, 'name': 'Clients'})
ok: [swl2_corporate] => (item={'id': 99, 'name': 'Management'})
ok: [swl2_server1] => (item={'id': 99, 'name': 'Management'})
ok: [swl3_dist1] => (item={'id': 30, 'name': 'Clients'})
ok: [swl3_dist2] => (item={'id': 99, 'name': 'Management'})
ok: [swl3_dist1] => (item={'id': 99, 'name': 'Management'})

TASK [vlan : Crear SVIs for VLAN Layer 3] ****
changed: [swl2_clients] => (item={'name': 'vlan 99', 'address': '10.0.0.10', 'vrrp_id': '10', 'stp_prio': 32})
changed: [swl2_corporate] => (item={'name': 'vlan 99', 'address': '10.0.0.10', 'vrrp_id': '10', 'stp_prio': 32})
changed: [swl3_dist2] => (item={'name': 'vlan 99', 'address': '10.0.0.10', 'vrrp_id': '10', 'stp_prio': 32})
changed: [swl2_server1] => (item={'name': 'vlan 99', 'address': '10.0.0.10', 'vrrp_id': '10', 'stp_prio': 32})
changed: [swl3_dist1] => (item={'name': 'vlan 99', 'address': '10.0.0.10', 'vrrp_id': '10', 'stp_prio': 32})
changed: [swl3_dist2] => (item={'name': 'vlan 10', 'address': '10.0.0.20', 'vrrp_id': '20', 'stp_prio': 32})
changed: [swl3_dist1] => (item={'name': 'vlan 10', 'address': '10.0.0.20', 'vrrp_id': '20', 'stp_prio': 32})
changed: [swl2_clients] => (item={'name': 'vlan 20', 'address': '10.0.0.254', 'vrrp_id': '20', 'stp_prio': 32})
changed: [swl2_corporate] => (item={'name': 'vlan 20', 'address': '10.0.0.254', 'vrrp_id': '20', 'stp_prio': 32})
changed: [swl3_dist2] => (item={'name': 'vlan 20', 'address': '10.0.0.254', 'vrrp_id': '20', 'stp_prio': 32})
changed: [swl3_dist1] => (item={'name': 'vlan 20', 'address': '10.0.0.254', 'vrrp_id': '20', 'stp_prio': 32})

```

Figura 7.8: Ejecución del playbook con el rol "vlan" en Ansible 1

```

TASK [vlan : No shutdown] *****
changed: [swl2_corporate] => (item={'name': 'vlan 99', 'address': '10.0.10.254', 'vrrp_id': '10', 'stp_prio': 32})
[WARNING]: To ensure idempotency and correct diff the input present in the running configuration on device
changed: [swl2_clients] => (item={'name': 'vlan 99', 'address': '10.0.10.254', 'vrrp_id': '10', 'stp_prio': 32})
changed: [swl3_dist2] => (item={'name': 'vlan 99', 'address': '10.0.20.254', 'vrrp_id': '20', 'stp_prio': 32})
changed: [swl2_server1] => (item={'name': 'vlan 99', 'address': '10.0.30.254', 'vrrp_id': '30', 'stp_prio': 32})
changed: [swl3_dist1] => (item={'name': 'vlan 99', 'address': '10.0.10.254', 'vrrp_id': '10', 'stp_prio': 32})
changed: [swl3_dist2] => (item={'name': 'vlan 10', 'address': '10.0.10.254', 'vrrp_id': '10', 'stp_prio': 0})
changed: [swl3_dist1] => (item={'name': 'vlan 10', 'address': '10.0.10.254', 'vrrp_id': '10', 'stp_prio': 0})
changed: [swl3_dist2] => (item={'name': 'vlan 20', 'address': '10.0.20.254', 'vrrp_id': '20', 'stp_prio': 32})
changed: [swl3_dist1] => (item={'name': 'vlan 20', 'address': '10.0.20.254', 'vrrp_id': '20', 'stp_prio': 0})
changed: [swl3_dist2] => (item={'name': 'vlan 30', 'address': '10.0.30.254', 'vrrp_id': '30', 'stp_prio': 32})
changed: [swl3_dist1] => (item={'name': 'vlan 30', 'address': '10.0.30.254', 'vrrp_id': '30', 'stp_prio': 0})

PLAY RECAP *****
swl2_clients : ok=3    changed=2    unreachable=0    failed=0
swl2_corporate : ok=3    changed=2    unreachable=0    failed=0
swl2_server1 : ok=3    changed=2    unreachable=0    failed=0
swl3_dist1 : ok=3    changed=2    unreachable=0    failed=0
swl3_dist2 : ok=3    changed=2    unreachable=0    failed=0

```

Figura 7.9: Ejecución del playbook con el rol "vlan" en Ansible 2

role/acl/...main.yml

```

1 - name: Enable inter-Vlan routing
2   cisco.ios.ios_config:
3     lines:
4       - ip routing
5
6
7 - name: Declare the ACLs
8   cisco.ios.ios_acls:
9     config:
10      - afi: "ipv4"
11        acls: "{{ acl_list.acls }}"
12      when: acl_list.acls is defined
13
14 - name: Apply the ACL inbound on the destination SVI
15   cisco.ios.ios_acl_interfaces:
16     config: "{{ acl_list.dest_vlan }}"

```

17 | when: acl_list.dest_vlan is defined

Ahora el rol se centra en las listas de control de acceso o [ACL](#). Lo primero es activar el enrutamiento en el *switch*, en nuestro caso para enrutar tráfico entre distintas [VLANs](#), y luego restringir quienes pueden acceder a las distintas zonas desde las [ACL](#). Estas se declaran a continuación mediante, de nuevo, el uso de un módulo preexistente de la colección y recorriendo la lista de [ACL](#) en las variables del *host*. Si no hay ninguna definida se omiten las tareas. Para finalizar se aplican las listas de control de acceso a las zonas correspondientes.

Podemos observar la ejecución de las tareas de este rol en la Figura 7.10.

```
PLAY [Setting Ip Routing and Acls] ****
TASK [acl : Enable inter-Vlan routing] ****
[WARNING]: To ensure idempotency and correct diff the input
present in the running configuration on device
changed: [swl3_dist2]
changed: [swl3_dist1]

TASK [acl : Declare the ACLs] ****
changed: [swl3_dist2]
changed: [swl3_dist1]

TASK [acl : Apply the ACL inbound on the destination SVI]
changed: [swl3_dist2]
changed: [swl3_dist1]

PLAY RECAP ****
swl3_dist1           : ok=3    changed=3    unreachable=0    errors=0
swl3_dist2           : ok=3    changed=3    unreachable=0    errors=0
```

Figura 7.10: Ejecución del playbook con el rol "acl" en Ansible 1

role/dist_switch/...main.yml

```
1 - name: Add physical members to Port-channel
2   ios_lag_interfaces:
3     config: "{{ port_channels }}"
4   vars:
5     ansible_command_timeout: 120
6   loop: "{{ port_channels }}"
7   when: port_channels is defined
8
9 - name: Configure VRRP on each SVI
10  ios_config:
11    lines: |
12      interface {{ item.name }}
```

```

13     vrrp {{ item.vrrp_id }} ip {{ item.vrrp_ip }}
14     vrrp {{ item.vrrp_id }} priority {{ item.vrrp_prio }}
15     vrrp {{ item.vrrp_id }} preempt
16     spanning-tree port-priority {{ item.stp_prio }}
17     loop: "{{ svls }}"
18     when: item.vrrp_prio is defined
19
20 - name: Convert uplink to a routed port (no switchport +
21   description)
21   cisco.ios.ios_interfaces:
22     config:
23       - name: "{{ uplink.interface }}"
24         mode: layer3
25         state: merged
26     when: uplink is defined
27
28 - name: Configure P2P uplink L3 to core router
29   cisco.ios.ios_13_interfaces:
30     config:
31       - name: "{{ uplink.interface }}"
32         ipv4:
33           - address: "{{ uplink.ip }}/{{ uplink.mask }}"
34     when: uplink is defined
35
36 - name: Spin up OSPFv2 process
37   cisco.ios.ios_ospfv2:
38     config:
39       processes:
40         - process_id: "{{ item.id }}"
41           network: "{{ item.networks }}"
42           passive_interfaces: "{{ item.passive_interfaces }}"
43     loop: "{{ ospf }}"
44     when: ospf is defined

```

El último de los roles de nuestro proceso de automatización con Ansible está dirigido al grupo de *switches* de distribución. En este se añaden puertos a un grupo *PortChannel*[\[31\]](#)[\[16\]](#), se configura el protocolo de redundancia [VRRP](#)[\[33\]](#)[\[16\]](#) haciendo uso de líneas de comando en texto plano, ya que no existe un modulo en la colección capaz de realizar esta tarea completa. Se definen los puertos hacia un *router* (u otros *switches*) con *links* de capa 3 y sus configuraciones adicionales y por ultimo se establece el protocolo [OSPF](#) para estos *links* creados en caso de que, como siempre, esté declarado en las variables de *host*.

Podemos observar fragmentos de la ejecución de las tareas de este rol en las Figuras [7.11](#) y [7.12](#).

```

PLAY [Distibution Switch Configuration] *****

TASK [dist_switch : Add physical members to Port-channel] *
changed: [swl3_dist1] => (item={'name': 'Port-channel 1', '|',
', 'mode': 'active'}])
changed: [swl3_dist2] => (item={'name': 'Port-channel 1', '|',
', 'mode': 'active'}))

TASK [dist_switch : Configure VRRP on each SVI] ****
skipping: [swl3_dist1] => (item={'name': 'vlan 99', 'address': '0.0.10.254', 'vrrp_id': '10', 'stp_prio': 0})
skipping: [swl3_dist2] => (item={'name': 'vlan 99', 'address': '0.0.10.254', 'vrrp_id': '10', 'stp_prio': 32})
changed: [swl3_dist1] => (item={'name': 'vlan 10', 'address': '0.0.10.254', 'vrrp_id': '10', 'stp_prio': 0})
changed: [swl3_dist2] => (item={'name': 'vlan 10', 'address': '0.0.10.254', 'vrrp_id': '10', 'stp_prio': 32})
changed: [swl3_dist1] => (item={'name': 'vlan 20', 'address': '0.0.20.254', 'vrrp_id': '20', 'stp_prio': 32})
changed: [swl3_dist2] => (item={'name': 'vlan 20', 'address': '0.0.20.254', 'vrrp_id': '20', 'stp_prio': 0})
changed: [swl3_dist1] => (item={'name': 'vlan 30', 'address': '0.0.30.254', 'vrrp_id': '30', 'stp_prio': 0})

```

Figura 7.11: Ejecución del playbook con el rol "dist_switch" en Ansible 1

```

TASK [dist_switch : Convert uplink to a routed port (no switchport + description)] *
changed: [swl3_dist1]
changed: [swl3_dist2]

TASK [dist_switch : Configure P2P uplink L3 to core router] ****
changed: [swl3_dist1]
changed: [swl3_dist2]

TASK [dist_switch : Spin up OSPFv2 process] ****
changed: [swl3_dist1] => (item={'id': '1', 'networks': [{"address": '10.0.0.0', 'interface': 'g0/1', 'set_interface': 'ospf1', 'status': 'up'}, {"address": '10.0.0.1', 'interface': 'g0/1', 'set_interface': 'ospf1', 'status': 'up'}], 'passive_interfaces': {'default': True, 'interface': 'g0/1', 'set_interface': 'ospf1', 'status': 'down'})
changed: [swl3_dist2] => (item={'id': '1', 'networks': [{"address": '10.0.0.0', 'interface': 'g0/1', 'set_interface': 'ospf1', 'status': 'up'}, {"address": '10.0.0.1', 'interface': 'g0/1', 'set_interface': 'ospf1', 'status': 'up'}], 'passive_interfaces': {'default': True, 'interface': 'g0/1', 'set_interface': 'ospf1', 'status': 'down'}}
```

Figura 7.12: Ejecución del playbook con el rol "dist_switch" en Ansible 2

PLANTILLA VARIABLES HOST

```

1 # Switch Variables
2 ansible_host: 10.0.99.5
3
4 # Interfaces
5 unused_ports: "gi1/2-3,gi2/0-3,gi3/0-2"
6 management_port: "gi3/3"
7

```

```

8 # Vlans
9 vlans:
10 - {id: 10, name: Servers}
11 - {id: 20, name: Corporate}
12 - {id: 30, name: Clients}
13 - {id: 99, name: Management}
14
15 # SVIs (y VRRP con vrrp_prio + vrrp_ip + vrrp_id)
16 svsis:
17 - {name: "vlan 99", address: "10.0.99.5", mask: "24"}
18 - {name: "vlan 10", address: "10.0.10.253", mask: "24",
19   vrrp_prio: "90", vrrp_ip: "10.0.10.254", vrrp_id: "10",
20   stp_prio: 32}
21 - {name: "vlan 20", address: "10.0.20.253", mask: "24",
22   vrrp_prio: "90", vrrp_ip: "10.0.20.254", vrrp_id: "20",
23   stp_prio: 32}
24 - {name: "vlan 30", address: "10.0.30.253", mask: "24",
25   vrrp_prio: "90", vrrp_ip: "10.0.30.254", vrrp_id: "30",
26   stp_prio: 32}
27
28 acl_list:
29   accls:
30     - name: "FROM-10"
31       acl_type: extended
32       aces:
33         - {sequence: 5, grant: deny, protocol: ip, source:
34           {address: "10.0.10.0", wildcard_bits: "0.0.0.255"}, destination:
35           {address: "10.0.30.0", wildcard_bits: "0.0.0.255"}}
36         - {sequence: 10, grant: permit, protocol: ip, source: {any:
37             true}, destination: {any: true}}
38     - name: "FROM-20"
39       acl_type: extended
40       aces:
41         - {sequence: 5, grant: deny, protocol: ip, source:
42           {address: "10.0.20.0", wildcard_bits: "0.0.0.255"}, destination:
43           {address: "10.0.30.0", wildcard_bits: "0.0.0.255"}}
44         - {sequence: 15, grant: permit, protocol: ip, source: {any:
45             true}, destination: {any: true}}
46     - name: "FROM-30"
47       acl_type: extended
48       aces:
49         - {sequence: 5, grant: deny, protocol: ip, source:
50           {address: "10.0.30.0", wildcard_bits: "0.0.0.255"}, destination:
51           {address: "10.0.10.0", wildcard_bits: "0.0.0.255"}}
52         - {sequence: 10, grant: deny, protocol: ip, source:
53           {address: "10.0.30.0", wildcard_bits: "0.0.0.255"}, destination:
54           {address: "10.0.10.0", wildcard_bits: "0.0.0.255"}}
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
826
827
828
828
829
829
830
831
832
833
834
835
835
836
837
837
838
838
839
839
840
841
842
843
844
844
845
846
846
847
847
848
848
849
849
850
851
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
157
```

```

39      {address: "10.0.20.0", wildcard_bits: "0.0.0.255"}}
40          - {sequence: 15, grant: permit, protocol: ip, source: {any:
41              true}, destination: {any: true}}
42          - name: "FROM-99"
43              acl_type: extended
44              aces:
45                  - {sequence: 5, grant: permit, protocol: ip, source:
46                      {address: "10.0.99.0", wildcard_bits: "0.0.0.255"}, destination:
47                      {address: "10.0.99.0", wildcard_bits: "0.0.0.255"}}
48                  - {sequence: 10, grant: deny, protocol: ip, source: {any:
49                      true}, destination: {any: true}}
50
51 dest_vlan:
52     - name: "vlan 10"
53         access_groups: [{acls: [{name: "FROM-10", direction: "in"}], afi: "ipv4"}]
54     - name: "vlan 20"
55         access_groups: [{acls: [{name: "FROM-20", direction: "in"}], afi: "ipv4"}]
56     - name: "vlan 30"
57         access_groups: [{acls: [{name: "FROM-30", direction: "in"}], afi: "ipv4"}]
58     - name: "vlan 99"
59         access_groups: [{acls: [{name: "FROM-99", direction: "in"}], afi: "ipv4"}]
60
61 # Access ports
62 access_ports:
63     - {interface: "g3/3", mode: access, vlan: 99}
64
65 # Trunk ports (Añadir portchannel si necesario)
66 trunk_ports:
67     - {interface: "gi0/0", mode: trunk, native_vlan: 999,
68         allowed_vlans: "10,20,99", encapsulation: "dot1q"}
69     - {interface: "gi0/2", mode: trunk, native_vlan: 999,
70         allowed_vlans: "10,20,99", encapsulation: "dot1q"}
71     - {interface: "gi0/3", mode: trunk, native_vlan: 999,
72         allowed_vlans: "30,99", encapsulation: "dot1q"}
73     - {interface: "{{ port_channels[0].name }}", mode: trunk,
74         native_vlan: 999, allowed_vlans: "10,20,30,99", encapsulation:
75             "dot1q"}
76
77 # PortChannel
78 port_channels:
79     - name: "Port-channel 1"

```

```

71    members:
72      - {member: "g1/0", mode: "active"}
73      - {member: "g1/1", mode: "active"}
74
75 # Router link
76 uplink: {interface: "g0/1", ip: "10.0.0.3", mask: "29"}
77
78 # OSPF (añade a networks las redes a compartir con ospf, pej:
79   vlans...)
80 ospf:
81   - id: "1"
82     networks:
83       - {address: "10.0.0.0", wildcard_bits: "0.0.0.7", area: "1"}
84       - {address: "10.0.10.0", wildcard_bits: "0.0.0.255", area:
85         "1"}
86       - {address: "10.0.20.0", wildcard_bits: "0.0.0.255", area:
87         "1"}
88       - {address: "10.0.30.0", wildcard_bits: "0.0.0.255", area:
89         "1"}
90
91     passive_interfaces: {default: true, interface: {name: "{{"
92       uplink.interface }}", set_interface: false}}

```

7.2 Aprovisionamiento del servidor en la nube

terraform_compute

```

1 resource "oci_core_instance" "ubuntu_instance" {
2   # Required
3   availability_domain =
4   data.oci_identity_availability_domains.ads.availability_domains[0].name
5   compartment_id =
6   "ocid1.compartment.oc1..aaaaaaaa2tb3oxhzdvg33syh2eqodtqxyeen-
7   4dhhfh3xjh1torpfipglbeq"
8   shape = "VM.Standard.A1.Flex"
9   shape_config {
10     ocpus = "1"
11     memory_in_gbs = "12"
12   }
13
14   source_details {
15     source_id =

```

"ocid1.image.oc1.eu-madrid-1.aaaaaaaa6zbjjysuxmjf6of32fg3rdzdsas-
6w3skwdy3ub5jhm5qststakvbg"

```

16         source_type = "image"
17     }
18
19     # Optional
20     display_name = "ubuntu-zerotier"
21     create_vnic_details {
22         assign_public_ip = true
23         subnet_id =
24         "ocid1.subnet.oc1.eu-madrid-1.aaaaaaaaawf3ecocwygkhepgnu7u4e6-
25             yyxxo477zbi3miahtxxvz2u4rk7za"
26     }
27     metadata = {
28         ssh_authorized_keys = file("../ssh/ssh-instance.pub")
29     }
30     preserve_boot_volume = false
31
32     provisioner "remote-exec" {
33         inline = ["echo 'Wait until SSH is ready'"]
34         connection {
35             type = "ssh"
36             user = "ubuntu" # or opc for Oracle Linux
37             private_key = file("../ssh/ssh-instance")
38             host = oci_core_instance.ubuntu_instance.public_ip
39         }
40     }

```

Este es el ultimo fichero personalizado que hay que realizar tras seguir la guía en la documentación inicial de integración de Oracle Cloud[17][40] con Terraform[4]. Aquí se establece el tipo de estancia, sus capacidades, información obligatoria relacionada con el usuario, el nombre de la instancia, la posibilidad de tener IP pública y se añade al archivo de *authorized keys* del servidor nuestra llave pública, lo que nos permite conectarnos al servidor usando la correspondiente llave privada. Por último se espera a que el servicio de **SSH** esté funcionando y accesible, ya que a continuación vamos a utilizar Ansible para instalar y configurar ciertos servicios y la conexión **SSH** es primordial para realizarlo.

ansible_zerotier_mosquitto

```

1 ---
2 - name: Install and configure ZeroTier
3   hosts: zerotier
4   become: true
5   vars:
6     zerotier_network_id: "885033839019e1a8"
7     mosquitto_user: admin
8     mosquitto_pass: mosquitto

```

```

9     password_file: /etc/mosquitto/passwd
10    mosquitto_conf_file: /etc/mosquitto/mosquitto.conf
11
12  tasks:
13    - name: Instalar prerequisitos
14      ansible.builtin.apt:
15        name:
16          - curl
17          - gnupg
18          - ca-certificates
19          - lsb-release
20        state: present
21        update_cache: yes
22
23    - name: Descargar y de-armor ZeroTier GPG
24      ansible.builtin.shell: |
25        set -e
26        curl -fsSL
27        https://download.zerotier.com/contact%40zerotier.com.gpg \
28        | gpg --dearmor --yes \
29        > /usr/share/keyrings/zerotier-archive-keyring.gpg
30      args:
31        creates: /usr/share/keyrings/zerotier-archive-keyring.gpg
32
33    - name: Añadir ZeroTier repositorio apt
34      ansible.builtin.apt_repository:
35        repo: >
36          deb
37          [signed-by=/usr/share/keyrings/zerotier-archive-keyring.gpg]
38          https://download.zerotier.com/debian/{{{
39            ansible_lsb.codename }}}
40            {{ ansible_lsb.codename }} main
41            filename: zerotier
42            state: present
43            update_cache: yes
44
45    - name: Instalar zerotier
46      ansible.builtin.apt:
47        name: zerotier-one
48        state: present
49
50    - name: Habilitar y lanzar el servicio
51      ansible.builtin.systemd:
52        name: zerotier-one
53        enabled: true
54        state: started

```

```

52
53     - name: Unirse a la red ZeroTier
54         command: zerotier-cli join {{ zerotier_network_id }}
55         register: join_result
56         changed_when: "'200 join OK' in join_result.stdout or
57 'already a member' in join_result.stdout"
58         failed_when: "'ERROR' in join_result.stdout"
59
60     - name: Resultado
61         debug:
62             var: join_result.stdout
63
64     - name: Instalar Mosquitto
65         apt:
66             name:
67                 - mosquitto
68             state: present
69             update_cache: yes
70
71     - name: Crear fichero passwd
72         file:
73             path: "{{password_file}}"
74             state: touch
75             owner: mosquitto
76             group: mosquitto
77             mode: '0640'
78
79     - name: Modificar archivo de contraseña con usuario y contraseña
80         ansible.builtin.command: >
81             mosquitto_passwd -b {{ password_file }} {{ mosquitto_user }}
82             {{ mosquitto_pass }}
83
84     - name: Añadir autenticación y logs al fichero mosquitto.conf
85         blockinfile:
86             path: "{{ mosquitto_conf_file }}"
87             block: |
88                 allow_anonymous false
89                 password_file {{ password_file }}
90                 listener 1883
91                 log_type error
92                 log_type warning
93
94     - name: Resesetar y habilitar Mosquitto
95         systemd:
96             name: mosquitto
97             state: restarted

```

```

96      enabled: yes
97
98  - name: Allow MQTT (1883/TCP) before default REJECT
99    ansible.builtin.iptables:
100      chain: INPUT
101      protocol: tcp
102      destination_port: 1883
103      jump: ACCEPT
104      action: insert      # position 5 (classic syntax)
105      comment: "MQTT broker"
106      rule_num: 5
107      state: present
108      become: yes

```

Las tareas de Ansible actualizan e instalan los prerequisitos necesarios y siguen parte de la documentación inicial para instalar la [VPN Zerotier](#)[9] y unirnos a nuestra preexistente red en una cuenta de Zerotier. Por último, instala el servicio del bróker [MQTT](#)[6], asigna los privilegios necesarios al archivo de contraseñas y crea un usuario y contraseña para proteger el acceso desconocido a nuestro Bróker. Los últimos ajustes modifican la configuración de Mosquitto[18] para no permitir accesos anónimos y emplear el archivo de contraseña, reinicia el servicio y añade una regla de *firewall* usando *iptables*[41] que permite las conexiones al puerto del servicio de Mosquitto desde fuera del servidor.

deploy.sh

```

1 #!/bin/bash
2
3 set -e
4
5 echo "Applying Terraform..."
6 cd tf-compute
7 terraform init
8 terraform apply -auto-approve
9
10 echo "Getting instance IP..."
11 INSTANCE_IP=$(terraform output -raw instance-public-ip)
12 cd ../ansible
13
14 echo "Generating Ansible inventory..."
15 echo "[zerotier]" > hosts.ini
16 echo "$INSTANCE_IP ansible_user=ubuntu
17     ansible_ssh_private_key_file=../ssh/ssh-instance" >> hosts.ini
18
19 echo "Running Ansible playbook..."
20 ansible-playbook -i hosts.ini zerotier-playbook.yml

```

Para integrar el aprovisionamiento del servidor en la nube con Terraform y su configuración posterior con Ansible se hace uso de un pequeño *script* en *bash*. El *script* inicia y ejecuta al archivo de Terraform, guarda la IP pública del servidor mediante un comando de Terraform y la utiliza a continuación para configurar el *host* y las variables en Ansible junto con la llave privada para que sea posible conectarse al servidor. El *script* termina con la ejecución del *playbook* de Ansible referenciando su archivo de *host* correspondiente.

Podemos ver los resultados de la ejecución del *script* en *bash* en las Figuras 7.13 y 7.14.

```
Commands will detect it and remind you to do so if necessary.
data.oci_identity_availability_domains.ads: Reading...
data.oci_identity_availability_domains.ads: Read complete after 1s [id=IdentityAvailabilityDomainsDa
Terraform used the selected providers to generate the following execution plan. Resource actions are
following symbols:
+ create

Terraform will perform the following actions:

# oci_core_instance.ubuntu_instance will be created
+ resource "oci_core_instance" "ubuntu_instance" {
    + availability_domain          = "ryQk:EU-MADRID-1-AD-1"
    + boot_volume_id                = (known after apply)
    + capacity_reservation_id       = (known after apply)
    + compartment_id                = "ocid1.compartment.oc1..aaaaaaaa2tb3oxhzdvg33syh2eqod"
    + compute_cluster_id            = (known after apply)
    + dedicated_vm_host_id          = (known after apply)
    + defined_tags                  = (known after apply)
    + display_name                  = "ubuntu-zerotier"
    + extended_metadata              = (known after apply)
    + fault_domain                  = (known after apply)
    + freeform_tags                 = (known after apply)
    + hostname_label                = (known after apply)
    + id                            = (known after apply)
    + image                          = (known after apply)
    + instance_configuration_id     = (known after apply)
    + ipxe_script                   = (known after apply)
    + is_cross numa_node             = (known after apply)
    + is_pv_encryption_in_transit_enabled = (known after apply)
    + launch_mode                    = (known after apply)
    + metadata                       = {
        + "ssh_authorized_keys" = <<-EOT
          ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDAssMiHGHc22KkECPRBc7N7P4/7m3FCA3K4zoEuna/R5To
          N1DfL5Ij5A...[REDACTED]...KvGPTUllAGCHUHhNLoNccExHuvLTCG
        -EOT
      }
  }
```

Figura 7.13: Ejecución de Terraform-Ansible para el servidor en la nube 1

```

TASK [Instalar prerequisitos] *****
ok: [143.47.58.30]

TASK [Descargar y de-armor ZeroTier GPG] *****
changed: [143.47.58.30]

TASK [Añadir ZeroTier repositorio apt] *****
changed: [143.47.58.30]

TASK [Instalar zerotier] *****
changed: [143.47.58.30]

TASK [Habilitar y lanzar el servicio] *****
ok: [143.47.58.30]

TASK [Unirse a la red ZeroTier] *****
changed: [143.47.58.30]

TASK [Resultado] *****
ok: [143.47.58.30] => {
    "join_result.stdout": "200 join OK"
}

TASK [Instalar Mosquitto] *****
changed: [143.47.58.30]

TASK [Crear fichero passwd] *****
changed: [143.47.58.30]

TASK [Modificar archivo de contraseña con usuario y contraseña] *****
changed: [143.47.58.30]

TASK [Añadir autenticación y logs al fichero mosquitto.conf] *****
changed: [143.47.58.30]

TASK [Resesetar y habilitar Mosquitto] *****
changed: [143.47.58.30]

TASK [Allow MQTT (1883/TCP) before default REJECT] *****
changed: [143.47.58.30]

PLAY RECAP *****
143.47.58.30 : ok=14    changed=10    unreachable=0    failed=0

```

Figura 7.14: Ejecución de Terraform-Ansible para el servidor en la nube 2

undeploy.sh

```

1 #!/bin/bash
2
3 set -e
4
5 echo "Applying Terraform..."
6 cd tf-compute
7 terraform destroy -auto-approve

```

Esta es una manera sencilla de eliminar la instancia del servidor en la nube creada con Terraform.

7.3 Pasos necesarios en la red industrial

modbus_sim.py

```

1 #!/usr/bin/env python3
2 from pymodbus.server import StartTcpServer
3 from pymodbus.datastore import ModbusSequentialDataBlock,
4     ModbusServerContext
5 import psutil, time, threading
6
7 class CpuDataBlock(ModbusSequentialDataBlock):
8     async def async_getValues(self, fx, address, count=1):
9         return super().getValues(address, count)
10
11    async def async_setValues(self, fx, address, values):
12        super().setValues(address, values)
13
14 block = CpuDataBlock(0, [0]*10)           # HR0-9
15 context = ModbusServerContext(slaves={1: block}, single=False)
16
17 def update_regs():
18     while True:
19         cpu = int(psutil.cpu_percent()*10)
20         frequency = int(psutil.cpu_freq().current*10)
21         ram = psutil.virtual_memory()
22         ram_percent =
23             int((ram.total-ram.available)/ram.total*100*10)
24         disk = int(psutil.disk_usage('C:\\\\').percent*10)
25         values = [cpu, frequency, ram_percent, disk]
26         print(f"cpu_usage = {cpu/10}   cpu_freq = {frequency/10}
27             ram_usage = {ram_percent/10}   disk_usage = {disk/10}")
28
29         block.setValues(0, values)
30         time.sleep(2)
31
32 threading.Thread(target=update_regs, daemon=True).start()
33
34 #slave id 1
35 StartTcpServer(context, address=("0.0.0.0", 1502))
```

Para la generación de datos, a falta de un sensor o semejantes se ha decidido emplear un portátil como dispositivo IoT que ejecute el programa que se observa arriba en Python. Este

programa, a grandes rasgos, obtiene métricas del procesador y las comparte como registros simulando ser un servidor Modbus[5]. Se escogió ésta solución porque es lo más parecido a las tecnologías que utilizan los PLC y otros dispositivos en los entornos industriales en la realidad.

7.4 Configuración Teltonika RUT901

The screenshot shows two main sections of the Teltonika ZeroTier configuration interface:

- "TFG_RUT" instance configuration:**
 - Enable: A toggle switch set to "on".
 - Node ID: 7793c678fa
- Network configuration:**
 - A table with columns: Network name, Network ID, Port, Enabled, and Actions.
 - One row is listed: TFG_NETWORK, 885033839019e1a8, 9993, Enabled (toggle switch set to "on"), and Actions (Edit, Delete).

Figura 7.15: Teltonika ZeroTier

En la Pestaña ”Services - VPN - Zerotier” (después de instalar el *plugin* Zerotier) añadimos el id de la red, como se indica en la Figura 7.15, y autorizamos al *router* externamente en ZeroTier.

Server ID * 1

Address * 192.168.2.196

Port * 1502

Timeout 1

Always reconnect off

Number of timeouts * 0

Frequency Period

Delay 0

Period * 1

Figura 7.16: Teltonika Modbus Client TCP

En la pestaña "Services - Modbus - Modbus TCP Client" se añade la dirección del portátil de la red local que ejecuta el *script* en Python de generación de datos, el cual utiliza el puerto por defecto 1502 para el servidor Modbus en este caso. Luego se puede regular la velocidad de refresco en la recogida de datos si se desea, nosotros lo hemos establecido a solo un segundo. Todas estas configuraciones las observamos en la Figura 7.16

| Name | Data type | Function | First register number | Register count / Values | Broadcast | Storage on channel | Enable | Actions |
|------|------------|-----------|-----------------------|-------------------------|---|---|--|---------|
| p. | 16bit I... | Read h... | 1 | 4 | <input checked="" type="checkbox"/> off | <input checked="" type="checkbox"/> off | <input checked="" type="checkbox"/> on | Delete |

Search...
Read holding registers (3)

Figura 7.17: Teltonika Modbus Client Request

En el apartado "Request" indicamos los registros de datos que se quieren obtener, en nuestro caso son cuatro valores, con tipo 16 bits *Integer* y la función 3 de lectura de registros. Podemos ver la petición en la interfaz web del *router* en la Figura 7.17.

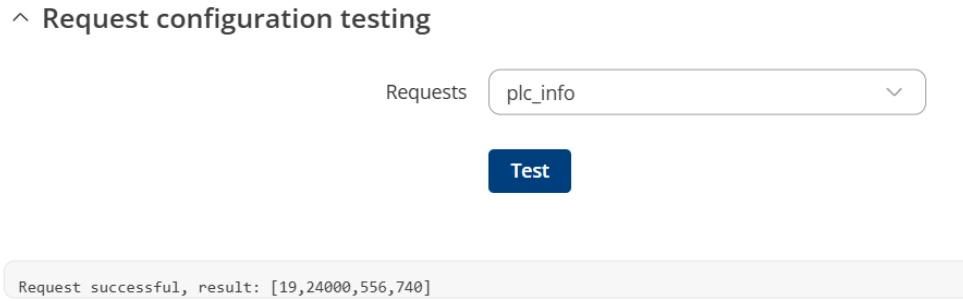


Figura 7.18: Teltonika Modbus Request Test

Podemos probar que funciona la recogida con el botón de "Test" al final de la página tal y como se muestra en la Figura 7.18.

The screenshot shows the "Data to Server" configuration section. It includes fields for "Type" (set to "MQTT"), "Server address" (172.30.1.100), "Port" (1883), "Keepalive" (60), "Topic" (router1), "Client ID" (RUT901), and "QoS" (0). There are also toggle switches for "Enable secure connection" (off) and "Use credentials" (on), along with input fields for "Username" (admin) and "Password" (redacted).

Figura 7.19: Teltonika Data to Server MQTT

En la pestaña "Data to Server" se añade la información del servidor con el bróker MQTT (dirección, puerto, usuario y contraseña), el tópico al que suscribirse para escribir datos y opcionalmente un ID y un nivel de QoS. Podemos observar la configuración del servidor en la Figura 7.19

^ "cpu_data" data configuration

The screenshot shows the configuration interface for a 'cpu_data' data configuration. The fields are as follows:

- Name: cpu_data
- Type: Modbus
- Data filtering: Request name
- Request name: plc_info
- Segment count: 1
- Send as object: off
- Format type: Custom
- Format string: {"date": "%date%", "name": "%name%", "data": %data%, "ip": "%ip%"} (highlighted in red)

Figura 7.20: Teltonika Dato to Server MQTT 2

Luego se añade el tipo de dato Modbus y se selecciona el nombre con el que se hayan guardado los registros que se reciben del servidor. A mayores se pueden especificar filtros para los datos y un formato específico de cadena para enviar, aunque JSON es el formato por defecto esto nos da mucha mayor libertad. La Figura 7.20 nos muestra la configuración final de los datos que se van a enviar.

7.5 Configuración del servidor local

mqtt_to_influx.py

```

1  #!/usr/bin/env python3
2  import json, paho.mqtt.client as mqtt
3  from influxdb_client import InfluxDBClient, Point, WritePrecision
4  from influxdb_client.client.write_api import SYNCHRONOUS
5
6
7  INFLUX_URL  = "http://localhost:8086"
8  TOKEN        = "#####"
9  ORG          = "TFG_EMPRESA"
10 BUCKET       = "data"
11
12 client = InfluxDBClient(url=INFLUX_URL, token=TOKEN, org=ORG)
13 write  = client.write_api(write_options=SYNCHRONOUS)
14

```

```

15| def on_connect(client, userdata, flags, reason_code, properties):
16|     print(f"Connected with result code {reason_code}")
17|     client.subscribe("router1")
18|     client.subscribe("home/temperature")
19|
20| def on_message(_, __, msg):
21|     payload = msg.payload.decode()
22|     print(msg.topic)
23|     if msg.topic == "router1":
24|
25|         json_data = json.loads(payload)
26|         data_array = json_data['cpu_data']['data']
27|         data = [float(x)/10 for x in data_array]
28|
29|         p = (
30|             Point("cpu_info")
31|                 .tag("ip", json_data['cpu_data']['ip'])
32|                 .field("cpu_usage", data[0])
33|                 .field("cpu_freq", data[1])
34|                 .field("ram_usage", data[2])
35|                 .field("disk_usage", data[3])
36|         )
37|         print(p)
38|         write.write(bucket=BUCKET, record=p,
39|         write_precision=WritePrecision.S)
40|
41| m = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
42| m.username_pw_set(username="admin", password="mosquitto")
43| m.on_connect = on_connect
44| m.on_message = on_message
45| m.connect("172.30.1.100", 1883, 60)
46| m.subscribe("router1")
47| m.loop_forever()

```

Se utiliza nuevamente otro programa en Python, aunque se podrían utilizar herramientas orientadas a estas situaciones como Telegraf[42], hemos decidido volver a utilizar Python porque además de ayudarnos a explorar todas sus capacidades, es mucho más personalizable que cualquier otra herramienta vista en el mercado. Aquí lo que hace, utilizando las librerías paho.mqtt[43] y influxdb_client[44] es suscribirse a nuestro bróker MQTT con usuario y contraseña, busca el tópico "router1" y obtiene los datos de ese mismo instante en JSON, los cuales utiliza para formar un paquete de datos compatible con la base de datos InfluxDB[19] de la cuál necesita dirección, organización, *token api* generado y por último el *bucket* donde se guardan los datos.

EL programa se podría correr como un servicio en Linux añadiendo un archivo .service al

directorio /etc/systemd/system/ con el siguiente contenido:

```

1 [Unit]
2 Description=MQTT topics to local influxdb
3 After=network-online.target
4 Wants=network-online.target
5
6 [Service]
7 Type=simple
8 User=debian
9 ExecStart=/usr/bin/env/bin/python3 /usr/bin/mqtt_to_db.py
10 Restart=on-failure
11 RestartSec=5
12 Environment=PYTHONUNBUFFERED=1
13
14 [Install]
15 WantedBy=multi-user.target

```

Reemplazamos las rutas de Python y el *script* por las correspondientes en cada caso. Se ejecuta `sudo systemctl daemon-reload` para recargar los servicios y podemos luego habilitar, correr y ver el estatus del servicio con `sudo systemctl (start/enable/status) file.service`. Si el *script*, el servidor MQTT y la generación de los datos son correctos, el estatus del servicio nos debería mostrar algo similar a la Figura 7.21.

```

debian@debian:~$ sudo systemctl status mqtt
● mqtt.service - MQTT topics to local influxdb
    Loaded: loaded (/etc/systemd/system/mqtt.service; enabled; preset: enabled)
    Active: active (running) since Tue 2025-06-24 19:14:02 UTC; 16s ago
      Main PID: 4297 (python3)
         Tasks: 1 (limit: 1743)
        Memory: 29.0M
          CPU: 267ms
        CGroup: /system.slice/mqtt.service
                └─4297 /usr/bin/env/bin/python3 /usr/bin/mqtt_to_db.py

Jun 24 19:14:13 debian python3[4297]: router1
Jun 24 19:14:13 debian python3[4297]: cpu_info, ip=172.30.1.10 cpu_freq=2496,cpu
Jun 24 19:14:14 debian python3[4297]: router1
Jun 24 19:14:14 debian python3[4297]: cpu_info, ip=172.30.1.10 cpu_freq=2496,cpu
Jun 24 19:14:15 debian python3[4297]: router1
Jun 24 19:14:15 debian python3[4297]: cpu_info, ip=172.30.1.10 cpu_freq=2496,cpu
Jun 24 19:14:16 debian python3[4297]: router1
Jun 24 19:14:16 debian python3[4297]: cpu_info, ip=172.30.1.10 cpu_freq=2496,cpu
Jun 24 19:14:17 debian python3[4297]: router1
Jun 24 19:14:17 debian python3[4297]: cpu_info, ip=172.30.1.10 cpu_freq=2496,cpu
lines 1-20/20 (END)

```

Figura 7.21: Script Python como servicio Linux

```

1 sudo apt update && sudo apt upgrade
2 #seguir
https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/
for grafana

```

```

3  sudo systemctl daemon-reload
4  sudo systemctl start grafana-server
5  sudo systemctl enable grafana-server
6  #seguir https://docs.influxdata.com/influxdb/v2/install/?t=Linux
   o mejor
   https://medium.com/yavar/install-and-setup-influxdb-on-ubuntu-20-04-22-04-3d6e090e
7  sudo systemctl start influxdb
8  sudo systemctl enable influxdb
9  sudo apt install python3.11-venv, pip
10 sudo python -m venv /opt/venv #(instalar dependencias)

```

Instalación de Grafana[20], InfluxDB[44] y Python para servidor en Linux. En Windows se puede descargar y ejecutar el instalador que nos ofrecen directamente y el proceso es mucho más rápido y sencillo.

7.6 Configuración de OPNsense

Primero configuraremos la dirección IPv4 de la red local, que se puede hacer tanto desde el terminal como la interfaz web de OPNsense. Ésta última la observamos en la Figura 7.22.

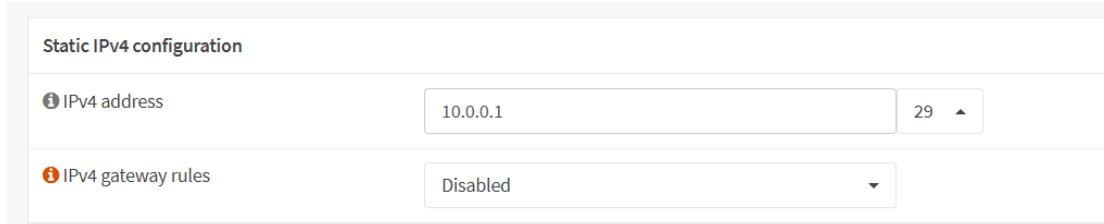


Figura 7.22: OPNsense configuración de interfaz LAN IPv4

Luego creamos una interfaz *bridge* a la que asignaremos los puertos opcionales como en la Figura 7.23, para que todos formen parte de la misma red local

| Interfaces: Other Types: Bridge | |
|---------------------------------|------------------|
| Interface | Members |
| bridge0 | OPT1, OPT2, OPT3 |

Figura 7.23: OPNsense creación de un bridge LAN

Podemos también crear en este momento la conexión a la [VPN](#) Zerotier, después de ac-

tualizar OPNsense e instalar el *plugin* ZeroTier, para lo cual que solo tenemos que añadir el código de identificación de la red y ser autorizados externamente. Esta sencilla configuración la podemos apreciar en la Figura 7.24.



Figura 7.24: OPNsense configuración de la red VPN ZeroTier

Ahora en el apartado de interfaces, el cual podemos observar en la Figura 7.25, asignamos el *bridge* creado anteriormente a la [LAN](#) para que conforme los puertos de la red local del *router* y también asignamos a un nuevo puerto opcional (OPT4 en este caso) el dispositivo correspondiente a ZeroTier que se creó automáticamente.

| Interfaces: Assignments | | |
|-------------------------|------------|-------------------------------------|
| Interface | Identifier | Device |
| [LAN] | lan | bridge0 () |
| [OPT1] | opt1 | vtnet1 (0c:e3:4d:88:00:01) |
| [OPT2] | opt2 | vtnet0 (0c:e3:4d:88:00:00) |
| [OPT3] | opt3 | vtnet3 (0c:e3:4d:88:00:03) |
| [OPT4] | opt4 | zt8gk1jge81jod8 (aa:68:18:16:fe:2c) |
| [WAN] | wan | vtnet2 (0c:e3:4d:88:00:02) |

Figura 7.25: OPNsense configuración de las interfaces

Para facilitar la gestión y reglas de las redes e interfaces en el *router* vamos a crear distintos alias y grupos correspondientes a la red de ZeroTier, a las subredes de las [VLANs](#) y a las interfaces *bridge* y ZeroTier, como vemos en ambas Figuras 7.26 y 7.27

| Firewall: Aliases | | | | |
|--------------------------|-------------------------------------|----------------|------------|-------------------------------|
| Aliases | | GeoIP settings | | |
| <input type="checkbox"/> | Enabled | Name | Type | Description |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | ZT | Host(s) | ZeroTier Adress 172.30.1.3 |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | ZT_full_remote | Network(s) | 192.168.2.0/24 172.30.1.1/24 |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | ZT_USERS | Network(s) | 10.0.10.0/24 10.0.20.0/24 |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | ZT_remote | Network(s) | 172.30.1.0/24 |

Figura 7.26: OPNsense creación de alias para facilitar su gestión

| Firewall: Groups | | |
|--------------------------|---------------|------------------|
| <input type="checkbox"/> | Name | Members |
| <input type="checkbox"/> | ZerotierGroup | OPT4 |
| <input type="checkbox"/> | bridge | OPT1, OPT2, OPT3 |
| <input type="checkbox"/> | openvpn | |
| <input type="checkbox"/> | enc0 | |

Figura 7.27: OPNsense creación de grupos para facilitar su gestión

Ahora solo quedan un par de configuraciones finales. En este instante creamos las reglas de *firewall* en las interfaces correspondientes en las que pretendamos bloquear y/o permitir en nuestro caso el uso de la red [VPN](#) según la dirección IP de origen y destino. Estas reglas, que en nuestro caso bloquean el acceso a la [VPN](#) a direcciones que no pertenezcan a las subredes de las [VLANs](#) 10 y 20, permiten (después de aplicarse el bloqueo anterior) todo el tráfico a las de direcciones de la red 10.0.0.0/8 y por último en la interfaz de ZeroTier permitimos que los dispositivos de la red [VPN](#) se comuniquen con nuestro *router*. Estas reglas las apreciamos en las Figuras 7.28 y 7.29.

| Protocol | Source | Port | Destination | Port | Gateway |
|--------------------------|------------------|------|-------------------|------|---------|
| <input type="checkbox"/> | | | | | |
| IPv4 * | ! ZT_USERS | * | ZT | * | * |
| IPv4 * | ZT_USERS | * | ZT | * | * |
| IPv4 * | 10.0.0.0/8 | * | * | * | * |
| IPv6 * | LAN net | * | * | * | * |
| pass | block | | reject | | |
| pass (disabled) | block (disabled) | | reject (disabled) | | |

Figura 7.28: OPNsense reglas firewall de la interfaz LAN

| Protocol | Source | Port | Destination | Port | Gateway |
|--------------------------|------------------|------|-------------------|------|---------|
| <input type="checkbox"/> | | | | | |
| IPv4 * | 172.30.1.0/24 | * | OPT4 address | * | * |
| pass | block | | reject | | |
| pass (disabled) | block (disabled) | | reject (disabled) | | |

Figura 7.29: OPNsense reglas firewall de la interfaz de ZeroTier

Como último paso relacionado con la conectividad de la [VPN](#) vamos a crear dos reglas [NAT outbound](#) que permitan a ciertas redes o dispositivos acceder a al destino seleccionado utilizando la IP del *router*, es decir, los dispositivos del exterior piensan que se comunican únicamente con el *router* pero los paquetes se traducen en su llegada al *router* a la dirección del dispositivo de la red local que envío originalmente las peticiones. Podemos observar las reglas [NAT outbound](#) para la interfaz [WAN](#) y la interfaz de la [VPN](#) ZeroTier en la Figura 7.30.

Firewall: NAT: Outbound

Mode

- Automatic outbound NAT rule generation (no manual rules can be used)
- Hybrid outbound NAT rule generation (automatically generated rules are applied)
- Manual outbound NAT rule generation (no automatic rules are being generated)
- Disable outbound NAT rule generation (outbound NAT is disabled)

Save

Manual rules

| | Interface | Source | Source Port | Destination | Destination Port | NAT Address |
|--------------------------|---------------|----------|-------------|----------------|------------------|-------------------|
| <input type="checkbox"/> | ZerotierGroup | ZT_USERS | * | ZT_full_remote | * | Interface address |
| <input type="checkbox"/> | WAN | any | * | * | * | Interface address |

▶ Enabled rule

Figura 7.30: OPNsense configuración de la traducción NAT hacia el exterior

Por último vamos a configurar OSPF para adquirir y compartir las rutas con los *switches de distribución*. Para ello le asignamos la interfaz **LAN**, que si recordamos la conforman varios puertos a los que irán conectados los *switches*, y compartimos el segmento de la red en el que se encuentra nuestra **LAN** para que el resto de dispositivos aprendan la ruta hacia el *router* OPNsense. Ambas configuraciones se muestran en las Figuras 7.31 y 7.32

Routing: OSPF

General **Networks** **Interfaces** **Prefix Lists** **Route Maps**

| Enabled | Interface Name | Network Type | Authentication Type |
|-------------------------------------|----------------|--------------------------------|---------------------|
| <input checked="" type="checkbox"/> | LAN | Broadcast multi-access network | None |

Figura 7.31: Selección de la interfaz OSPF

Routing: OSPF

General **Networks** **Interfaces** **Prefix Lists** **Route Maps**

| Enabled | Network Address | Mask | Area |
|-------------------------------------|-----------------|------|---------|
| <input checked="" type="checkbox"/> | 10.0.0.0 | 29 | 0.0.0.1 |

Figura 7.32: Configuración de las redes a compartir con OSPF

7.7 Configuración de Zerotier

Zerotier es gratis hasta 10 dispositivos a través de su web, solo hay que registrarse o iniciar sesión con una cuenta de Google y se crea automáticamente una red de prueba con nombre por defecto que podemos comenzar a editar. Una vez creada el comportamiento por defecto se puede deshabilitar o cambiar, pero quien intente conectarse a tu red mediante el código que la identifica tiene que ser autorizado manualmente para poder usar la red. Podemos observar en las Figuras 7.33 y 7.34 el servidor en la nube antes y después de ser autorizado en la red ZeroTier respectivamente.

| | Edit | Auth | Address | Name/Desc | Managed IPs | Last Seen | Version | Physical IP |
|--------------------------|-------------------------------------|--------------------------------------|---------------------------------|----------------|-------------|------------|---------|----------------|
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | ✗ | 54A68F5916 aa:b5:bf:1fda:25 | | | 1 minute | 1.14.2 | |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | ✓ | 7793C678FA aa:96:8a:56:fb:c9 | RUT_INDUSTRIAL | 172.30.1.1 | 1 minute | 1.14.0 | 91.117.229.232 |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | ✓ | 8901867D1F aa:68:18:16:fe:2c | Gns3_opnsense | 172.30.1.3 | 3 hours | 1.14.2 | 170.253.25.251 |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | ✓ | D95A892C17 aa:38:43:19:af:24 | | 172.30.1.10 | 23 minutes | 1.14.2 | 170.253.25.251 |

Figura 7.33: Zerotier servidor sin autorización

The screenshot shows the ZeroTier web interface for managing a network named "TFG_NETWORK". At the top, there's a header with the ZeroTier logo and the network name. Below it, a "Members" section displays four authorized members. The table has columns for Edit, Auth, Address, Name/Desc, Managed IPs, Last Seen, Version, and Physical IP. The members listed are:

| | Edit | Auth | Address | Name/Desc | Managed IPs | Last Seen | Version | Physical IP |
|--------------------------|-------------------------------------|-------------------------------------|---------------------------------|----------------|--------------|------------|---------|----------------|
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | 54A68F5916 aa:b5:bf:1f:da:25 | OCI SERVER | 172.30.1.100 | 1 minute | 1.14.2 | 79.72.60.98 |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | 7793C678FA aa:96:8a:56:fb:c9 | RUT_INDUSTRIAL | 172.30.1.1 | 1 minute | 1.14.0 | 91.117.229.232 |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | 8901867D1F aa:68:18:16:fe:2c | Gns3_opnsense | 172.30.1.3 | 3 hours | 1.14.2 | 170.253.25.251 |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | D95A892C17 aa:38:43:19:af:24 | | 172.30.1.10 | 25 minutes | 1.14.2 | 170.253.25.251 |

Figura 7.34: ZeroTier servidor autorizado

7.8 Limitaciones en el entorno físico

Es importante mencionar que a pesar de tener 5 *switches* Cisco a disposición, no se puede replicar exactamente el proceso de automatización en éste entorno debido a lo siguiente:

- Sólo los *switches* de distribución, que son de capa 3 y más modernos que los otros, tienen compatibilidad con [SSH](#).
- La versión de sistema operativo de cisco (IOS) de los *switches* de distribución es la 12, que presenta problemas de compatibilidad a la hora de utilizar herramientas de automatización como Ansible para ciertos módulos de cisco, como otras personas también exponen[45][46].

Ante éstos contratiempo se decidió, en vez de intentar reemplazar los modelos incompatibles por otros que si lo soporten, automatizar lo mínimo posible y configurar manualmente la mayor parte de la red física, ayudándose de las configuraciones presentes en los *switches* del laboratorio virtual para replicar el estado de la red y comparar comportamientos, sensaciones y ventajas/desventajas entre ambos entornos dentro de lo posible.

Capítulo 8

Operación

FINALMENTE simulamos distintos usos de la red para comprobar que todo está desplegado y funcionando correctamente: monitorizamos métricas con Grafana, accedemos a la red industrial desde la red corporativa, probamos los protocolos y herramientas seleccionadas...

8.1 Generación de Datos y MQTT

Comprobamos que los datos se estén generando correctamente y siendo enviados por el *router* Teltonika al bróker MQTT del servidor en la nube correctamente.

```
PS D:\modbus_sim> & d:/modbus_sim/env/Scripts/python.exe d:/modbus_sim/modbus_sim.py
cpu_usage = 0.0  cpu_freq = 2400.0  ram_usage = 59.7  disk_usage = 74.0
cpu_usage = 2.6  cpu_freq = 1520.0  ram_usage = 59.6  disk_usage = 74.0
cpu_usage = 3.7  cpu_freq = 2400.0  ram_usage = 59.6  disk_usage = 74.0
cpu_usage = 2.1  cpu_freq = 2400.0  ram_usage = 59.5  disk_usage = 74.0
```

Figura 8.1: Datos generados por el simulado modbus en Python

En la Figura 8.1 podemos apreciar la salida por pantalla del programa en Python que genera los datos y simula un servidor Modbus en el que los recoge el *router*, como vimos anteriormente en la Figura 7.18, para luego enviarlos al bróker MQTT.

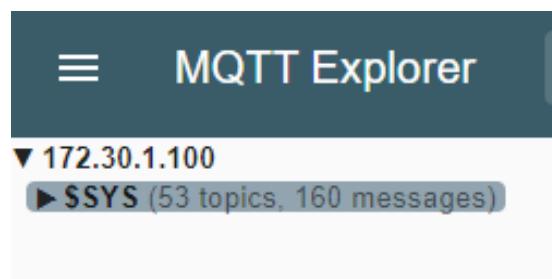


Figura 8.2: Datos sin escribir en el bróker MQTT

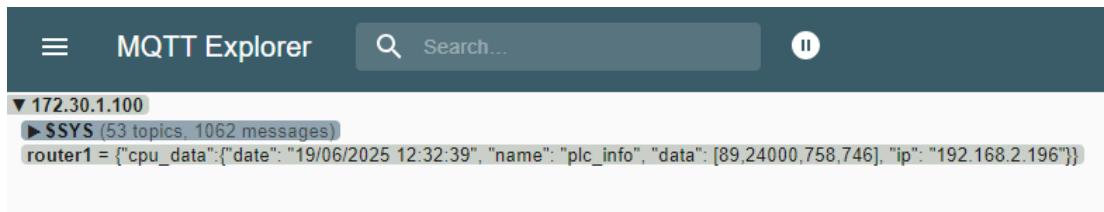


Figura 8.3: Datos escritos en el bróker MQTT

Las Figuras 8.2 y 8.3 comparan el contenido de tópicos del bróker Mosquitto antes y después de comenzar e generar datos. Podemos verlo representado a través de una interfaz gráfica gracias a la herramienta [MQTT Explorer](#).

8.2 Recepción de datos y monitorización

Además de que en la figura 7.21 se comprueba el correcto funcionamiento del *script* de Python como servicio Linux encargado de extraer los datos e insertarlos en la base de datos InfluxDB, a continuación comprobamos visualmente que los datos existan y se representen en Grafana (conectada a Influxdb) en la Figura 8.4.

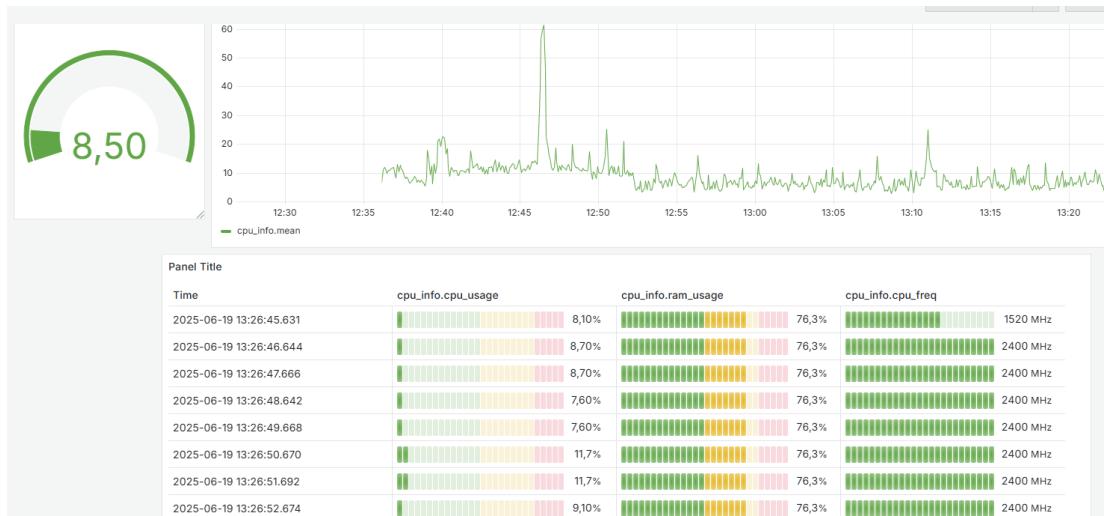


Figura 8.4: Grafana Vista de Tablero

8.3 Comprobación de Control de Acceso y Firewall

Para tener una idea previa de las direcciones que existen a la hora de realizar las pruebas hacemos un listado de las importantes:

- En la **VLAN** 10 se encuentra el servidor con dirección IP 10.0.10.100 y otro PC con dirección 10.0.10.101
- En la **VLAN** 20 hay dos PC con direcciones 10.0.20.100-101
- En la **VLAN** 30 dos PC con direcciones 10.0.30.100-101
- En la **VLAN** 99 un PC con dirección 10.0.99.100 además del resto de switches en el rango 10.0.99.1-5
- En la red **VPN** Zerotier como observamos en la Figura 7.34, el *router* Teltonika corresponde al 172.30.1.1 (y 192.168.2.196 la dirección privada del dispositivo IoT conectado en su red local), el servidor en la nube tiene el 172.30.1.100 y el *router* OPNsense el 172.30.1.3 (y 10.0.0.1 en la red local).

Ahora comprobaremos mediante *pings* en cada **VLAN** la conectividad entre ellas, con Internet, con algún dispositivo de la **VPN** y, en caso de ser este último *ping* correcto, con el dispositivo **IoT** de la red remota industrial. En caso de la **VLAN** 20 también probaremos a acceder a las interfaces web de Grafana e InfluxDB del servidor local. Por último probaremos desde un dispositivo ajeno conectado a la **VPN** si tiene conectividad con el *router* OPNsense y luego si puede acceder a la red local de la oficina.

8.3.1 VLAN 10

```
debian@debian:~$ ip r
default via 10.0.10.254 dev ens4 onlink
10.0.10.0/24 dev ens4 proto kernel scope link src 10.0.10.100
debian@debian:~$ ping 10.0.10.101 -c 1
PING 10.0.10.101 (10.0.10.101) 56(84) bytes of data.
64 bytes from 10.0.10.101: icmp_seq=1 ttl=64 time=4.44 ms

--- 10.0.10.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.441/4.441/4.441/0.000 ms
debian@debian:~$ ping 10.0.20.101 -c 1
PING 10.0.20.101 (10.0.20.101) 56(84) bytes of data.
64 bytes from 10.0.20.101: icmp_seq=1 ttl=63 time=3009 ms

--- 10.0.20.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3009.274/3009.274/3009.274/0.000 ms
debian@debian:~$ ping 10.0.30.100 -c 1
PING 10.0.30.100 (10.0.30.100) 56(84) bytes of data.
From 10.0.10.252 icmp_seq=1 Packet filtered

10.0.20.101 ping statistics
```

Figura 8.5: Pruebas de conectividad de la VLAN 10 (1)

```

--- 10.0.30.100 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

debian@debian:~$ ping 1.1.1.1 -c 1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=126 time=21.0 ms

--- 1.1.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 21.031/21.031/21.031/0.000 ms
debian@debian:~$ ping 172.30.1.1 -c 1
PING 172.30.1.1 (172.30.1.1) 56(84) bytes of data.
64 bytes from 172.30.1.1: icmp_seq=1 ttl=62 time=465 ms

--- 172.30.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 465.069/465.069/465.069/0.000 ms
debian@debian:~$ ping 192.168.2.196
PING 192.168.2.196 (192.168.2.196) 56(84) bytes of data.
64 bytes from 192.168.2.196: icmp_seq=1 ttl=125 time=267 ms

```

Figura 8.6: Pruebas de conectividad de la VLAN 10 (2)

Según los requisitos la zona de servidores debería poder acceder a Internet, a la [VPN](#) (y al dispositivo [IoT](#) de la red local industrial a mayores en este caso) y al resto de [Virtual Local Area Network \(VLAN\)](#) excepto la 30. Todo esto se prueba correctamente como observamos en las Figuras 8.5 y 8.6.

8.3.2 VLAN 20

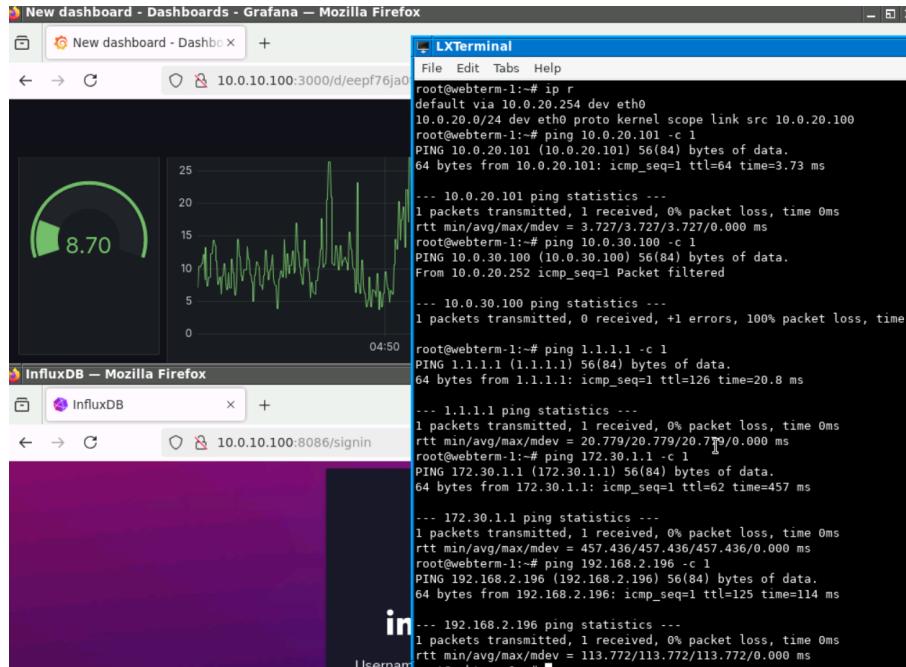


Figura 8.7: Pruebas de conectividad de la VLAN 20

Gracias a la Figura 8.8 nos cercioramos de que la **VLAN** 20 puede acceder al servidor de la **VLAN** 10 y al igual que en la prueba anterior a todas las redes previstas (Internet, **VPN**, dispositivo **IoT**) excepto la **VLAN** 30.

8.3.3 VLAN 30

```

PC1> show ip
NAME      : PC1[1]
IP/MASK   : 10.0.30.100/24
GATEWAY   : 10.0.30.254
DNS       :
MAC       : 00:50:79:66:68:01
LPORT     : 10009
RHOST:PORT : 127.0.0.1:10010
MTU:      : 1500

PC1> ping 10.0.30.101 -c 1
84 bytes from 10.0.30.101 icmp_seq=1 ttl=64 time=4.013 ms

PC1> ping 10.0.20.101 -c 1
*10.0.30.252 icmp_seq=1 ttl=255 time=5.688 ms (ICMP type:3, code:13, Communication administratively prohibited)

PC1> ping 10.0.10.101 -c 1
*10.0.30.252 icmp_seq=1 ttl=255 time=21.147 ms (ICMP type:3, code:13, Communication administratively prohibited)

PC1> ping 172.30.1.1 -c 1
172.30.1.1 icmp_seq=1 timeout

PC1> ping 1.1.1.1 -c 1
84 bytes from 1.1.1.1 icmp_seq=1 ttl=126 time=23.481 ms

```

Figura 8.8: Pruebas de conectividad de la VLAN 30

La **VLAN** 30 solo debe tener conectividad con su misma subred y con Internet, sin gozar de acceso a la **VPN**. La prueba en la Figura 8.8 confirma el funcionamiento esperado e incluso se podemos ver como actuaron las **ACL** al intentar acceder a otras **VLAN**.

8.3.4 VLAN 99

```

debian@debian:~$ ip a | grep 'inet 10\.\d'
    inet 10.0.99.100/24 brd 10.0.99.255 scope global ens5
debian@debian:~$ ping 10.0.10.100 -c 1
ping: connect: Network is unreachable
debian@debian:~$ ping 10.0.20.100 -c 1
ping: connect: Network is unreachable
debian@debian:~$ ping 10.0.30.100 -c 1
ping: connect: Network is unreachable
debian@debian:~$ ping 1.1.1.1 -c 1
ping: connect: Network is unreachable
debian@debian:~$ ping 10.0.99.5 -c 1
PING 10.0.99.5 (10.0.99.5) 56(84) bytes of data.
64 bytes from 10.0.99.5: icmp_seq=1 ttl=255 time=16.9 ms

--- 10.0.99.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 16.930/16.930/16.930/0.000 ms
debian@debian:~$ 

```

Figura 8.9: Pruebas de conectividad de la VLAN 99

La VLAN 99 solamente puede acceder a su misma red y esto lo comprobamos en la Figura 8.9.

8.3.5 Otros

```
PS C:\> ping 172.30.1.3 -n 1
Haciendo ping a 172.30.1.3 con 32 bytes de datos:
Respueta desde 172.30.1.3: bytes=32 tiempo=35ms TTL=64

Estadisticas de ping para 172.30.1.3:
  Paquetes: enviados = 1, recibidos = 1, perdidos = 0
              (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
  Minimo = 35ms, Maximo = 35ms, Media = 35ms
PS C:\> ping 10.0.0.1 -n 1

Haciendo ping a 10.0.0.1 con 32 bytes de datos:
Tiempo de espera agotado para esta solicitud.

Estadisticas de ping para 10.0.0.1:
  Paquetes: enviados = 1, recibidos = 0, perdidos = 1
              (100% perdidos),
PS C:\> ping 10.0.10.100 -n 1

Haciendo ping a 10.0.10.100 con 32 bytes de datos:
Tiempo de espera agotado para esta solicitud.
```

Figura 8.10: Pruebas de conectividad de otros dispositivos

Desde un dispositivo ajeno a la red pero conectado a la [VPN](#) se comprueba, como vemos en la Figura 8.10, que tiene conectividad con el *router* OPNsense pero no con su red local, tal y como esperábamos.

8.4 Comprobación de Protocolos

8.4.1 OSPF

| Routing: Diagnostics: OSPF | | | | | |
|----------------------------|---------------|--------------|-----------|------------|------------------|
| Overview | Routing Table | Database | Neighbors | Interfaces | Metrics |
| 10.0.99.4 | 1 | Full/Backup | 36640 | 10.0.0.2 | bridge0:10.0.0.1 |
| 10.0.99.5 | 1 | Full/DROther | 39692 | 10.0.0.3 | bridge0:10.0.0.1 |

Figura 8.11: Prueba de vecinos OSPF en OPNsense

| Routing: Diagnostics: OSPF | | | | | |
|----------------------------|--------------|---------------|----------|-------------------|------------|
| | Overview | Routing Table | Database | Neighbors | Interfaces |
| Type | Network | Cost | Area | Via | |
| N | 10.0.0.0/29 | 10 | 0.0.0.1 | Directly Attached | |
| N | 10.0.10.0/24 | 11 | 0.0.0.1 | 10.0.0.2 | |
| N | 10.0.10.0/24 | 11 | 0.0.0.1 | 10.0.0.3 | |
| N | 10.0.20.0/24 | 11 | 0.0.0.1 | 10.0.0.2 | |
| N | 10.0.20.0/24 | 11 | 0.0.0.1 | 10.0.0.3 | |
| N | 10.0.30.0/24 | 11 | 0.0.0.1 | 10.0.0.2 | |
| N | 10.0.30.0/24 | 11 | 0.0.0.1 | 10.0.0.3 | |

Figura 8.12: Prueba de rutas OSPF en OPNsense

En las Figuras 8.11 y 8.12 podemos apreciar efectivamente que el *router* OPNsense detecta ambos *switches* como vecinos OSPF y adquiere las rutas de las VLAN que estos le comparten.

```
swl3_dist1#show ip ospf neighbor
Neighbor ID      Pri  State            Dead Time    Address          Interface
10.0.99.5        1    FULL/DROTHER   00:00:39    10.0.0.3        GigabitEthernet0/1
192.168.4.11     1    FULL/DR       00:00:36    10.0.0.1        GigabitEthernet0/1
swl3_dist1#
swl3_dist1#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
      a - application route
      + - replicated route, % - next hop override

Gateway of last resort is 10.0.0.1 to network 0.0.0.0

O*E2  0.0.0.0/0 [110/10] via 10.0.0.1, 00:22:35, GigabitEthernet0/1
      10.0.0.8 is variably subnetted, 10 subnets, 3 masks
C     10.0.0.0/29 is directly connected, GigabitEthernet0/1
L     10.0.0.2/32 is directly connected, GigabitEthernet0/1
C     10.0.10.0/24 is directly connected, Vlan10
L     10.0.10.252/32 is directly connected, Vlan10
```

Figura 8.13: Prueba de funcionamiento del protocolo OSPF en switch

La Figura 8.13 no muestra cómo uno de los *switches* de distribución verifica también que el protocolo OSPF funciona. Detecta al otro *switch* y al *router* como vecinos y adquiere la ruta por defecto 0.0.0.0 hacia la dirección IP del *router* como se puede ver a su izquierda por "O+E2".

8.4.2 VRRP

```

swl3_dist2#show vrrp br
Interface      Grp Pri Time  Own Pre State    Master addr    Group addr
Vl10          10  90  3648     Y  Master  10.0.10.253  10.0.10.254
Vl20          20  90  3648     Y  Master  10.0.20.253  10.0.20.254
Vl30          30  90  3648     Y  Master  10.0.30.253  10.0.30.254
swl3_dist2#
*Jun 25 05:32:14.761: %VRRP-6-STATECHANGE: Vl10 Grp 10 state Master -> Backup
*Jun 25 05:32:14.763: %VRRP-6-STATECHANGE: Vl20 Grp 20 state Master -> Backup
*Jun 25 05:32:14.769: %VRRP-6-STATECHANGE: Vl30 Grp 30 state Master -> Backup
swl3_dist2#show vrrp br
Interface      Grp Pri Time  Own Pre State    Master addr    Group addr
Vl10          10  90  3648     Y  Backup  10.0.10.252  10.0.10.254
Vl20          20  90  3648     Y  Backup  10.0.20.252  10.0.20.254
Vl30          30  90  3648     Y  Backup  10.0.30.252  10.0.30.254
swl3_dist2#show logging | include VRRP
*Jun 25 05:29:35.958: %VRRP-6-STATECHANGE: Vl10 Grp 10 state Init -> Backup
*Jun 25 05:29:35.960: %VRRP-6-STATECHANGE: Vl20 Grp 20 state Init -> Backup
*Jun 25 05:29:35.961: %VRRP-6-STATECHANGE: Vl30 Grp 30 state Init -> Backup
*Jun 25 05:29:39.608: %VRRP-6-STATECHANGE: Vl10 Grp 10 state Backup -> Master
*Jun 25 05:29:39.614: %VRRP-6-STATECHANGE: Vl20 Grp 20 state Backup -> Master
*Jun 25 05:29:39.629: %VRRP-6-STATECHANGE: Vl30 Grp 30 state Backup -> Master
*Jun 25 05:29:42.459: %VRRP-6-STATECHANGE: Vl10 Grp 10 state Master -> Backup
*Jun 25 05:29:42.467: %VRRP-6-STATECHANGE: Vl20 Grp 20 state Master -> Backup
*Jun 25 05:29:42.473: %VRRP-6-STATECHANGE: Vl30 Grp 30 state Master -> Backup
*Jun 25 05:30:55.643: %VRRP-6-STATECHANGE: Vl30 Grp 30 state Backup -> Master
*Jun 25 05:30:55.664: %VRRP-6-STATECHANGE: Vl10 Grp 10 state Backup -> Master
*Jun 25 05:30:56.315: %VRRP-6-STATECHANGE: Vl20 Grp 20 state Backup -> Master
*Jun 25 05:32:14.761: %VRRP-6-STATECHANGE: Vl10 Grp 10 state Master -> Backup
*Jun 25 05:32:14.763: %VRRP-6-STATECHANGE: Vl20 Grp 20 state Master -> Backup
*Jun 25 05:32:14.769: %VRRP-6-STATECHANGE: Vl30 Grp 30 state Master -> Backup

```

Figura 8.14: Prueba de funcionamiento del protocolo VRRP

Para finalizar hemos apagado y encendido el *switch* que actuaba como maestro VRRP, y en el otro *switch* hemos visualizado correctamente los cambios de *backup* a *master* y lo mismo en orden inverso cuando se encendió de nuevo el *switch* apagado, tal y como podemos observar en la Figura 8.14.

Capítulo 9

Conclusiones

9.1 Resumen

LEGADO al último capítulo de la memoria del proyecto, es el momento de repasar todo lo realizado, extraer nuestras conclusiones, comparativas y opiniones respecto a las tecnologías o métodos empleados, qué hemos aprendido de todo el trabajo y qué mejoras con vista al futuro le podríamos realizar a nuestro proyecto.

9.2 Lecciones Aprendidas

Abordamos el diseño de una red empresarial con éxito empleando la metodología [PP-DIOO](#), que nos permitió organizar las ideas y requisitos generales en un diseño bien estructurado y definido que facilitó la posterior implementación de la red. Hemos utilizado gran variedad de tecnologías y herramientas como Grafana para crear paneles y visualizaciones de datos en tiempo real o períodos temporales provenientes de distintas fuentes de datos e InfluxDB como base de datos de series temporales orientada al almacenamiento de datos con marcas de tiempo como métricas de sensores, lo cual nos descubre una potente base de datos no relacional para casos de uso específicos como éste. Hemos investigado e implementado una manera de integrar protocolos muy utilizados en ámbitos industriales como la transmisión de datos a través de Modbus y el sistema de suscripciones y publicaciones en los bróker [MQTT](#) que nos ha permitido ser capaces de transportar registros de datos entre ubicaciones remotas con una gran facilidad gracias también a tecnologías actuales como los servidores en la nube que nos permiten realizar proyectos tan interesantes como estos. En el diseño de redes hemos trabajado con buenas prácticas para estructurar y gestionar nuestra topología, segmentando distintos grupos de trabajadores y dispositivos con distintos niveles de privilegios en [VLANs](#) que se han aprovechado para asignar restricciones de acceso según lo estipulado con anterioridad por los requisitos iniciales. Tecnologías como [OSPF](#), [VRRP](#) y [PortChannel](#) las utilizamos

par aportar redundancia, tolerancia a fallos y transmisión de rutas entre dispositivos de manera dinámica. Una de nuestras mayores satisfacciones personales durante este trabajo ha sido tener la oportunidad de descubrir y trabajar con el *router/firewall* OPNsense y explorar la cantidad de opciones que ofrece para gestionar y monitorizar una red, aunque al no ser un tema principal en este proyecto y tener una curva de aprendizaje inicial elevada no hemos utilizado ni mostrado su repertorio de funcionalidades más avanzadas.

Todo lo anterior hemos sido capaces de implementarlo de manera automatizada con herramientas como Ansible, Terraform y Python, lo que nos permite agilizar el despliegue de la misma u otra red igual a la planteada hasta necesitar solo unos cuantos minutos y la posibilidad de cambiar la configuración desde sus ficheros de variables en unos segundos para aplicar distintas configuraciones a la red.

9.3 Comparativa de entornos

Otro tema interesante se encuentra en la comparativa de trabajar en un mismo proyecto con un entorno virtual y físico. Si bien surgieron limitaciones relacionadas con los dispositivos de la red física, a la hora de la implementación manual no hemos distinguido cambios notables, pero si que podemos destacar ciertas ventajas y desventajas que son posibles de apreciar con el uso continuado de un tipo de entorno:

- El entorno virtual nos ofrece muchas más variedades de opciones, sobretodo si hay una clara limitación financiera, y otorga la capacidad de llevarnos los proyectos con nosotros o trabajar de manera remota sobre el laboratorio virtual de manera sencilla y sin opción a aislarla de la red por un fallo de configuración.
- Trabajar con el entorno físico es lo ideal para conocer y familiarizarse con el *hardware* de los dispositivos y para evitar accidentes inesperados debido a ciertas opciones que pueden no existir o funcionar como en la imagen virtual.
- La conexión en físico a varios dispositivos que pueden perder la conexión en cualquier momento resulta frustrante e incluso retrasa los avances que uno podría tener en el entorno virtual.
- La potencia y rendimiento en el laboratorio virtual es uno de sus mayores lastres. Aunque en gran medida dependa del equipo que ejecute las herramientas de virtualización, la emulación puede no ser perfecta y debido a la arquitectura física real es posible que nunca pueda llegar a serlo. Esto en conjunto provoca que de vez en cuando se encuentren errores de emulación o externos que provocan la caída, no respuesta o mal funcionamiento de los equipos, lo que a veces nos hizo vernos obligados a crear nuevas máquinas virtuales desde cero y borrar las perjudicadas.

Es posible que una mezcla entre lo físico y lo virtual para los proyectos, según las necesidades específicas que requieran, sea la mejor opción para aprovechar las ventajas de los ambos mundos. Es sorprendente la facilidad con la que se puede hacer interactuar a una red virtualizada de más de 10 dispositivos con una red real operativa o parte de ella.

9.4 Trabajo futuro

Muchas opciones tan buenas o mejores como las implementadas quedaron descartadas por falta de tiempo, complejidad o falta de material necesario. Las más interesantes para mejorar este proyecto en un futuro son las siguientes:

- Aplicar redundancia en *switches* con tecnologías de *stack* virtuales que permiten tratar a ambos como un mismo dispositivo lógico y permitiría por ejemplo realizar un *PortChanel* hacia el *router* con los cables de cada *switch* en un mismo grupo.
- Implementar sistema de *backups*, *nic bonding* para los servidores, dos *switches* de acceso en las zonas más críticas...
- Implementar sistemas de monitorización del estado y seguridad de la red como **IDS**, **Intrusion Prevention System (IPS)** o **Security Information and Event Management (SIEM)**

Apéndices

Apéndice A

Material adicional

A.1 Ansible

Aquí se muestran los ficheros de Ansible empleados para hacer copias de seguridad de manera automática de los *switches* y *router* de la red para guardar un estado concreto al que poder volver si las pruebas y cambios realizados posteriormente no son de nuestro agrado.

role/switch_backup/tasks/main.yml

```
1 - name: Backup config (when mode is backup)
2   cisco.ios.ios_config:
3     backup: yes
4     when: mode == 'backup'
5     register: backup_result
6
7 - name: Save backup locally
8   fetch:
9     src: "{{ backup_result.backup_path }}"
10    dest: "{{ backup_file }}{{ inventory_hostname }}.cfg"
11    flat: yes
12    when: mode == 'backup'
13
14 - name: Restore config from file (when mode is restore)
15   cisco.ios.ios_config:
16     src: "{{ backup_file }}{{ inventory_hostname }}.cfg"
17     when: mode == 'restore'
```

role/switch_backup/var/var.yml

```
1 backup_file: "/home/debian/ansible_local/backups/"
```

```
1 - hosts: all
2   gather_facts: no
```

```

3
4 vars_prompt:
5   - name: mode
6     prompt: "Choose action for router/switches (backup|restore)"
7   - name: dev_type
8     prompt: "Choose to what apply the action (router|switch|all)"
9
10 tasks:
11   - name: OPNsense router Backup/Restore
12     include_role:
13       name: backup_router
14     when:
15       - inventory_hostname in groups['opnsense']
16       - dev_type in ['router', 'all']
17
18   - name: Cisco switch Backup/Restore
19     include_role:
20       name: backup_switch
21     when:
22       - inventory_hostname in (groups['12switches'] +
23         groups['13switches'])
24       - dev_type in ['switch', 'all']

```

role/router_backup/tasks/main.yml

```

1 - name: Find current old backup
2   find:
3     paths: "{{ backup_folder }}"
4     patterns: "config*"
5   register: old_backup
6
7 - name: Delete old backups
8   file:
9     path: "{{ item.path }}"
10    state: absent
11   with_items: "{{ old_backup.files }}"
12   when: old_backup.matched > 0 and mode == "backup"
13
14 - name: Download current config.xml
15   ansible.builtin.uri:
16     url: "https://{{ ansible_host }}/api/core/backup/download/this"
17     method: GET
18     force_basic_auth: yes
19     url_username: "{{ api_key }}"
20     url_password: "{{ api_secret }}"
21     dest: "{{ backup_folder }}"
22     validate_certs: no

```

```

23     status_code: 200
24 when: mode == "backup"
25
26 - name: Restore previously saved config
27   ansible.builtin.uri:
28     url: "https://{{ ansible_host
29       }}/api/core/backup/revert_backup/config-{{"
30       old_backup.files[0].path.split('-')[2] }}"
31     method: GET
32     force_basic_auth: yes
33     url_username: "{{ api_key }}"
34     url_password: "{{ api_secret }}"
35     validate_certs: no
36     status_code: 200
37     when: mode == "restore"

```

role/router_backup/var/vars.yml

```

1 backup_folder: "/home/debian/ansible_local/backups/"

```

Estos roles ofrecen al usuario, una vez ejecute el *playbook* configurado, dos opciones (*backup/restore*) que el usuario puede introducir por teclado para especificar si se quiere realizar una copia de seguridad o restaurar una ya realizada y por último seleccionar a que *hosts* quiere realizarle copias de seguridad (*switches/router/all*).

A.2 Grafana e InfluxDB

En esta sección se muestran imágenes mostrando las interfaces de Grafana e InfluxDB y ciertas configuraciones que no pudieron ponerse en el trabajo por exceso de páginas.

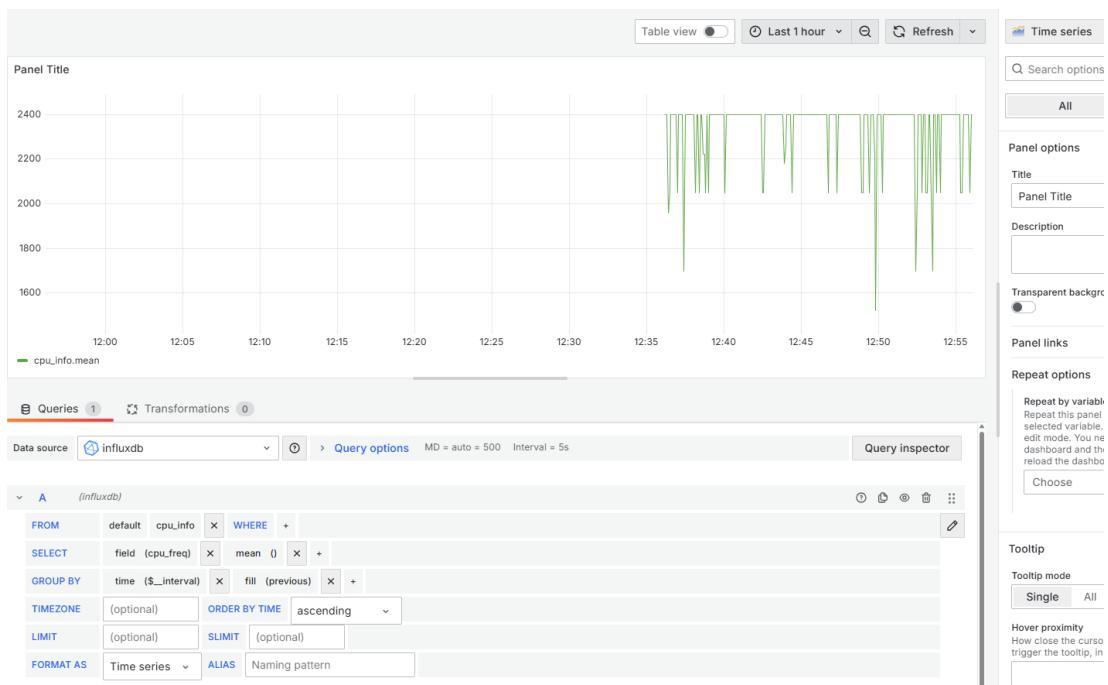


Figura A.1: Creación de un panel simple en Grafana

The screenshot shows the Grafana configuration interface at localhost:3000/connections/datasources/edit/depf69lj5is5cc. The left sidebar is open, showing the 'Data sources' section selected. The main panel is titled 'HTTP' and contains fields for 'URL' (set to `http://localhost:8086`), 'Allowed cookies' (with a placeholder 'New tag (enter key to add)' and an 'Add' button), and 'Timeout' (set to 'Timeout in seconds'). Below this is the 'Auth' section, which includes 'Basic auth' (enabled), 'With Credentials' (disabled), 'TLS Client Auth' (disabled), 'With CA Cert' (disabled), 'Skip TLS Verify' (disabled), and 'Forward OAuth Identity' (disabled). Under 'Basic Auth Details', the 'User' field is set to 'admin' and the 'Password' field contains a masked value. The 'Custom HTTP Headers' section has a '+ Add header' button. At the bottom is the 'InfluxDB Details' section, which includes a note about database access and a warning about data isolation.

Figura A.2: Añadiendo InfluxDB como fuente de datos de Grafana (1)

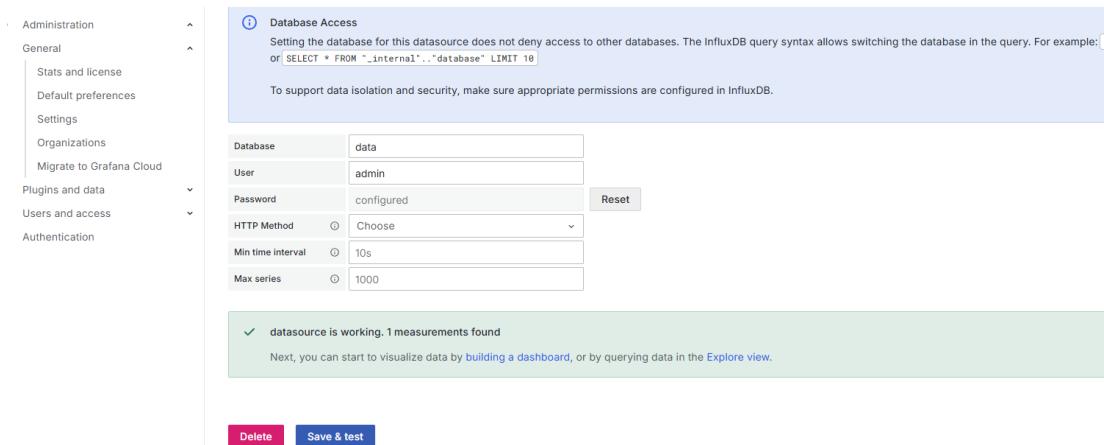


Figura A.3: Añadiendo InfluxDB como fuente de datos de Grafana (2)

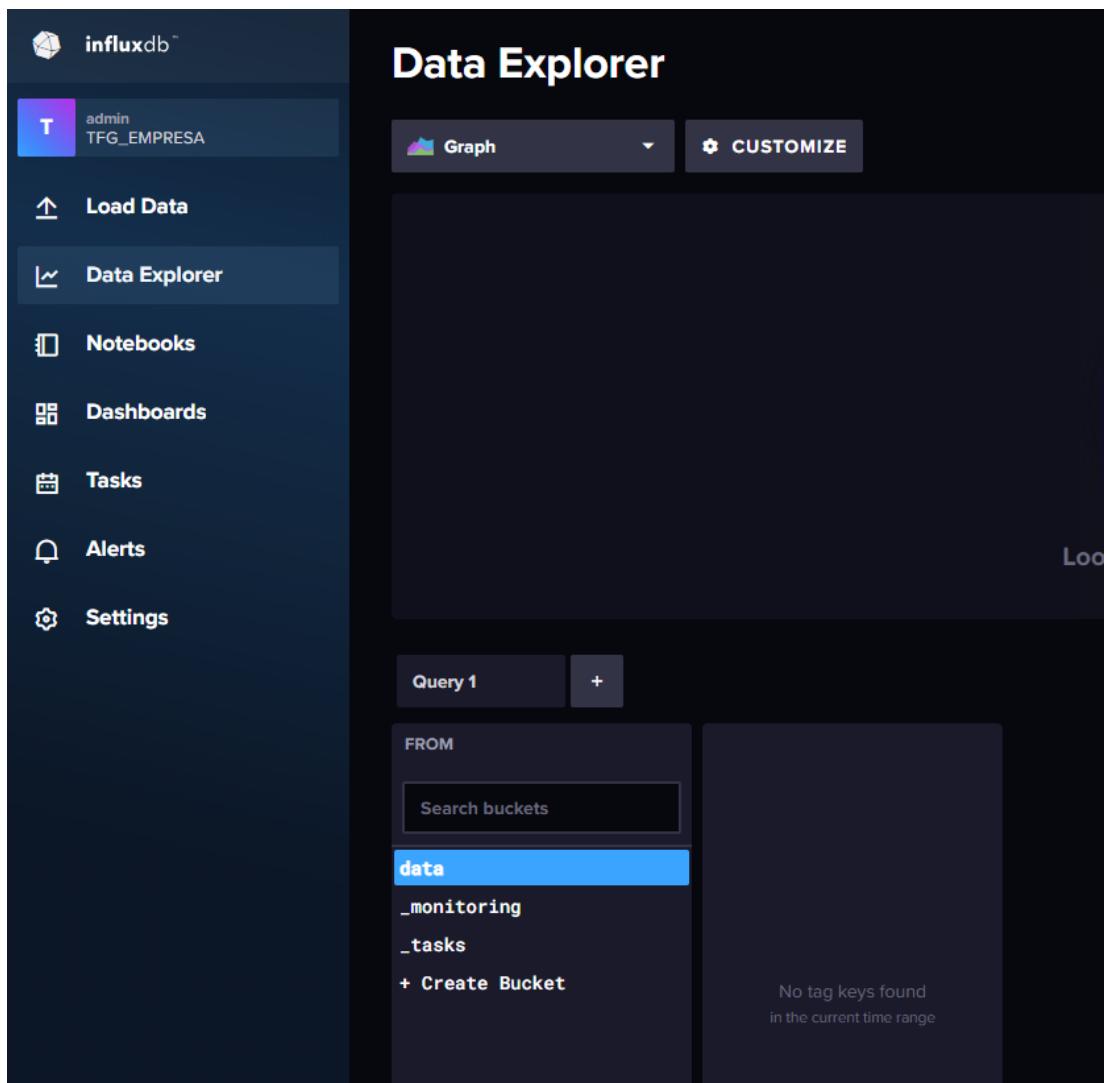


Figura A.4: Interfaz general de InfluxDB

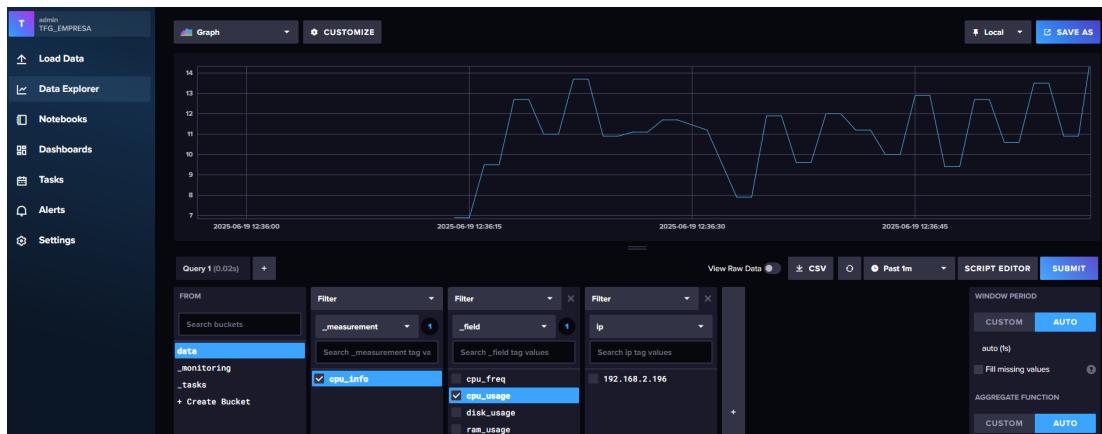


Figura A.5: Visualización de datos desde la interfaz InfluxDB

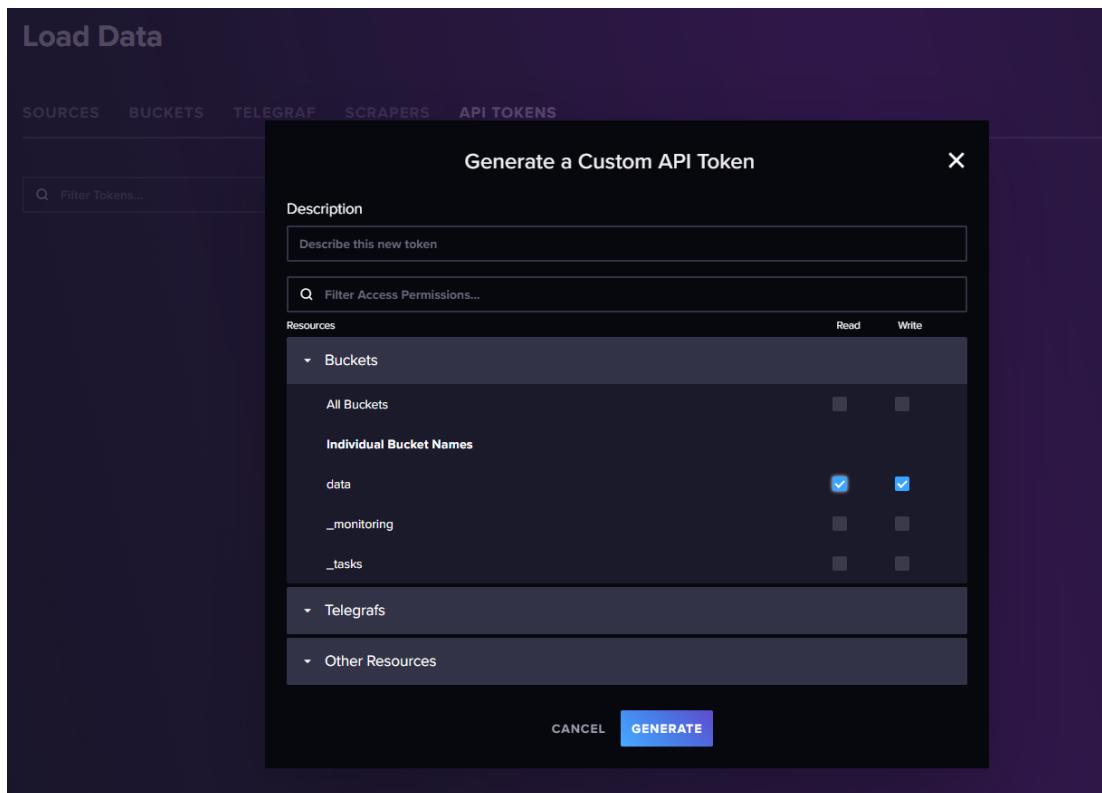


Figura A.6: Creación de un token api en InfluxDB

A.3 Oracle Cloud

Aquí mostramos un ejemplo de lo que se vería en las instancias de nuestra cuenta de Oracle Cloud después de ejecutar el *script* para el despliegue automática de la instancia



The screenshot shows a table of instances. The columns are: Name, State, Public IP, Private IP, Shape, OCPU count, and Memory (GB). There is one row visible, representing an instance named "ubuntu-zerotier" which is "Running". Its public IP is 79.72.60.98, private IP is 10.0.0.251, shape is VM.Standard.A1.Flex, OCPU count is 1, and memory is 12 GB.

| <input type="checkbox"/> | Name | State | Public IP | Private IP | Shape | OCPUs | Memory (GB) |
|--------------------------|---------------------------------|---------|-------------|------------|---------------------|-------|-------------|
| <input type="checkbox"/> | ubuntu-zerotier | Running | 79.72.60.98 | 10.0.0.251 | VM.Standard.A1.Flex | 1 | 12 |

Figura A.7: Instancia corriendo de Oracle Cloud

Apéndice B

Imágenes dispositivos físicos

Aquí mostramos fotos con los dispositivos de red físicos utilizados en algún momento del proyecto:

APÉNDICE B. IMÁGENES DISPOSITIVOS FÍSICOS

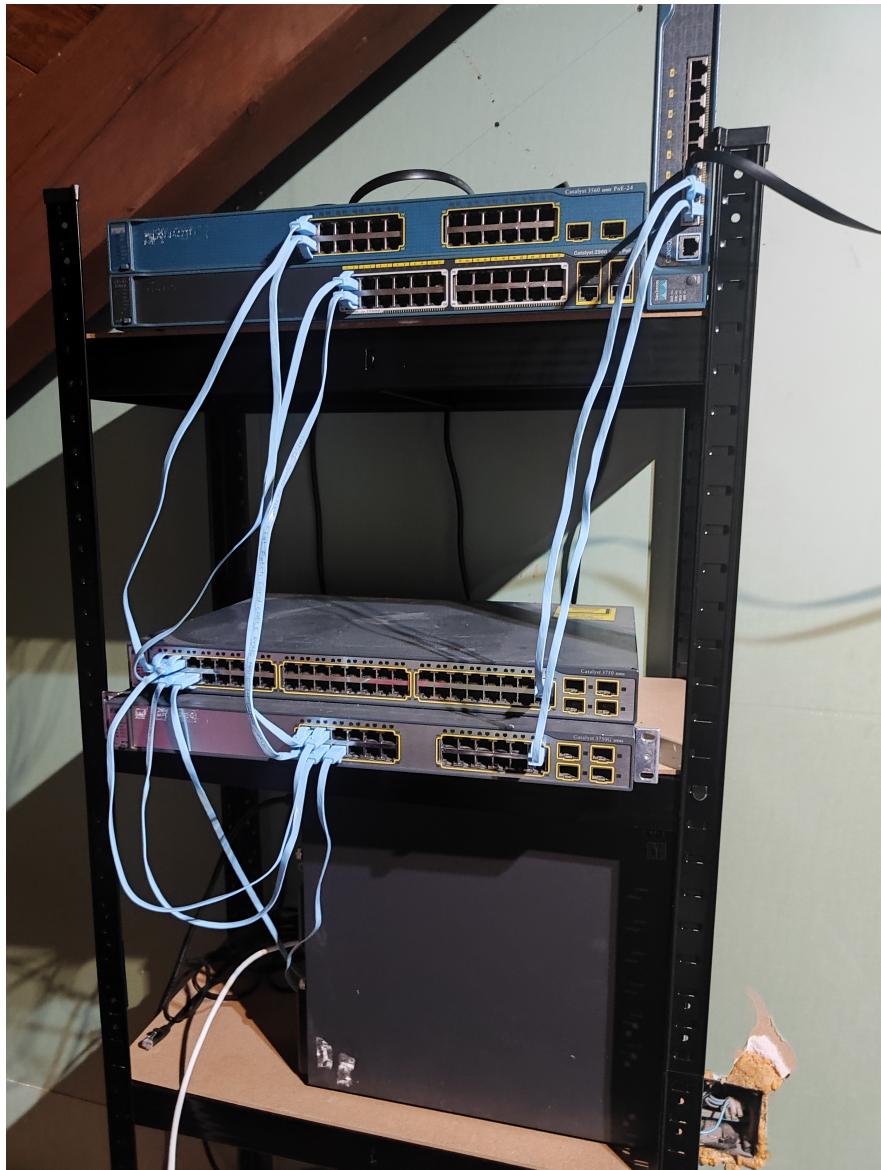


Figura B.1: Switches Cisco y router OPNsense reales

APÉNDICE B. IMÁGENES DISPOSITIVOS FÍSICOS

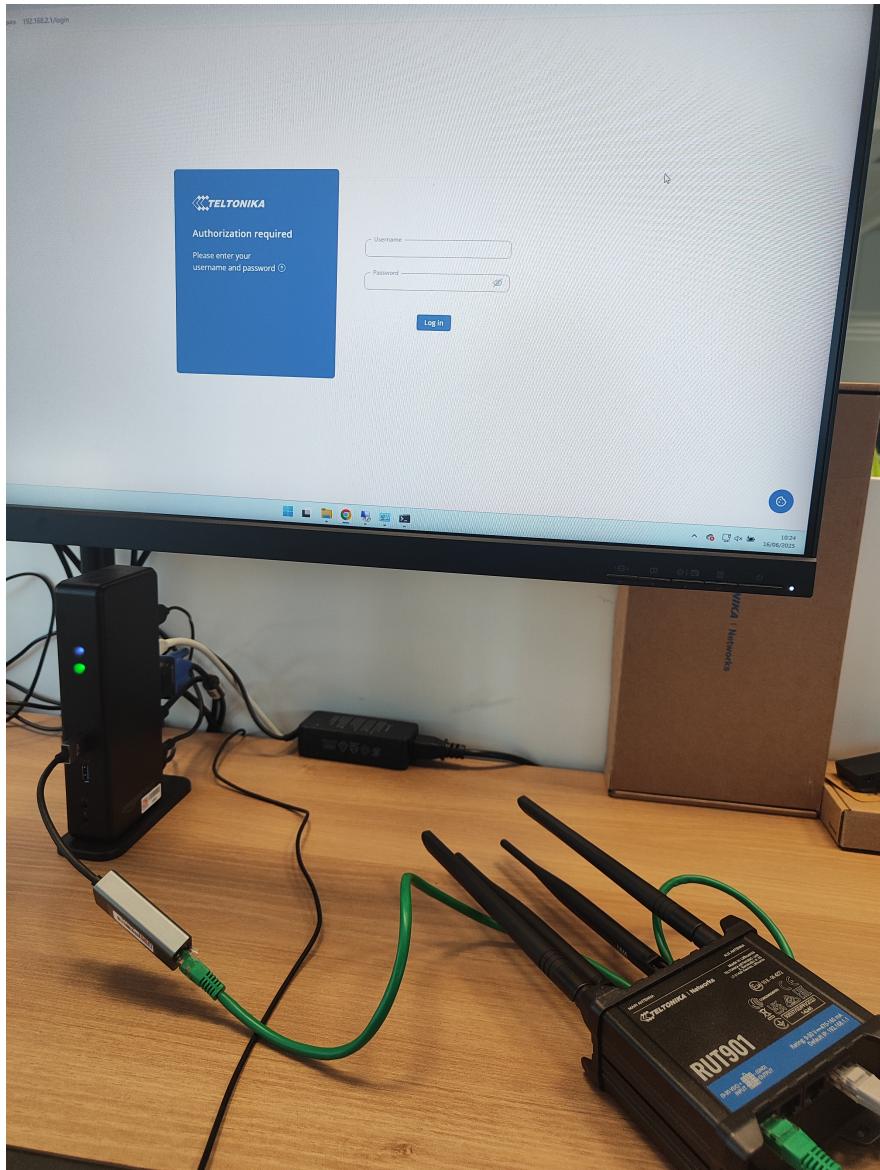


Figura B.2: Router Teltonika RUT901 de la red industrial

Lista de acrónimos

ACL Access Control List. [1](#), [6](#), [29](#), [44](#), [74](#)

CGNAT Carrier-Grade NAT. [14](#), [17](#)

CML Cisco Modelings Labs. [18](#)

DMZ demilitarized zone. [14](#)

DNS Domain Name System. [5](#)

HMI Human-Machine Interface. [1](#), [11](#)

IDS Intrusion Detection System. [18](#), [80](#)

IGP Interior Gateway Protocol. [8](#)

IoT Internet of Things. [1](#), [6](#), [11](#), [17](#), [19](#), [21](#), [56](#), [72–74](#)

IPS Intrusion Prevention System. [80](#)

IT Tecnología de las Informaciones. [1](#), [4](#), [11](#)

LACP Link Aggregation Control Protocol. [27](#)

LAN Local Area Network. [7](#), [25](#), [64](#), [67](#)

LSA Link State Advertisements. [9](#)

M2M Machine-to-Machine. [6](#)

MQTT Message Queueing Telemetry Transport. [1](#), [2](#), [5](#), [6](#), [15](#), [16](#), [19–21](#), [24](#), [53](#), [59](#), [61](#), [62](#), [70](#), [71](#), [78](#)

MSTP Multiple Spanning Tree Protocol. 8

NAT Network Address Translation. 7, 14, 18, 66

OCI Oracle Cloud Infrastructure (OCI). 14

OSPF Open Shortest Path First. 1, 8, 9, 18, 29, 45, 76, 78

OT Operational Technology. 1

PBR Policy Based Routing. 18

PLC Programmable Logic Controller. 1, 5, 11, 57

PPDIOO Preparar Planear Diseñar Identifica Operar Optimizar. 1, 9, 10, 78

PVST+ Per-Vlan Spanning Tree. 8

PYME Pequeña y Mediana Empresa. 12, 16

QoS Quality of Service. 6, 18, 59

RAM Random Access Memory. 18

RSA Rivest, Shamir y Adleman. 25, 30, 33

RSTP Rapid Spanning Tree Protocol. 8

SD-WAN Software Defined Wide Area Network. 7

SIEM Security Information and Event Management. 80

SSH Secure Shell. 4, 25, 33, 34, 36, 39, 50, 69

STP Spanning Tree Protocol. 8, 18, 26–28

SVI Switched Virtual Interface. 18, 28, 29, 33, 42

VLAN Virtual Local Area Network. 1, 4, 7–9, 15, 16, 18, 24, 26–29, 32, 33, 38, 42, 44, 64, 65, 72–76, 78

VPN Virtual Private Network. 1, 2, 5, 7, 14, 15, 17, 18, 20, 21, 24, 53, 63, 65, 66, 72–75

VRP Virtual Router Redundancy Protocol. 1, 28, 29, 45, 77, 78

WAN Wide Area Network. 7, 66

Bibliografía

- [1] C. Systems, “Campus lan and wireless lan design guide,” 2023, cisco Validated Design. [En línea]. Disponible en: <https://www.cisco.com/c/en/us/td/docs/solutions/CVD/Campus/cisco-campus-lan-wlan-design-guide.html>
- [2] Ansible Documentation Team, “Ansible getting started: Introduction,” Ansible Documentation, 2025. [En línea]. Disponible en: https://docs.ansible.com/ansible/latest/getting_started/introduction.html
- [3] ——, “Ansible cisco.ios.vlans module,” Ansible Documentation, 2025. [En línea]. Disponible en: https://docs.ansible.com/ansible/latest/collections/cisco/ios/vlans_module.html
- [4] H. Developer, “Terraform documentation,” HashiCorp, 2025. [En línea]. Disponible en: <https://developer.hashicorp.com/terraform/docs>
- [5] M. Organization, “Modbus application protocol specification v1.1b3,” Modbus Organization, 2006. [En línea]. Disponible en: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf
- [6] O. M. TC, “Mqtt version 5.0: Oasis standard,” OASIS, 2025. [En línea]. Disponible en: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html#_Introduction
- [7] I. Cisco Systems, “Configuring access control lists,” Cisco Systems, 2012. [En línea]. Disponible en: https://www.cisco.com/en/US/docs/switches/datacenter/nexus5000/sw/configuration/guide/cli_rel_4_1/Cisco_Nexus_5000_Series_Switch_CLI_Software_Configuration_Guide_chapter21.pdf
- [8] C. Community, “Cisco access control lists (acl),” Cisco, 2025. [En línea]. Disponible en: <https://community.cisco.com/t5/networking-knowledge-base/cisco-access-control-lists-acl/ta-p/4182349>

- [9] I. ZeroTier, “Zerotier documentation,” ZeroTier, Inc., 2025. [En línea]. Disponible en: <https://docs.zerotier.com/what/>
- [10] P. A. N. Cyberpedia, “What is sd-wan?” Palo Alto Networks, 2023. [En línea]. Disponible en: <https://www.paloaltonetworks.com/cyberpedia/what-is-sd-wan>
- [11] W. Odom, *CCNA 200-301 Official Cert Guide, Volume 1*, 2nd ed. Cisco Press, 2020.
- [12] C. L. N. Community, “Vlan 1 and vlan hopping attack,” Cisco Systems, Inc., 2019. [En línea]. Disponible en: <http://learningnetwork.cisco.com/s/blogs/a0D3i000002SKPREA4/vlan1-and-vlan-hopping-attack>
- [13] J. Moy, “Ospf version 2,” Internet Engineering Task Force (IETF), 1998. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc2328>
- [14] T. Li, D. A. Maltz, and C. L. Partridge, “Ospf for ipv6 (ospfv3),” Internet Engineering Task Force (IETF), 2008. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc5340>
- [15] C. Systems, “Small enterprise design profile – chapter 2: Network foundation,” 2020. [En línea]. Disponible en: https://www.cisco.com/c/dam/en/us/td/docs/solutions/Enterprise/Small_Enterprise_Design_Profile/chap2sba.pdf
- [16] D. H. Brad Edgeworth, Ramiro Garza and J. Gooley, *CCNP and CCIE Enterprise Core ENCOR 350-401 Official Cert Guide*, 1st ed. Cisco Press, 2020.
- [17] O. Corporation, “Oracle cloud infrastructure documentation,” Oracle, 2025. [En línea]. Disponible en: <https://docs.oracle.com/en-us/iaas/Content/home.htm>
- [18] E. Foundation, “Eclipse mosquitto documentation,” Eclipse Foundation, 2025. [En línea]. Disponible en: <https://mosquitto.org/documentation/>
- [19] InfluxData, “Influxdb documentation,” InfluxData, 2025. [En línea]. Disponible en: <https://docs.influxdata.com/influxdb/v2/>
- [20] G. Labs, “Grafana documentation,” Grafana Labs, 2025. [En línea]. Disponible en: <https://grafana.com/docs/grafana/latest/>
- [21] —, “Available data sources,” Grafana Labs, 2025. [En línea]. Disponible en: <https://grafana.com/docs/grafana/latest/datasources/>
- [22] G. Team, “Gns3 documentation,” GNS3, 2025. [En línea]. Disponible en: <https://docs.gns3.com/docs/>

- [23] O. Team, “Opnsense documentation,” OPNsense, 2025. [En línea]. Disponible en: <https://docs.opnsense.org>
- [24] I. Cisco Systems, “Cisco modeling labs – iosvl2 documentation,” Cisco Systems, 2025. [En línea]. Disponible en: <https://developer.cisco.com/docs/modeling-labs/iosvl2/>
- [25] T. Networks, “Rut901 5g industrial lte router user guide,” Teltonika Networks, 2025. [En línea]. Disponible en: <https://wiki.teltonika-networks.com/view/RUT901>
- [26] B. O. del Estado, “Resolución de 4 de abril de 2025, de la dirección general de trabajo, por la que se registra y publica el xix convenio colectivo estatal de empresas de consultoría, tecnologías de la información y estudios de mercado y opinión pública,” 2025. [En línea]. Disponible en: [https://www.boe.es/eli/es/res/2025/04/04/\(10\)](https://www.boe.es/eli/es/res/2025/04/04/(10))
- [27] AnsibleGuy, “ansibleguy.opnsense collection documentation,” AnsibleGuy, 2025. [En línea]. Disponible en: <https://opnsense.ansibleguy.net/>
- [28] K. Byers, “Netmiko: Multi-vendor ssh python library,” PyPI, 2025. [En línea]. Disponible en: <https://pypi.org/project/netmiko/>
- [29] J. Postel and J. Reynolds, “Telnet protocol specification,” Internet Engineering Task Force (IETF), 1983. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc854>
- [30] I. Cisco Systems, “Vlan best practices and security tips for cisco business routers,” Cisco Systems, 2017. [En línea]. Disponible en: <https://www.cisco.com/c/en/us/support/docs/smb/routers/cisco-rv-series-small-business-routers/1778-tz-VLAN-Best-Practices-and-Security-Tips-for-Cisco-Business-Routers.html>
- [31] ——, “Configuring port channels—cisco nexus 5500 interfaces configuration guide,” 2021. [En línea]. Disponible en: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5500/sw/interfaces/7x/b_5500_Interfaces_Config_Guide_Release_7x/configuring_port_channels.pdf
- [32] P. Seaman, “Link aggregation: Next generation of ethernet,” IEEE 802.3ad Task Force, 1999. [En línea]. Disponible en: https://www.ieee802.org/3/ad/public/mar99/seaman_1_0399.pdf
- [33] I. Cisco Systems, “Configuring vrrp,” 2021. [En línea]. Disponible en: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5500/sw/interfaces/7x/b_5500_Interfaces_Config_Guide_Release_7x/configuring_port_channels.pdf
- [34] C. Systems, “Data center infrastructure 2.5 design guide – chapter 6 (access layer design),” Cisco Systems, 2013. [En línea]. Disponible en: https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCInfra_6.html#wp1097523

- [35] P. S. Foundation, “Módulo argparse — facilidades para procesar argumentos de línea de comandos,” Python Software Foundation, 2025. [En línea]. Disponible en: <https://docs.python.org/es/3/library/argparse.html>
- [36] ——, “Módulo re — operaciones con expresiones regulares,” Python Software Foundation, 2025. [En línea]. Disponible en: <https://docs.python.org/es/3.13/library/re.html>
- [37] A. Community, “ansible.utils collection,” Ansible Galaxy, 2025. [En línea]. Disponible en: <https://galaxy.ansible.com/ui/repo/published/ansible/utils/>
- [38] ——, “ansible.netcommon collection,” Ansible Galaxy, 2025. [En línea]. Disponible en: <https://galaxy.ansible.com/ui/repo/published/ansible/netcommon/>
- [39] ——, “cisco.ios collection — ansible documentation,” Ansible Documentation, 2025. [En línea]. Disponible en: <https://docs.ansible.com/ansible/latest/collections/cisco/ios/index.html>
- [40] Oracle, “Oci terraform for beginners — introduction,” Oracle Cloud Infrastructure, 2025. [En línea]. Disponible en: <https://docs.oracle.com/en/learn/oci-terraform-for-beginners/index.html#introduction>
- [41] T. N. Project, “iptables — administration tool for ipv4 packet filtering and nat,” netfilter.org, 2025. [En línea]. Disponible en: <https://netfilter.org/projects/iptables/>
- [42] InfluxData, “Telegraf documentation,” InfluxData, 2025. [En línea]. Disponible en: <https://docs.influxdata.com/telegraf>
- [43] E. Foundation, “paho-mqtt — python mqtt client library,” Eclipse IoT, 2025. [En línea]. Disponible en: <https://pypi.org/project/paho-mqtt/>
- [44] InfluxData, “influxdb-client-python — official influxdb 2.x client library,” InfluxData, 2025. [En línea]. Disponible en: <https://github.com/influxdata/influxdb-client-python>
- [45] Anonymous, “Are ansible modules for cisco ios not compatible with ios 12?” Reddit, 2023. [En línea]. Disponible en: https://www.reddit.com/r/ansible/comments/11gazjo/are_ansible_modules_for_cisco_ios_not_compatible
- [46] a. Contributors, “Issue #906: Duplicate “configure terminal” invocation in ios_config module,” 2025. [En línea]. Disponible en: <https://github.com/ansible-collections/cisco.ios/issues/906>