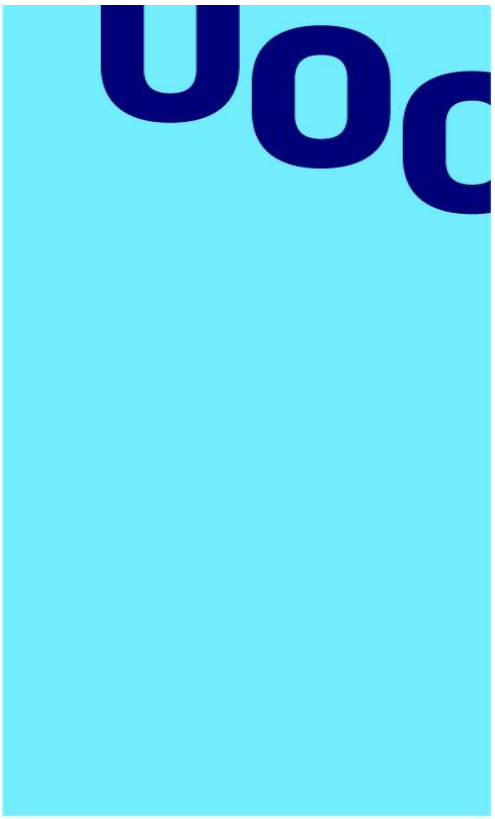
Desarrollo de un servicio API REST para simplificar la administración de redes multi-vendor y multi-entorno

Administración de redes y sistemas operativos



Universitat Oberta de Catalunya

David Torrejón Vázquez

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Miguel Martín Mateo

Consultor

Título del trabajo:	Desarrollo de un servicio API REST para simplificar la administración de redes multi-vendor y multi-entorno
Nombre del autor:	David Torrejón Vázquez
Nombre del consultor:	Miguel Martín Mateo
Fecha de entrega (mm/aaaa):	01/2024
Área del trabajo final:	Administración de redes y sistemas operativos
Titulación:	Ingeniería de Tecnologías y Servicios de Telecomunicación

RESUMEN DEL TRABAJO

Actualmente nos encontramos ante un boom tecnológico que está cambiando la forma en la cual se despliegan y administran las redes telemáticas, debido a la evolución hacia la automatización de redes, así como a la migración hacia la nube de todo tipo de servicios y sistemas, dando lugar a entornos híbridos (cloud y on-premise), donde nos encontramos con elementos de red tanto físicos como virtuales de distintos fabricantes.

Esta revolución, trae consigo la aparición de todo tipo de herramientas y metodologías de trabajo, que están cambiado el rol del ingeniero de redes tradicional denominado *ClickOps*, hacia la figura del ingeniero *NetDevOps*, el cual requiere de un alto grado de conocimientos técnicos y aprendizaje continuo.

Para simplificar este escenario y agilizar las tareas de administración de las redes actuales, este proyecto tiene como objetivo desarrollar un prototipo de una API REST en Python que pueda ser consumida a través de Internet, permitiendo gestionar equipos de red independientemente del fabricante y entorno (cloud y on-premise).

ABSTRACT

We are currently facing a technological boom, changing the way in how networks are being deployed and managed, due to evolution towards network automation, as well as all kinds of services and systems migration to the cloud, giving rise to hybrid environments (cloud and on-premise), where we find both physical and virtual network elements from different vendors.

This revolution brings the spawning of many tools, as well as work methodologies, which have changed the traditional network administrator role aka ClickOps, towards NetDevOps engineer, needing a high technical knowledge degree and continuous learning.

In order to simplify this landscape and ease network tasks, this project aims to develop a REST API prototype in Python programming language, to be consumed through the Internet, and allowing to manage network elements multi-vendor and multi-environment (cloud and on-premise).

ÍNDICE

1. Motivación del proyecto

Actualmente, nos encontramos ante un boom tecnológico que está cambiando la forma en la cual se despliegan y administran las redes telemáticas, debido a la evolución hacia la automatización de redes, así como a la migración hacia la nube de todo tipo de servicios y sistemas, dando lugar a entornos híbridos (cloud y on-premise), donde nos encontramos con elementos de red tanto físicos como virtuales de distintos fabricantes.

Esta revolución, trae consigo la aparición de todo tipo de herramientas y metodologías de trabajo, que están cambiado el rol del ingeniero de redes tradicional denominado ClickOps, hacia la figura del ingeniero NetDevOps, el cual requiere de un alto grado de conocimientos técnicos y aprendizaje continuo.

Para simplificar este escenario y agilizar las tareas de administración de las redes actuales, este proyecto tiene como objetivo desarrollar un prototipo funcional de una API REST en Python que pueda ser consumida a través de Internet, permitiendo gestionar equipos de red independientemente del fabricante, modelo y entorno (cloud y on-premise).

2. Contexto actual y justificación del proyecto

La metodología NetDevOps y el despliegue de infraestructura de red como código (NaC), están revolucionando la forma tradicional de administrar las redes.

Se han desarrollado muchas herramientas dentro del campo, pero también se han generado infinidad librerías de código y APIs, que requieren al nuevo rol de *NetDevOps Engineer* buenos conocimientos de diferentes lenguajes de programación, así como una elevada inversión de tiempo en leer e interpretar la documentación, y desarrollar código.

Una forma de facilitar el trabajo al *NetDevOps Engineer*, es ofrecer un servicio que le permita abstraerse de los trabajos de codificación para interactuar con el equipamiento de red, además de simplificar las tareas de Integración continua y Entrega continua (CI/CD) asociadas a dichos trabajos.

Por ello, nace la motivación de desarrollar una aplicación de código abierto que:

- Permita, no simplemente hacer uso de ciertas librerías, sino que el uso de estas y muchas otras más sea algo transparente para el usuario, el cual envía la misma orden a diferentes equipos de diferentes *vendor* consumiendo un mismo endpoint de la API, simplificando las operaciones.

- Ofrecer dicha aplicación como un SaaS, de tal manera que pueda ser utilizado con facilidad por parte de los usuarios, sin tener que invertir tiempo en despliegue y mantenimiento de infraestructura.

3. Objetivos del proyecto

Los principales objetivos de este proyecto son:

- Simplificar las tareas de administración a los ingenieros NetDevOps, ofreciendo una interfaz común para la gestión de cualquier elemento de red

- Que l@s ingenier@s puedan enfocarse en las tareas de automatización sin tener que invertir tiempo en estudiar librerías, realizar desarrollos y conocer las diferentes herramientas

- Ofrecer esta interfaz a través de una API REST, como un servicio público, donde cada cliente disponga de su propio tenant y pueda gestionar de forma aislada su red

- Facilitar la integración con herramientas de automatización de red, NaC (network as code), CI/CD, CVS, monitorización, backup…

- Engendrar un proyecto colaborativo que pueda llegar a ser un referente en el ámbito de la administración y automatización de redes.

4. Metodología de trabajo

El proyecto a desarrollar, tiene un componente relativamente alto de incertidumbre, lo que hace que el uso de metodologías que están orientadas a proyectos donde el trabajo es más predictivo y definible, no terminan de encajar.

Por ello, para la realización de este proyecto, se va a emplear un enfoque basado en metodologías ágiles, con un ciclo de vida iterativo. Este ciclo de vida permite se enfoca en realizar una entrega única del proyecto, pero a su vez permite obtener feedback continuo del trabajo que se está realizando, minimizando de esta manera riesgos e incertidumbre, y a la vez permitiendo entregar un producto de mayor calidad.

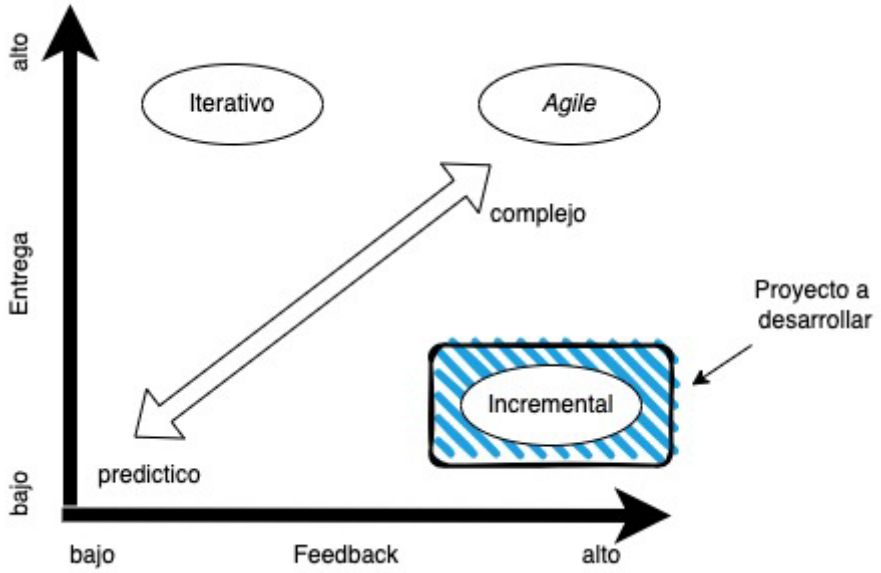


Figura 3.1 Metodologías vs ciclo de vida de un proyecto

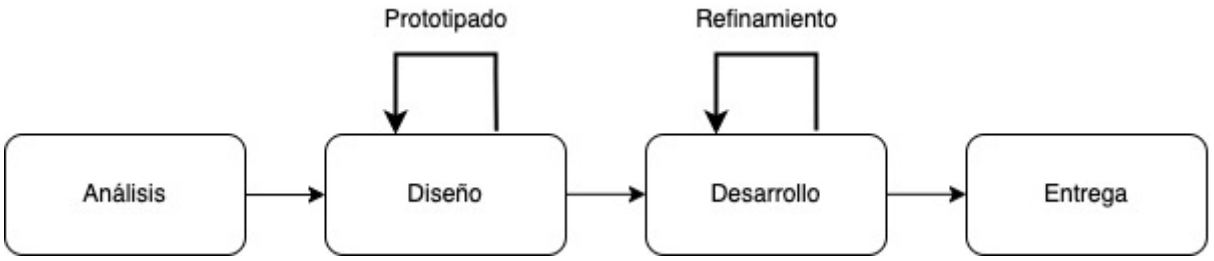


Figura 3.2: Ciclo de vida incremental

En línea con el ciclo de vida que se ha visto que mejor se ajusta a este proyecto, se ha desarrollado la planificación del cronograma, la cual se divide en tres sprints, con las respectivas entregas que conforman el producto. Este se encuentra definido en el apartado número 6, Plan de trabajo.

5. Alcance

Aunque como se ha comentado en el punto anterior, en este proyecto va a desarrollar con un enfoque ágil con ciclo de vida incremental, es necesario disponer de una planificación upfront de alto nivel, que defina el product backlog.

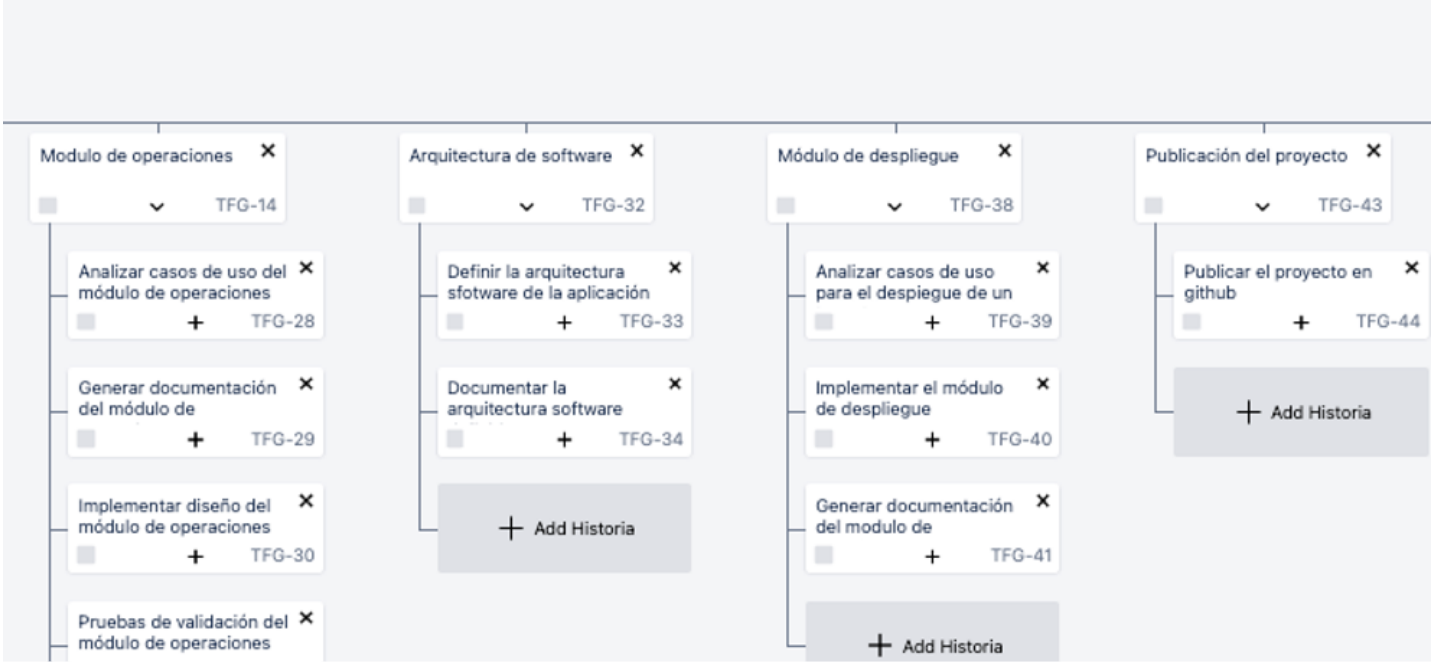
Al tratarse desde un punto de vista predictivo, nos hemos basado en los procesos de planificación de gestión del alcance del PMBOK (Project Management Base of Knowledgment), del PMI.

5.1. Descripción del alcance

El proyecto abarca los siguientes puntos:

- Realizar un análisis de las soluciones existentes en el mercado, que permita definir y desplegar la arquitectura del sistema necesario para la aplicación a desarrollar
- Desplegar la arquitectura definida para la aplicación, así como las dependencias técnicas necesarias (aplicaciones de terceros, módulos, librerías…), que permitan el desarrollo, testeo y puesta en producción de esta.
- Desarrollar una API Rest en Python, que permita utilizar un grupo acotado de funcionalidades de un elemento de red
- Implementar un módulo de gestión de usuarios
- Desarrollar un sistema que permita establecer una comunicación segura, entre una instancia y los elementos de red del cliente.
- Implementar un módulo que permita gestionar la conectividad con los elementos de red (altas, bajas, modificaciones…), de un cliente u organización
- Desarrollar un módulo de envío de comandos, que permita ejecutar órdenes específicas de una marca/modelo
- Publicar en Github el código correspondiente a la API, con el ánimo de crear una comunidad que permita evolucionar el proyecto, integrando más elementos de red y funcionalidades.

5.2. EDT



5.3. Diccionario de la EDT

[TFG-2] Análisis de la arquitectura de sistemas	
Estado:	Tareas por hacer

Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Analizar, definir y dibujar la arquitectura de sistemas, haciendo uso del modelo C4 y UML.

[TFG-3] Despliegue de la arquitectura de sistemas definida	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Implementar el despliegue de la arquitectura definida

[TFG-4] Instalación de software y aplicaciones de terceros	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Llevar a cabo la instalación del software de terceros necesario en cada uno de los elementos que componen la arquitectura definida.

[TFG-8] Desplegar equipamiento para realizar pruebas	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Implementar el laboratorio, preferiblemente sobre la plataforma de virtualización GNS3

[TFG-9] Test conectividad Google Cloud <>OpenVPN Cloud<>Red de cliente	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

[TFG-15] Analizar casos de uso del módulo de comunicación	
Estado:	Tareas por hacer

Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint PEC 3		

Descripción

Realizar un análisis de los casos de uso para desarrollar el módulo de comunicación, así como los *endpoints* de la API.

Este debe de permitir establecer una comunicación segura entre el servicio correspondiente al *tenant* de cliente y la red o redes de cliente a gestionar.

[TFG-16] Pruebas End to End del módulo de comunicación	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint PEC 3		

Descripción

Realizar las pruebas de validación para asegurar el correcto funcionamiento, y obtener capturas e información como evidencias.

[TFG-17] Definir y diseñar el Laboratorio de equipos de red	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Definir el laboratorio de pruebas, necesario para simular una red de cliente

[TFG-19] Implementar diseño del módulo de comunicación	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint PEC 3		

Descripción

Realizar el desarrollo en Python, haciendo uso también de todas las tecnologías necesarias y descritas en TFG-15

[TFG-20] Analizar casos de uso del módulo de gestión de equipos de red	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint PEC 3		

Descripción

Realizar un análisis de los casos de uso para desarrollar el módulo de equipos de red, así como los *endpoints* de la API

[TFG-21] Generar documentación del módulo de gestión de equipos de red	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint PEC 3		

[TFG-22] Implementar diseño del módulo de gestión de equipos de red	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint PEC 3		

Descripción

Realizar el desarrollo en Python, siguiendo el análisis realizado en TFG-20

[TFG-24] Analizar casos de uso del módulo de usuario	
Estado:	Tareas por hacer

Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint PEC 3		

Descripción

Realizar un análisis de los casos de uso para desarrollar el módulo de usuarios, así como los *endpoints* de la API, y el sistema de autenticación de estos.

[TFG-26] Implementar el módulo de usuarios

Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint PEC 3		

Descripción

Realizar el desarrollo en Python, siguiendo el análisis realizado en TFG-24

[TFG-27] Generar documentación del módulo de usuarios

Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon

Sprint:	Sprint PEC 3

Descripción

Actualizar la documentación del sistema siguiendo el modelo C4, así como revisar y actualizar la documentación de la API en *swagger*

[TFG-28] Analizar casos de uso del módulo de operaciones	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint Entrega final		

[TFG-29] Generar documentación del módulo de operaciones	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint Entrega final		

Descripción

Actualizar la documentación del sistema siguiendo el modelo C4, así como revisar y actualizar la documentación de la API en swagger

[TFG-30] Implementar diseño del módulo de operaciones	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint Entrega final		

Descripción

Realizar el desarrollo en Python, siguiendo el análisis realizado en TFG-28

[TFG-31] Pruebas de validación del módulo de operaciones	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint Entrega final		

Descripción

Realizar una batería de pruebas que permitan validar el correcto funcionamiento de las operaciones implementadas.

Para ello se debe hacer uso del laboratorio de red de cliente, previamente creado en TFG-7

[TFG-33] Definir la arquitectura software de la aplicación	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Realizar un análisis de las posibles arquitecturas de software, y ver cual es la que mejor se adapta a las necesidades presentes y futuras.

Se ha de tener en cuenta que debe permitir añadir nuevos equipos de red y funcionalidades de forma rápida y sencilla

[TFG-34] Documentar la arquitectura software definida	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Actualizar documento de proyecto con la información obtenida y generar mapa de la arquitectura de software.

[TFG-39] Analizar casos de uso para el despliegue de un tenant	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Analizar los casos de uso necesarios, y definir los *endpoints*, que permitan crear un *tenant*, levantando un servicio aislado.

En la respuesta, se debe establecer de qué modo se conecta el tenant del cliente con la red de este. La implementación se realizará posteriormente (TFG-11 To Do Módulo de comunicación entre *cloud* y red cliente).

[TFG-40] Implementar el módulo de despliegue	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Realizar el desarrollo en Python, siguiendo el análisis realizado en TFG-39

[TFG-41] Generar documentación del módulo de despliegue	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	SPRINT PEC 2		

Descripción

Actualizar la documentación del sistema siguiendo el modelo C4, así como revisar y actualizar la documentación de la API en swagger

[TFG-42] Generar documentación del sistema de comunicación	
Estado:	Tareas por hacer
Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint PEC 3		

Descripción

Actualizar la documentación del sistema siguiendo el modelo C4, así como revisar y actualizar la documentación de la API en swagger

[TFG-44] Publicar el proyecto en github	
Estado:	Tareas por hacer

Proyecto:	TFG

Tipo:	Historia	**Prioridad:**	Medium
Informador:	David Torrejon	**Responsable:**	David Torrejon
Sprint:	Sprint Entrega final		

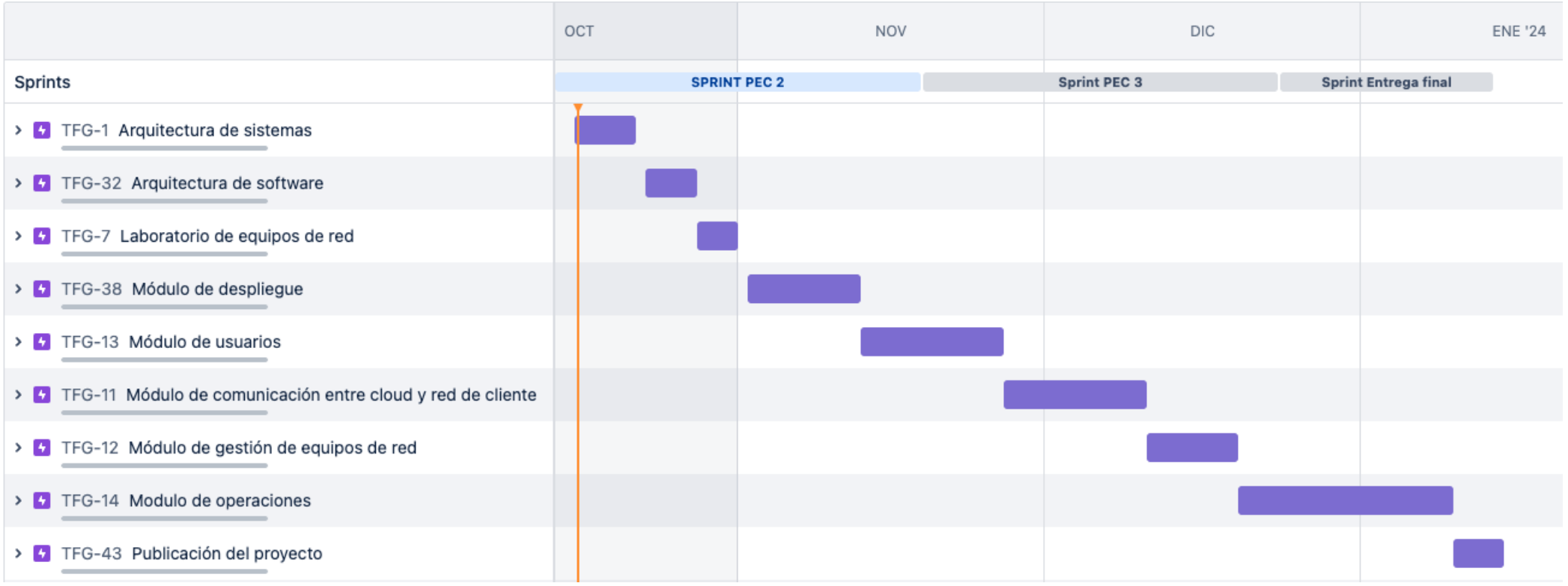
Descripción

Publicar el proyecto en Github con todo el código y documentación necesaria.

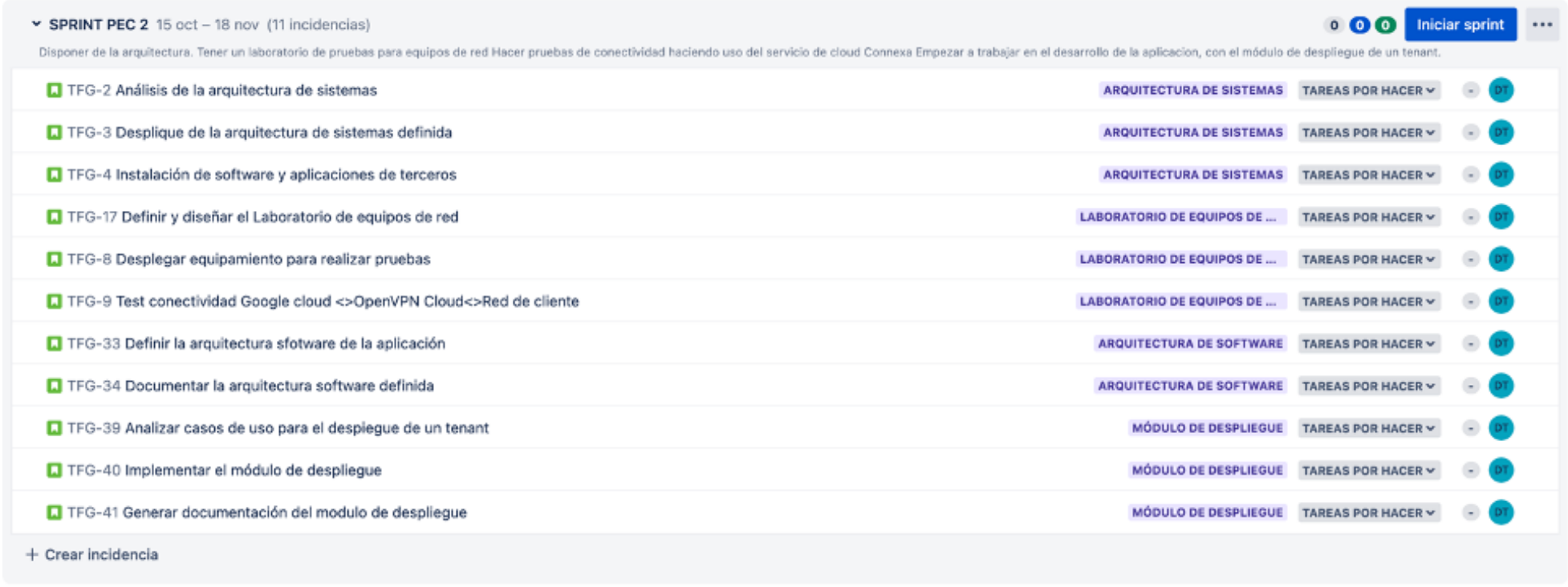
Tabla 5.3.1 Diccionario de la EDT

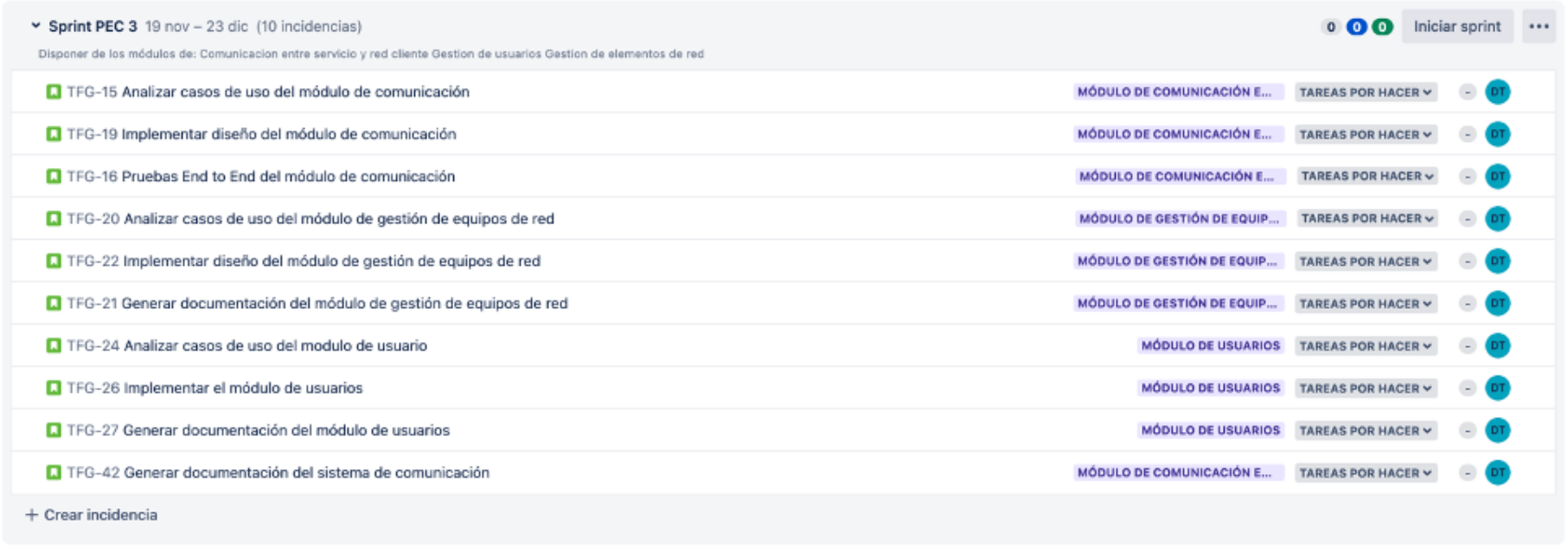
6. Plan de trabajo

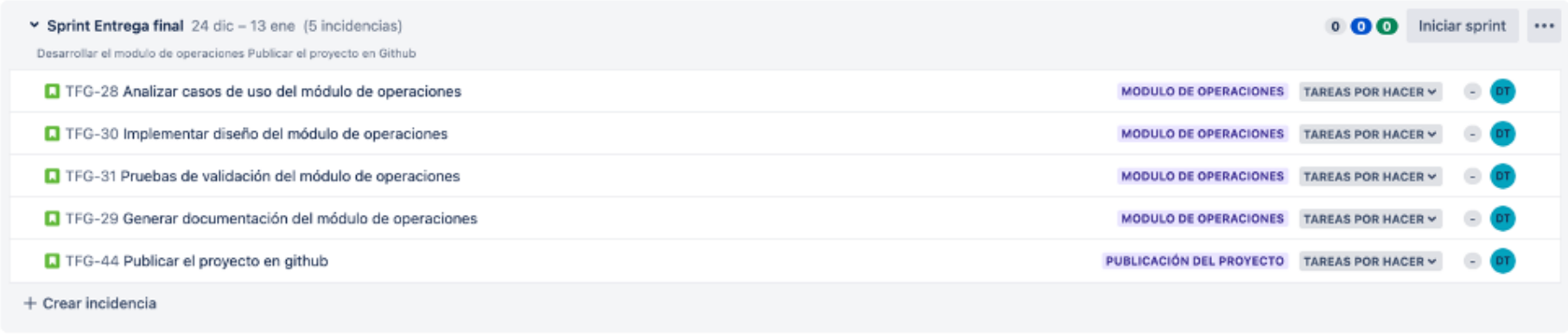
6.1. Roadmap



6.2. Sprint Planning




Sprint PEC 3 19 nov – 23 dic (10 incidencias)
0 0 0
Iniciar sprint
Disponer de los módulos de: Comunicacion entre servicio y red cliente Gestion de usuarios Gestion de elementos de red
TFG-15 Analizar casos de uso del módulo de comunicación
MÓDULO DE COMUNICACIÓN E...
TAREAS POR HACER
TFG-19 Implementar diseño del módulo de comunicación
MÓDULO DE COMUNICACIÓN E...
TAREAS POR HACER
TFG-16 Pruebas End to End del módulo de comunicación
MÓDULO DE COMUNICACIÓN E...
TAREAS POR HACER
TFG-20 Analizar casos de uso del módulo de gestión de equipos de red
MÓDULO DE GESTIÓN DE EQUIP...
TAREAS POR HACER
TFG-22 Implementar diseño del módulo de gestión de equipos de red
MÓDULO DE GESTIÓN DE EQUIP...
TAREAS POR HACER
TFG-21 Generar documentación del módulo de gestión de equipos de red
MÓDULO DE GESTIÓN DE EQUIP...
TAREAS POR HACER
TFG-24 Analizar casos de uso del modulo de usuario
MÓDULO DE USUARIOS
TAREAS POR HACER
TFG-26 Implementar el módulo de usuarios
MÓDULO DE USUARIOS
TAREAS POR HACER
TFG-27 Generar documentación del módulo de usuarios
MÓDULO DE USUARIOS
TAREAS POR HACER
TFG-42 Generar documentación del sistema de comunicación
MÓDULO DE COMUNICACIÓN E...
TAREAS POR HACER
Crear incidencia


Sprint Entrega final 24 dic – 13 ene (5 incidencias)
0 0 0
Iniciar sprint
Desarrollar el modulo de operaciones Publicar el proyecto en Github
TFG-28 Analizar casos de uso del módulo de operaciones
MODULO DE OPERACIONES
TAREAS POR HACER
TFG-30 Implementar diseño del módulo de operaciones
MODULO DE OPERACIONES
TAREAS POR HACER
TFG-31 Pruebas de validación del módulo de operaciones
MODULO DE OPERACIONES
TAREAS POR HACER
TFG-29 Generar documentación del módulo de operaciones
MODULO DE OPERACIONES
TAREAS POR HACER
TFG-44 Publicar el proyecto en github
PUBLICACIÓN DEL PROYECTO
TAREAS POR HACER
Crear incidencia

7. Análisis de riesgos

7.1. Análisis cualitativo

7.1.1. Matriz de probabilidad/impacto

PROBABILIDAD IMPACTO	MUY BAJO	BAJO	MEDIO	ALTO	MUY ALTO
MUY BAJO					
BAJO					
MEDIO					
ALTO					
MUY ALTO					

Tabla 8.1.1.1 Matriz de probabilidad/impacto

7.1.2. Descripción de los riesgos

CÓDIGO	RIESGO	DESCRIPCION	PROB.	IMP.
R-TFG-2.1	Complejidad en el análisis y definición de la arquitectura	Dada la complejidad existente a la hora de definir inicialmente la arquitectura, existe riesgo de no cumpla con las necesidades finales	Medio	Medio
R-TFG-3.1	Problemas de despliegue de sistemas	Debido al grado de incertidumbre del proyecto, es posible que salgan futuros requerimientos durante la ejecución, que requiera de modificaciones	Medio	Bajo
R-TFG-4.1	Incompatibilidad software de terceros	A la hora de realizar la instalación y configuración de programas y servicios de terceros, pueden surgir incompatibilidades	Medio	Medio
R-TFG-8.1	Problemas de compatibilidad en GNS3	GNS3 puede tener problemas de compatibilidad con algún tipo de imagen de equipo de red	Bajo	Bajo
R-TFG-9.1	Problemas validación de conectividad con Cloud Connexa	Cloud Connexa es un servicio con poco tiempo en el mercado	Muy bajo	Medio
R-TFG-9.2	Problemas con el uso de la API de Cloud Connexa	La API que ofrece actualmente Cloud Connexa está en fase beta	Bajo	Alto
R-TFG-29.1	Complejidad en el desarrollo del módulo de comunicaciones	Problemas de automatización del proceso, para configurar una VPN de nivel 3 entre el servidor	Medio	Alto

		que presta el servicio de API y la red a gestionar del cliente		
R-TFG-28.1	Complejidad en el análisis y definición de los endpoints de la API para elementos de red	El diseño de los endpoints y los objetos JSON de intercambio, pueden dar lugar a problemas de escalabilidad, falta de información o fallos de configuración	Medio	My Alto

Taba 8.1.2.1 Riesgos del proyecto

7.2. Plan de contingencia

CÓDIGO	TIPO RESPUESTA	RESPUESTA
R-TFG-2.1	Mitigar	Revisar en cada implementación necesidades reales y futuras para poder ir realizando las posibles adaptaciones de forma ordenada y escalonada
R-TFG-3.1	Corregir	Realizar un análisis de posibles alternativas
R-TFG-4.1	Corregir	Realizar un análisis de posibles alternativas
R-TFG-8.1	Aceptar	Valorar posibles alternativas de equipos a integrar en la API.
R-TFG-9.1	Corregir	Buscar proveedores de servicios alternativos
R-TFG-9.2	Corregir	Buscar proveedores de servicios alternativos
R-TFG-29.1	Mitigar	Servicios alternativos que permitan a automatización
R-TFG-28.1	Mitigar	Realizar un versionado que permita realizar cambios y mantener versiones antiguas funcionando

Tabla 8.2.1 Respuesta a los riesgos del proyecto

8. Elección de las principales aplicaciones, servicios, y herramientas de trabajo

A continuación, se describen brevemente las principales herramientas, aplicaciones y servicios seleccionados para llevar a cabo el proyecto.

Software y Aplicaciones de Terceros

Ansible https://www.ansible.com/. Herramienta de automatización de TI que permite gestionar configuraciones, aplicaciones y sistemas de manera eficiente, que emplearemos para automatizar tareas de configuración y despliegue.

Terraform https://www.terraform.io/. Herramienta IaC para gestionar y provisionar infraestructura en la nube.

Cisco IOSv https://gns3.com/cisco-iosv. Imagen para simular equipos de red Cisco, concretamente switches de capa 2 y capa 3, compatible con GNS3.

Pfsense https://www.pfsense.org/. Solución de Firewall virtual basada en FreeBSD, y compatible con GNS3

Linux Ubuntu https://ubuntu.com/download/server. Distribución gratuita de sistema operativo linux, que se emplea para el despliegue de servidores.

Python 3.11 https://www.python.org/ . Lenguaje de programación ampliamente usado en desarrollo backend, automatización y scripting, entre otros.

FastAPI https://fastapi.tiangolo.com/es/. Framework moderno que permite desarrollar APIs con Python, que tienen una amplia aceptación y una gran comunidad en Internet.

Servicios Cloud y plataformas de virtualización

Google Cloud Platform https://console.cloud.google.com/. Proporciona implementación y gestión de servicios cloud, como máquinas virtuales, seguridad…

OpenVpn Cloud Connexa https://openvpn.net/cloud-vpn/. Solución para implementar site-to-site VPN basadas en cloud

MongoDB Atlas https://www.mongodb.com/atlas/database. Servicio cloud de base de datos que proporciona escalabilidad y alta disponibilidad.

GNS3 https://www.gns3.com/. Plataforma de simulación de redes que permite emular equipos para pruebas y desarrollo, antes de implementarlas en un entorno de producción.

Vmware Fusion https://www.vmware.com/es/products/fusion.html Software para virtualización de sistemas en PC, que permite ejecutar máquinas virtuales con diferentes SSOO.

GitHub https://github.com/. Servicio que permite principalmente, el control de versiones de código, así como publicar y compartir proyectos software.

Virtio https://docs.oasis-open.org/virtio/virtio/v1.2/virtio-v1.2.html. Sistema de virtualización *Open Source* que permite gestionar máquinas virtuales sobre entornos linux

Docker https://www.docker.com/. Plataforma de contenedores que permite el despliegue de aplicaciones aisladas y encapsuladas en contenedores

Herramientas de trabajo

JetBrains Pycharm https://www.jetbrains.com/es-es/pycharm/ IDE específico para Python que ofrece características avanzadas y plugins para el uso de herramientas de terceros como *Mermaid* o *JSON Crack*

Jira cloud https://www.atlassian.com/es/software/jira. Herramienta de gestión de proyectos y seguimiento que se ofrece en modelo SaaS.

Mermaid Diagramming and Charting tool https://mermaid.js.org/. Herramienta enfocada en la documentación como código (DaC), que permite crear diagramas y gráficos a partir de código.

Typora https://typora.io/. Editor Markdown que permite escribir documentación como código (DaC), también compatible con mermaid.

PostMan https://www.postman.com/. Herramienta de testeo, prueba y documentación de APIs

Robo3T https://robomongo.org/. Aplicación similar a MongoDB Compass, pero con un interfaz mas intuitiva y herramientas gráficas para manejo de las bases de datos.

MongoDB Compass https://www.mongodb.com/try/download/compass. Herramienta oficial de MongoDB para manejo de las bases de datos.

9. Diseño de la arquitectura propuesta

9.1. Arquitectura de sistemas

La arquitectura está compuesta principalmente por tres sistemas.

El sistema de despliegue, se encarga de provisionar el servicio de forma aislada e independiente para cada uno de los clientes. En cuanto al sistema de Tenants, este se encarga de proporcionar el servicio de API a cada uno de los clientes. El último sistema, es la propia infraestructura de cliente, la cual está formada principalmente por el conector generado y entregado por el sistema de despliegue, y la propia red a gestionar del cliente.

En el mapa de contexto del anexo de diagramas, se puede observar de forma visual estos sistemas, y las principales interacciones entre estos.

Respecto al despliegue de la arquitectura, se ha decido por un aparte, emplear servicios de computación en la nube, para el despliegue de los servidores que va a alojar la aplicación, y por otra el uso de varios servicios de terceros que se detallan a continuación.

Base de Datos. Se ha decido optar por MongoDB Atlas, dado que ofrece una capa gratuita que cumple con los requisitos de la aplicación, y así se evita tener que desplegar y mantener un SGBD propio. Esta se utilizará principalmente para almacenar toda la información relacionada con tenants, usuarios y equipos.

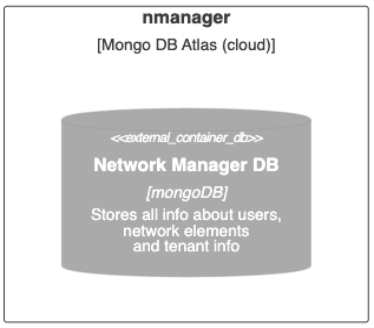


Figura 9.1-1 Network manager DB

Servicio cloud VPN. Para establecer comunicaciones seguras entre las instancias de cliente (*tenants*), y las redes de estos, se ha decidido emplear un servicio cloud denominado Cloud Connexa. Este, dispone de una API la cual nos permite la automatización de la configuración dentro del proceso de despliegue de un *tenant*.

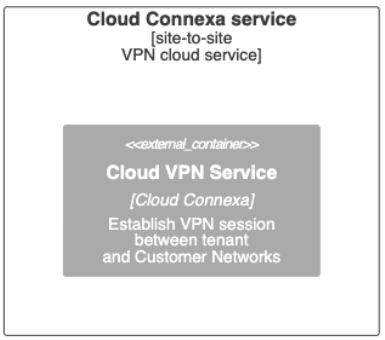


Figura 9.1-2 Cloud VPN Service

Servicio de *email*. Al igual que con la decisión tomada para la base de datos, resulta más sencillo consumir un servicio de mail para cubrir la necesidad de realizar envíos puntuales, que desplegar y mantener un servidor propio.

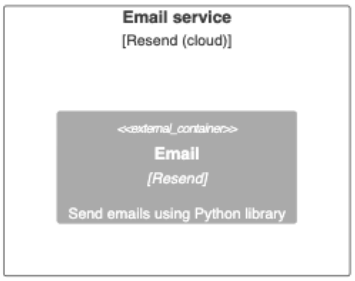


Figura 9.1-3 Email service

Servicios de computación en la nube. De la misma manera, y siguiendo con la línea de emplear servicios cloud, para eliminar trabajos de despliegue y mantenimiento, se ha optado por desplegar la infraestructura de sistemas en Google Cloud Platform. Dentro de esta, se hará uso de los servicios de *Compute Engine*, que permite desplegar VM's (máquinas virtuales), a través de una interfaz Web, una API o empleando Terraform.

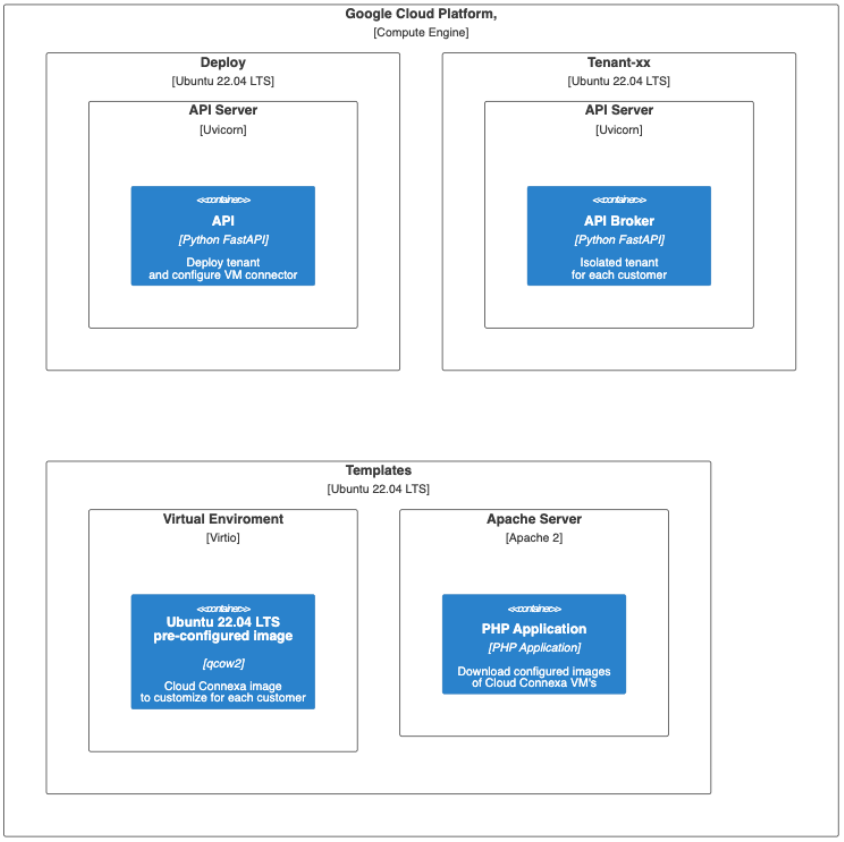


Figura 9.1-4 Despliegue de infraestructura en Google Cloud Platform

En el diagrama de despliegue del anexo de diagramas, se dispone del mapa completo.

9.1.1. Alta de servicios, instalación de software y otras dependencias

Para implementar la arquitectura diseñada, se requiere de la configuración de servicios de terceros. A continuación, se describen todos aquellos que forman parte del diseño de la arquitectura.

MongoDB Atlas

Se trata de un servicio de base de datos no relacional en la nube, del cual se requiere disponer de una url de acceso y unas credenciales para poder acceder. También existe la posibilidad de emplear una API REST, pero se considera más factible emplear la librería Pymongo de Python.

A continuación se muestra el detalle de cómo establecer una conexión.

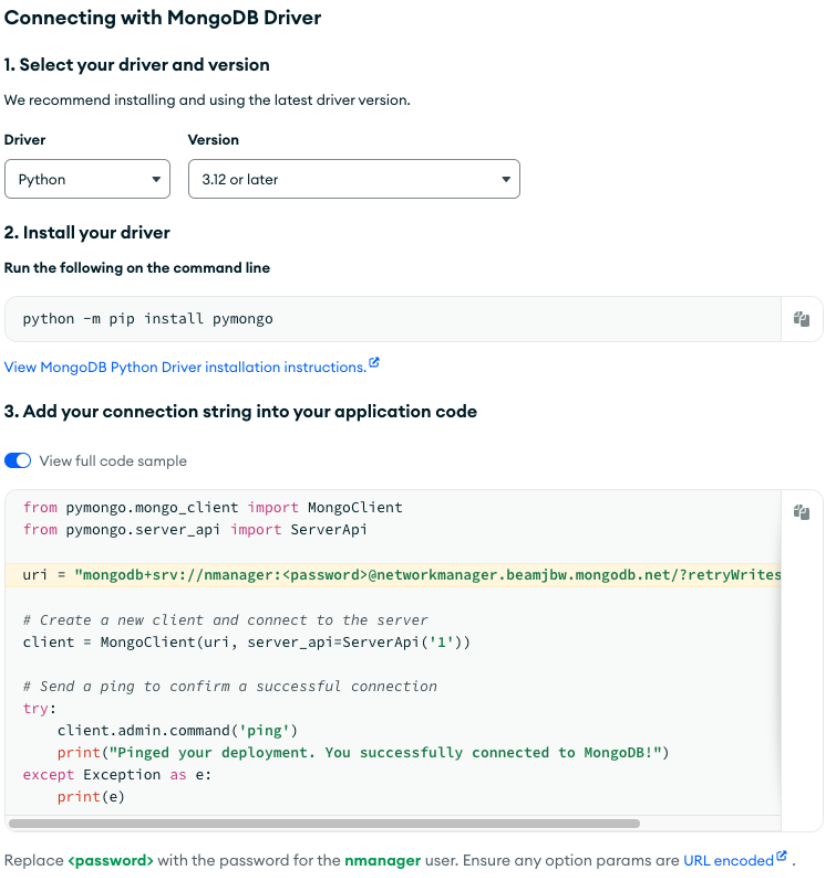


Figura 9.1.1-1 Configuración Pymongo para acceso a MongoDB Atlas

Se dispone del anexo de Configuración del servicio MongoDB Atlas, donde se detalla cómo darse de alta en el servicio.

OpenVNP Cloud Connexa

Para disponer del servicio de cloud connexa, simplemente se ha de realizar el registro en la siguiente Web: https://myaccount.openvpn.com/signin/cvpn/domain-select

Se selecciona un nombre de dominio

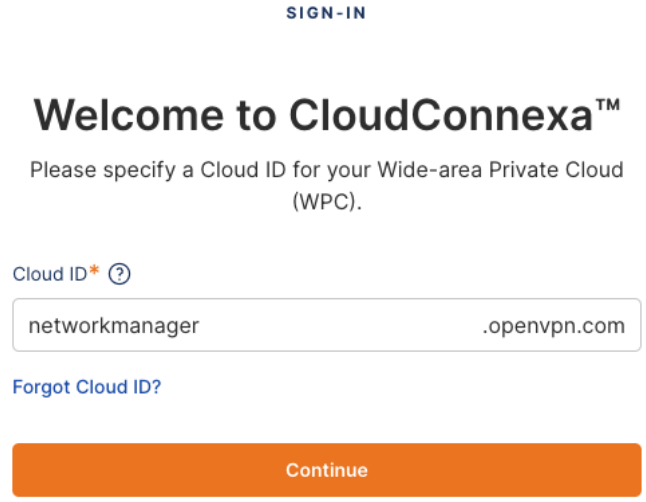


Figura 9.1.1-2 Registro subdominio en Cloud Connexa

Se introducen las credenciales de acceso

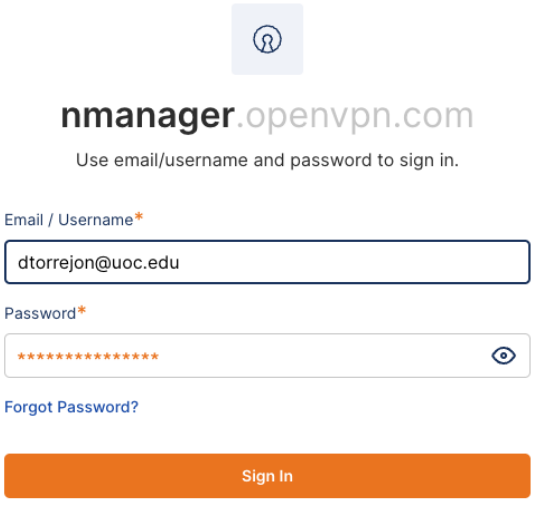


Figura 9.1.1-3 Registro usuario en Cloud Connexa

En el menú se selecciona la opción API, y se generan unas credenciales que permitan el uso de la misma.

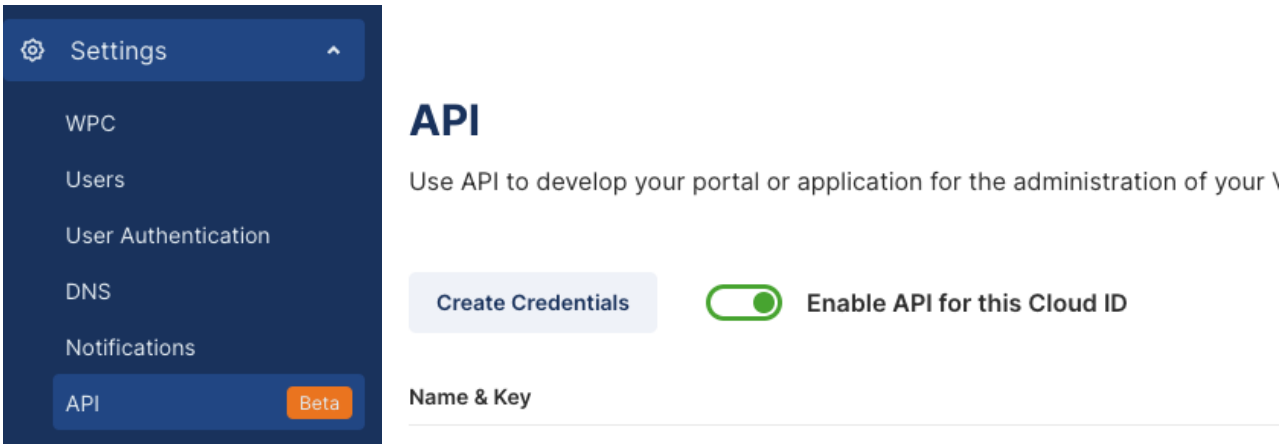


Figura 9.1.1-4 Crear credenciales para uso de la API en Cloud Connexa

La configuración de este servicio se realizará desde el sistema de despliegue, en función de la información proporcionada por cada cliente a la hora de crear el servicio. Se explicará en detalle en el apartado 12.1. Módulo de despliegue.

La documentación de la API esta disponible en el siguiente enlace: https://openvpn.net/cloud-docs/developer/cloudconnexa-api.html

Servicio de Email (RESEND)

Para disponer del servicio de mail, simplemente debe aplicarse el registro en la siguiente Web: https://resend.com/signup

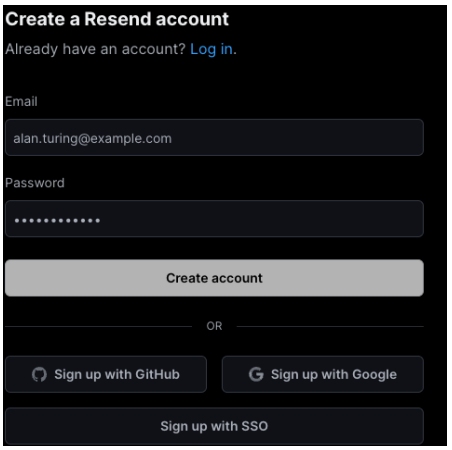


Figura 9.1.1-5 Registro servicio email *Resend*

Se crea un equipo

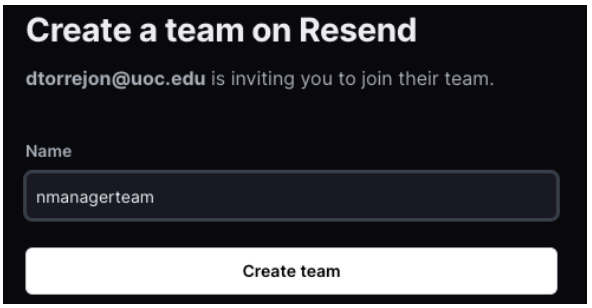


Figura 9.1.1-6 Creación equipo en servicio email Resend

Se selecciona la opción “Add an API Key” y se obtiene una plantilla para el envío de emails en python

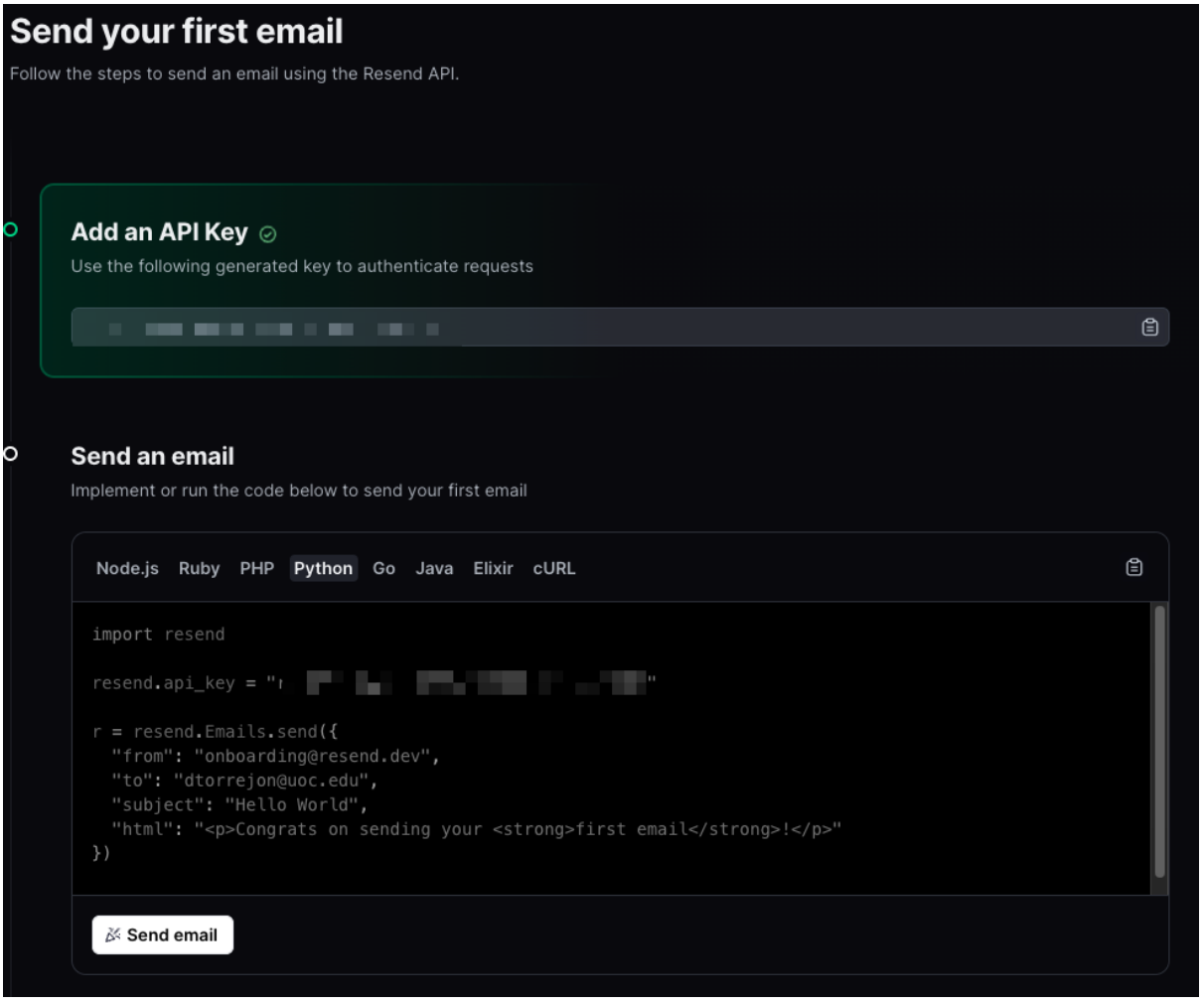


Figura 9.1.1-7 Plantilla en python para uso del servicio email Resend

Servicio de computación en la nube

Para el despliegue de los servidores necesarios en la nube se ha elegido Google Cloud Platform, ya que los servicios a utilizar de compute engine ofrecen unos precios ligeramente más competitivos que Azure o EC2 para los requerimientos de cpu, disco, y RAM.

Para registrarse, simplemente hay que validarse con una cuenta de gmail.

Lo primero a realizar, es añadir una clave RSA pública, que permita desde un servidor u otro pc que disponga de clave privada, poder conectarse directamente por SSH.

Para ello vamos a la opción "Metadatos" dentro de "Compute Engine", se selecciona la opción de editar, y se añade la clave pública.

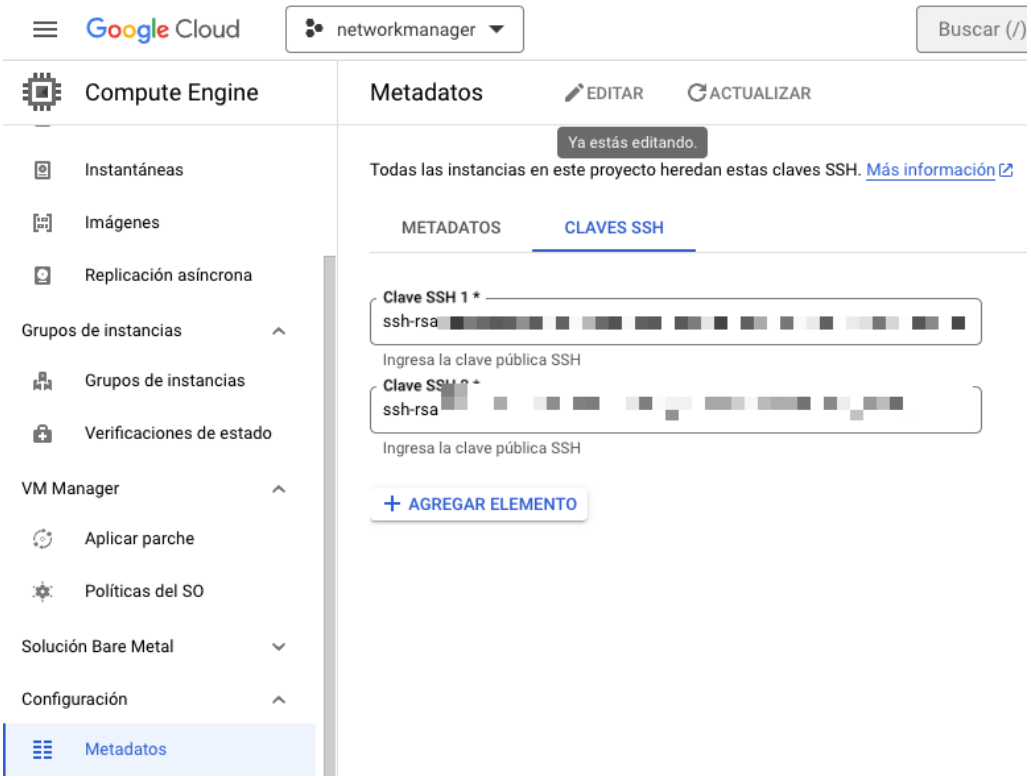


Figura 9.1.1-8 Asignar clave pública RSA en GCP

A continuación, se despliegan y configuran los servidores *Deploy* y *Templates*. La misión de *Deploy*, es configurar el servicio para establecer la VPN con la infraestructura del cliente, levantar una instancia dedicada con el servicio de API, mientras que la de *Templates*, es la de generar las VM's que servirán a los clientes como conectores para levantar las correspondientes VPNs con las instancias que albergan el API Broker.

En esta imagen se puede observar estos servidores, y una instancia correspondiente a un *tenant* de cliente, denominada fstech.

Figura 9.1.1-9 Instancias en GCP

Para desplegar los servidores, se puede hacer a través de la interfaz Web, la consola de Google, o una API proporcionada.

En este caso, se hará uso de la API junto con una serie de plantillas, que nos permitirá automatizar el despliegue de servidores para aquellos casos que se pueda requerir.

Respecto al detalle de la configuración, se dispone del anexo 21.6 Guia para el despliegue y configuración de los servidores en Google Cloud Platform

9.2. Arquitectura de software

Tal como se puede deducir de los apartados anteriores, este sistema está basado en un estilo arquitectónico orientado a microservicios y a REST.

REST, se basa en un conjunto de principios y restricciones, que permiten identificar de forma única a cada recurso y manipularlo, a través de *url`s*. Dichos recursos se representan a través de objetos representados con estándares como XML o JSON, y además carecen de estado.

En este caso concreto, está basado en el protocolo HTTP y sus métodos GET, POST, PUT, PATCH, y DELETE, por lo que se trata de un estilo arquitectónico RESTful.

Respecto a los microservicios, se trata de dividir la aplicación en una serie de servicios independientes que trabajan de forma autónoma. Esto permite que cada uno pueda ser desarrollado de forma independiente, y que realizar cambios en uno de los servicios no pueda llegar a afectar al resto de la aplicación. La forma de comunicación más común entre servicios es a través de APIs REST. En nuestro caso, todos los servicios se

comunican a través de API RESTful, exceptuando RESEND y MongoDB Atlas, para los cuales se dispone de librerías específicas en Python.

En cuanto al patrón arquitectónico elegido para la implementación de las aplicaciones de *Deploy* y *Tenant*, es una arquitectura hexagonal, también conocida como arquitectura de puertos y adaptadores.

El hecho de elegir este patrón frente a otros como MVC que permiten un desarrollo mucho más rápido, es la independencia entre capas, que permite aislar el core de la aplicación de las implementaciones específicas, para conectar con cualquier servicio. Aunque el tamaño de la aplicación a desarrollar es pequeño, se ha tenido en consideración que encaja muy bien, con la necesidad de generar una implementación específica para equipo de red del cliente. Esto hace que sea fácilmente escalable, y que se pueda desarrollar e integrar cualquier implementación desarrollada por cualquier persona de forma independiente.

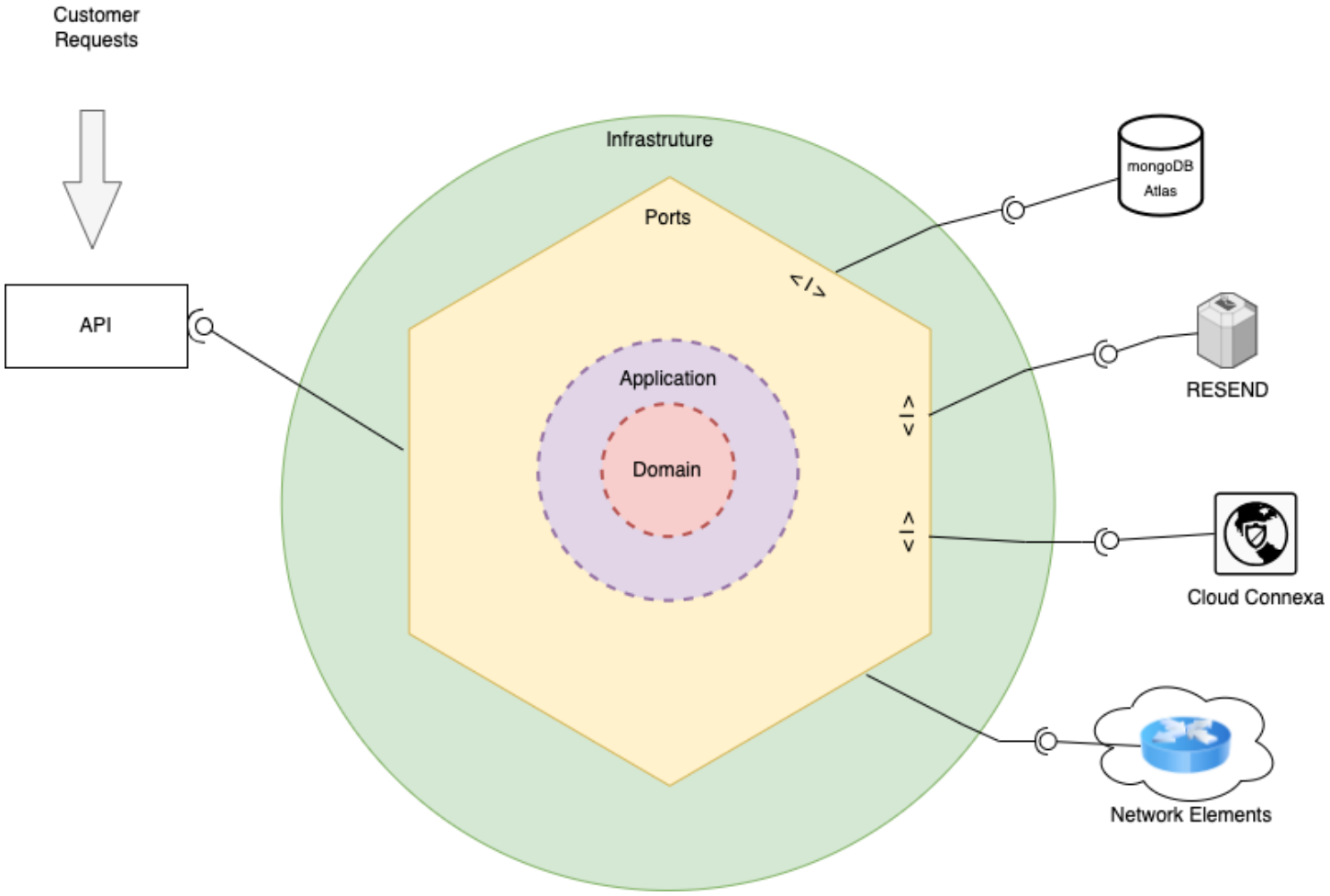


Figura 9.2-1 Arquitectura software de la aplicación

9.2.1. Frameworks de desarrollo

Los frameworks más empleados hoy en día para desarrollo de APIs con Python son *Django* y *Flask*. Pero en los últimos años ha tomado mucha fuerza un nuevo framework denominado *FastAPI*. Este, oferece frente al resto, las siguientes mejoras:

- Mejor rendimiento y velocidad que otros frameworks de Python
- Uso de tipado estático, que ayuda a reducir errores y problemas de seguridad
- Documentación automática de la API con *swagger*
- Validacion sencilla de los esquemas de datos
- Sintaxis clara y sencilla

En el siguiente link se puede consultar la documentación oficial de *FastAPI*. https://fastapi.tiangolo.com/

10. Diseño del laboratorio de red para simulaciones y *test*

Para poder simular una red de cliente, y realizar las pruebas de validación, es necesario disponer de una laboratorio. El hecho de disponer de elementos físicos resulta un problema a nivel de costes, tiempo de implementación de la red, y disponibilidad, para realizar las pruebas pertinentes.

Por ello, se ha optado por implementar un laboratorio virtual, haciendo uso de la herramienta GNS3 y Vmware Fusion.

10.1. Entorno de virtualización

GNS3 es una plataforma de simulación de redes de código abierto, que permite emular redes sin necesidad de un hardware físico específico. Utiliza imágenes de sistemas operativos de cualquier tipo de equipo para recrearlos de manera virtual en un entorno de laboratorio. Los principales elementos que se pueden emular son routers, switches, firewalls… etc

En resumen, las principales características de GNS3 son:

- Emulación de Dispositivos de Red, como ya se ha comentado anteriormente.

- Ofrece una interfaz gráfica para diseñar y configurar topologías de red y equipos.

- Uso de imágenes de sistemas operativos reales, permitiendo emular el comportamiento de los elementos de red.

VMware Fusion es un software de virtualización para PC's Mac, que permite ejecutar todo tipo de sistemas operativos, como Windows, Linux y otros. En este caso, se emplea para ejecutar la máquina virtual encargada de gestionar los equipos de red creados en el laboratorio de cliente en GNS3.

10.2. Instalación de software, imágenes y dependencias.

Para poder comenzar con la configuración del laboratorio, se necesita previamente realizar la instalación del entorno. Estes se compone principalmente de:

- Software GNS3. Se debe instalar el cliente con el entorno gráfico.

- VMware Fusion. Para permitir la ejecución de la máquina virtual que gestione los equipos de red del laboratorio

- Instalar y configurar las plantillas de los equipos de red correspondientes, con las imágenes de sus sistemas operativos.

Para proceder con la instalación del entorno se deben descargar los programas correspondientes a Vmware y GNS3, así como la máquina virtual que vienen ya preconfigurada para el propio entorno de GNS3.

https://www.gns3.com/software/download

https://www.gns3.com/software/download-vm

https://www.vmware.com/es/products/fusion/fusion-evaluation.html

Una vez descargado el software, es importante importar la máquina virtual en Vmware, y configurar este como entorno de virtualización en GNS3.

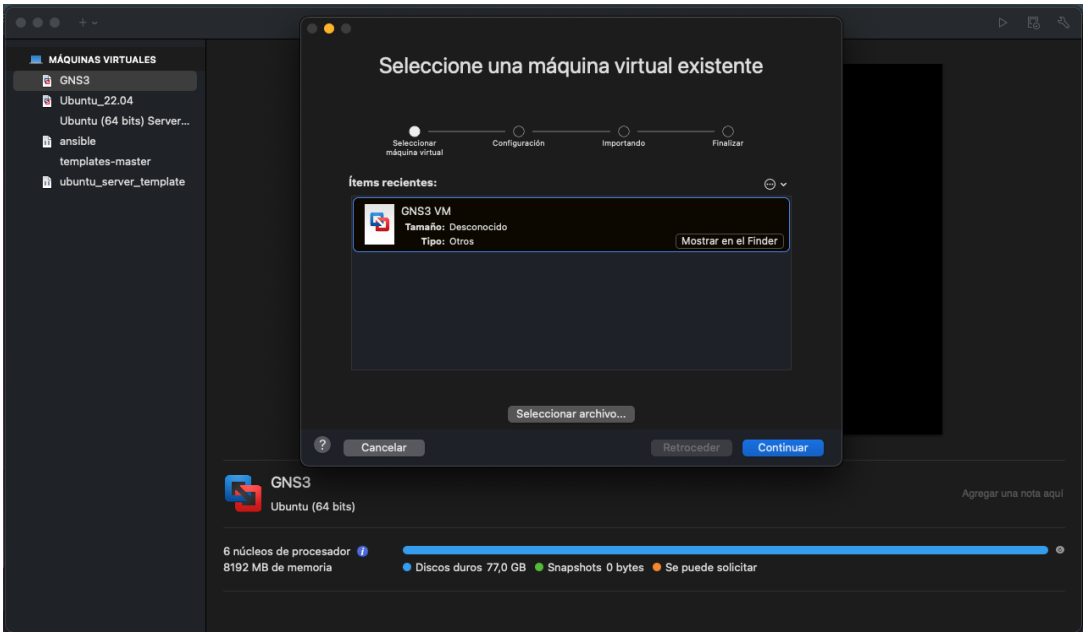


Figura 10.2-1 Importación de imagen VM GNS3 a Vmware

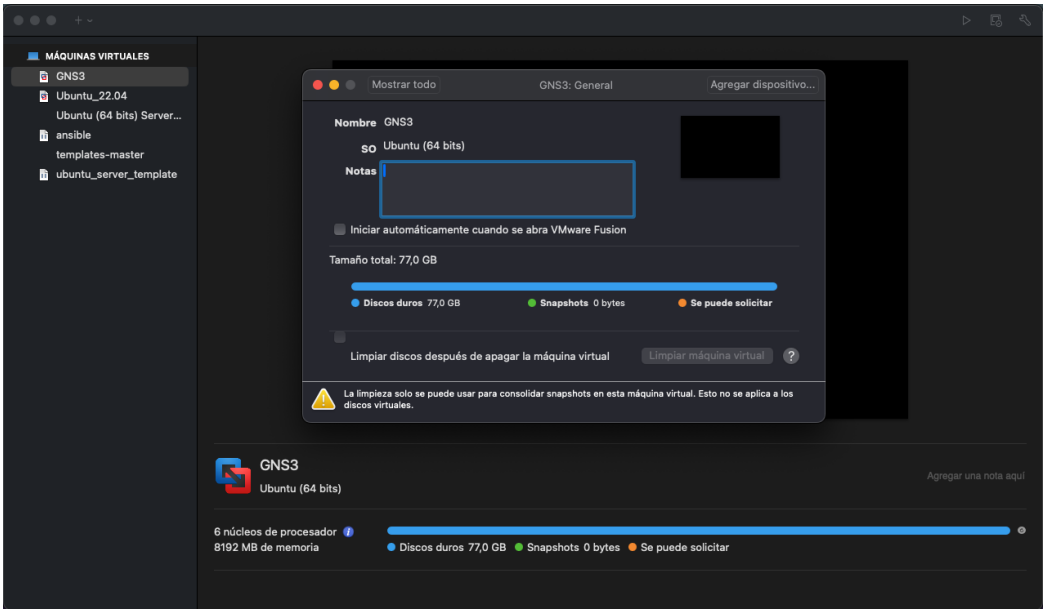


Figura 10.2-2 Información sobre la VM importada a GNS3

Dentro de GNS3, en el menú “GNS3” -> “Preferences”, se deben configurar los campos como se indica a continuación, para usar VMware como motor de virtualización. Las vCPUs y RAM a asignar dependerá de los recursos físicos disponibles del equipamiento donde se ejecute la VM.

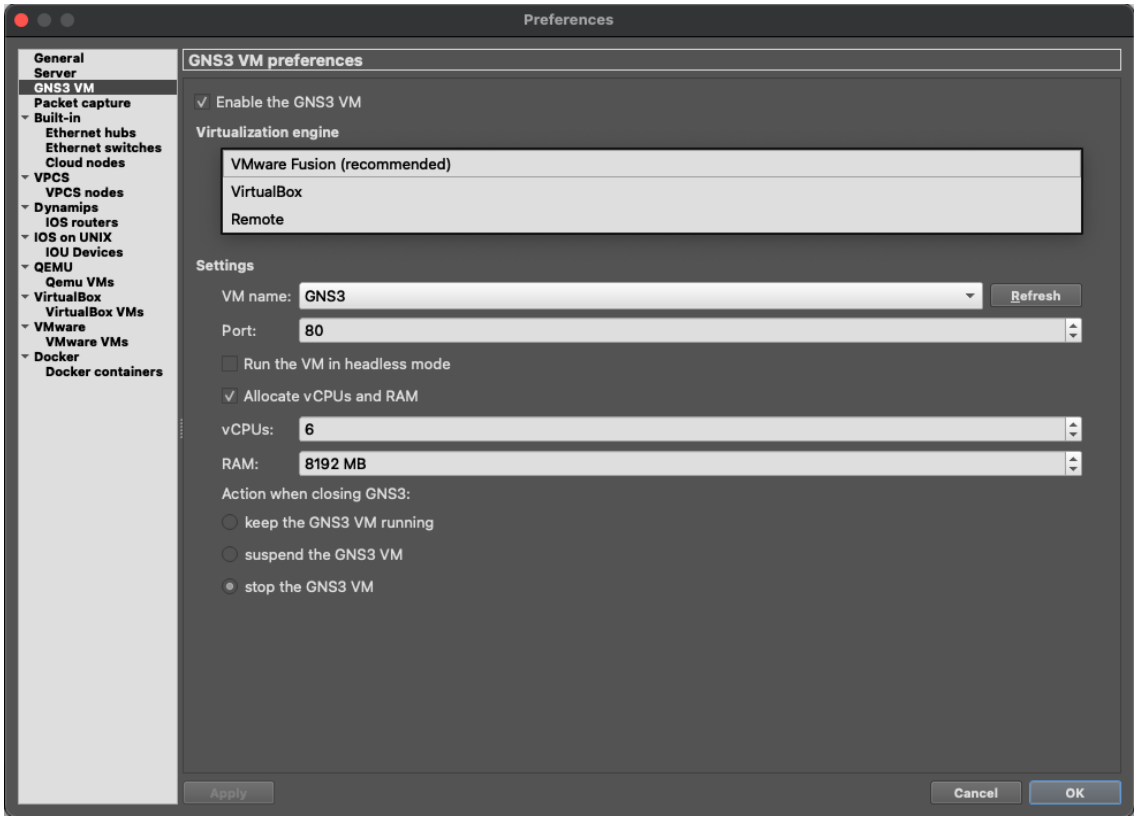


Figura 10.2.-3 Configuración Vmware como motor de virtualización

En el apartado 10.5 Instalación y configuración de equipos en GNS3, se explicará la configuración específica de cada imagen, para cada equipo del laboratorio.

10.3. Elementos que forman el laboratorio

A continuación, se detallan todos los elementos que forman el laboratorio de la imagen.

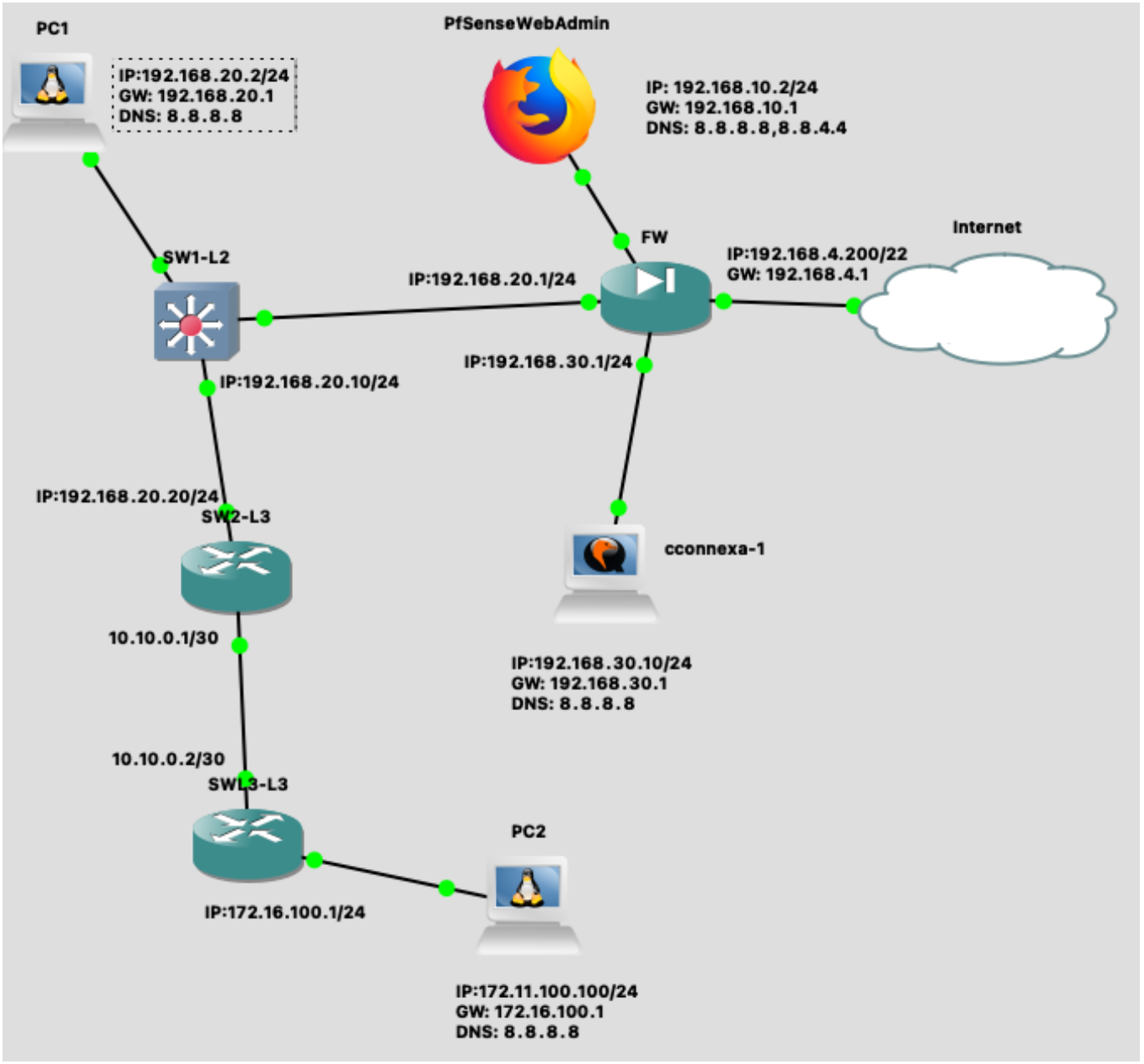


Figura 10.3-1 laboratorio de red cliente en GNS3

Los elementos que componen el laboratorio son:

- **Cloud**. Proporciona acceso a Internet a través de una de las interfaces de red de la VM que hace de motor de virtualización

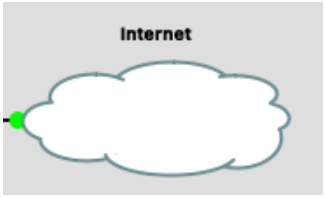


Figura 10.3-2 Acceso a Internet en GNS3

- **FW y PfSenseWebAdmin**. Se trata de una imagen del firewall *Pfsense Community Edition*. Para poder acceder al interfaz web de administración, se ha desplegado un Navegador firefox.

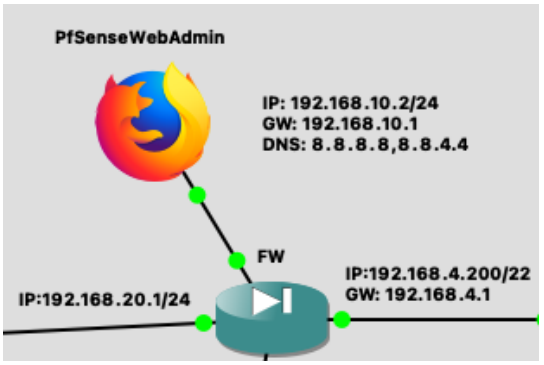


Figura 10.3-3 Firewall y WebAdmin en GNS3

- **PC1 y PC2.** Simulan dispositivos conectados a las diferentes redes para poder validar pruebas de configuración y realizar *tests* conectividad

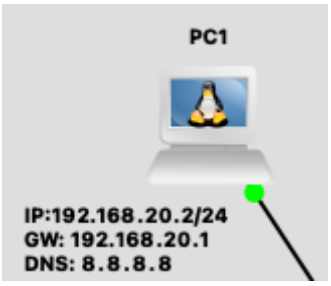


Figura 10.3-4 PCs emulados en GNS3

- **SW1-L2.** Switch con imagen del SO IOSvL2, que emula un switch cisco de capa 2.

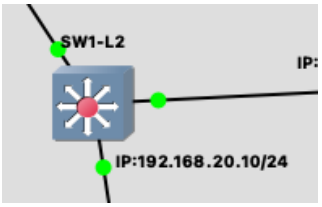


Figura 10.3-5 Switch capa 3 en GNS3

- **SW2-L3 y SW3-L3**. Switches con imagen del Mikrotik RouterOS 7.8 y Cisco IOSvL3, y ambos emulan un switch cisco de capa 3. En el caso de SW2-L3, dispone además de la cli, de una API REST.

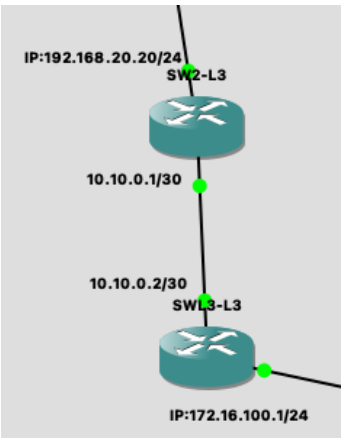


Figura 10.3-6 Switches de capa 3 en GNS3

- **Cconnexa-1**. VM que hace de conector con la instancia de cliente que se ejecuta en google cloud. proporciona una comunicación segura entre los dos puntos.

Figura 10.3-7 Conector Cloud Connexa desplegado en GNS3

10.4. Topología de la red

Dado que el objetivo del laboratorio es poder realizar pruebas, no se ha definido una topología de red con una arquitectura concreta, sino que se ha optado por desplegar un switch para delimitar la red externa de la interna, un switch de capa 2 para hacer las validaciones de los diferentes endpoints a implementar en la API, y 2 switches de capa 3 que permitan validar configuraciones de protocolos de capa 3.

Con estos elementos, añadiendo los PCs, y las interconexiones realizadas, tal como se muestra en la figura 10.3-1, se dispone de todos los elementos necesarios para validar la API a implementar.

10.5. Instalación y configuración de equipos en GNS3

A continuación, se pasa a detallar cómo se realiza la instalación y configuración de los elementos que no se encuentran disponibles de forma predeterminada en GNS3, y que debe realizarse aparte.

Firewall

En GNS3, se selecciona la opción “New Template”.

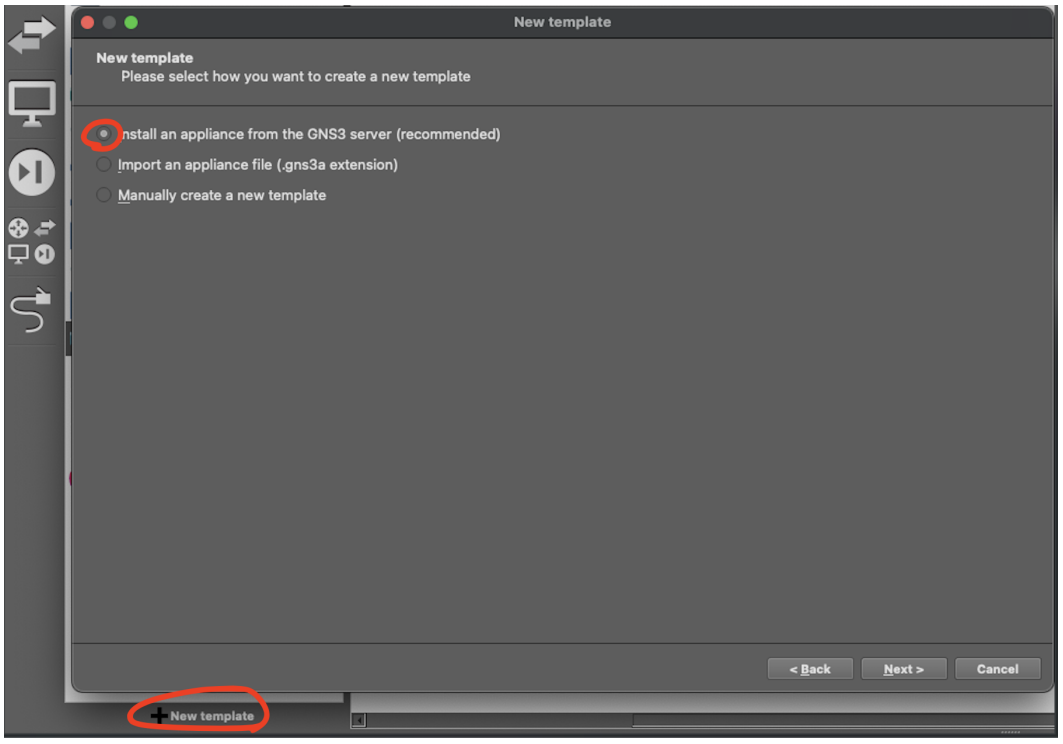


Figura 10.5-1 CRear nuevo template en GNS3

Seleccionar la opción de pfSense

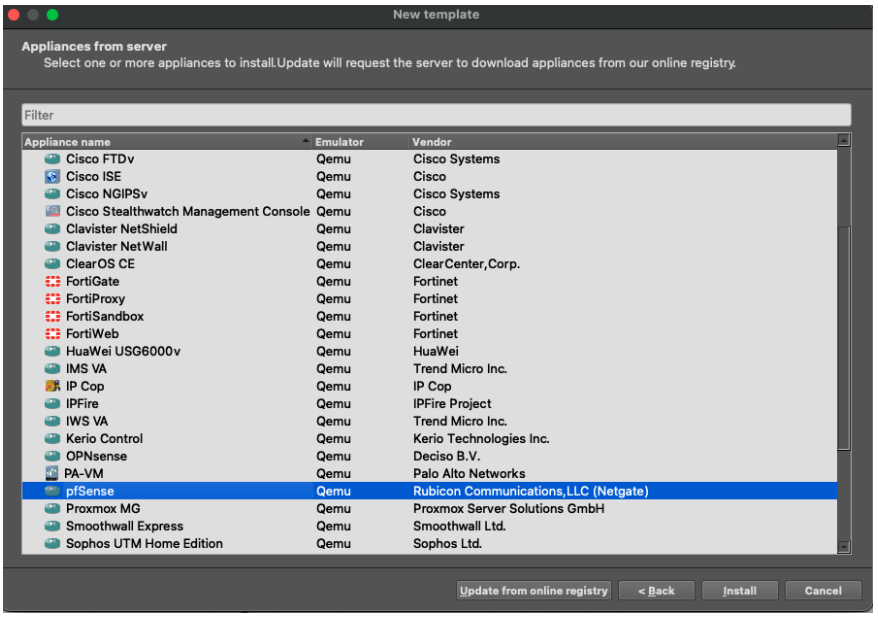


Figura 10.5-2 Instalar templete en GNS3

Se importa la imagen previamente descargada de la Web de [PfSense]

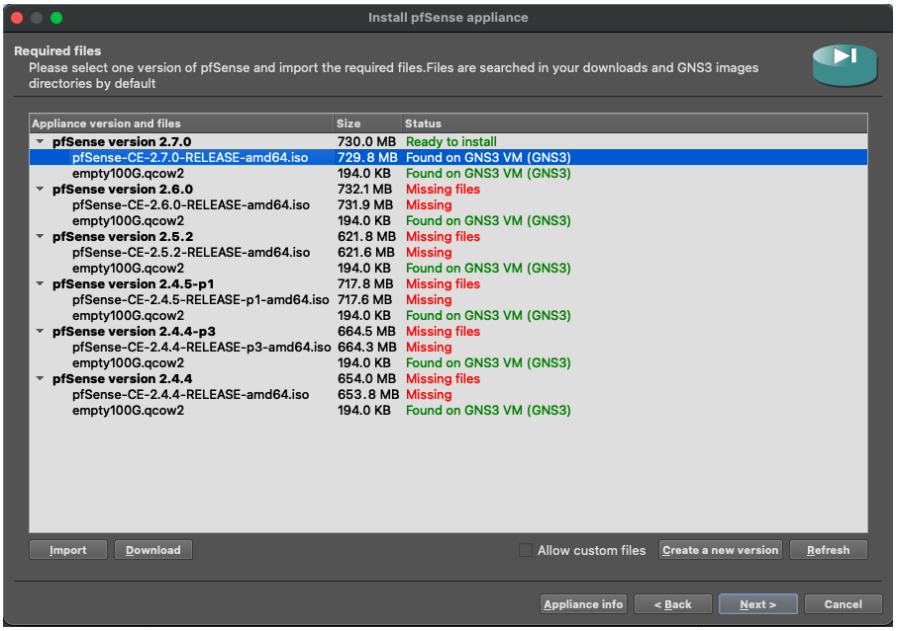


Figura 10.5-3 Importar imagen en GNS3

Ya se dispone de un template que permite instanciar firewalls con el SO

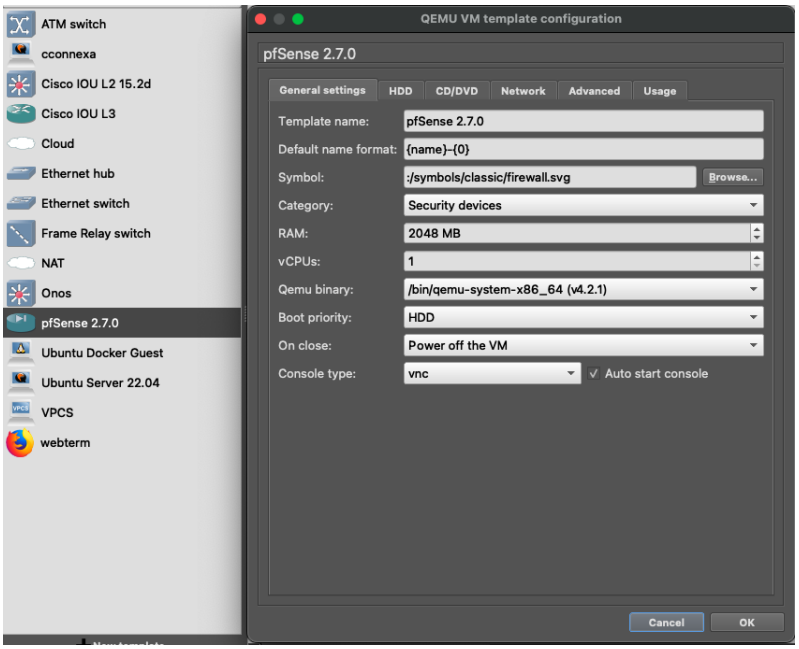


Figura 10.5-4 Propiedades tempate pfSense en GNS3

Una vez que se dispone del template, se crea un firewall, se inicializa y se configuran las IPs de los interfaces, según corresponda en cada caso. Para el laboratorio en cuestión, se configuran las siguientes IPs e interfaces.

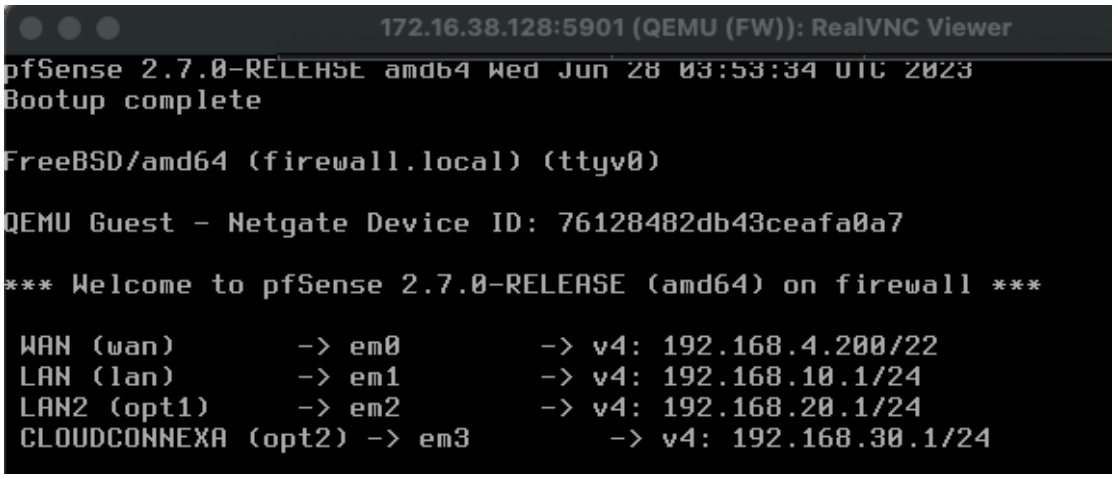


Figura 10.5-5 Interfaces configuradas en firewall PfSense

Switches

Para crear los templates con las imágenes de IOsvL2, L3 y RouterOS, se ha seguido el mismo procedimiento explicado anteriormente para GNS3.

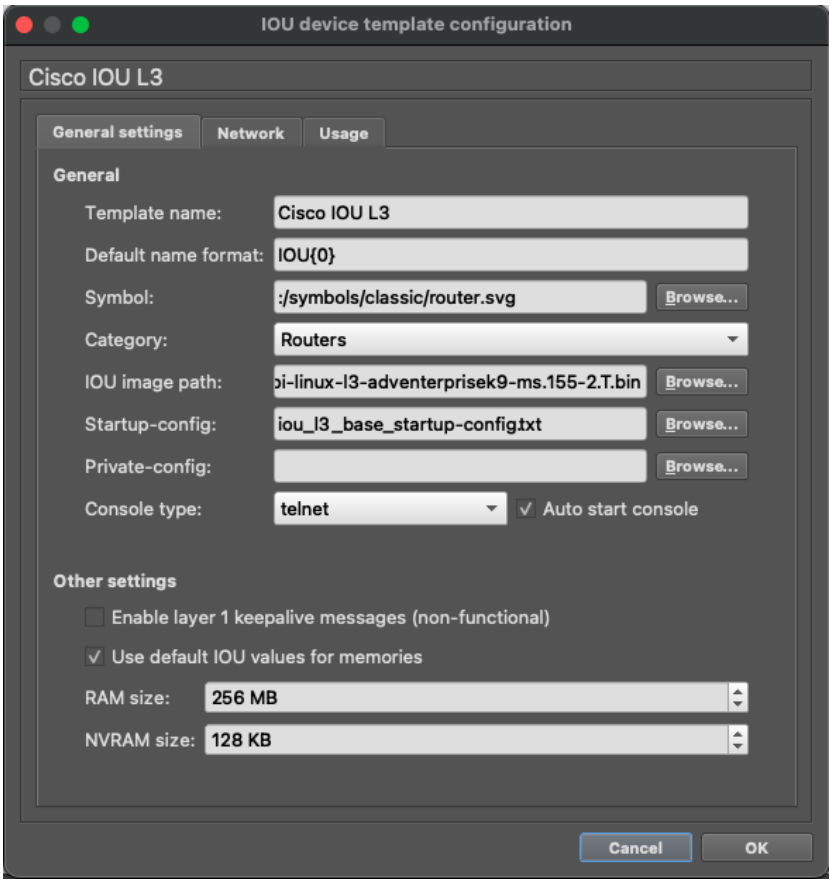


Figura 10.5-6 Configuración template Cisco IOSv

Respecto a la configuración inicial, se detalla a modo de ejemplo, la aplicada para el switch de capa 2 (SW1-2)

```
#console line configuration
enable
configure terminal
line console 0
password uoc
login
loggin synchronous
```

```
exec-timeout 60 0
exit
#vty line configuration
line vty 0 4
password uoc
login
loggin synchronous
exec-timeout 60 0
exit
#hostname configuration
hostname sw1
exit
#VLAN 1 interface configuration
configure terminal
interface vlan 1
ip address 192.168.20.10 255.255.255.0
description sw2-uoc-lab
no shutdown
ip default-gateway 192.168.20.1
exit
#ssh configuration
config t
username uoc password uoc
enable secret labuoc
service password-encryption
ip domain-name uoc.local
crypto key generate rsa
2048
ip ssh version 2
line vty 0 4
transport input ssh
login local
# configurar ruta por defecto para verse con otras subredes
ip route 0.0.0.0 0.0.0.0 192.168.20.1
```

Conector Cloud Connexa

En lo que respecta al conector, este puede ser descargado del servidor de ficheros que almacena las imágenes generadas en formato -vmdk y .qcow2

Esta imagen, debe ser importada y configurada en GNS3, para usar con el entorno de virtualización Qemu.

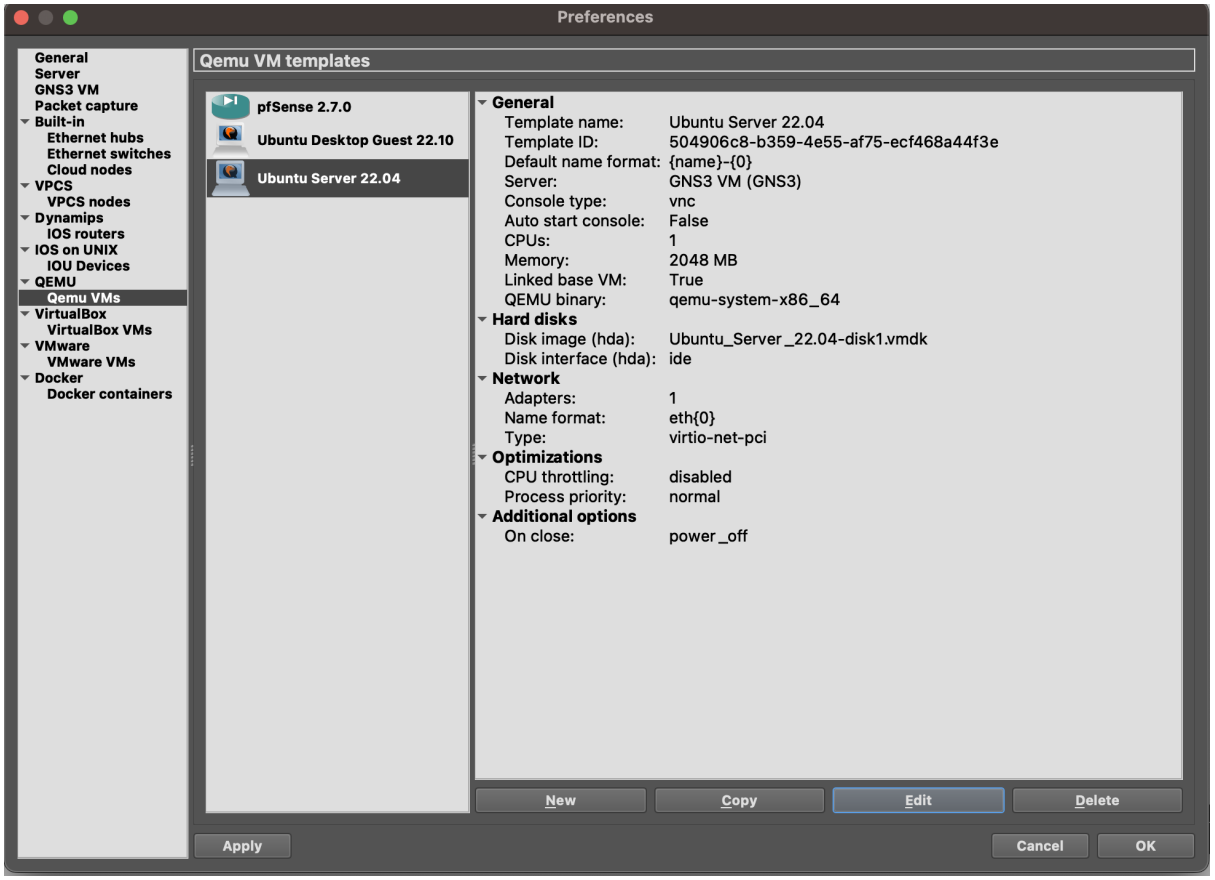


Figura 10.5-7 Configuracion template VM Ubuntu Server 22-04 LTS en GNS3

Dentro del anexo 21.3.2. Instalación de servidor Ubuntu, se explica entre otras, los paso a seguir para importar una máquina virtual en GNS3.

Dado que la VM generada por el servidor Deploy ya viene preconfigurada con los parámetros facilitados en a la hora de solicitar la creación del servicio, simplemente hay que conectarla a algún otro dispositivo con salida a Internet y desde donde disponga de visibilidad con las redes de gestión.

En el caso del laboratorio de GNS3, se conectará al puerto físico 3 del firewall.

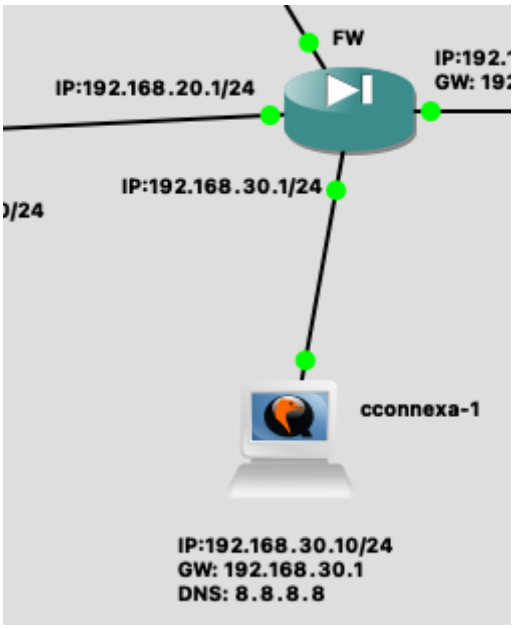


Figura 10.5-8 Conexionado entre VM conector Cloud Connexa y el Firewall

10.6. Pruebas de conectividad

Es importante validar el proceso a seguir, para realizar la configuración del servicio Cloud Connexa, dado que posteriormente tendremos que integrarlo dentro del servidor Deploy, y así realizar la configuración tanto del lado servidor como cliente, de forma totalmente transparente al usuario.

El primer paso a realizar será realizar una configuración de pruebas, siguiendo la documentación proporcionada por el proveedor del servicio. Esta se encuentra ubicada en https://openvpn.net/cloud-docs/administrator/index.html

Para facilitar la configuración de este, se dispone una guia con un ejemplo de configuración, en el anexo 21.3.3 Instalación y configuración del servicio Cloud Connexa

Una vez configurados ambos extremos siguiendo la guía, se procede a realizar un PING entre servidor y una máquina de la red cliente en GNS3, y así poder validar la conectividad entre ambos puntos.

Antes, se debe revisar que ambos extremos se encuentran online.

Figura 10.6-1 VM creada para test conectividad de Cloud Connexa en lado servidor

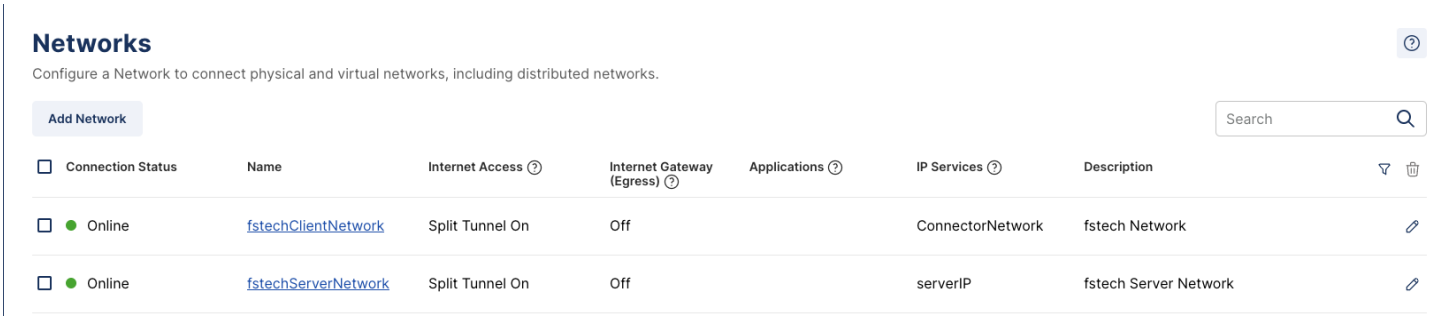


Figura 10.6-2 Estado los conectores en lado cliente y lado servidor

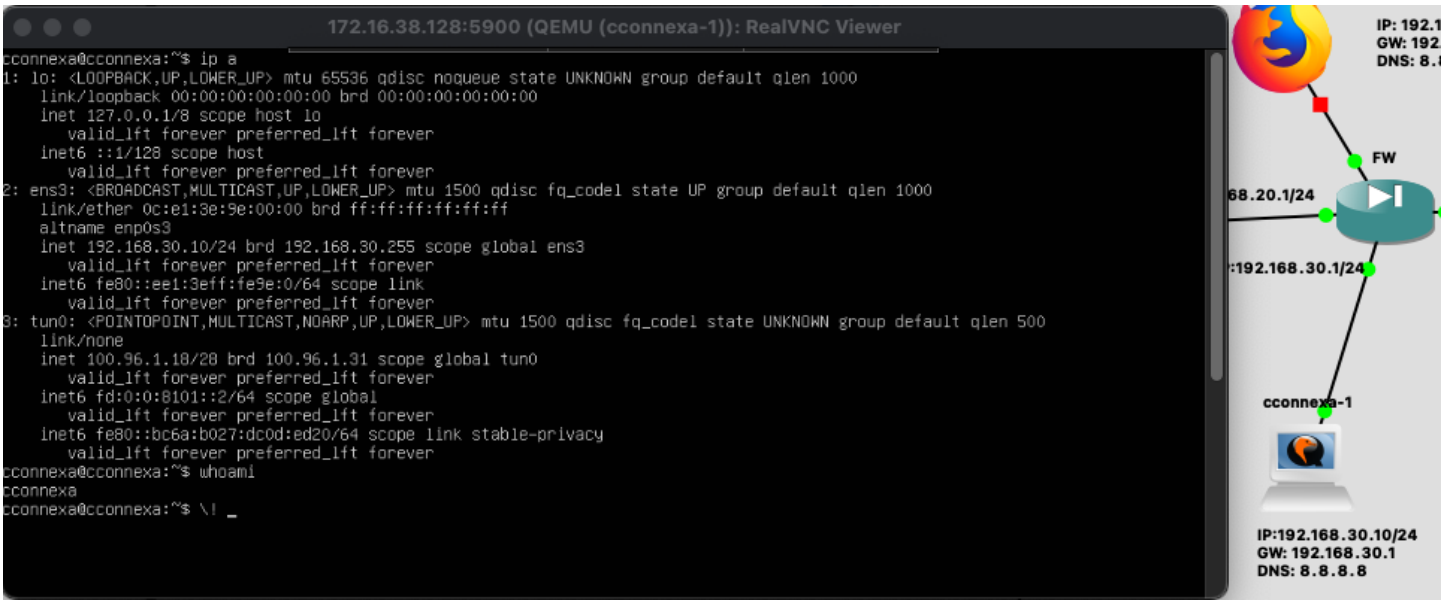
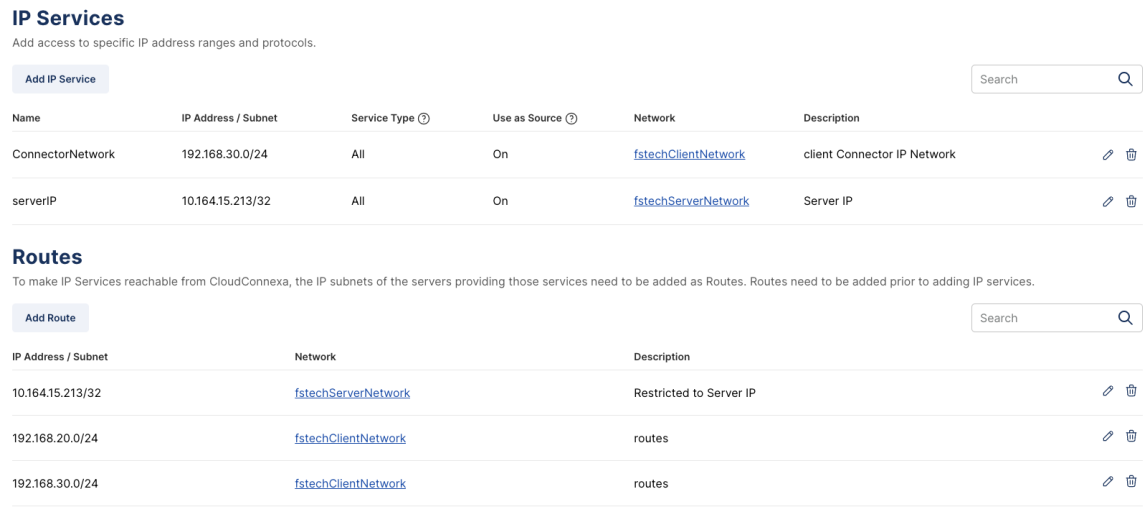


Figura 10.6-3 configuración de red en conector lado cliente

A continuación, se debe revisar que la configuración de las rutas y servicios es la correcta



IP Services

Add access to specific IP address ranges and protocols.

Add IP Service | Search

Name	IP Address / Subnet	Service Type	Use as Source	Network	Description
ConnectorNetwork	192.168.30.0/24	All	On	fstechClientNetwork	client Connector IP Network
serverIP	10.164.15.213/32	All	On	fstechServerNetwork	Server IP

Routes

To make IP Services reachable from CloudConnexa, the IP subnets of the servers providing those services need to be added as Routes. Routes need to be added prior to adding IP services.

Add Route | Search

IP Address / Subnet	Network	Description
10.164.15.213/32	fstechServerNetwork	Restricted to Server IP
192.168.20.0/24	fstechClientNetwork	routes
192.168.30.0/24	fstechClientNetwork	routes

Figura 10.6-4 Configuración de las rutas e IPs de servicios en Cloud Connexa

Finalmente, realizamos un PING, a un equipo de la red de cliente, desde el servidor.

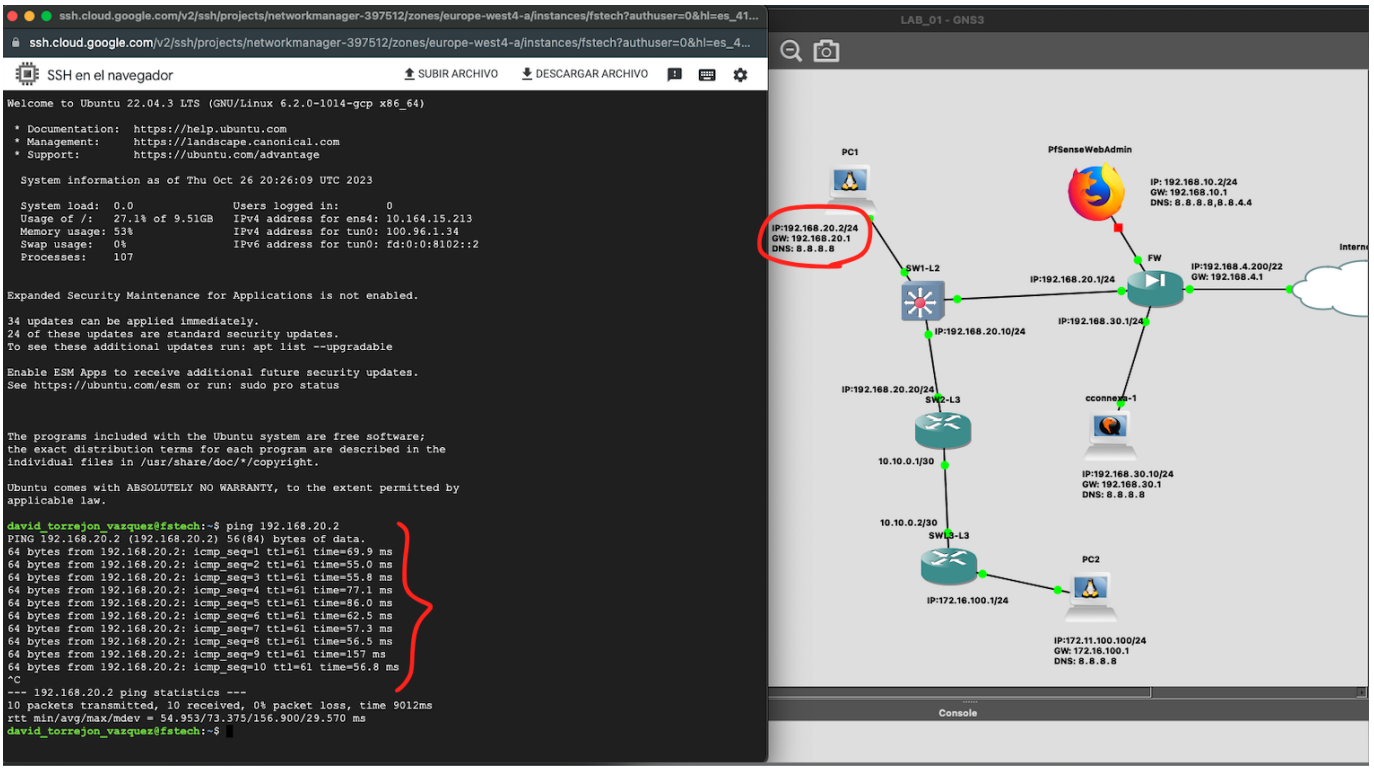


Figura 10.6-5 Test ping desde servidor a un equipo de la red de cliente

En lo que respecta al proceso de configuración a través de la API de Cloud Connexa, se describen a continuación los mensajes a enviar para configurar el servicio, con el objetivo de confirmar el correcto funcionamiento de la API de Cloud Connexa.

El primer mensaje a enviar será la solicitud de un Token para validarse.

```
curl --location --request
POST'https://dtorrejon.api.openvpn.com/api/beta/oauth/token?client_id=xxx.dtorr
ejon&client_secret=password' \
--header 'Accept: application/json'
```

La respuesta recibida en formato json es la siguiente:

```
{
    "access_token": "secret_token_kjhfdjsagfjkdhsa",
    "token_type": "bearer",
    "expires_in": 43199,
    "scope": "read write"
}
```

Ahora que ya tenemos el token, se procede a la configuración de una red. Para ello, enviaremos el siguiente JSON con una configuración de prueba:

```
curl --location 'https://dtorrejon.api.openvpn.com/api/beta/networks' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer secret_token_kjhfdjsagfjkdhsa' \
--data '{
  "connectors": [
    {
      "description": "test description",
      "name": "test",
      "vpnRegionId": "es-mad"
    }
  ],
  "description": "testinggg network",
  "egress": true,
```

```
  "internetAccess": "LOCAL",
  "name": "networknametestttttt",
  "routes": [
    {
      "allowEmbeddedIp": false,
      "description": "descriptionIP",
      "value": "1.1.1.1/32"
    }
  ]
}'
```

Obtenemos como respuesta el siguiente JSON, confirmando que se ha configurado correctamente


```
{
    "id": "71b5ac53-f637-4c22-ad29-3752860acc39",
    "name": "networknametestttttt",
    "description": "testinggg network",
    "egress": true,
    "internetAccess": "LOCAL",
    "routes": [
        {
            "id": "73e7cefe-449e-4520-bcdc-0f1e0d1d8f37",
            "type": "IP_V4",
            "subnet": "1.1.1.1/32",
            "description": "descriptionIP"
        }
    ],
    "connectors": [
        {
            "id": "f4f1fdc4-6c4a-438d-9ab1-0c4e0096a345",
            "networkItemId": "71b5ac53-f637-4c22-ad29-3752860acc39",
            "networkItemType": "NETWORK",
            "name": "test",
            "description": "test description",
            "vpnRegionId": "es-mad",
            "ipV4Address": "100.96.1.18/28",
```

```
            "ipV6Address": "fd:0:0:8101::2/64"
        }
    ],
    "systemSubnets": [
        "100.96.1.16/28",
        "fd:0:0:8101::/64"
    ]
}
```


A través del portal Web podemos observar que la red se ha configurado correctamente.

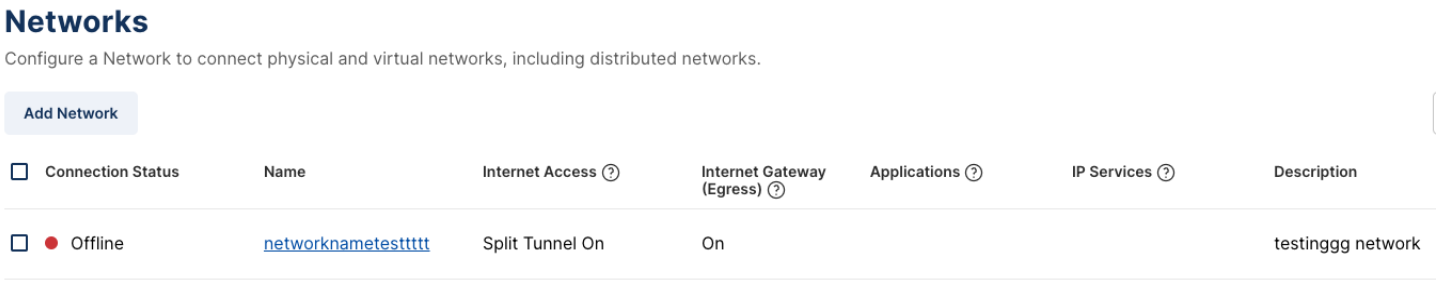


Figura 10.6-6 Creación de una red en Cloud Connexa vía API RESTful

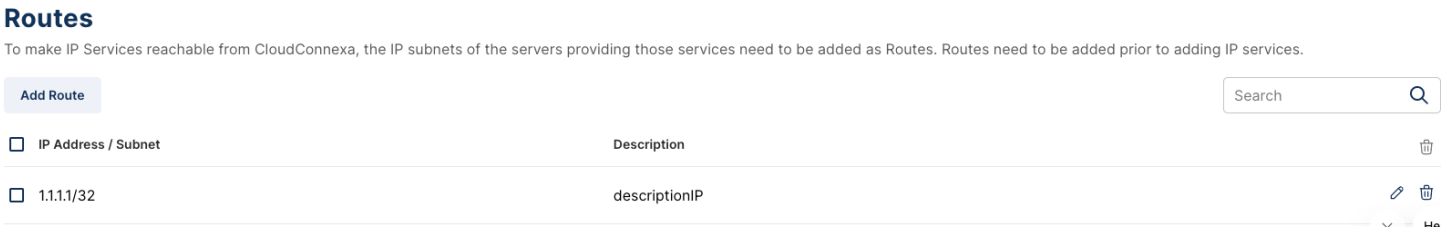


Figura 10.6-7 Creación de rutas y servicios IP en Cloud Connexa vía API RESTful

11. Diseño y desarrollo de la aplicación

11.1. Lenguaje de programación elegido

Python dispone de varias ventajas y características que lo hacen más eficiente que otros lenguajes, para el producto que estamos desarrollando. Además, este lenguaje de programación es líder en automatización de redes, y herramientas tan importantes como Ansible, se han desarrollado con este.

Las principales ventajas que aporta a este proyecto, son:

Legibilidad y Simplicidad. Su sintaxis es simple, el código se asemeja al lenguaje humano, y esto hace que se reduzca la probabilidad de errores y acelera el desarrollo.

Bibliotecas y Frameworks. Python cuenta con varias bibliotecas y frameworks dedicados al desarrollo de APIs que son simples de utilizar. En este caso, como ya se ha comentado anteriormente, usaremos FastAPI.

Multiplataforma. Python es compatible con múltiples sistemas operativos, lo que garantiza que se pueda ejecutar en diferentes plataformas, desde un Serverless como AWS Lambda, hasta una máquina windows.

Soporte. Se dispone de un amplio conjunto de recursos disponibles en línea, lo que facilita el poder encontrar soluciones a problemas comunes y obtener ayuda de forma rápida.

Escalabilidad. Funciona bien, tanto para crear desde pequeñas APIs, hasta grandes sistemas distribuidos de alta complejidad, lo cual está alineado con los objetivos del proyecto.

Seguridad. Dispone de actualizaciones constantes para mantenerse seguro y actualizado.

Librerías Específicas para automatización de redes. Python ofrece librerías específicas para la gestión de dispositivos de red, como Netmiko, Paramiko y NAPALM. Estas bibliotecas simplifican la automatización y la comunicación con dispositivos de diferentes fabricantes.

Integración con otras APIs. Se integra de manera natural con otras APIs Restful, además de disponer de "Requests", una librería muy sencilla de manejar

12. Documentación

12.1. Módulo de despliegue

El objetivo de este módulo, es ofrecer al usuario un endpoint que le permita desplegar un tenant, y conectarse a este de forma transparente.

Para ello, se debe de ejecutar una serie de tareas de forma ordenada que por un lado levante el servicio, y por otro ofrezca conectividad con este.

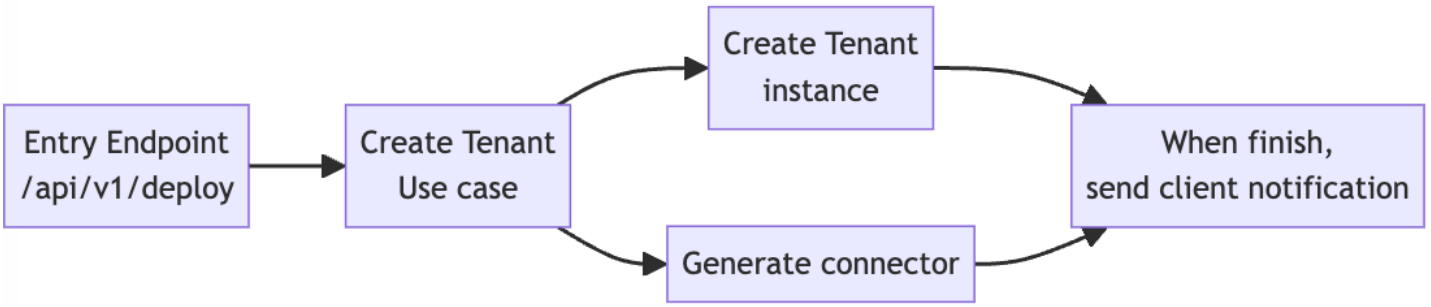


Figura 12.1-1 Diagrama de flujo del módulo de despliegue

Como podemos observar en el diagrama de flujo, se debe exponer un endpoint, el cual se podrá acceder de forma pública, para realizar la petición de crear el servicio.

Esta petición, ejecutará un caso de uso, denominado “Create Tenant", que por un lado generará la instancia que ejecute un servicio aislado para cada cliente, y por otro un método de conexión al servicio.

Una vez esté el servicio completamente creado, se debe enviar una notificación al cliente.

Al emplear arquitectura hexagonal, en el caso de uso, simplemente se encargará de ejecutar las implementaciones concretas que se le pasen como parámetros, lo que facilita agregar futuras. Respecto a la implementación a realizar para este proyecto, las instancias de cliente se levantarán sobre el servicio de “Compute Engine” de *Google Cloud*, como se ha explicado en anteriores apartados, y el conector está basado en el servicio de cloud connexa y una máquina virtual que hace de puerta de enlace en la red de cliente.

Respecto a la implementación concreta del conector, se encuentra detallada en el apartado 12.3 Módulo de comunicación.

En el apartado 21.1.3 Diagrama de componentes, se puede observar los componentes del contenedor API deploy server, y cómo interactúa con el resto de contenedores

12.2. Módulo de usuarios

Este módulo es el encargado de gestionar la administración de las cuentas de usuario.

Para poder identificarlos de forma única, se emplea el propio nombre de usuario para minimizar el número de consultas necesarias.

Existen tres roles:

- **Admin**. Puede realizar cualquier tipo de operación sobre los equipos de red, y administrar los usuarios
- **Editor**. Puede realizar cualquier tipo de operación sobre los equipos de red.
- **Viewer**. Solo puede realizar operaciones de consulta sobre los equipos de red (Ver estado de interfaces, realizar un PING…)

A continuación, se muestra un objeto JSON correspondiente a un nuevo usuario.

```
{
  "email": "user1@mymail.com",
  "name": "user",
  "password": "pass",
  "role": "editor",
  "surname": "one",
  "username": "user1"
}
```

En cuanto al JSON de respuesta, se puede observar que lógicamente no devuelve la contraseña.

```
{
  "email": "user1@mymail.com",
  "name": "user",
  "role": "editor",
  "surname": "one",
  "username": "user1"
}
```

Los datos se almacenan en una colección, dentro de una base de datos independiente, y todas las contraseñas se encuentran cifradas para dar más seguridad. Para ello, se ha implementado la clase *Hasher,* que lo que hace es aplicar un hash sobre la contraseña para almacenarla. En cuanto a la validación de los usuarios, se realiza la operación de *hash* sobre la contraseña proporcionada y se compara con el string almacenado.

```python
from passlib.context import CryptContext
from tenant.domain.ports.haser_interface import IHasher

class Hasher(IHasher):
    context = CryptContext(schemes=["bcrypt"], deprecated="auto")

    @classmethod
    def compare_hash(cls, plain_password: str, hashed_password: str) -> bool:
        return cls.context.verify(plain_password, hashed_password)

    @classmethod
    def generate_hash(cls, plain_text: str) -> str:
        return cls.context.hash(plain_text)
```

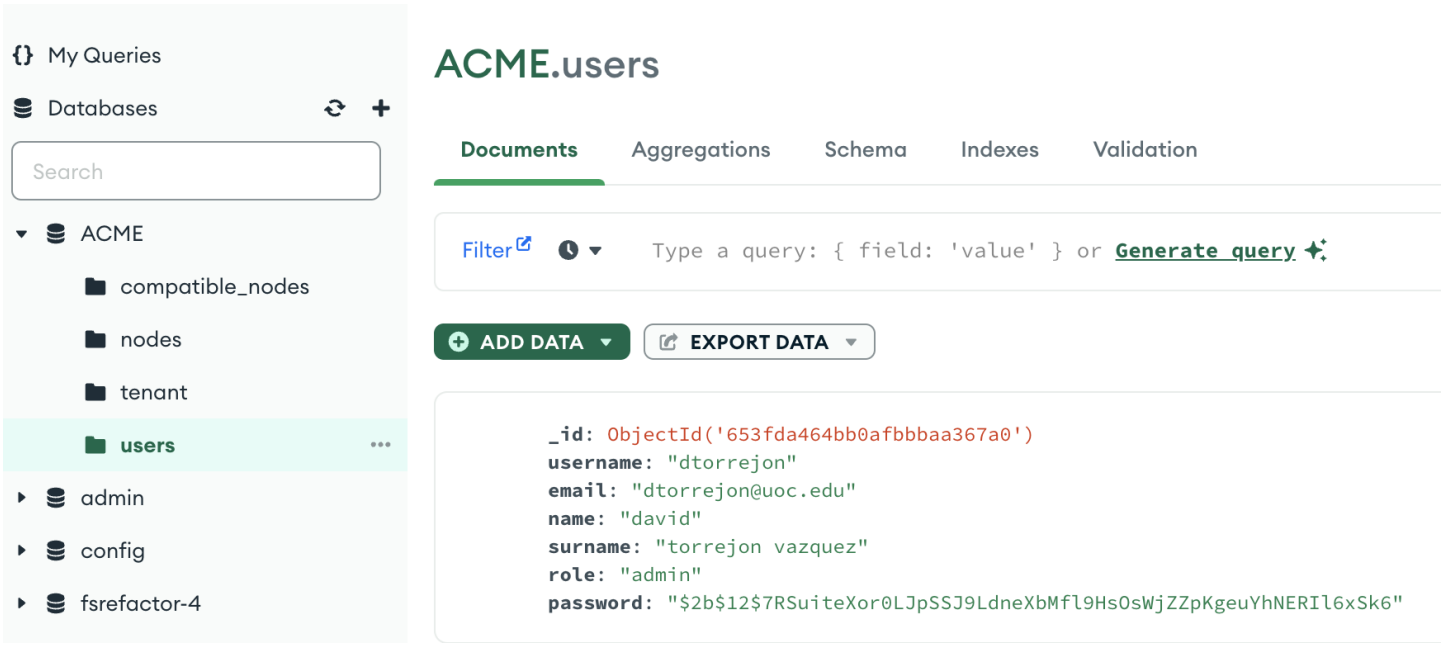


Figura 12.2-1 Colección de usuarios en MongoDB

En la siguiente imagen se puede observar la implementación de clases para crear un usuario y almacenarlo en la base de datos. Esta arquitectura es la que se ha seguido para implementar todos los casos de uso

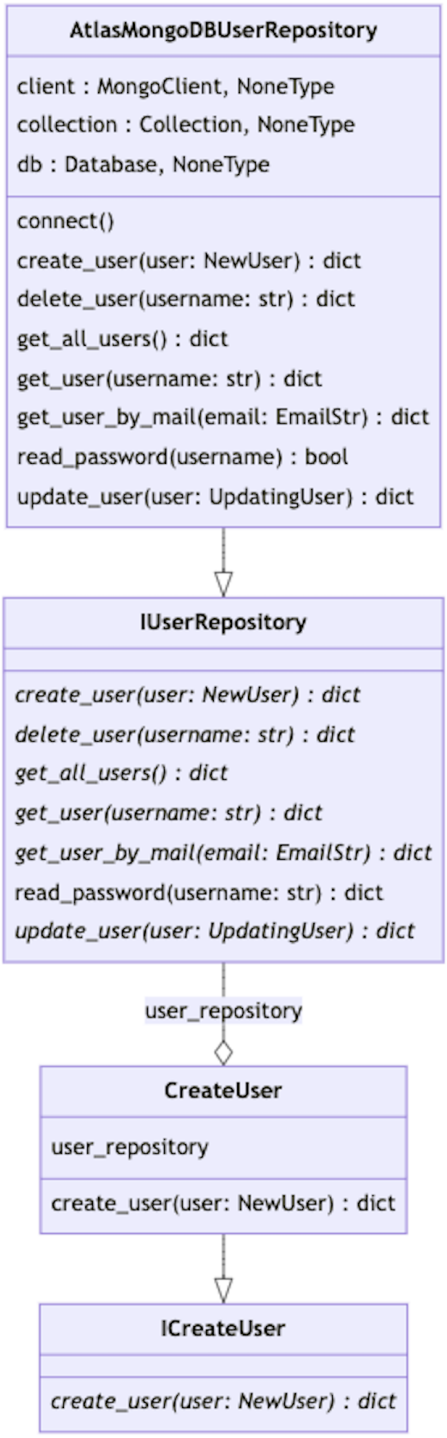


Figura 12.2-3 Diagrama UML de un caso de uso y su implementación

12.3. Módulo de comunicación

El principal objetivo de este módulo, es el de poder establecer comunicaciones entre el servidor y la red de cliente de forma segura a través de internet, mediante el establecimiento de una vpn site to site.

En un escenario típico, esto representa la necesidad de realizar una serie de configuraciones que permitan establecer el túnel, y dado que se está desarrollando una API que permita disponer de una capa de abstracción y facilite entre otras, la automatización, se ha determinado que es necesario hacer transparente este proceso de cara al usuario.

Para ello, se hace uso del servicio Cloud Connexa anteriormente descrito, haciendo uso de su API REST en la aplicación desarrollada para realizar el despliegue de los tenants.

El proceso que se sigue es el siguiente:

1. Se instancia una máquina virtual en Google cloud, y dentro del proceso de configuración se instala el agente de *openvpn*.

2. Con la información proporcionada por el cliente a la hora de crear el servicio, se configura el servicio de cloud connexa.

```
{
    "name": "company",
    "address": "Fondo Somella s/n",
    "phone": "9344657894",
    "user": {
        "name": "david",
        "email": "dtorrejon@uoc.edu",
        "username": "dtorrejon",
        "password": "1234",
        "surname": "torrejon vazquez"
    },
    "network_config": {
        "ip_cidr": "192.168.30.10/24",
        "gateway": "192.168.30.1",
        "allowed_networks": [
            "192.168.30.0/24",
```

```
            "192.168.20.0/24"
        ]
    }
}
```

3. Se configura una máquina virtual que hace de punto de conexión, la cual el cliente ejecuta dentro de su infraestructura.

12.4. Módulo de gestión de equipos de red

Este funciona de forma similar al de gestión de usuarios.

A continuación se muestra el objeto definido para crear un nodo

```
{
  "name": "string",
  "technology": "string",
  "vendor": "string",
  "model": "string",
  "softwareVersion": "string",
  "ipAddress": "string",
  "protocol": "ssh",
  "port": 0,
  "username": "string",
  "password": "string"
}
```

Como se puede observar en la siguiente imagen, la contraseña se almacena cifrada

Figura 12.4-1 Colección de nodos en MongoDB

A nivel de seguridad, implementa una clase denominada *PasswordCrypt,* que permite a través de clave simétrica, almacenar de forma segura las contraseñas de los nodos en la base de datos.

```
from cryptography.fernet import Fernet


class PasswordCrypt:

    def __init__(self):
        SECRET_KEY: bytes = b'ThisIsSecret'
        self.cipher_suite = Fernet(SECRET_KEY)

    def encode_password(self, password: str) -> str:
        return self.cipher_suite.encrypt(password.encode()).decode("utf-8")

    def decode_password(self, encrypted_password: str) -> str:
        return self.cipher_suite.decrypt(encrypted_password).decode("utf-8")

    @staticmethod
    def generate_secret_key() -> bytes:
        return Fernet.generate_key()
```

12.5. Módulo de operaciones

Este módulo permite realizar las tareas requeridas en los diferentes elementos de red, de forma estandarizada.

Para las operaciones implementadas, dentro del alcance de este proyecto, se han definido una serie de objetos que deberán de ser comunes a todos los elementos.

A continuación, se muestran algunos de los objetos definidos. Todos ellos se encuentran documentados en la propia documentación de la API, publicada y accesible desde Internet en el siguiente enlace.

PING

```
{
  "destination": "8.8.8.8",
  "packetsReceived": 4,
  "packetsTransmitted": 4,
  "percent": "100%",
  "roundTrip": {
    "avg": "38 ms",
    "max": "51 ms",
    "min": "26 ms"
  },
  "source": "192.168.20.10"
}
```


VLAN BRIEF


```
[
  {
    "vlan": 1,
    "name": "default",
    "status": "active",
    "ports": [
      "Et0/0",
      "Et0/1",
      "Et0/3",
      "Et1/0",
      "Et1/1",
      "Et1/3"
    ]
  },
  {
    "vlan": 99,
    "name": "VLAN0099",
    "status": "active",
    "ports": []
  },
  {
```

```
    "vlan": 789,
    "name": "VLAN0789",
    "status": "active",
    "ports": [
      "Et3/1"
    ]
  },
  {
    "vlan": 1002,
    "name": "fddi-default",
    "status": "act/unsup",
    "ports": []
  },
  {
    "vlan": 1003,
    "name": "token-ring-default",
    "status": "act/unsup",
    "ports": []
  },
  {
    "vlan": 1004,
    "name": "fddinet-default",
    "status": "act/unsup",
    "ports": []
  },
  {
    "vlan": 1005,
    "name": "trnet-default",
    "status": "act/unsup",
    "ports": []
  }
]
```


INTERFACE BRIEF


```
[
  {
```

```
    "id": 1,
    "destination": "0.0.0.0/0",
    "protocol": "STATIC_CANDIDATE_DEFAULT",
    "preference": 1,
    "cost": 0,
    "nextHop": "192.168.20.1",
    "interface": "192.168.20.1",
    "age": "-"
  },
  {
    "id": 2,
    "destination": "192.168.20.0/24",
    "protocol": "CONNECTED",
    "preference": 0,
    "cost": 0,
    "nextHop": "directly",
    "interface": "Vlan1",
    "age": "-"
  },
  {
    "id": 3,
    "destination": "192.168.20.10/32",
    "protocol": "LOCAL",
    "preference": 0,
    "cost": 0,
    "nextHop": "directly",
    "interface": "Vlan1",
    "age": "-"
  }
]
```


Respecto al diseño, se ha optado por implementar una serie de clases siguiendo el patrón *Strategy,* el cual permite a través de la composición, modificar el comportamiento de los objetos que permiten ejecutar las operaciones en diferentes elementos de red, en tiempo de ejecución. De esta forma, se simplifica y optimiza la arquitectura y el esfuerzo necesario para actualizar o añadir operaciones, así como implementaciones de nuevos elementos de red.

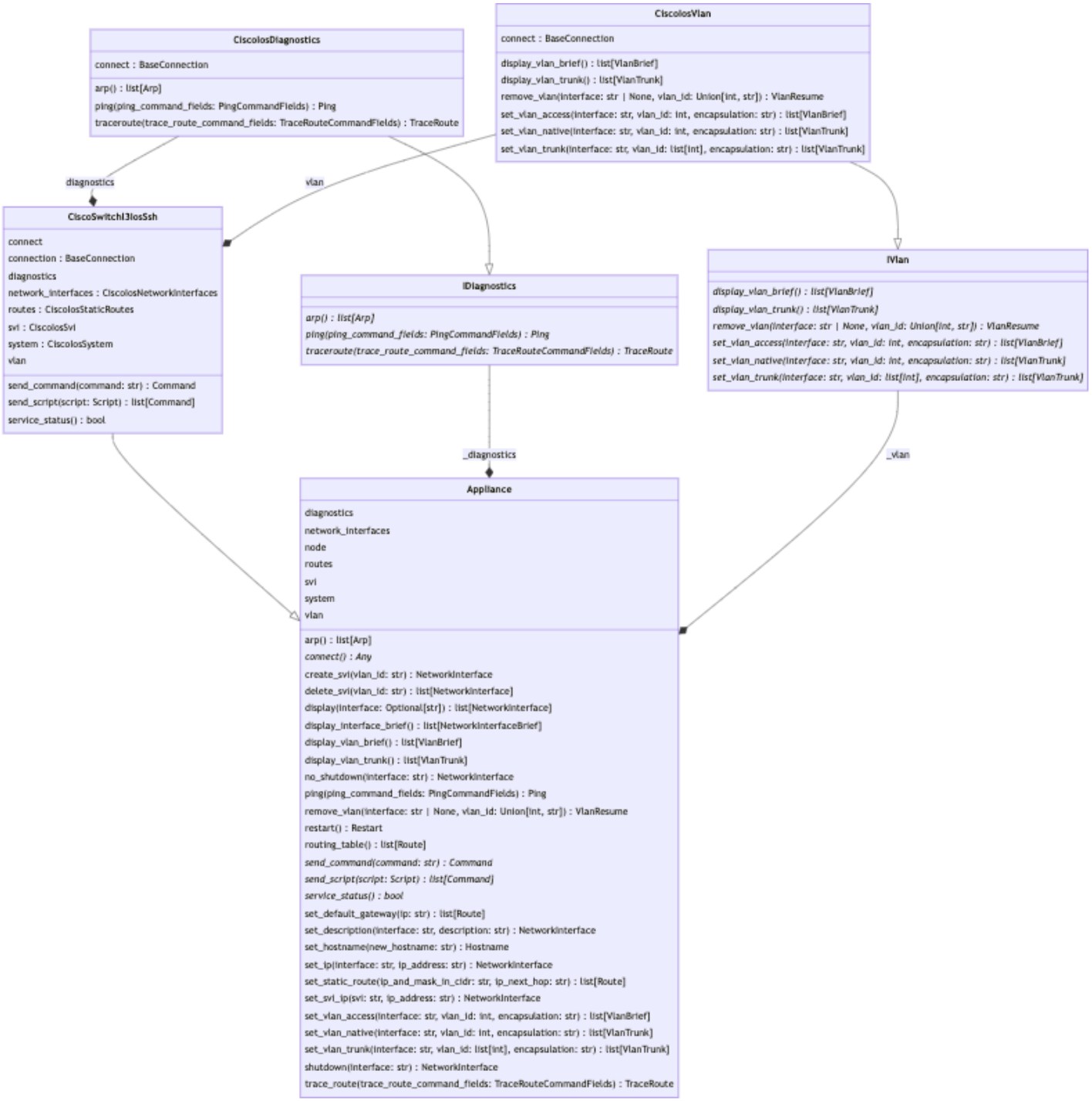


Figura 12.5-1 Diagrama de clases para la gestión de un elemento de red

13. Test y pruebas de validación

13.1. Pruebas unitarias

Las pruebas unitarias permiten validar el comportamiento de una parte de código de manera aislada e independiente, asegurando que cada unidad funcione como se espera. Estas pruebas son clave para garantizar la calidad del código y facilitar el mantenimiento de este, pues una forma simple de mejorarlo, consiste en añadir simplemente más pruebas.

Por otro lado, al realizar cambios y modificaciones sobre el código, permite validar que estos no afectan al funcionamiento.

Destacar que la suite de test se puede ejecutar de forma automática, en este caso haciendo uso de *pytest*.

Para poder asegurar el funcionamiento esperado del prototipo, se ha desarrollado una suite de test basada principalmente en validar los casos de uso, las implementaciones concretas para gestionar los elementos de red, y la implementación del repositorio, además de las clases implementadas para el cifrado de contraseñas y generación de tokens.

Al haber empleado una arquitectura de software hexagonal o en capas, hace que toda la lógica de negocio quede fuera de la implementación de los endpoints de la API, pues esta básicamente se encargará de llamar a cada uno de los casos de uso, enviando la información proporcionada por el usuario, y devolviendo la respuesta que le hace llegar cada caso de uso.

A continuación se añade una captura de la implementación de un test unitario, y una captura con la evidencia de que los test han pasado correctamente. Todos ellos se encuentran en la carpeta test del código.

```
import pytest
from unittest.mock import Mock, patch
from
tenant.infraestructure.adapters.repositories.atlas_mongo_db_node_repository
import AtlasMongoDBNodeRepository
from tenant.domain.schemas.node.node import Node
from tenant.domain.schemas.node.existing_node import ExistingNode
```

```
class TestAtlasMongoDBNodeRepository:

    @staticmethod

@patch('tenant.infraestructure.adapters.repositories.atlas_mongo_db_node_reposi
tory.MongoClient')
    def
test_should_return_node_response_when_save_is_called_with_valid_node(mock_mongo
_client):
        mock_collection = Mock()
        mock_collection.insert_one.return_value = True
        mock_collection.find_one.return_value = {"name": "Node Name", "vendor":
"Vendor", "technology": "Tech",
                                                 "model": "Model",
"softwareVersion": "1.0", "ipAddress": "192.168.1.1",
                                                 "protocol": "ssh", "port":
1234}

mock_mongo_client.return_value.__getitem__.return_value.__getitem__.return_valu
e = mock_collection

        repository = AtlasMongoDBNodeRepository()
        result = repository.save(
            Node(name="Node Name", vendor="Vendor", technology="Tech",
model="Model", softwareVersion="1.0",
                 ipAddress="192.168.1.1", protocol="ssh", port=1234))

        assert result.name == "Node Name"
        assert result.vendor == "Vendor"
        assert result.technology == "Tech"
        assert result.model == "Model"
        assert result.softwareVersion == "1.0"
        assert result.ipAddress == "192.168.1.1"
        assert result.protocol == "ssh"
        assert result.port == 1234

    @staticmethod
```

```
@patch('tenant.infraestructure.adapters.repositories.atlas_mongo_db_node_reposi
tory.MongoClient')
    def
test_should_return_empty_dict_when_retrieve_is_called_with_no_matching_node_nam
e(mock_mongo_client):
        mock_collection = Mock()
        mock_collection.find_one.return_value = None

mock_mongo_client.return_value.__getitem__.return_value.__getitem__.return_valu
e = mock_collection

        repository = AtlasMongoDBNodeRepository()
        result = repository.retrieve("Nonexistent Node Name")

        assert result == {}

    @staticmethod

@patch('tenant.infraestructure.adapters.repositories.atlas_mongo_db_node_reposi
tory.MongoClient')
    def
test_should_return_all_node_responses_when_retrieve_all_is_called(mock_mongo_cl
ient):
        mock_collection = Mock()
        mock_collection.find.return_value = [
            {"name": "Node Name 1", "vendor": "Vendor 1", "technology": "Tech
1", "model": "Model 1",
             "softwareVersion": "1.0", "ipAddress": "192.168.1.1", "protocol":
"telnet", "port": 23},
            {"name": "Node Name 2", "vendor": "Vendor 2", "technology": "Tech
2", "model": "Model 2",
             "softwareVersion": "2.0", "ipAddress": "192.168.2.2", "protocol":
"ssh", "port": 22}]

mock_mongo_client.return_value.__getitem__.return_value.__getitem__.return_valu
e = mock_collection

        repository = AtlasMongoDBNodeRepository()
```

```
        result = repository.retrieve_all()
        assert len(result) == 2
        assert result[0]["name"] == "Node Name 1"
        assert result[1]["name"] == "Node Name 2"
```

Como se puede observar, se hace uso de la clases Mock y patch, para poder simular los objetos y funciones.

En cuanto a la generación del código de pruebas, se ha hecho uso de una versión beta de Github Copilot chat, IA generativa en fase beta actualmente, con el objetivo de ayudar a reducir las horas en realizar tareas repetitivas para la generación de las plantillas de los test.

Figura 13.1-1 Evidencia Test unitarios

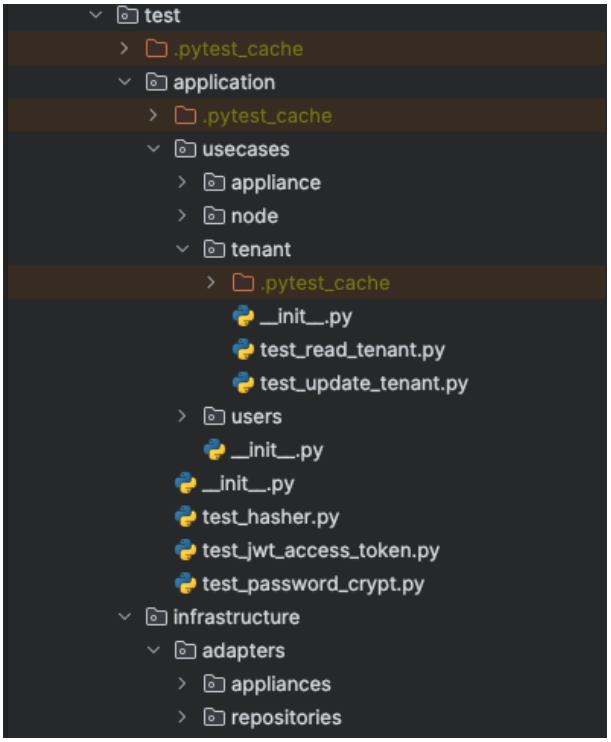


Figura 13.1-2 Estructura de carpetas de test unitarios

13.2. Pruebas de integración

Estas pruebas sirven para validar el correcto funcionamiento entre los diferentes componentes de la aplicación, así como la integración con modelos y aplicaciones de terceros.

En este caso en concreto, se dispone de *swagger* integrado dentro de *FastAPI*, una potente herramienta, que además de de documentar y validar que se cumple con los estándares de Open API, permite realizar operaciones sobre cada uno de los endpoints.

Por un lado, se realiza la prueba para el despliegue de un tenant, a través del endpoint disponible en el servidor "deploy". Con esto se podrá validar el correcto funcionamiento e integración con la API de Google Cloud, Cloud Connexa, y el servicio de correo electrónico. En cuanto al resto, se realiza una prueba sobre cada uno de los *endpoints* para asegurar que funcionan correctamente, una vez desplegado el tenant.

Los pre-requisitos para poder realizar estas pruebas son:

- Tener los servidores "deploy" y "templates-server" implementados.

- Disponer de una red con equipos gestionables, en este caso el laboratorio de GNS3

A continuación, se añaden algunas evidencias para demostrar el correcto funcionamiento de la aplicación.

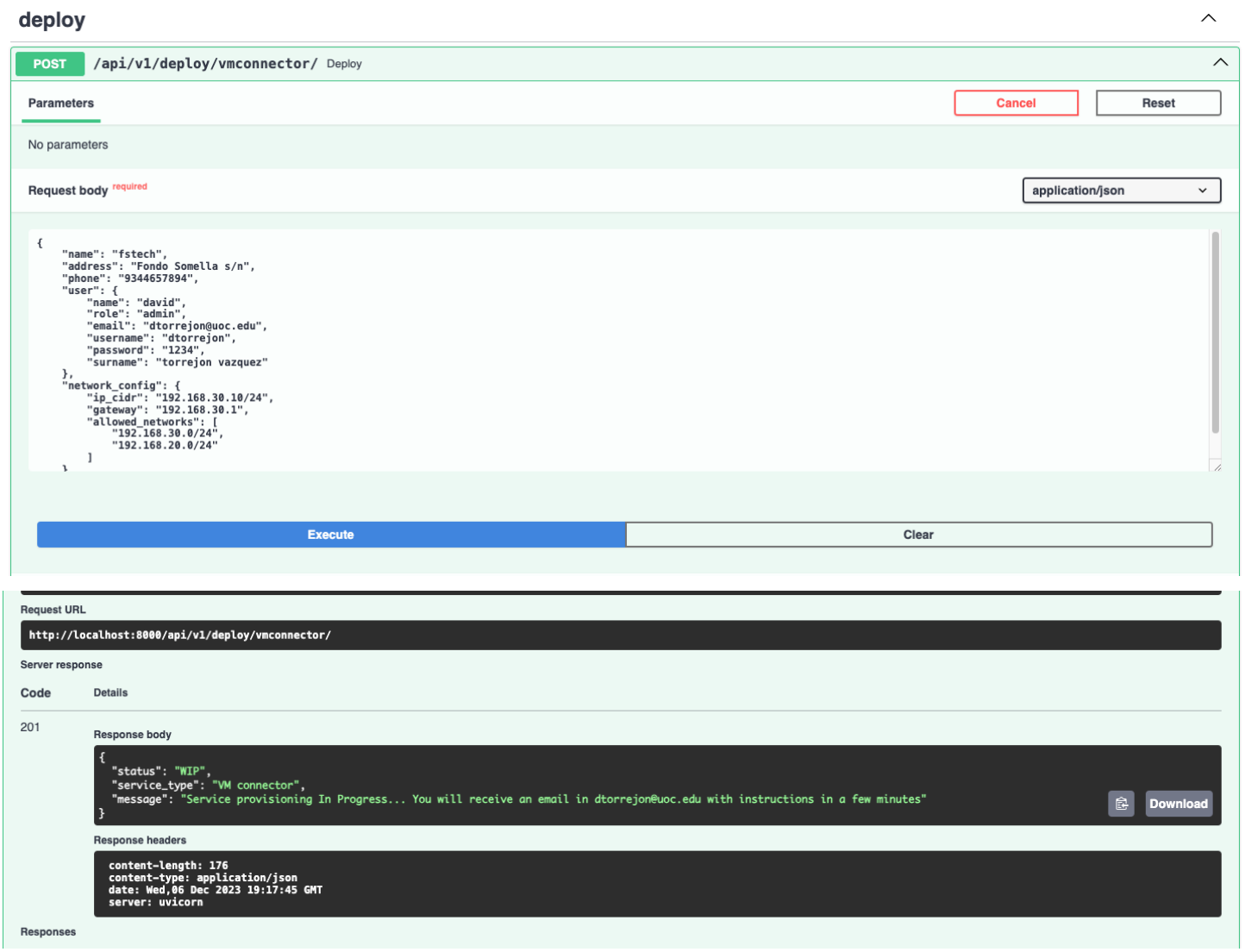


Figura 13.2-1 Creación de un tenant a través de la API

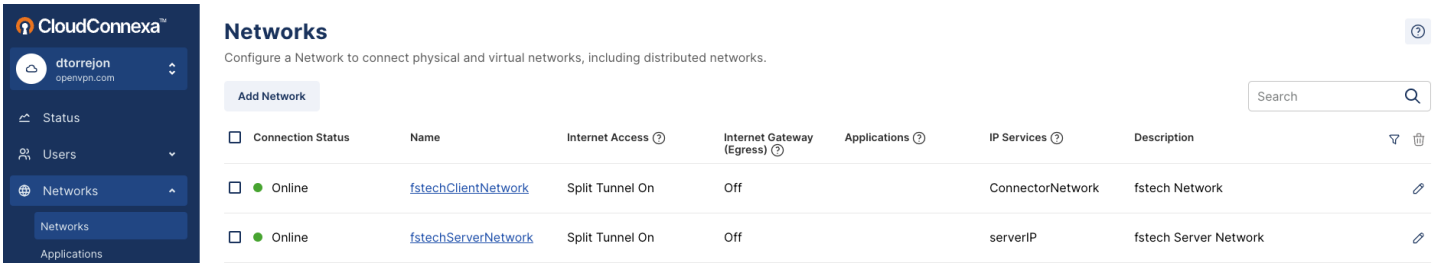


Figura 13.2-2 Verificación de la configuración de los conectores en Cloud Connexa

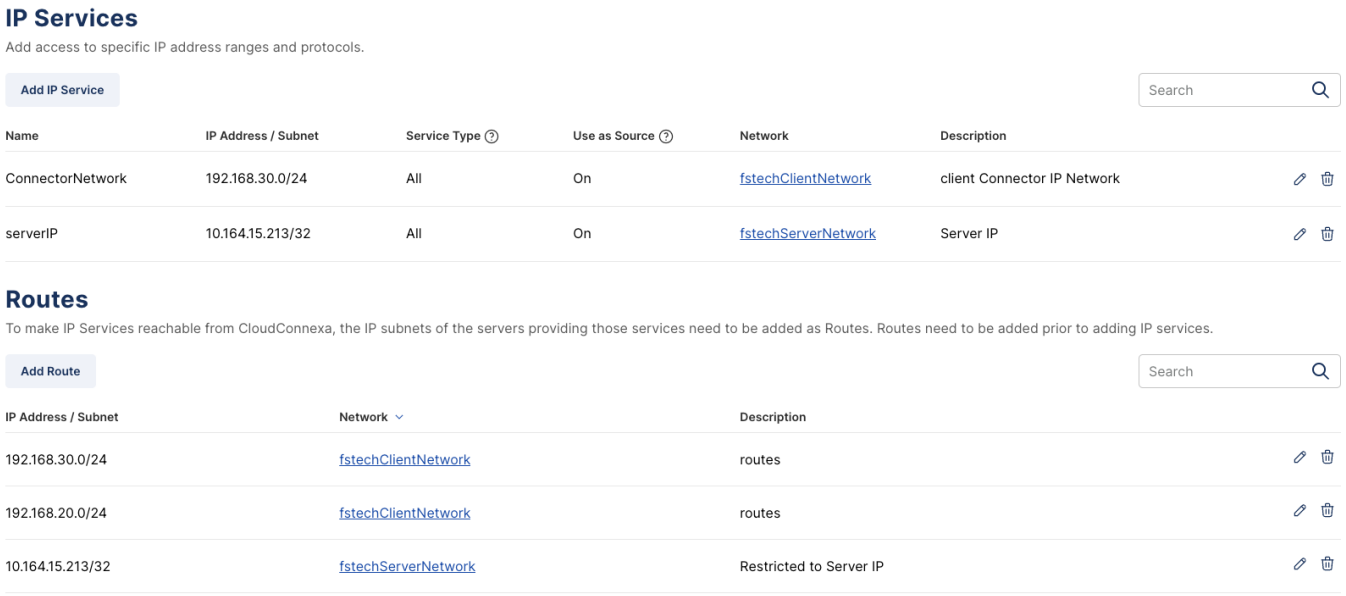


Figura 13.2-3 validación de la configuración de servicios y rutas en Cloud Connexa

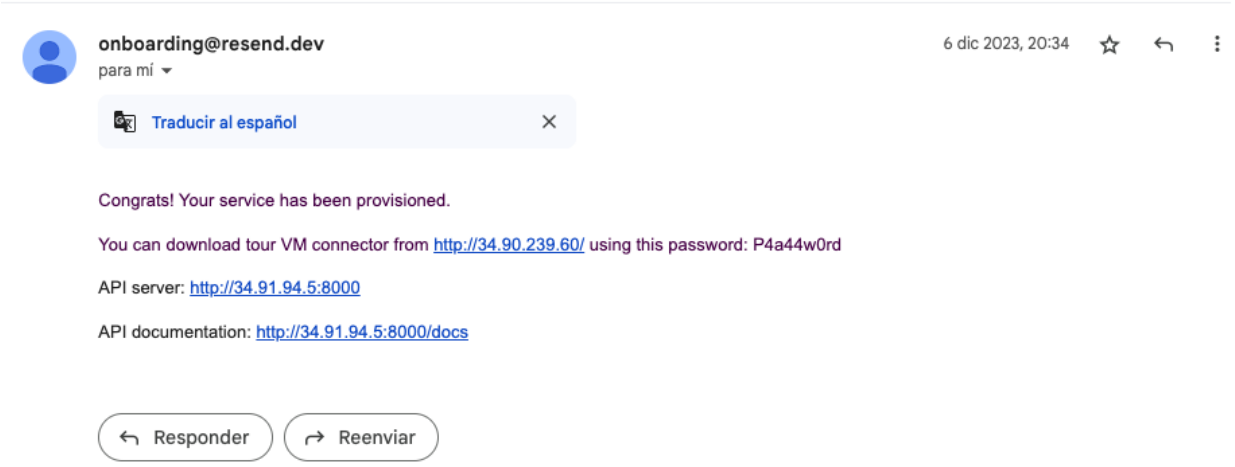


Figura 13.2-4 Email de confirmación de Tenant en servicio y link de descarga para la Vm del conector

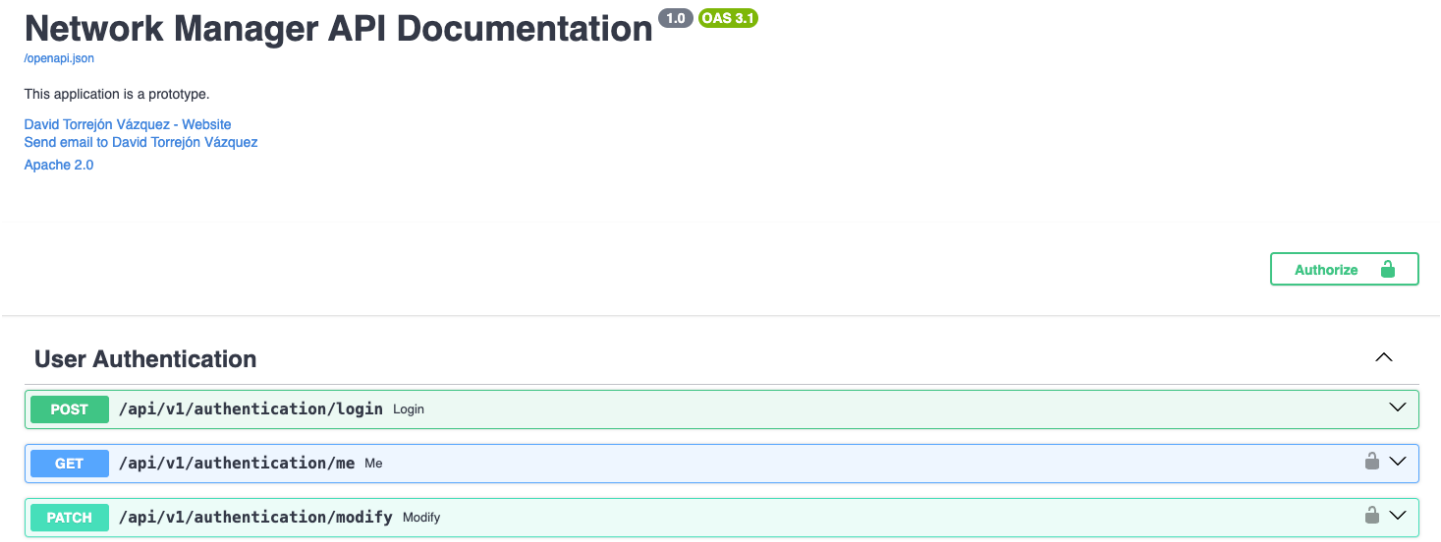


Figura 13.2-5 Endpoints de gestión del usuario

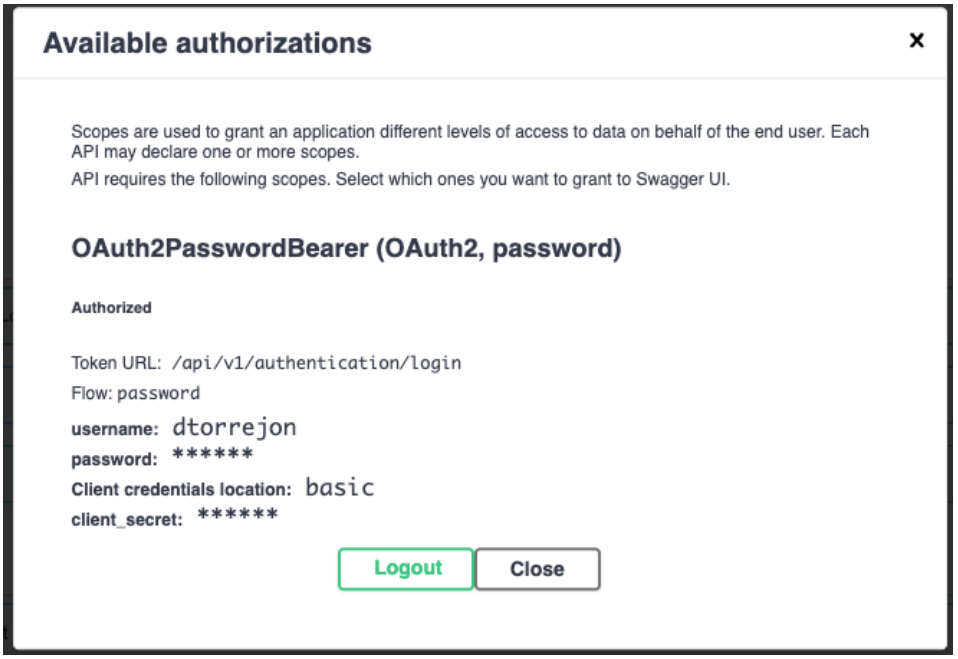


Figura 13.2-6 Acceso con credenciales

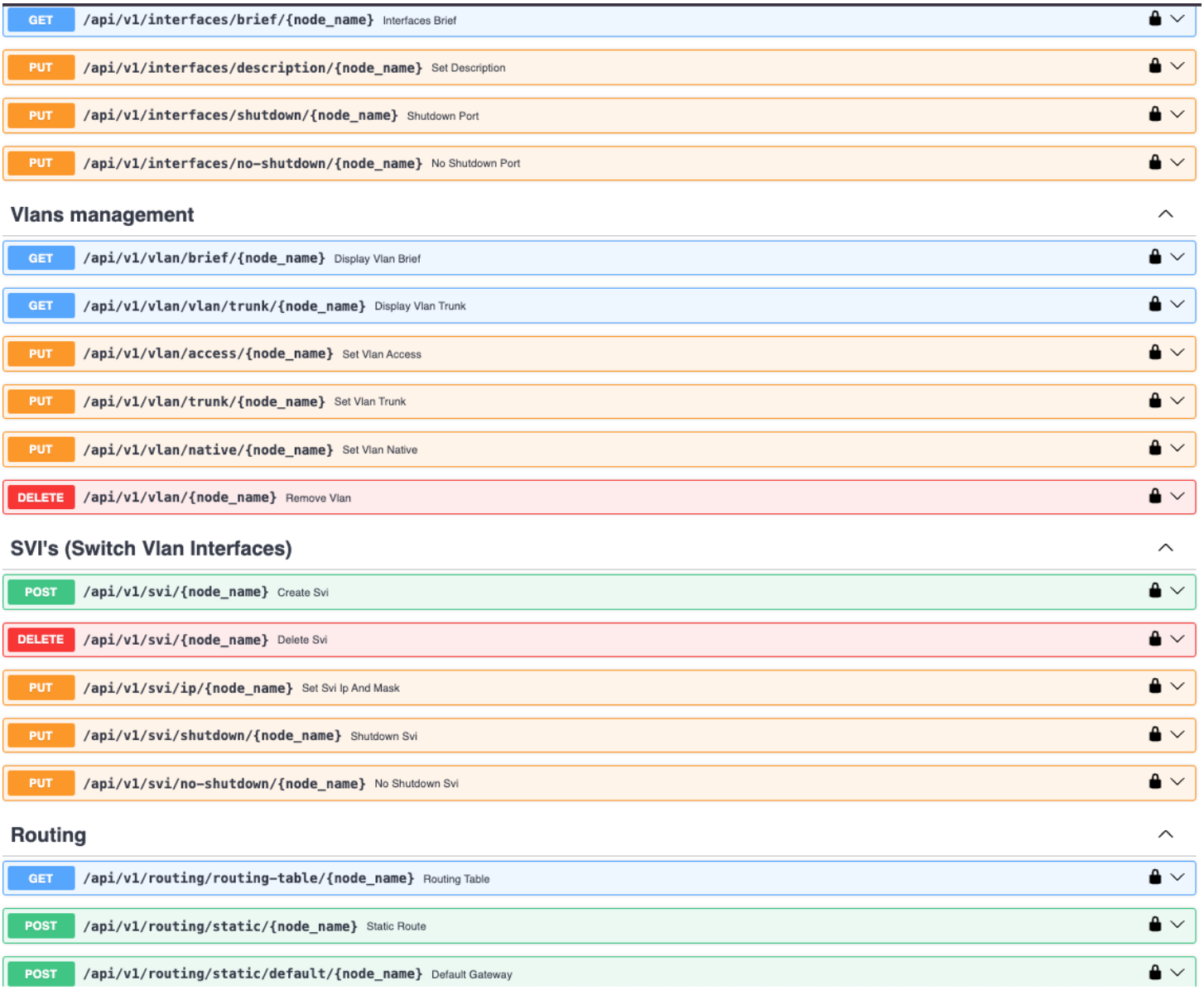


Figura 13.2-7 Algunos de los endpoints de la aplicación

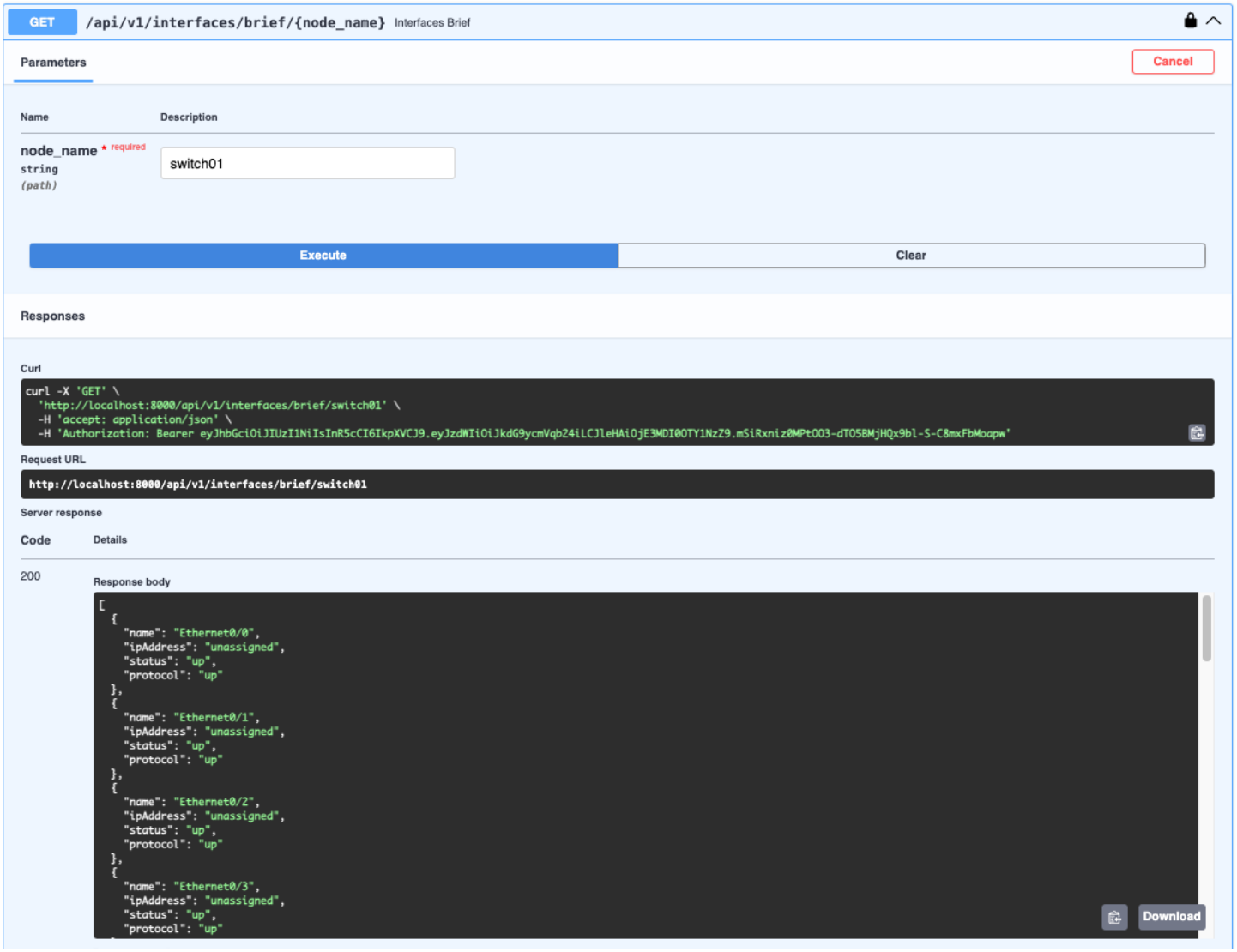


Figura 13.2-8 Petición de resumen de interfaces

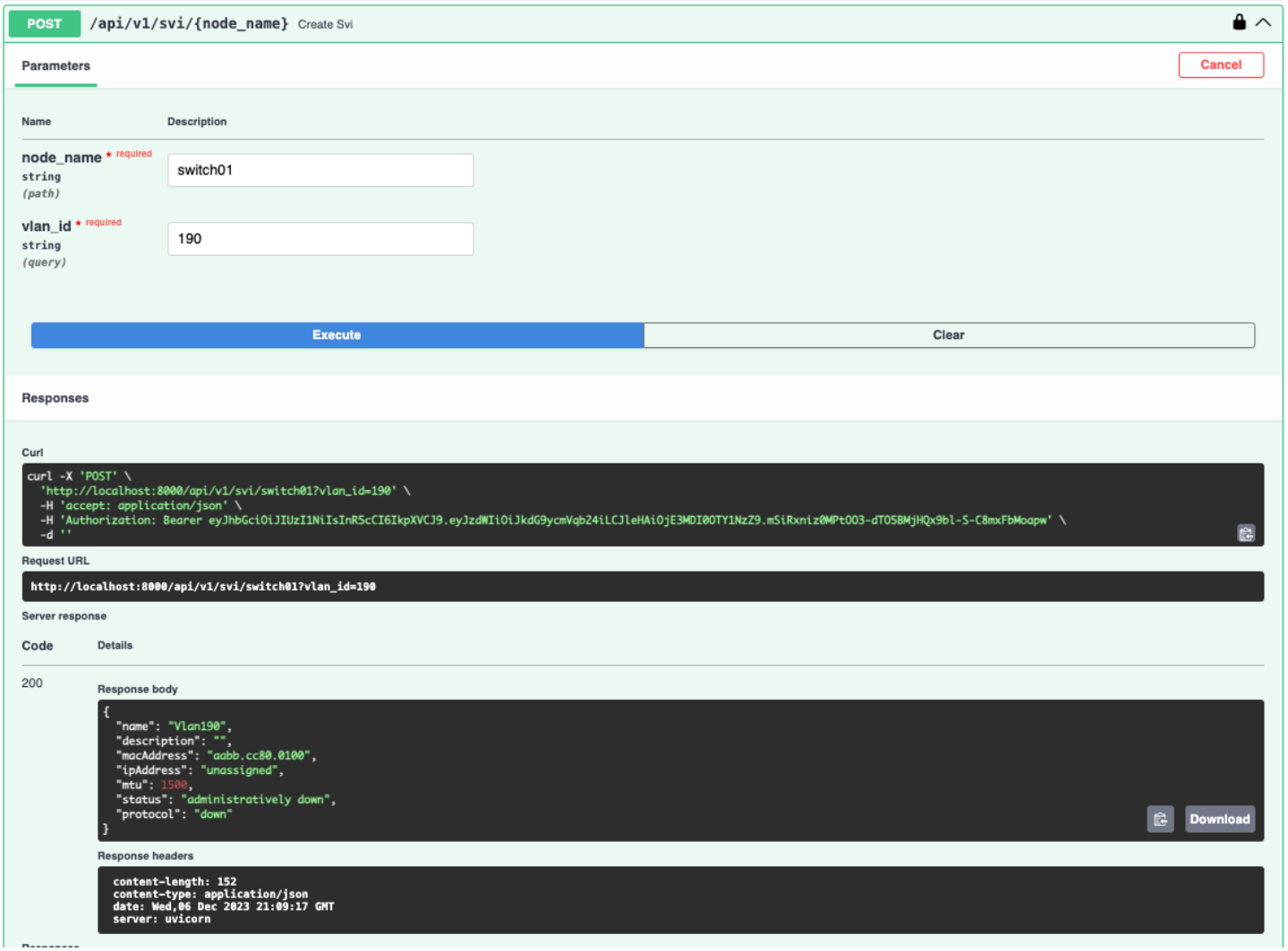


Figura 13.2-9 Creación de un SVI

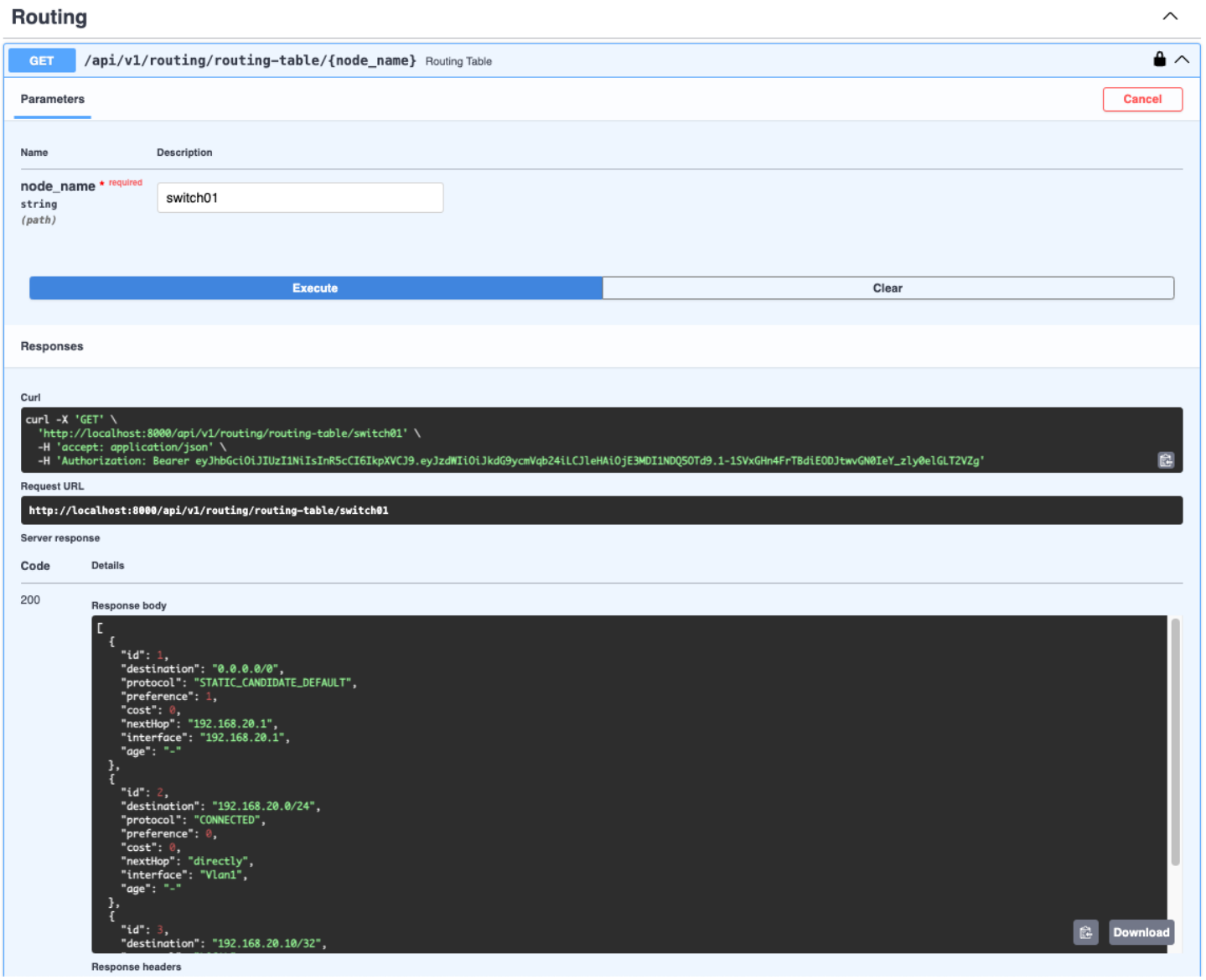


Figura 13.2-10 Petición de la tabla de rutas

Otra prueba importante es validar el funcionamiento, en el caso de que hayan diferentes operarios enviando órdenes al mismo elemento de red. Para ello, se ha desarrollado un script que lanza un paquete de operaciones de forma concurrente. A continuación, se añade tanto el código del script como una evidencia de que los resultados son los esperados, al no existir colisiones entre las diferentes peticiones.

```
import requests
from concurrent.futures import ThreadPoolExecutor
```

```
class ConcurrentTest:

    def __init__(self):
        url = "http://127.0.0.1:8000/api/v1/authentication/login"
        payload = {'username': 'dtorrejon','password': 'password'}
        files = []
        headers = {}
        response = requests.request("POST", url, headers=headers, data=payload,
files=files)
        self.token = response.json()["access_token"]

    def fetch_data(self, url):
        payload = {}
        headers = {'Authorization': f'Bearer {self.token}'}
        response = requests.request("GET", url, headers=headers, data=payload)
        return response.text

    def main_funct(self):
        urls = [

'http://localhost:8000/api/v1/diagnostics/ping/switch01/8.8.8.8?repeat_ping=4',
            'http://localhost:8000/api/v1/routing/routing-table/switch01',
            'http://localhost:8000/api/v1/diagnostics/arp/switch01',
            'http://localhost:8000/api/v1/routing/routing-table/switch01',

'http://localhost:8000/api/v1/diagnostics/ping/switch01/8.8.8.8?repeat_ping=4',
            'http://localhost:8000/api/v1/diagnostics/arp/switch01',

'http://localhost:8000/api/v1/diagnostics/traceroute/switch01/8.8.8.8?probe_cou
nt=3',
            'http://localhost:8000/api/v1/diagnostics/arp/switch01',

'http://localhost:8000/api/v1/diagnostics/traceroute/switch01/1.1.1.1?probe_cou
nt=3',

'http://localhost:8000/api/v1/diagnostics/ping/switch01/8.8.8.8?repeat_ping=4',
            'http://localhost:8000/api/v1/routing/routing-table/switch01',
```

```
            'http://localhost:8000/api/v1/routing/routing-table/switch01',
        ]

        with ThreadPoolExecutor(max_workers=10) as executor:
            results = list(executor.map(self.fetch_data, urls))

        for i, result in enumerate(results, 1):
            print(f'{i}: {result[:80]}...')


if __name__ == "__main__":
    test = ConcurrentTest()
    test.main_funct()
```

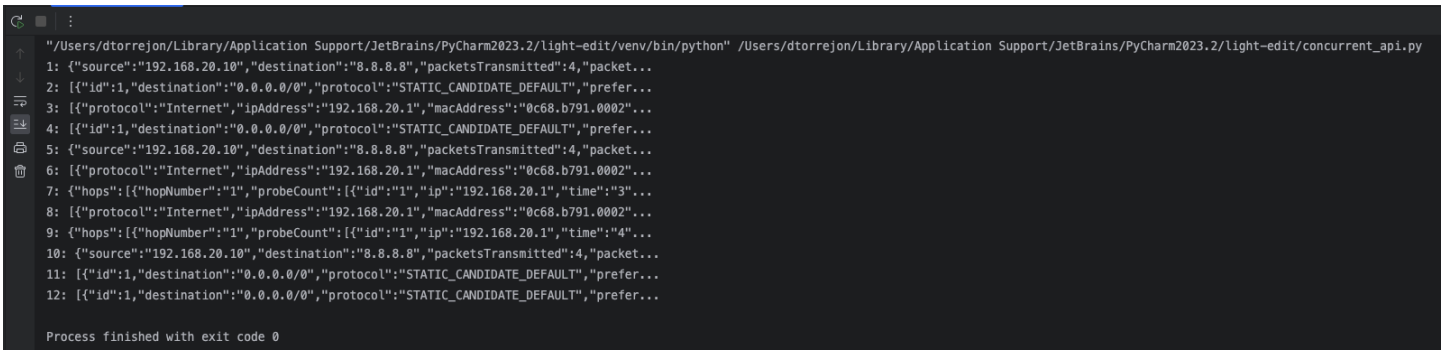


Figura 13.2-11 Respuesta a la ejecución del script con respuesta favorable

13.3. Pruebas de usuario

Aunque en este caso se trata de un prototipo funcional, y no se ha empleado una metodología para recopilar requisitos de usuario, si que es importante mencionarlas, pues como el objetivo es continuar a futuro con el desarrollo de este proyecto, se deberán tener en cuenta.

Básicamente, estas pruebas se centran en evaluar el cumplimiento de los requisitos y expectativas del usuario final, antes de su implementación en un entorno de producción. Las principales razones para llevar a cabo este tipo de pruebas, son:

- **Validación de Requisitos del Usuario**. Asegurar que satisface sus requisitos

- **Identificación de Problemas de Usabilidad**. Detectar posibles problemas de interfaz, diseño, experiencia del usuario…

- **Reducción de Errores en Producción.** identificar y corregir problemas durante las pruebas de usuario

- **Involucramiento del Usuario.** Involucrar activamente a los usuarios finales en el proceso de desarrollo

14. Publicación del proyecto

Se espera que este prototipo funcional siente las bases tanto de una idea que persigue facilitar la vida a los ingenieros de red. Por ello, se ha priorizado el diseño, por encima de la implementación de un gran número de funcionalidades.

La estandarización es posible, pues estamos trabajando con equipos que a nivel de comunicaciones son capaces de entenderse porque hablan el mismo idioma (protocolos de comunicación). Solo hay que seguir trabajando en un lenguaje común para la gestión de estos, pues al final todos responden lo mismo de diferente manera.

Por ello, se ha decidido publicar un proyecto en github con el objetivo de seguir desarrollándolo, y poder generar una comunidad que permita la evolución de la herramienta, así como poder enfocar los esfuerzos en aquellos desarrollos que puedan resultar de mayor interés para los usuarios.

Dado que la aplicación de Deploy está enfocada a ofrecer en modelo SaaS la API desarrollada, solamente será publicado en github el código correspondiente a la propia API, incluyendo un script que permita crear la base de datos con un usuario.

El link donde se encuentra publicado es: https://github.com/dtorrejon/networkmanager

15. Próximos pasos

Una vez finalizado el prototipo, es necesario saber qué pasos se han de dar para seguir desarrollando la idea. Por ello, además de las mejoras que se proponen en el apartado 16. Mejoras y propuesta de nuevas features, se debe de plantear la realización de un análisis de mercado y plan de desarrollo, que permitan definir una estrategia.

Para ello ,se debe de elaborar un plan estratégico que desarrolle los siguientes puntos clave:

- Análisis de negocio (Modelo Canvas)
- Misión, Visión y Valores
- Agentes y grupos de interés (stakeholders)
- Cadena de Valor de Porter
- Análisis del macroentorno: PESTEL
- Análisis del microentorno: Las 5 fuerzas de Porter
- Análisis DAFO
- Estrategia de negocio

16. Conclusiones, mejoras y propuesta de nuevas features

Como conclusiones al proyecto realizado, en función de los objetivos marcados, se puede decir que:

- Se ha desarrollado una herramienta que permite tener un interfaz común para una serie de funcionalidades de los elementos de red integrados
- También se ha conseguido desarrollar un servicio que permite ofrecer esta herramienta en modelo Saas, ofreciendo una comunicación segura de forma sencilla y transparente
- Facilita la automatización de redes, así integración con otras herramientas al devolver un mismo objeto para cada operación, independientemente del equipo de red
- Por contra, dado el alcance del proyecto, no se ha dispuesto de horas suficientes para poder realizar todas las tareas de refactorización, agregar mas test unitarios y evolucionar todo lo deseado la herramienta

REspecto a los puntos a seguir trabajando en el corto plazo, serían:

- Añadir más funcionalidades para operar los diferentes elementos de red.
- Implementar más marcas y modelos

- Evolucionar hacia la integración de equipos que trabajan con otras tecnologías, como FTTH
- Integración con IPAM, herramientas de backup…

17. Lista de figuras

18. Lista de tablas

19. Glosario de términos

API REST. Interfaz de programación de aplicaciones basada en el estilo arquitectonico REST.

Clave privada de cifrado. Código privado, que se emplea en criptografía para descifrar datos y/o firmar digitalmente.

Clave pública de cifrado. Código compartido que es utilizado en criptografía para cifrar informacion y/o verificar firmas digitales.

ClickOps. Método de gestión y operación, de redes y sistemas, realizado de forma manual

Cloud. Entorno de computación en la nube que ofrece servicios y recursos a través de Internet.

IaC (Infrastructure as Code). Provisión y gestión de infraestructura a través de código.

MVC (Model View Controller). Patrón de diseño de software que separa la lógica de la aplicación en tres componentes: Modelo, Vista y Controlador.

NaC (Network as Code). Provisión y gestión de redes a través de código.

NetDevOps. Integración de prácticas de desarrollo de software (DevOps) en el ámbito de redes.

On-Premise. Despliegues realizados en los propios servidores o instalaciones propias.

Patrón de software. Estándares de diseño de software, para dar solución a problemas comunes, y facilitar la reutilización de código.

PMBOK (Project Management Body of Knowledge): Conjunto de estándares y directrices en la gestión de proyectos, elaborados por el PMI (Project Management Institute).

PMI (Project Management Institute). Organización sin fines de lucro que promueve la gestión de proyectos basada principalmente en los estándares de esta.

RSA (Rivest-Shamir-Adleman). Algoritmo de criptografía basado en clave asimétrica, utilizado para la seguridad en la comunicación.

SaaS (Software as a Service). Modelo de distribución de software en el que las aplicaciones son desplegadas y accesibles, de forma transparente al usuario.

VM (virtual machine). Sistema informático emulado en software.

20. Bibliografía

Contributor. (2020, March 2). DevSecOps and DevNetOps: New heroes in the devops saga. DevOps.com. Retrieved from https://devops.com/devsecops-devnetops-new-heroes-devops-saga/

Dodin, R. (n.d.). applying devops to networks. Index - Applying DevOps to networks. Retrieved from https://netdevops.me/

FastAPI. FASTAPI. (n.d.). Retrieved from https://fastapi.tiangolo.com/

James Kelly Follow MCS: Cloud & DevOps Technologist; (Ad)Venturist: Sales and Marketing Guru. (n.d.). DevNetOps Overview. Share and Discover Knowledge on SlideShare. Retrieved from https://www.slideshare.net/JamesKelly68/devnetops-overview-80716277

José Ruiz Cristina Ingeniero de telecomunicación por la UPM. (n.d.). Qué Es DevOps (y sobre todo qué no es devops). Paradigma. Retrieved from https://www.paradigmadigital.com/techbiz/que-es-devops-y-sobre-todo-que-no-es-devops/

Leandro Pavón Serrano Titulado en Ingeniería técnica de Telecomunicaciones y estudiante de Máster en Ingeniería de Telecomunicación. Actualmente trabajo en Ingeniería de clientes, especializado en redes L. A. N. y W. F. E. tecnológico y nostálgico de mi anterior dedicación a la visión artificial y la robótica. (2020, May 25). NetDevOps: El Nuevo Ingeniero de Red. Think Big. Retrieved from https://empresas.blogthinkbig.com/netdevops-nuevo-ingeniero-de-red/

NetDevOps: Network to code: Network Automation Solutions. Network to Code. (2021, November 9). Retrieved from https://www.networktocode.com/solutions/netdevops/

NetDevOps: Next-Generation Network Engineer. Speaker Deck. (n.d.). Retrieved from https://speakerdeck.com/pichuang/netdevops-next-generation-network-engineer

Netpalm introduction - part 1. Technology Blog Wim. (2020, April 14). Retrieved from https://blog.wimwauters.com/networkprogrammability/2020-04-14_netpalm_introduction_part1/

Tbotnz. (n.d.). Tbotnz/netpalm: Rest Based Network Device broker. GitHub. Retrieved from https://github.com/tbotnz/netpalm

Atlassian. (n.d.). Software de Colaboración para equipos de software, TI y empresa. Atlassian. Retrieved from https://www.atlassian.com/es

Business VPN: Next-Gen VPN. OpenVPN. (2021, November 17). Retrieved from https://openvpn.net/

The developer Data Platform. MongoDB. (n.d.). Retrieved from https://www.mongodb.com/home

Getting started with GNS3: GNS3 documentation. GNS3 Docs. (n.d.). Retrieved from https://docs.gns3.com/docs/

Github - Let's build from here. GitHub. (n.d.). Retrieved from https://github.com/

Open source security. pfSense® - World's Most Trusted Open Source Firewall. (n.d.). Retrieved from https://www.pfsense.org/

Rtsoneva. (n.d.). Documentación de VMware vSphere. VMware Docs Home. Retrieved from https://docs.vmware.com/es/VMware-vSphere/index.html

Blancarte, O. (n.d.). Introducción a la arquitectura de software. Un enfoque práctico. 2020.

Chou, E. (2020). Mastering python networking: Your One-stop solution to using python for network automation, programmability, and DevOps. Packt Publishing.

Griffiths, M. (2018). Pmi-Acp Exam Prep: A Course in a book for passing the Pmi Agile Certified practitioner (Pmi-Acp) exam. RMC Publications.

OKASHA, K. A. R. I. M. (2020). Network automation cookbook: Proven and actionable recipes to automate and manage network devices ... using Ansible. PACKT Publishing Limited.

Parga Carlos Jiménez de. (2021). Uml, Arquitectura de aplicaciones en java, C++ Y Phython. RA-MA.

Project Management Institute. (2017). Guía práctica de Ágil.

Ratan, A., Chou, E., Kathiravelu, P., & Faruque, S. M. O. (2019). Python Network Programming: Conquer all your networking challenges with the powerful Python language. Packt Publishing Ltd.

Tragura, S. J. (2022). Building python microservices with fastapi: Build secure, scalable, and structured python microservices from design concepts to infrastructure. Packt Publishing.

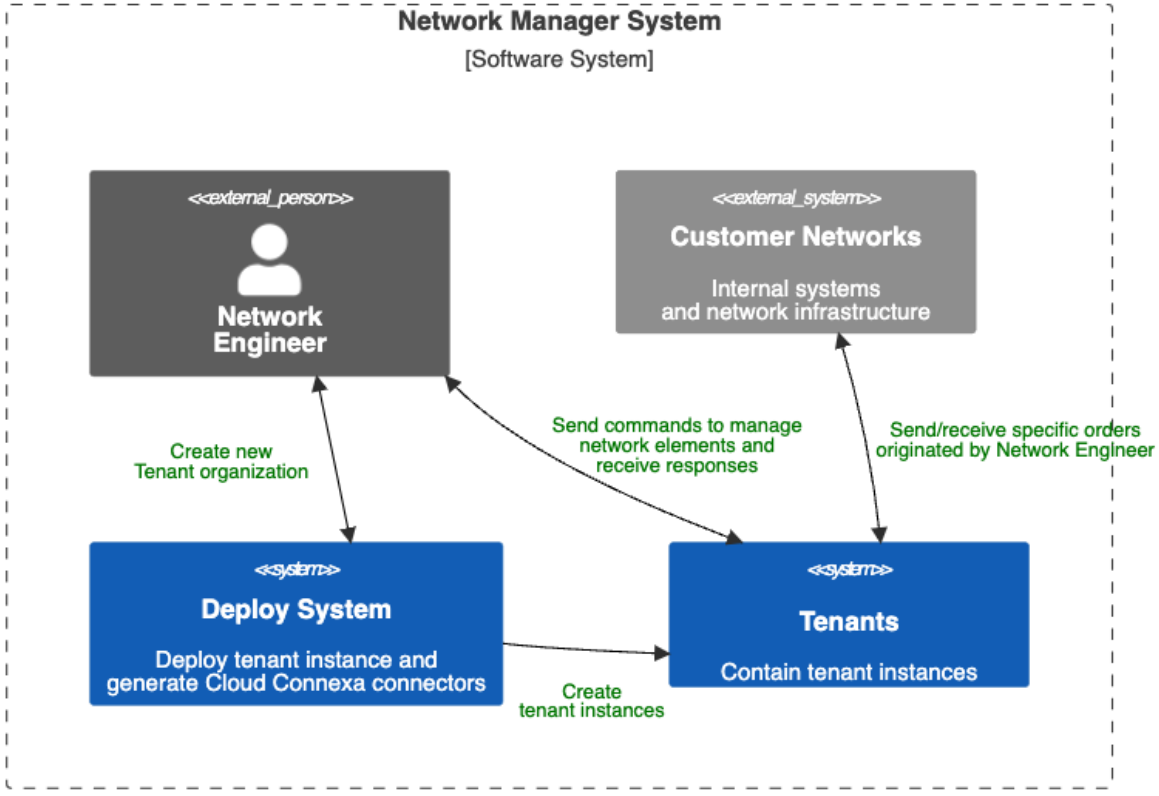
Gartner_Inc. (n.d.). Best networking practices in a DevOps World. Gartner. Retrieved from https://www.gartner.com/en/documents/3984713

Gartner_Inc. (n.d.). Market Guide for Network Automation. Gartner. Retrieved from https://www.gartner.com/document/4011752

21. Anexos

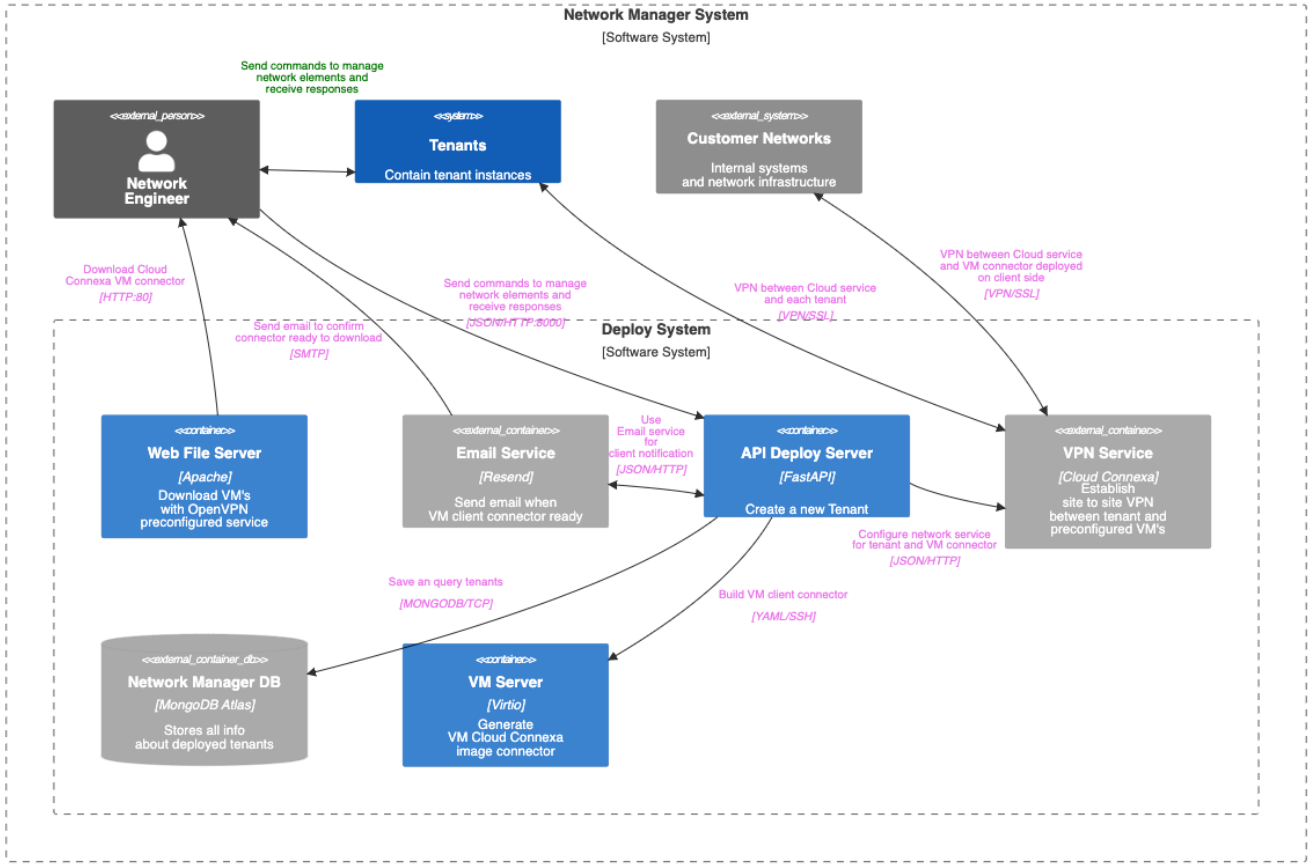
21.1. Diagramas

21.1.1. Diagrama de contexto

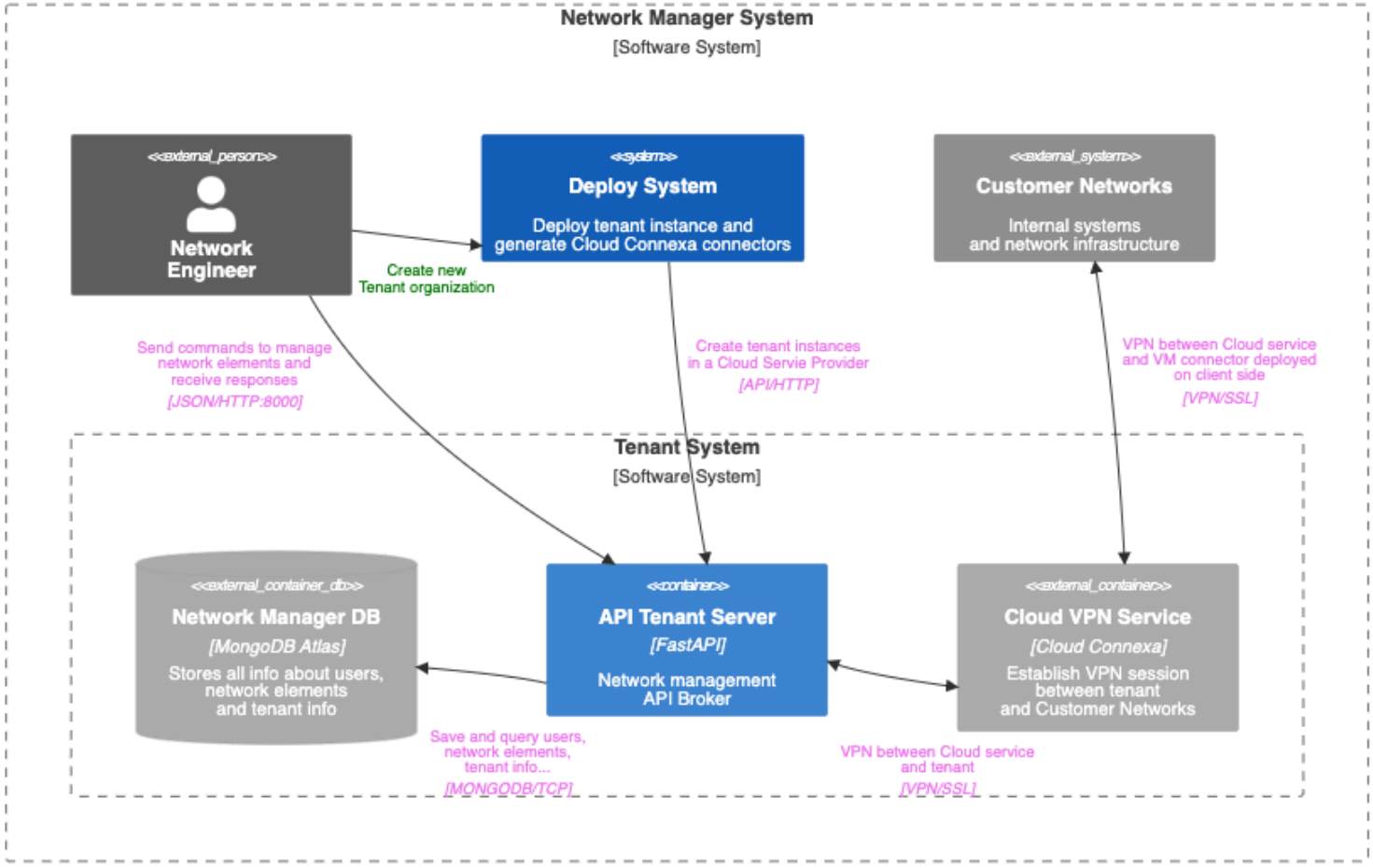


21.1.2. Diagrama de contenedores

21.1.2.1. Deploy System

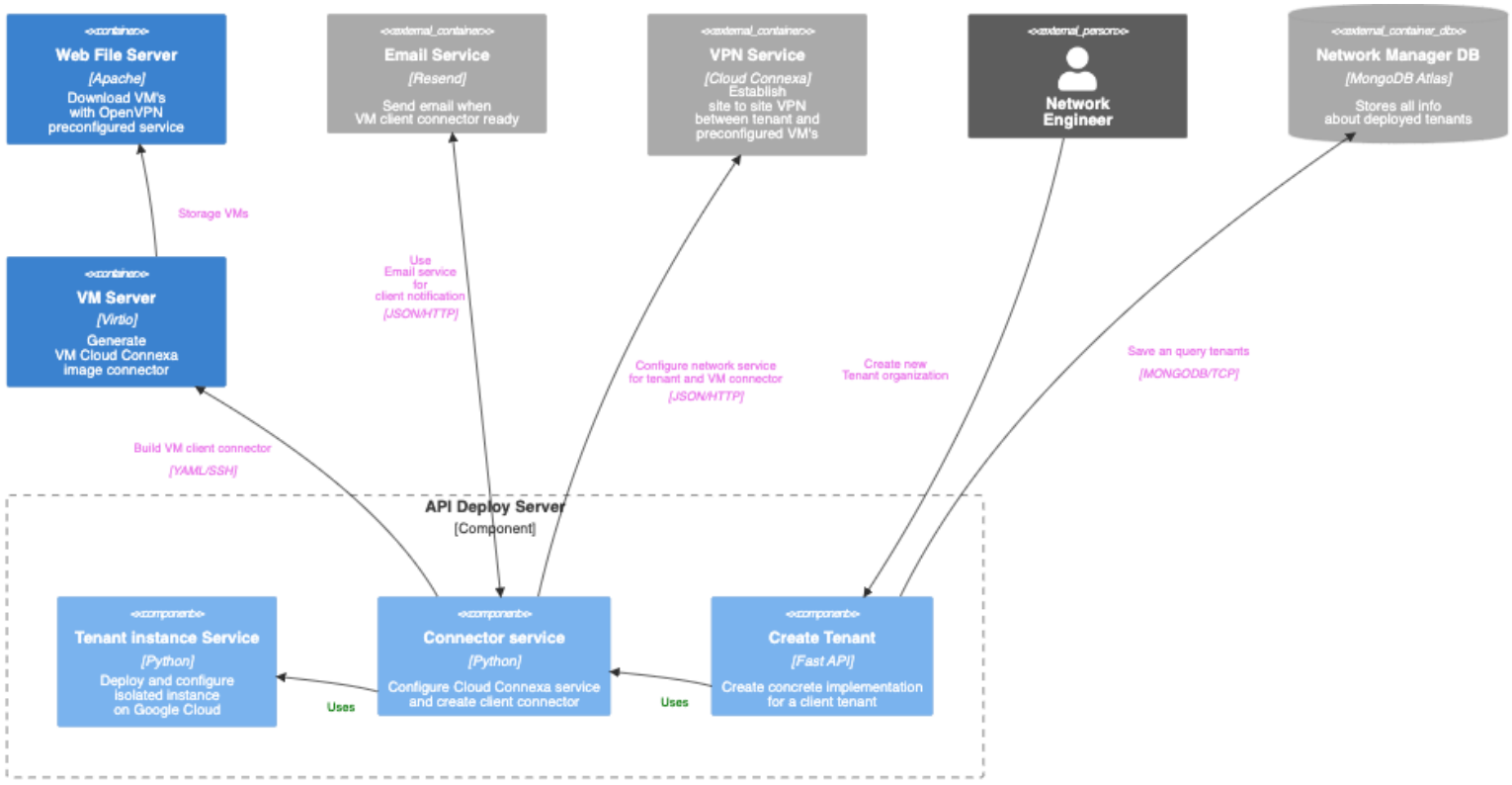


21.1.2.2. Tenants System

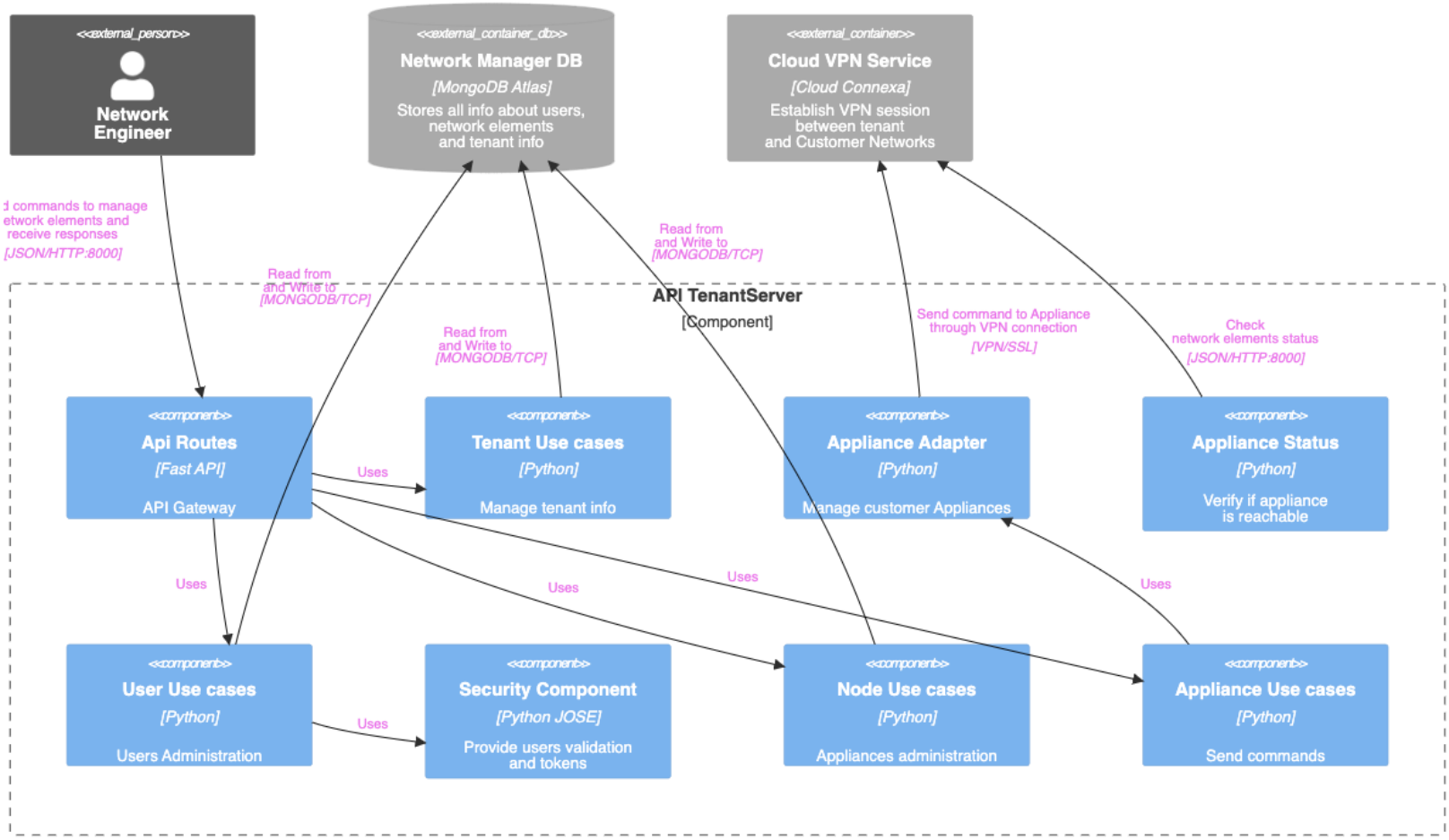


21.1.3. Diagrama de componentes

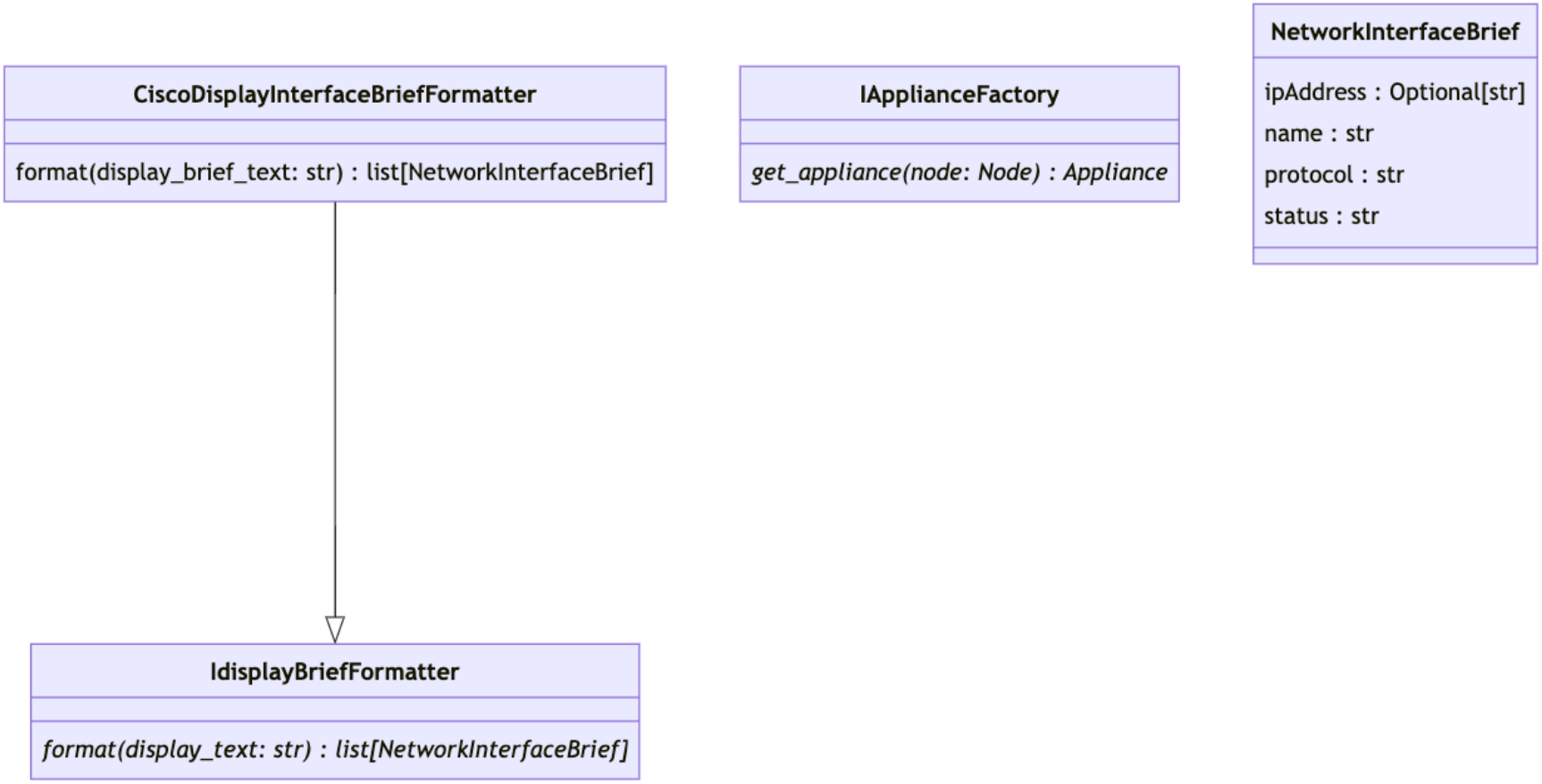
API Deploy Server

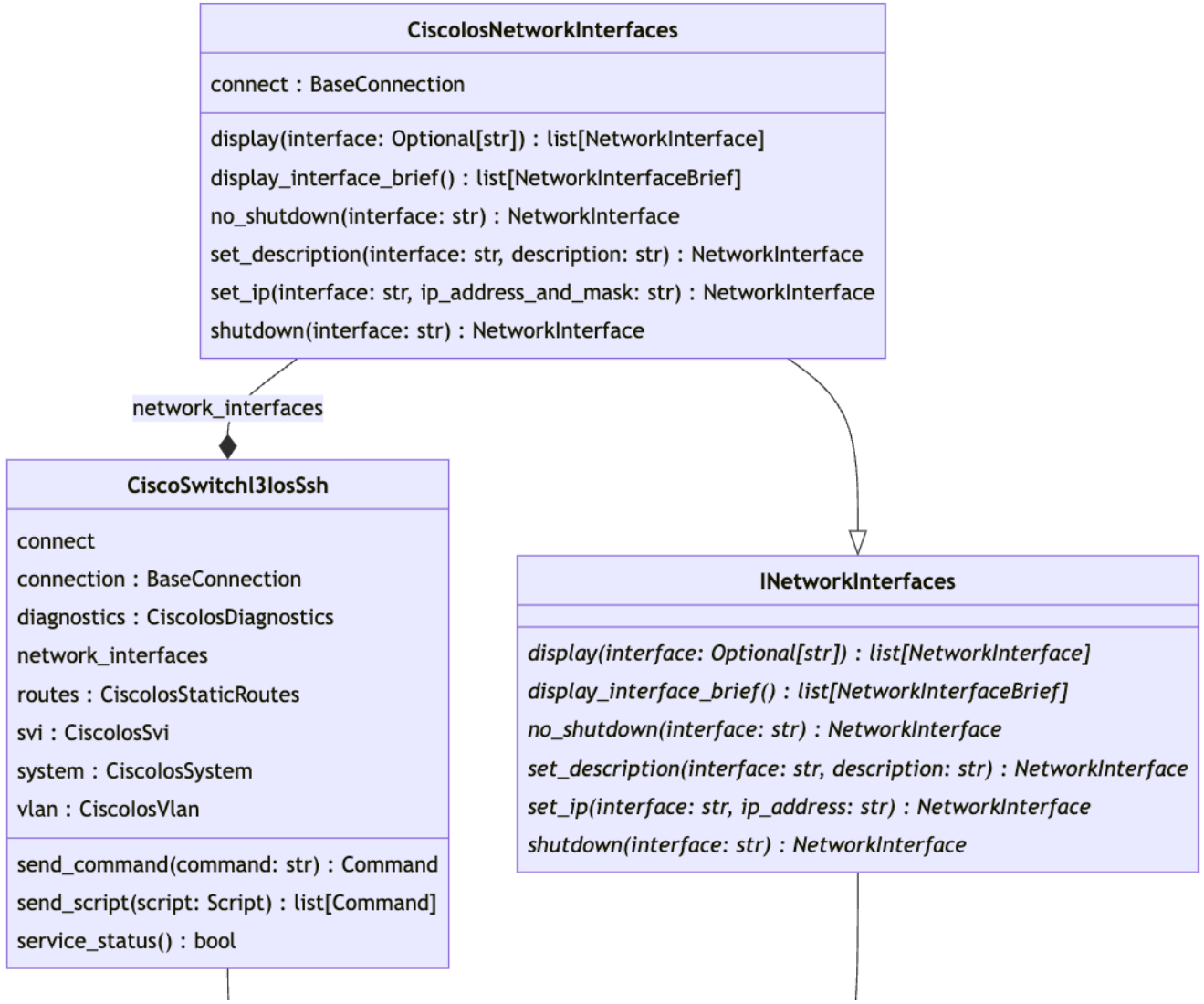


API Tenant Server



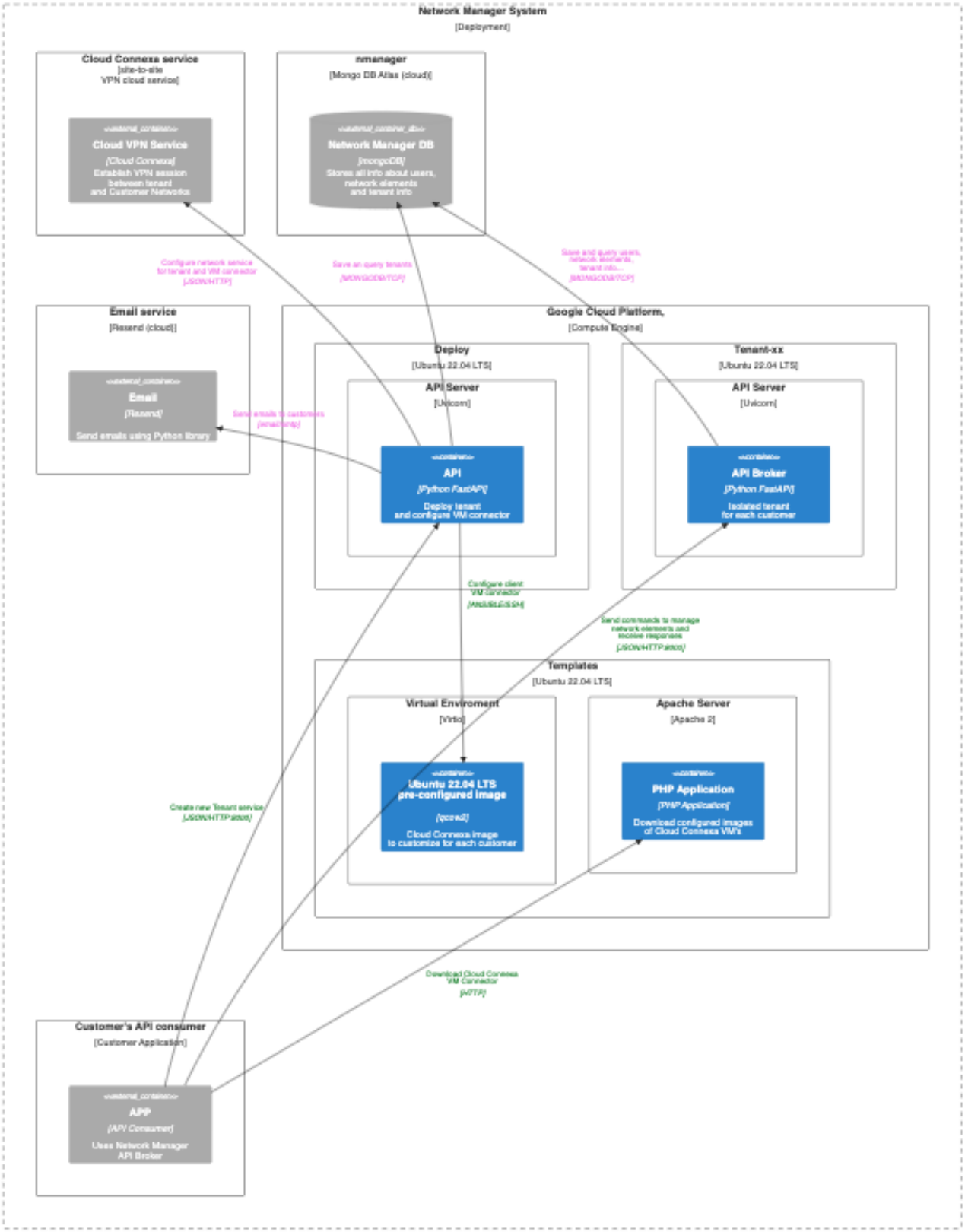
21.1.4. Diagrama de código




CiscolosNetworkInterfaces
connect : BaseConnection
display(interface: Optional[str]) : list[NetworkInterface]
display_interface_brief() : list[NetworkInterfaceBrief]
no_shutdown(interface: str) : NetworkInterface
set_description(interface: str, description: str) : NetworkInterface
set_ip(interface: str, ip_address_and_mask: str) : NetworkInterface
shutdown(interface: str) : NetworkInterface
network_interfaces
CiscoSwitchI3IosSsh
connect
connection : BaseConnection
diagnostics : CiscolosDiagnostics
network_interfaces
routes : CiscolosStaticRoutes
svi : CiscolosSvi
system : CiscolosSystem
vlan : CiscolosVlan
send_command(command: str) : Command
send_script(script: Script) : list[Command]
service_status() : bool
INetworkInterfaces
display(interface: Optional[str]) : list[NetworkInterface]
display_interface_brief() : list[NetworkInterfaceBrief]
no_shutdown(interface: str) : NetworkInterface
set_description(interface: str, description: str) : NetworkInterface
set_ip(interface: str, ip_address: str) : NetworkInterface
shutdown(interface: str) : NetworkInterface

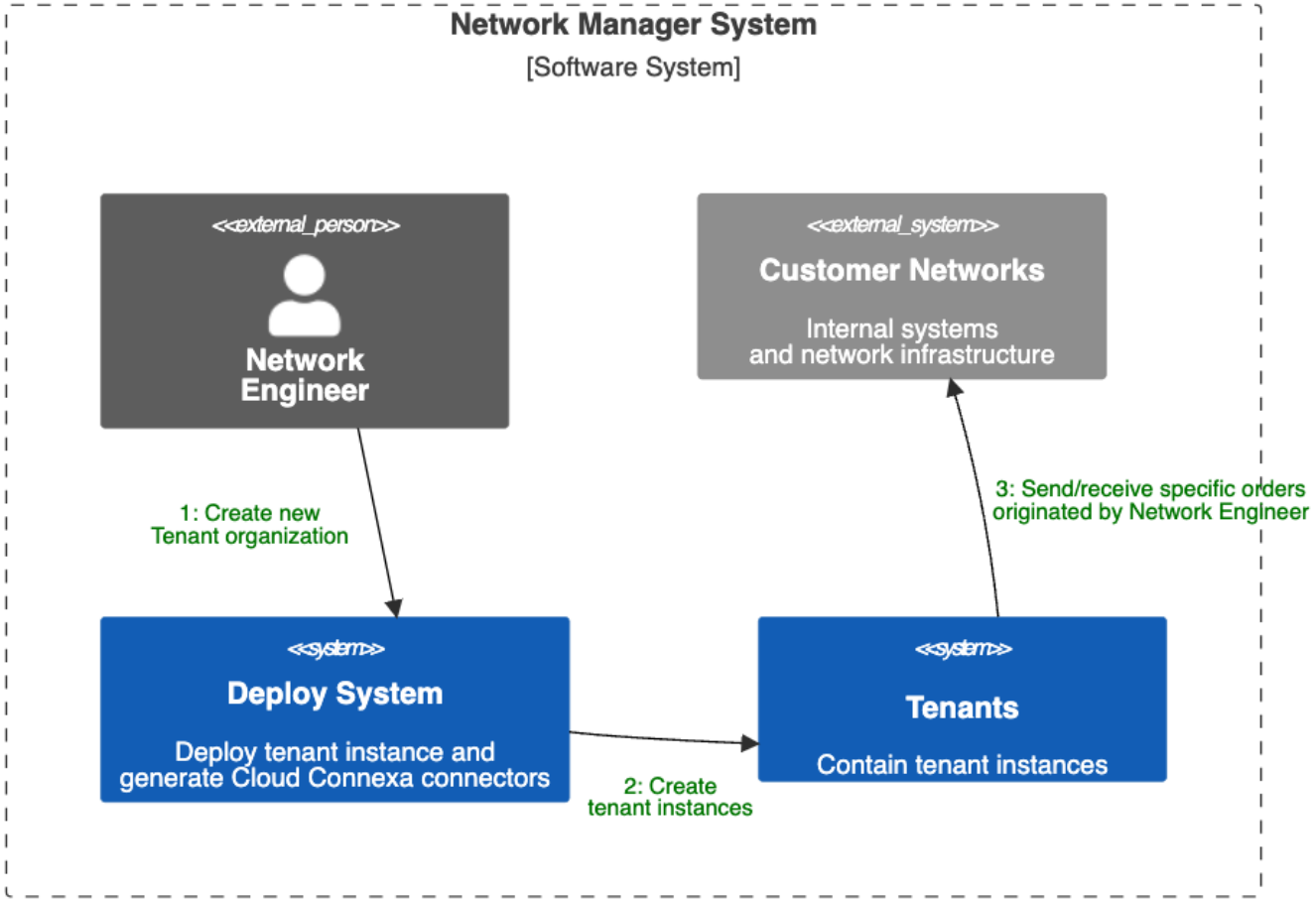

_network_interfaces
Appliance
diagnostics
network_interfaces
node
routes
svi
system
vlan
arp() : list[Arp]
connect() : Any
create_svi(vlan_id: str) : NetworkInterface
delete_svi(vlan_id: str) : list[NetworkInterface]
display(interface: Optional[str]) : list[NetworkInterface]
display_interface_brief() : list[NetworkInterfaceBrief]
display_vlan_brief() : list[VlanBrief]
display_vlan_trunk() : list[VlanTrunk]
no_shutdown(interface: str) : NetworkInterface
ping(ping_command_fields: PingCommandFields) : Ping
remove_vlan(interface: str | None, vlan_id: Union[int, str]) : VlanResume
restart() : Restart
routing_table() : list[Route]
send_command(command: str) : Command
send_script(script: Script) : list[Command]
service_status() : bool
set_default_gateway(ip: str) : list[Route]
set_description(interface: str, description: str) : NetworkInterface
set_hostname(new_hostname: str) : Hostname
set_ip(interface: str, ip_address: str) : NetworkInterface
set_static_route(ip_and_mask_in_cidr: str, ip_next_hop: str) : list[Route]
set_svi_ip(svi: str, ip_address: str) : NetworkInterface
set_vlan_access(interface: str, vlan_id: int, encapsulation: str) : list[VlanBrief]
set_vlan_native(interface: str, vlan_id: int, encapsulation: str) : list[VlanTrunk]
set_vlan_trunk(interface: str, vlan_id: list[int], encapsulation: str) : list[VlanTrunk]
shutdown(interface: str) : NetworkInterface
trace_route(trace_route_command_fields: TraceRouteCommandFields) : TraceRoute
appliance
DisplayBriefNetworkInterface
appliance
display_brief() : list[NetworkInterfaceBrief]
IDisplayBriefNetworkInterface
display_brief() : list[NetworkInterfaceBrief]

21.1.5. Diagrama de despliegue

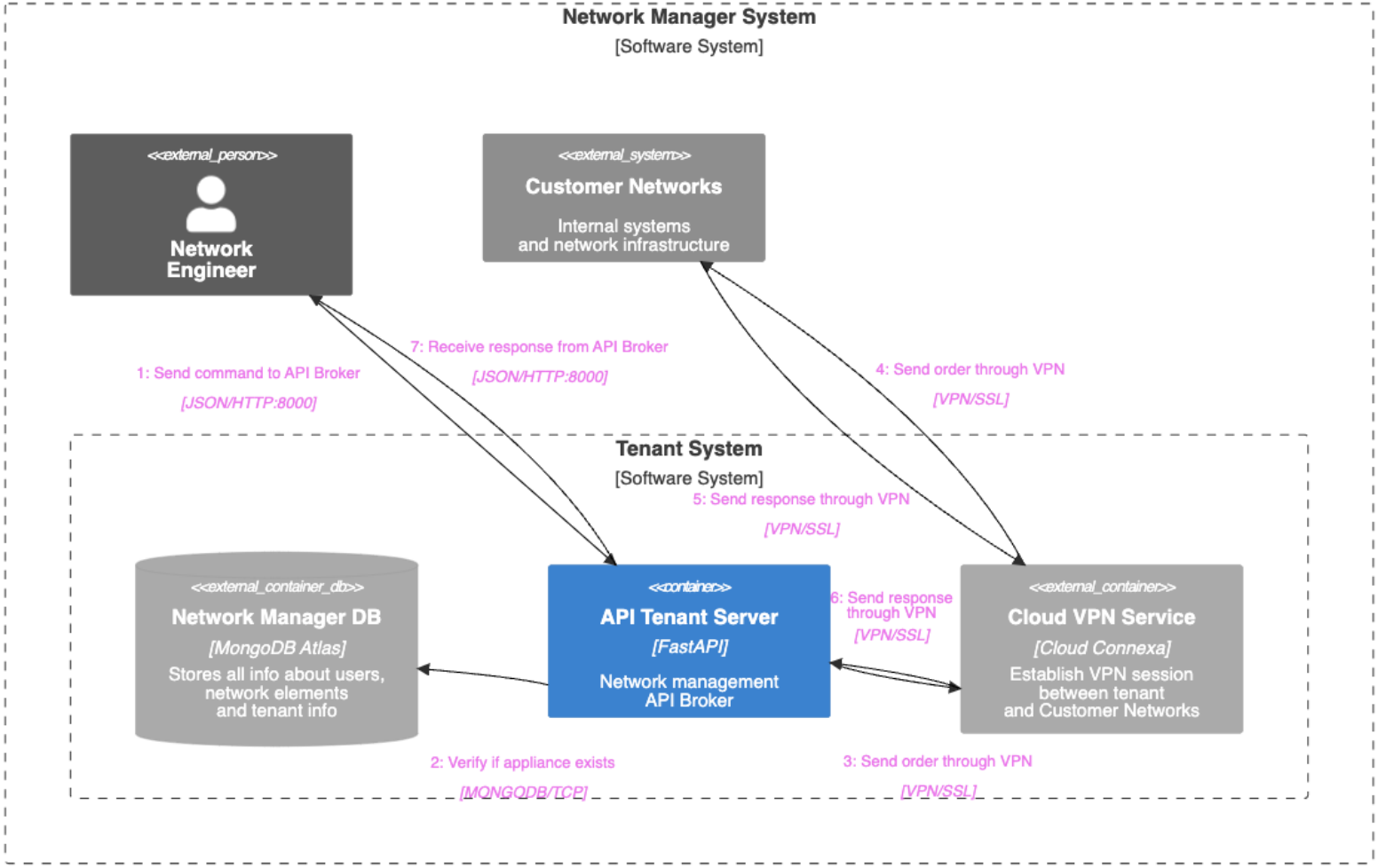


21.1.6. Diagramas dinámicos

21.1.6.1. Crear un nuevo tenant



21.1.6.2. Gestión de elementos de red



21.2. Documentación de la API

A la hora de proporcionar una documentación, es importante que esta se encuentre actualizada. Para ello, se ha publicado la API en Postman, para que se pueda consultar de forma online. Esta disponible en Network Manager API Documentation

Además, cada instancia dispone de su propia documentación, dado que el framework de FastAPI implementa tanto *swagger* como *redoc*. Estas pueden ser consultadas apuntando la los endpoints /docs y /redoc respectivamente.

21.3. Guia de Instalación y configuración de equipos virtuales en GNS3

21.3.1. Instalacion de PfSense

A continuación, se muestra una guía para la configuración inicial de pfsense

Una vez que arranca la imagen, debemos ir seleccionando las opciones que se van mostrando en las siguientes capturas.
La primera pantalla, debemos seleccionar la opción "*Install*"

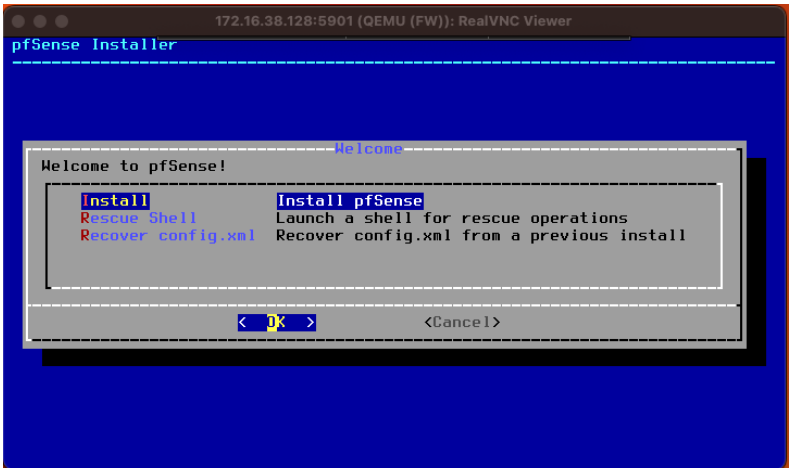


Figura 21.3.1-1 Guia instalación Pfsense. Install pfsense

Se selecciona el método de particionado

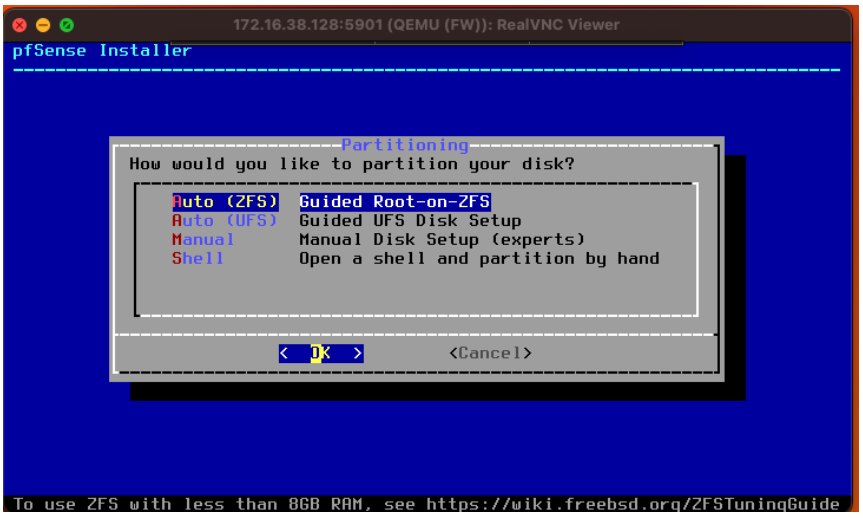


Figura 21.3.1-2 Guía instalación Pfsense. Partition disk

Procedemos a la instalación

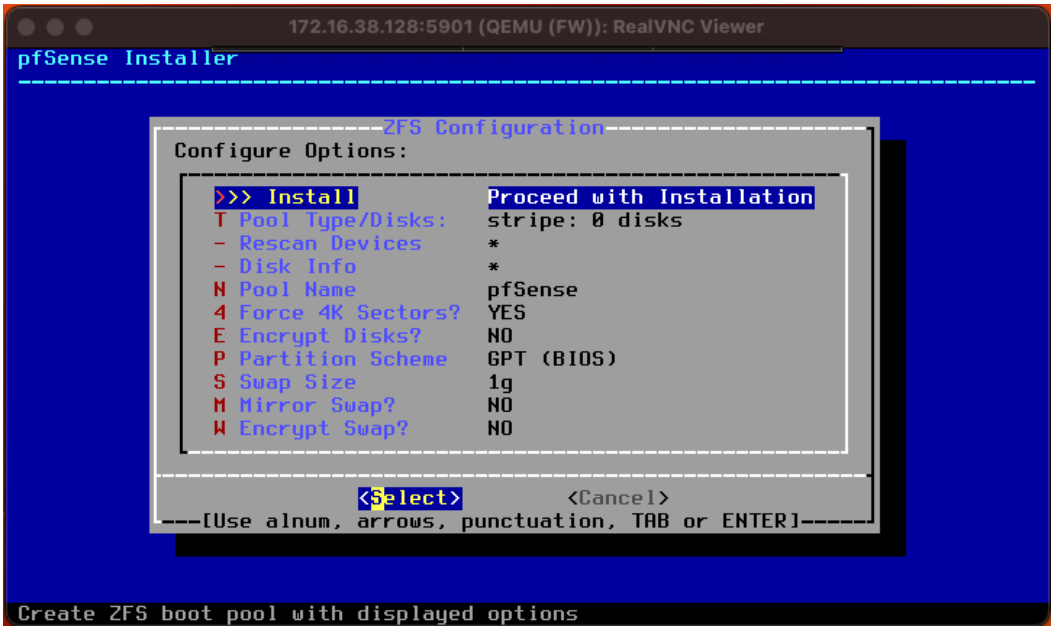


Figura 21.3.1-3 Guía instalación Pfsense. Configure options

Seleccionamos “stripe” como tipo de dispositivo virtual

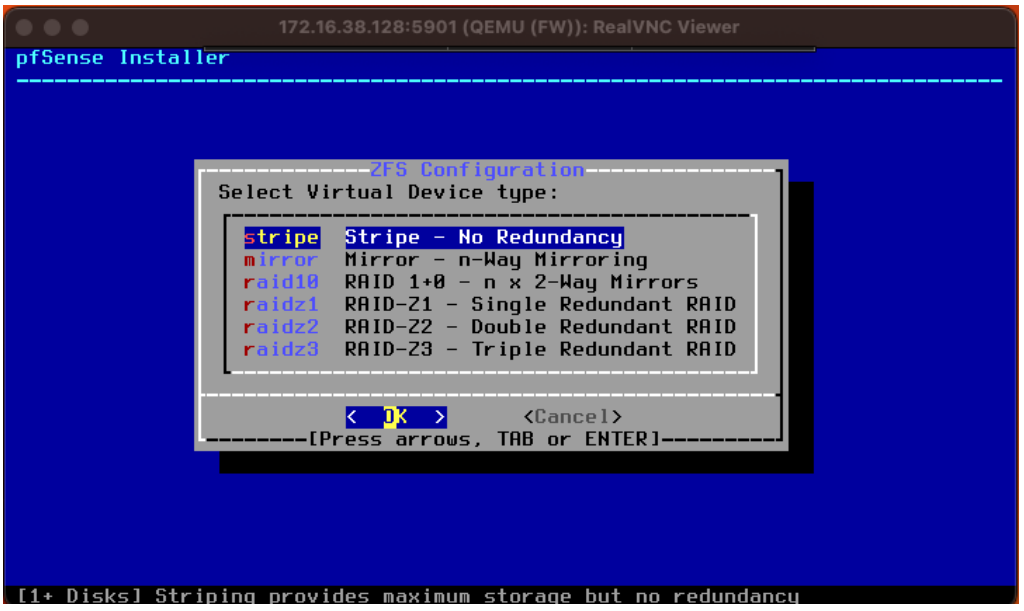


Figura 21.3.1-4 Guía instalación Pfsense. Virtual device type

Seleccionamos el tipo de bloque

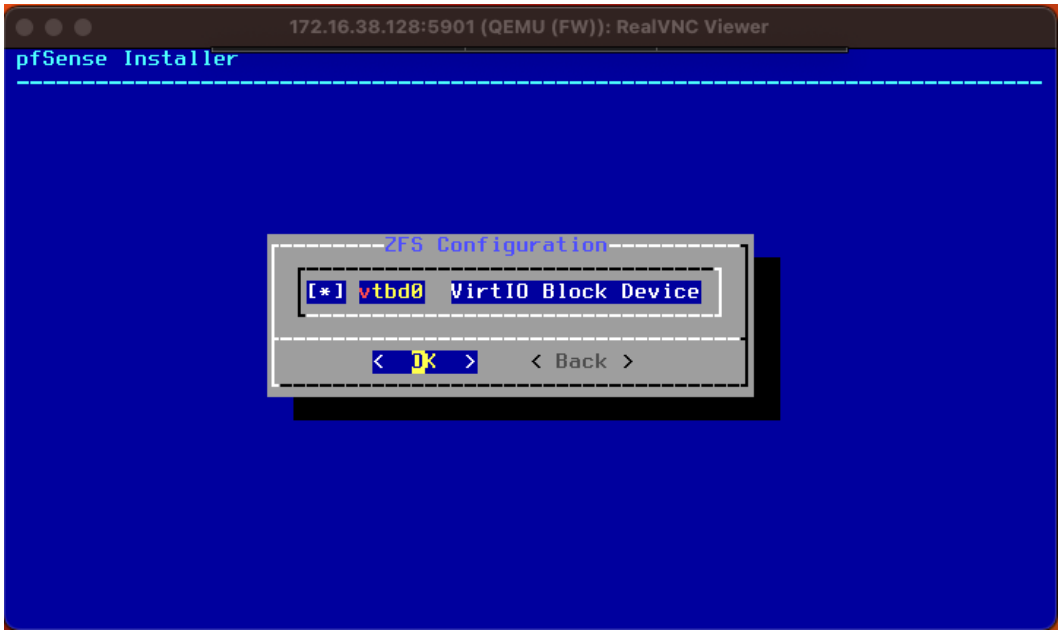


Figura 21.3.1-5 Guía instalación Pfsense. ZFS Configuration

Aplicamos la configuración seleccionada

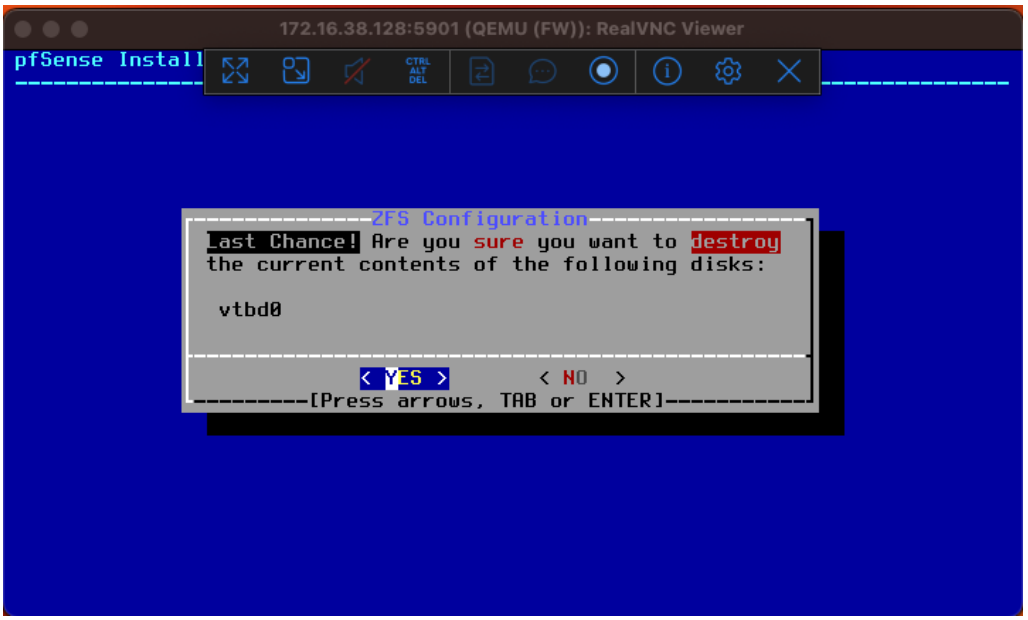


Figura 21.3.1-6 Guía instalación Pfsense. ZFS configuration

Reiniciamos el firewall

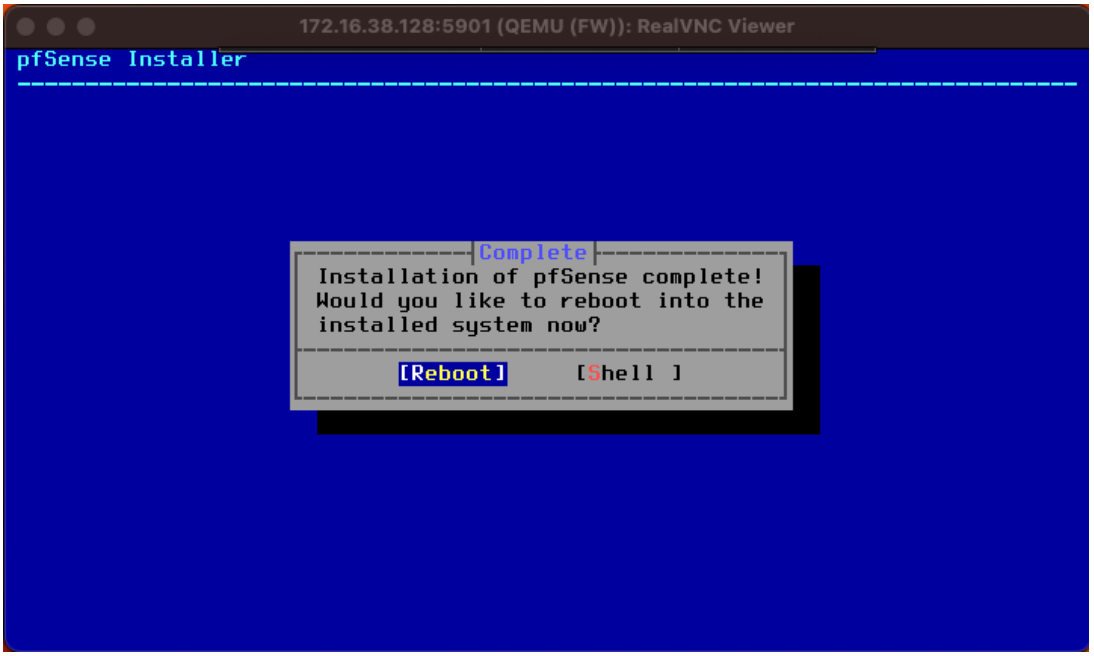


Figura 21.3.1-7 Guía instalación Pfsense. Installation complete

Tras el reinicio llegamos a la siguiente pantalla con la configuración por defecto. En este caso concreto ha obtenido la configuración por DHCP

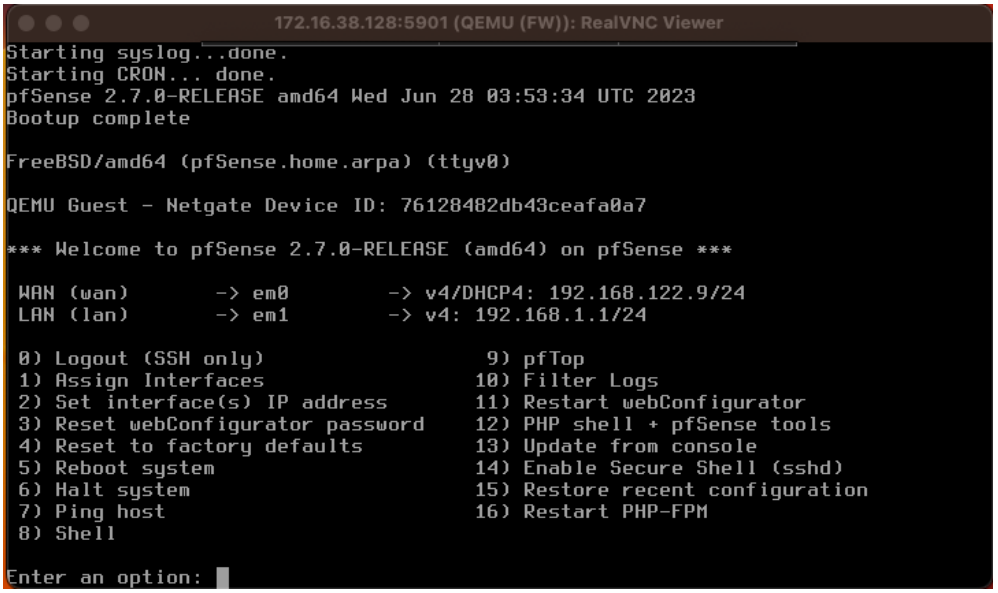


Figura 21.3.1-8 Guía instalación Pfsense. Consola tras reinicio

Se realiza PING a la IP 8.8.8.8 para verificar que se dispone de salida a Internet

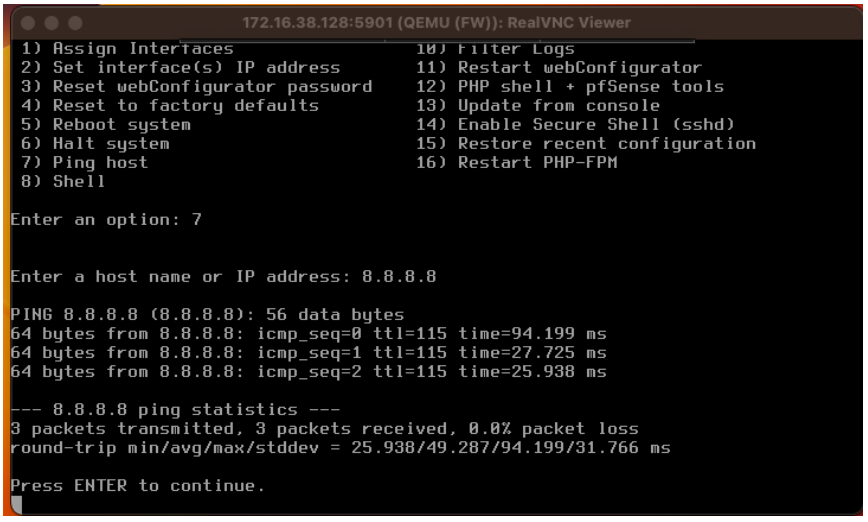


Figura 21.3.1-9 Guía instalación Pfsense. Prueba PING conectividad

Seleccionamos el interfaz WAN, deshabilitando la configuración IP vía DHCP, y asignando una IP dentro del rango como estática.

IP:192.168.4.200/22
GW: 192.168.122.1

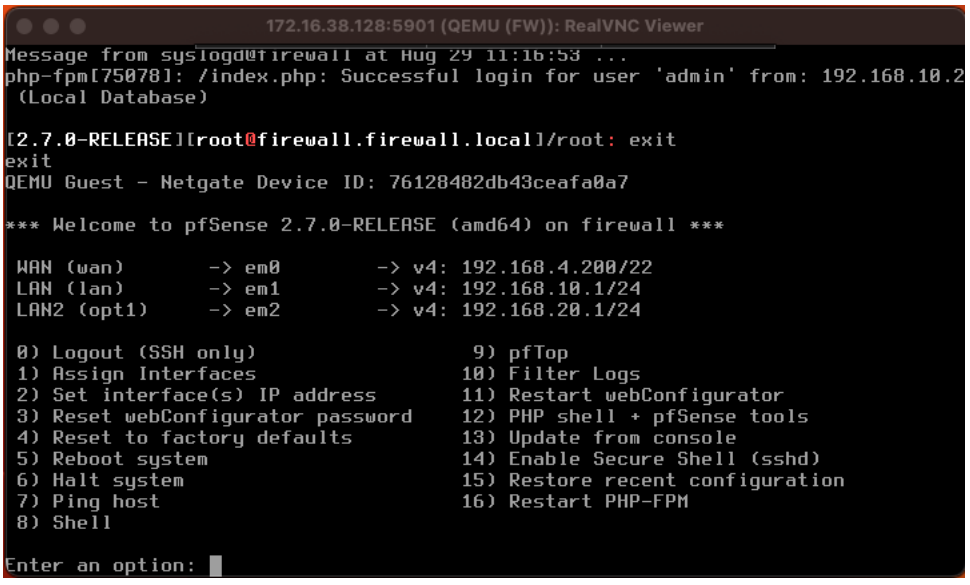


Figura 21.3.1-10 Guía instalación Pfsense. Configuración interfaz WAN

Se realiza de nuevo ping a la IP 8.8.8.8 para verificar que se sigue saliendo a Internet.

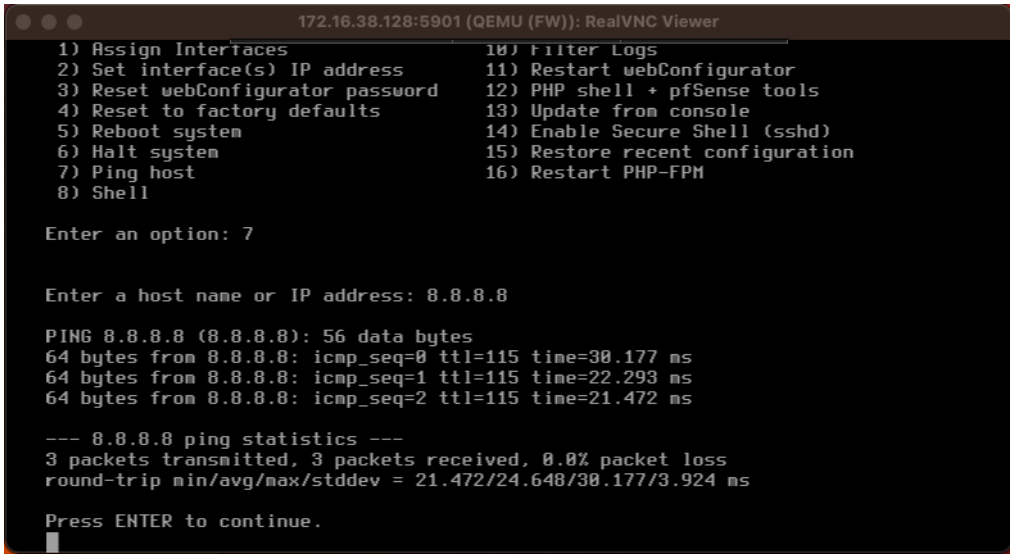


Figura 21.3.1-11 Guía instalación Pfsense. Prueba PING (2)

Asignamos al interfaz LAN (em1) la IP fija 192.168.10.1/24 , y en la em2 la IP 192.168.20.1/24

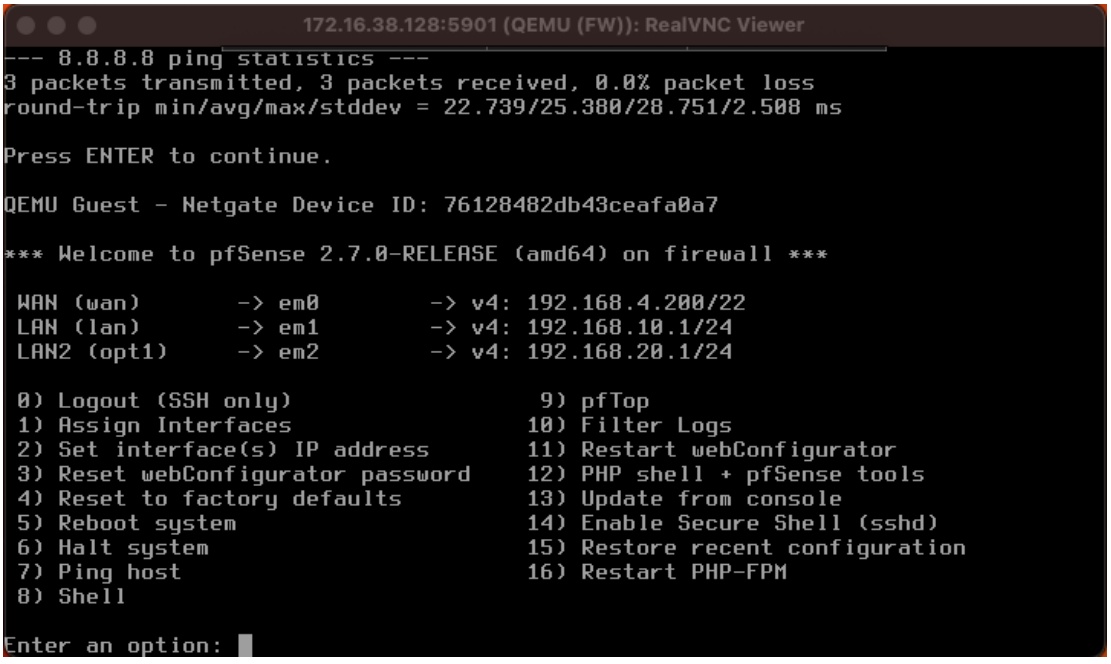


Figura 21.3.1-12 Guía instalación Pfsense. Configuración interfaz LAN en2

Ahora le asignamos un terminal Web en GNS3, realizamos la conexión con el firewall a través del interfaz en1 le configuramos a este una IP fija dentro del mismo rango de IPs que em1.

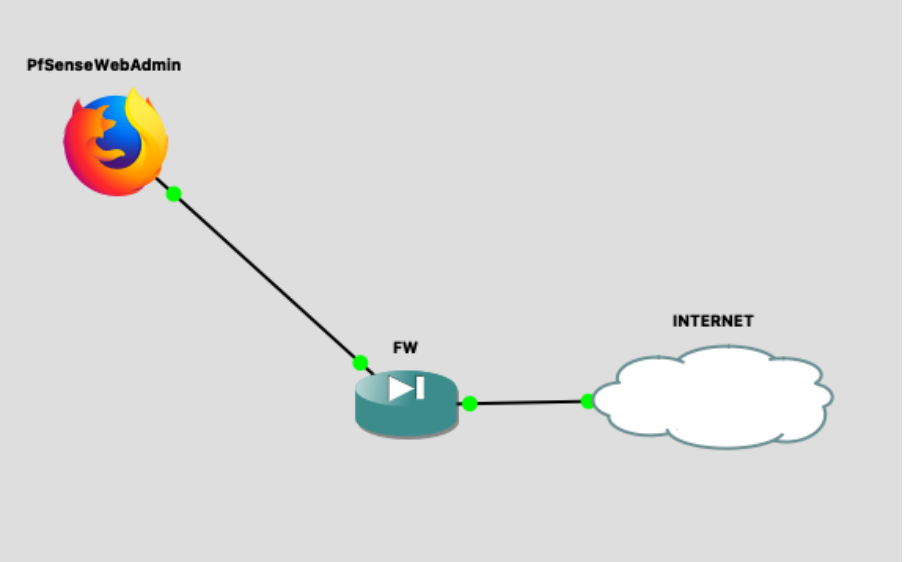


Figura 21.3.1-13 Guía instalación Pfsense. Diagrama conexionado en GNS3

Accedemos a la configuración de red

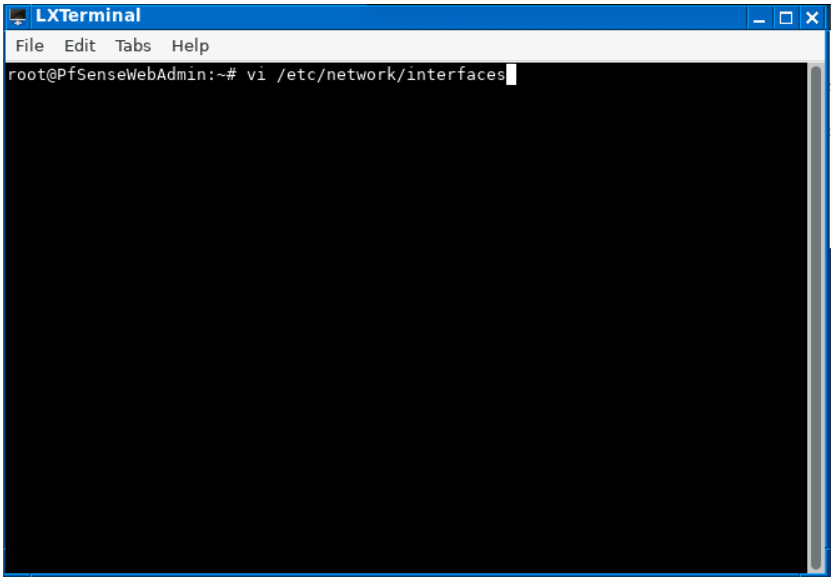


Figura 21.3.1-14 Guía instalación Pfsense. Configuración adaptador de red WebAdmin (1)

Modificamos la configuración para que se vea con el interfaz del firewall al que está conectado

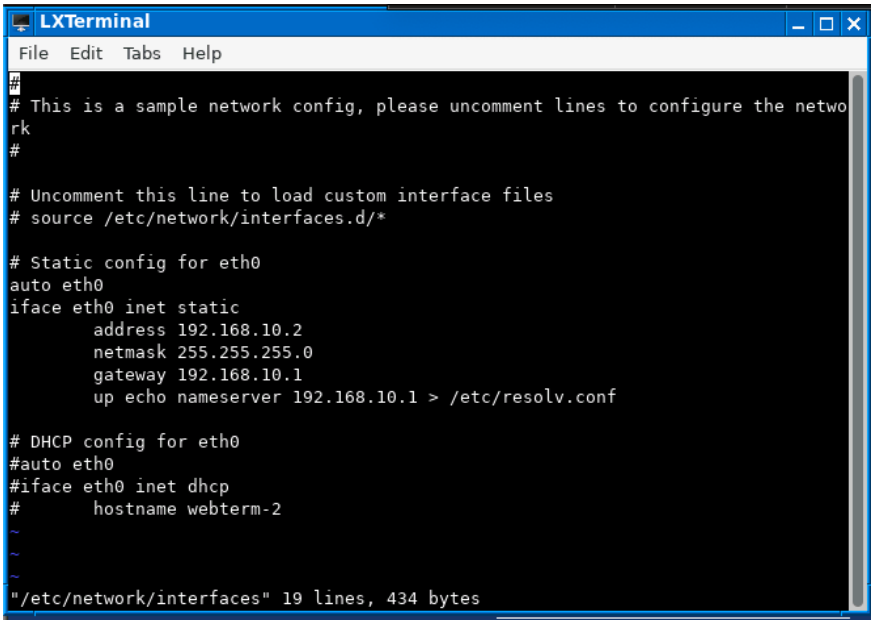


Figura 21.3.1-15 Guía instalación Pfsense. Configuración adaptador de red WebAdmin (2)

Ahora accedemos a través del navegador Web con las credenciales por defecto.

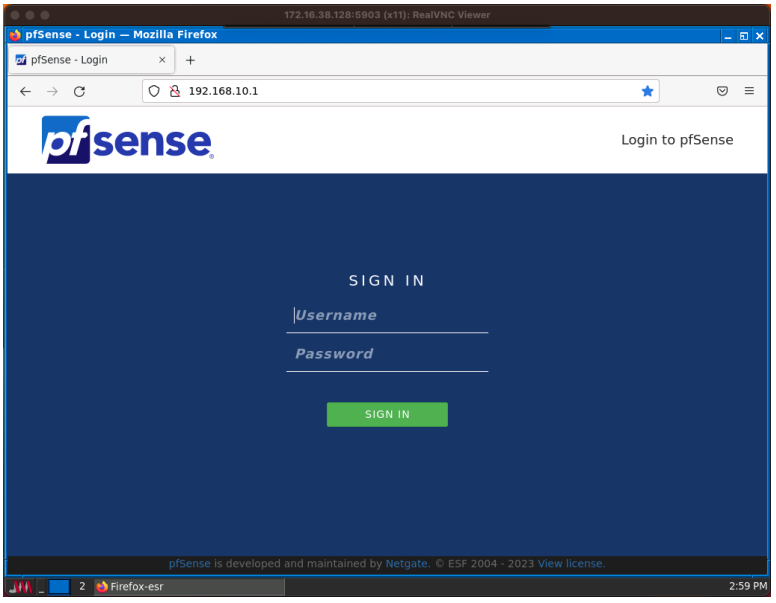


Figura 21.3.1-16 Guía instalación Pfsense. Acceso a Firewall por Web

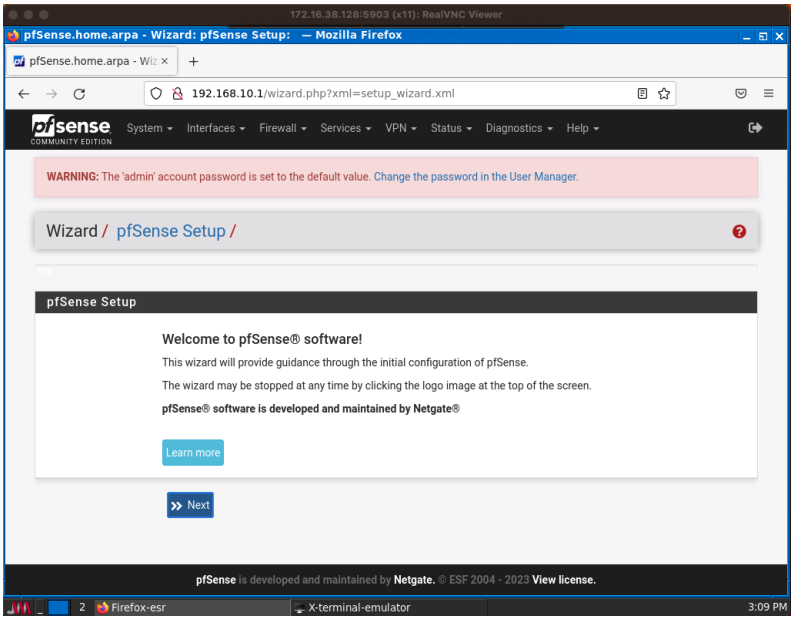


Figura 21.3.1-17 Guía instalación Pfsense. Configuración Web Firewall (1)

A continuación, pasamos a realizar la configuración inicial

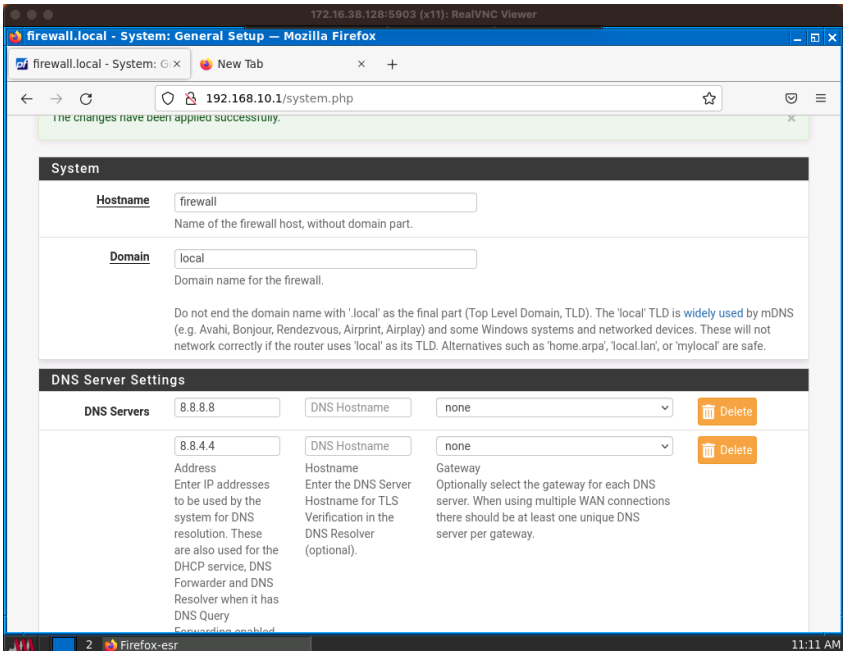


Figura 21.3.1-18 Guía instalación Pfsense. Configuración Web Firewall (2)

Hostname: firewall
Domain: local

Primary DNS Server: 8.8.8.8
Secondary DNS Server: 8.8.4.4

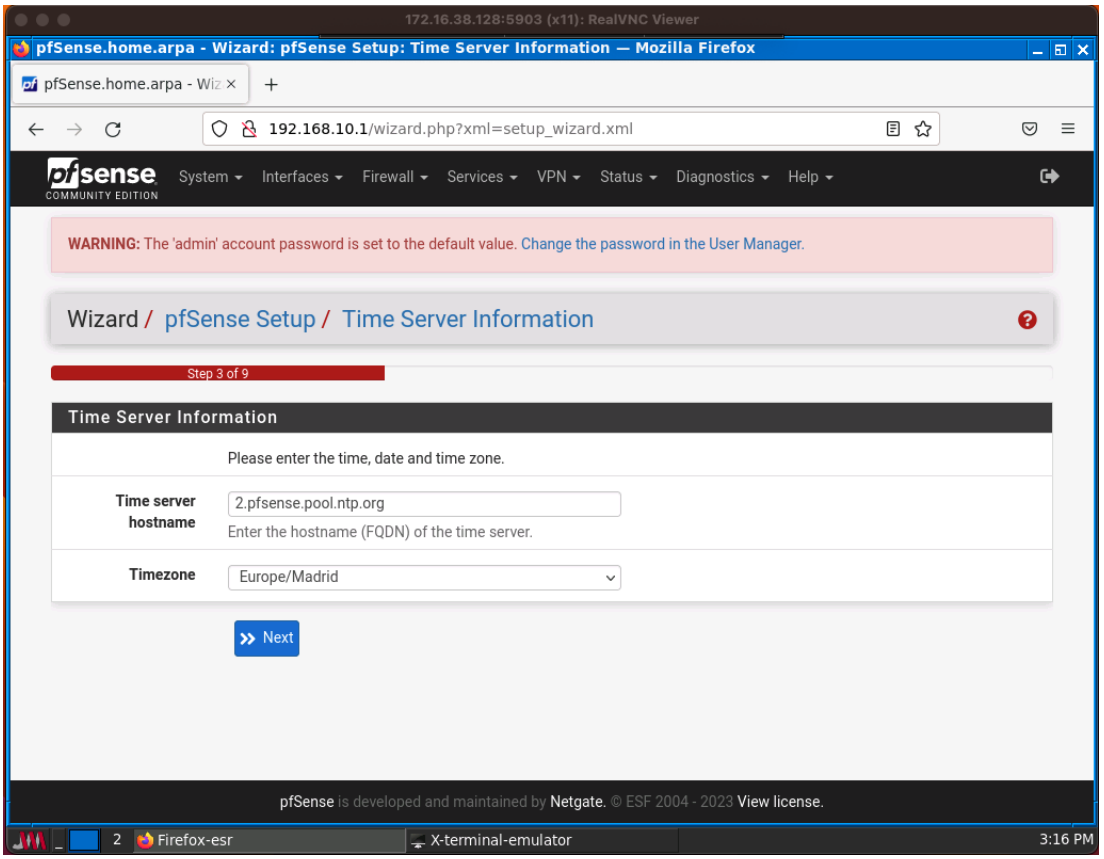


Figura 21.3.1-19 Guía instalación Pfsense. Configuración Web Firewall (3)

Time server hostname: 2.pfsense.pool.ntp.org
Timezone: Europe/Madrid

La configuración WAN la dejamos tal cual la tenemos actualmente configurada

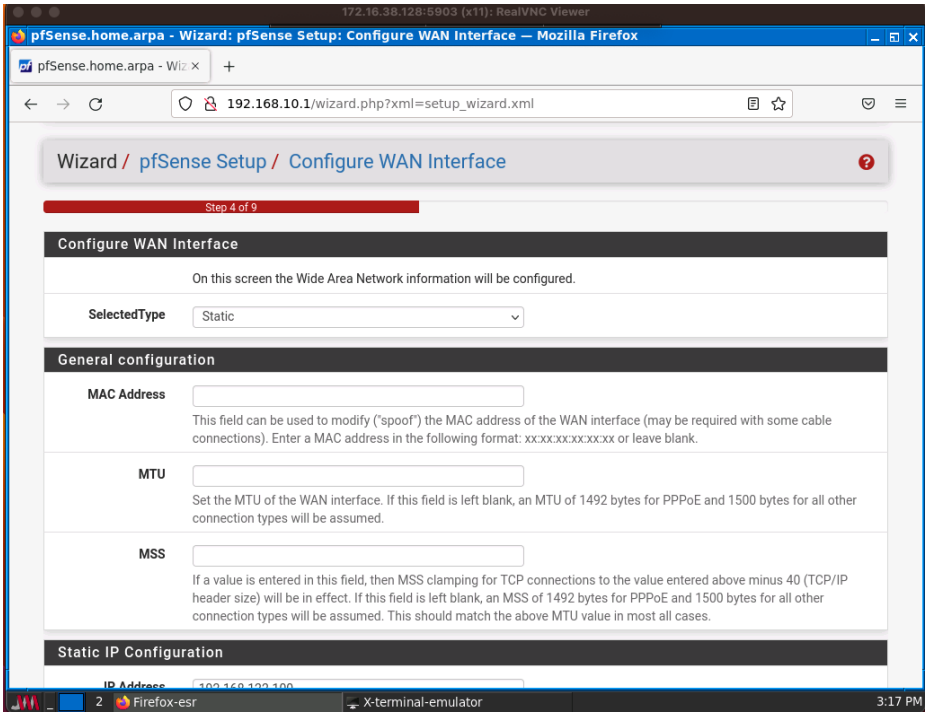


Figura 21.3.1-20 Guía instalación Pfsense. Configuración Web Firewall (4)

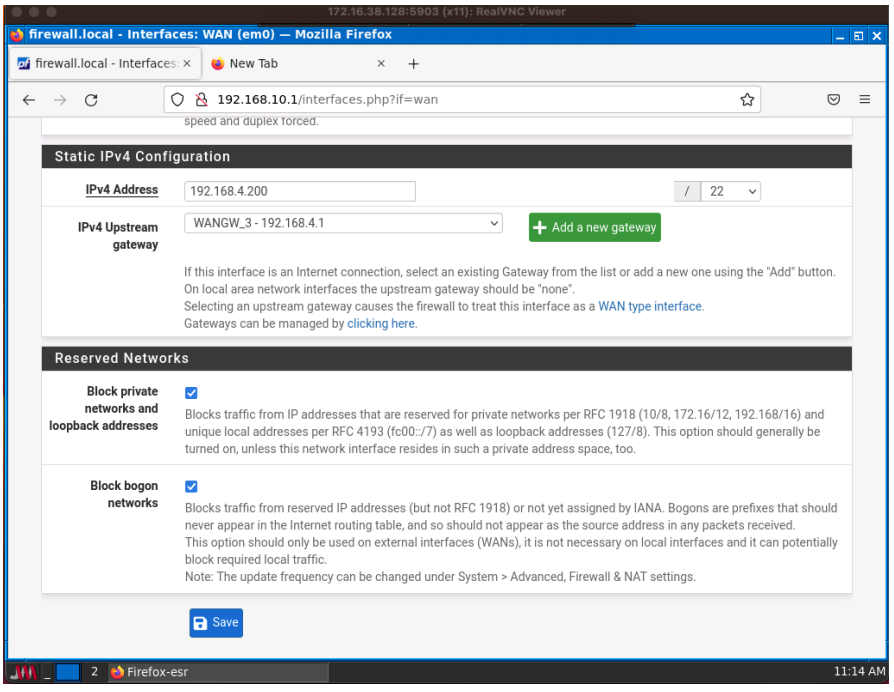


Figura 21.3.1-21 Guía instalación Pfsense. Configuración Web Firewall (5)

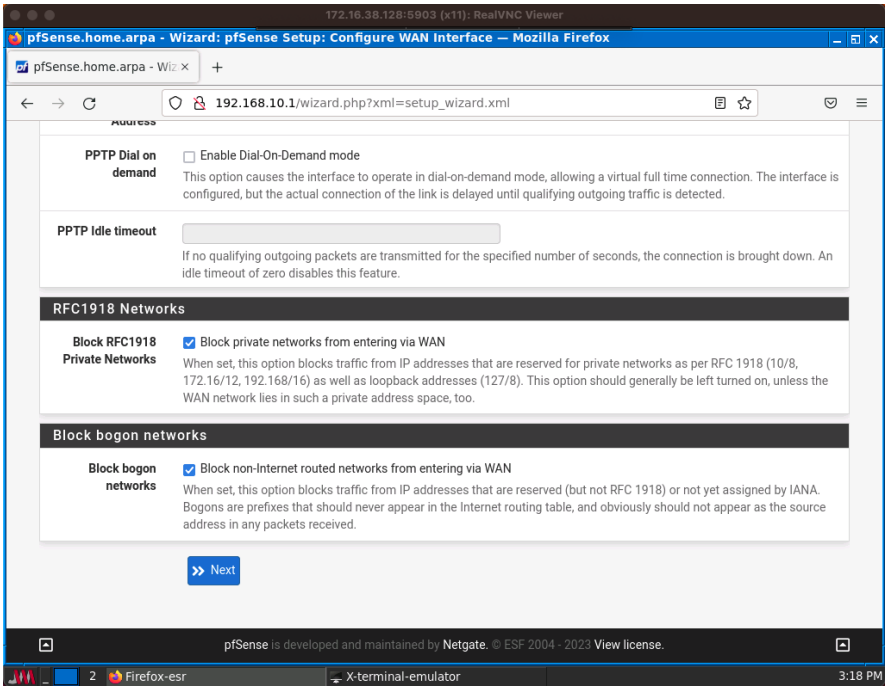


Figura 21.3.1-22 Guía instalación Pfsense. Configuración Web Firewall (6)

Lo mismo para el interfaz LAN em1

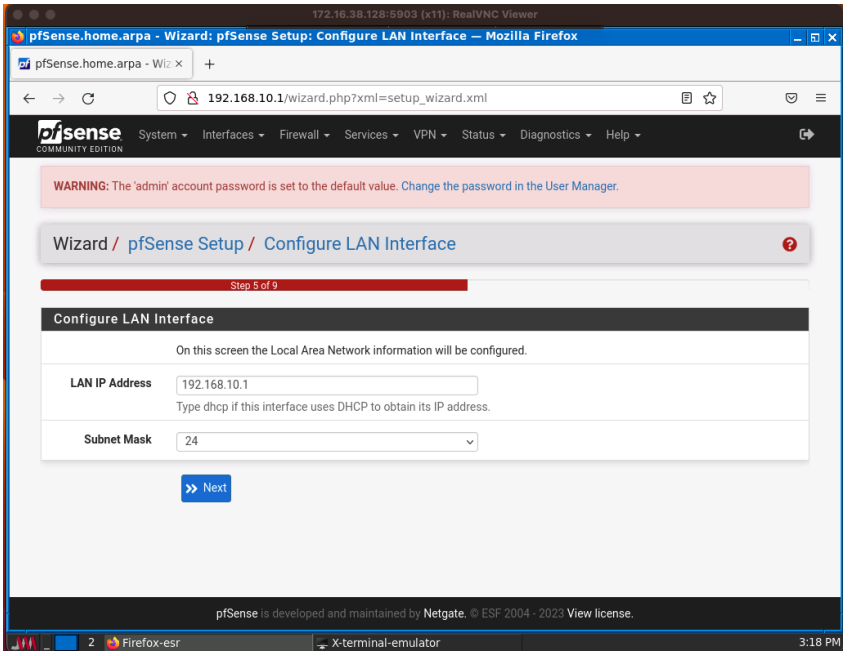


Figura 21.3.1-23 Guía instalación Pfsense. Configuración Web Firewall (7)

Cómo password dejamos actualmente el que viene por defecto (pfsense)

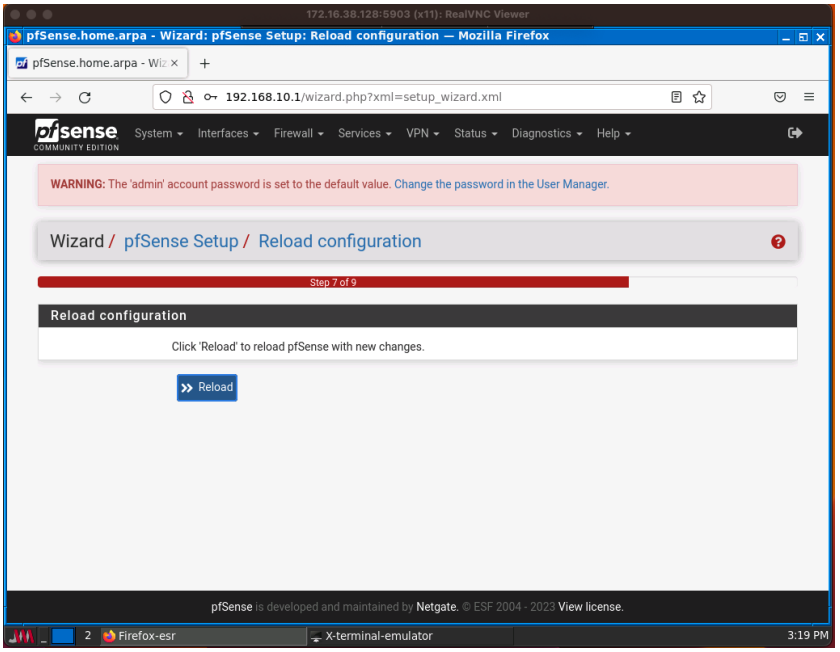


Figura 21.3.1-24 Guía instalación Pfsense. Configuración Web Firewall (8)

Y se da por finalizada la configuración inicial

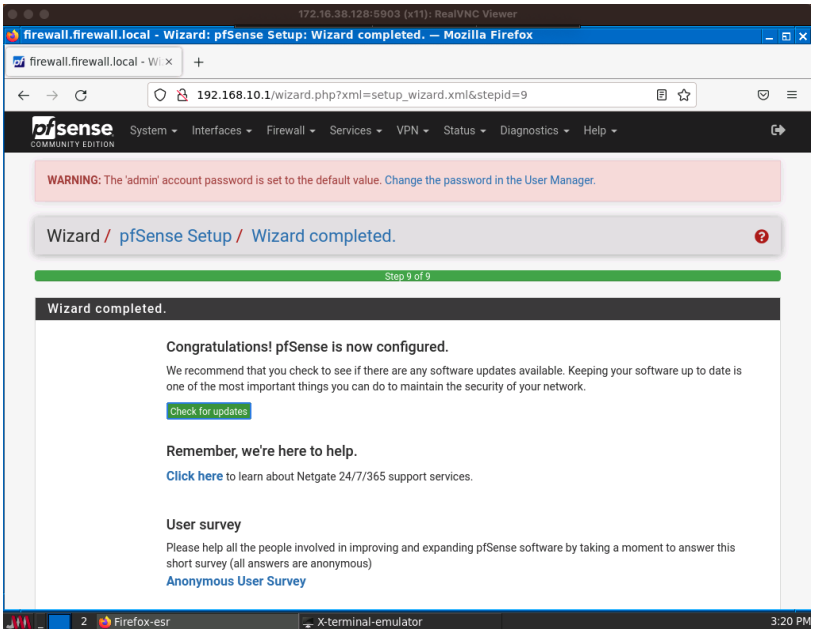


Figura 21.3.1-25 Guía instalación Pfsense. Configuración Web Firewall (9)

Para probar la salida a Internet desde el navegador, falta configurar los nameservers. Para ello, editamos el fichero de configuración resolv.conf con los servidores DNS

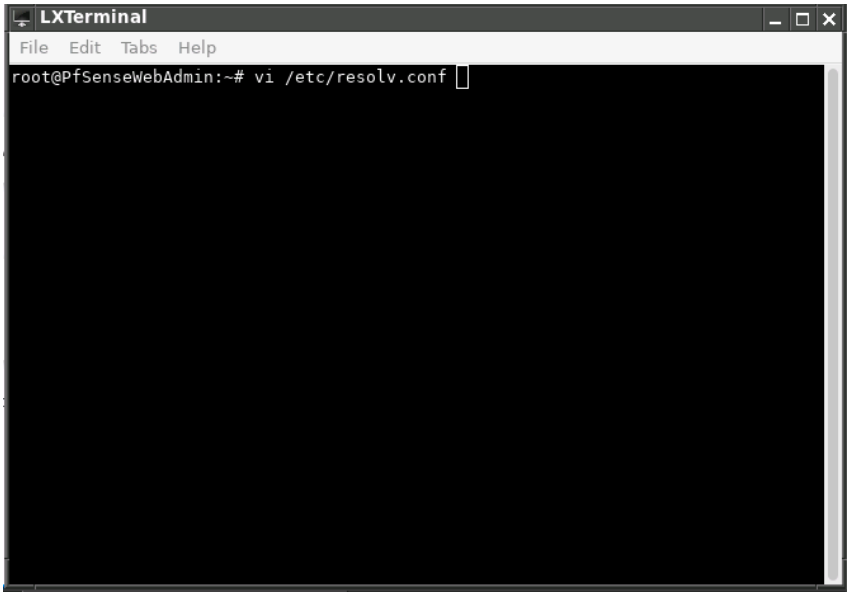


Figura 21.3.1-26 Guía instalación Pfsense. Configuracion DNS en WebAdmin (1)

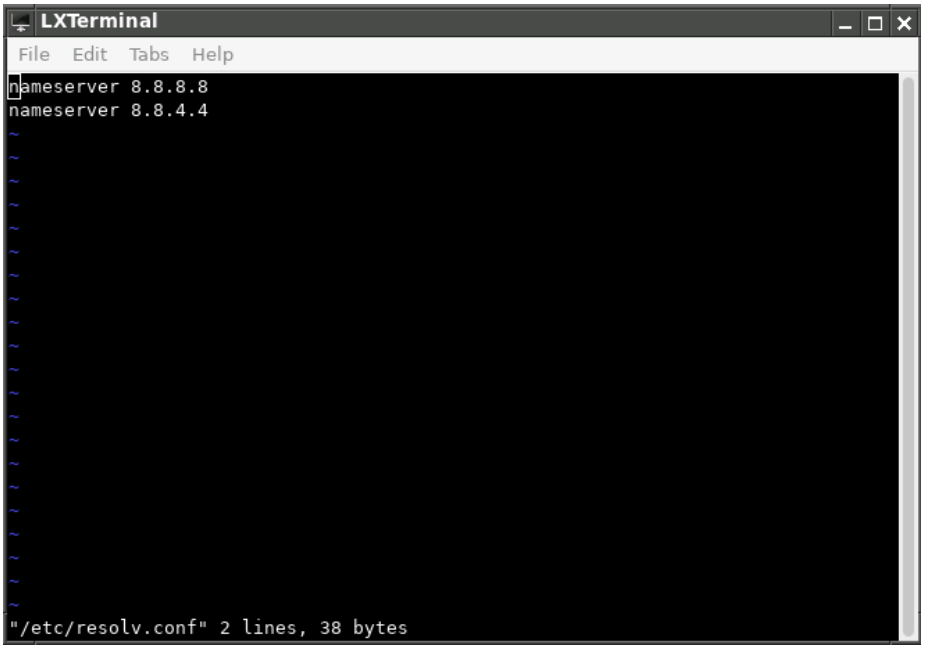


Figura 21.3.1-27 Guía instalación Pfsense. Configuracion DNS en WebAdmin (2)

Un vez configurados, se comprueba que el navegador tiene acceso a Internet

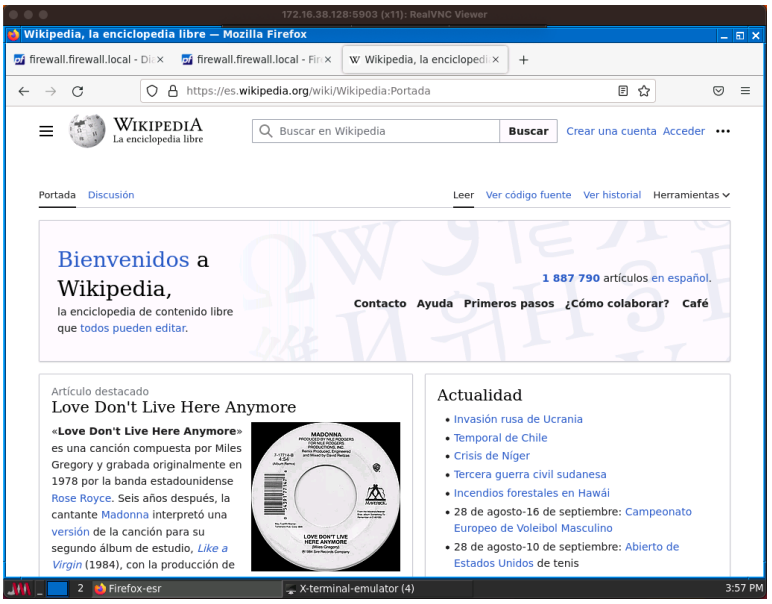


Figura 21.3.1-28 Guía instalación Pfsense. Comprobación salida a Internet desde WebAdmin

A continuación, se habilita el interfaz LAN (em2), para conectar un switch en el interface eth0/0, y un pc con SO ubuntu conectado a este mismo switch en el eth0/1

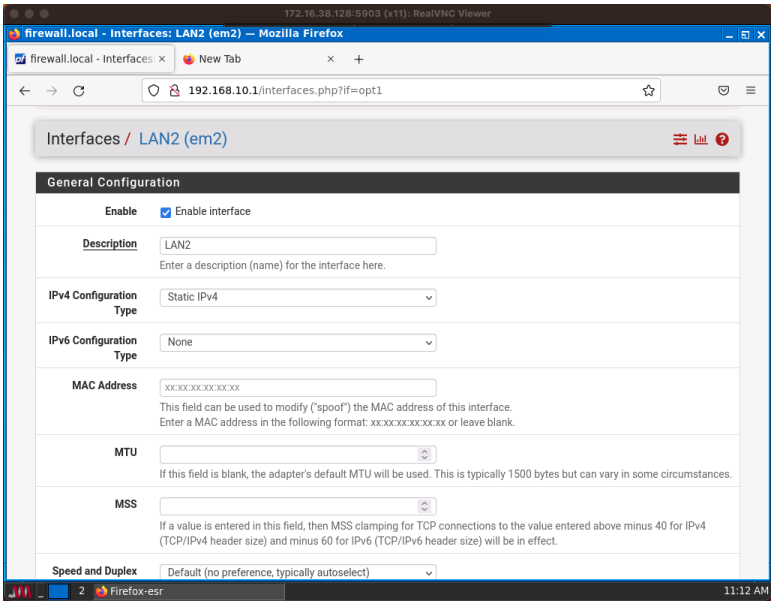


Figura 21.3.1-29 Guía instalación Pfsense. Configuración Web interfaz en2 (1)

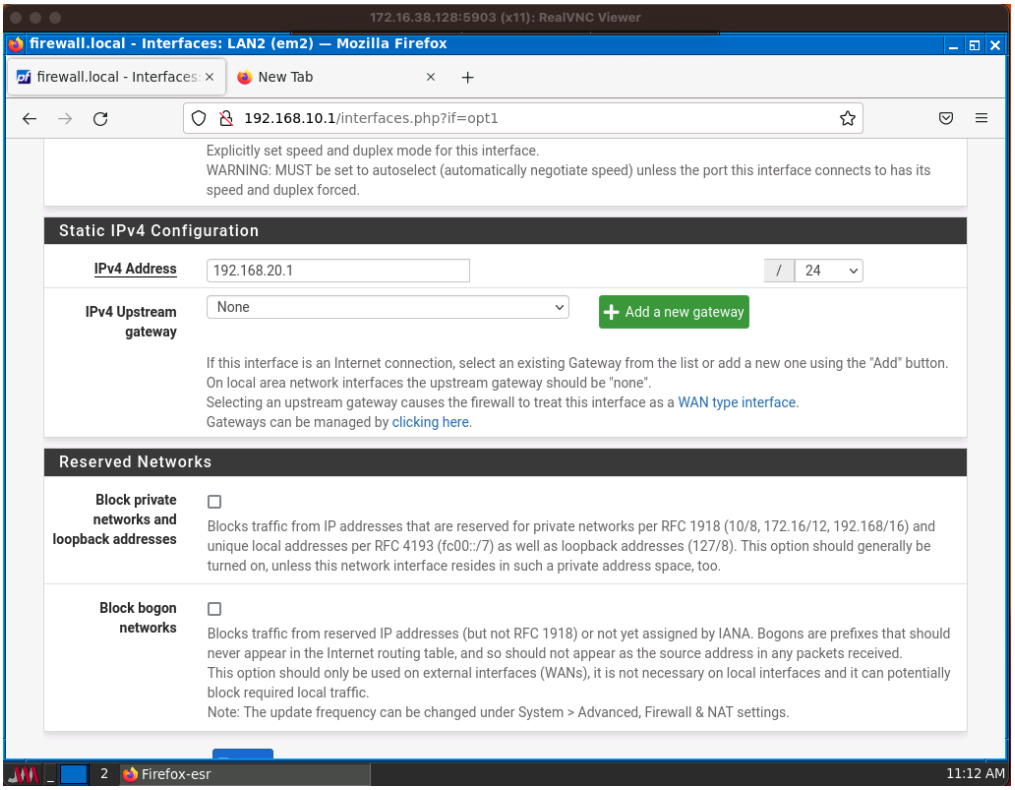


Figura 21.3.1-30 Guía instalación Pfsense. Configuración Web interfaz en2 (2)

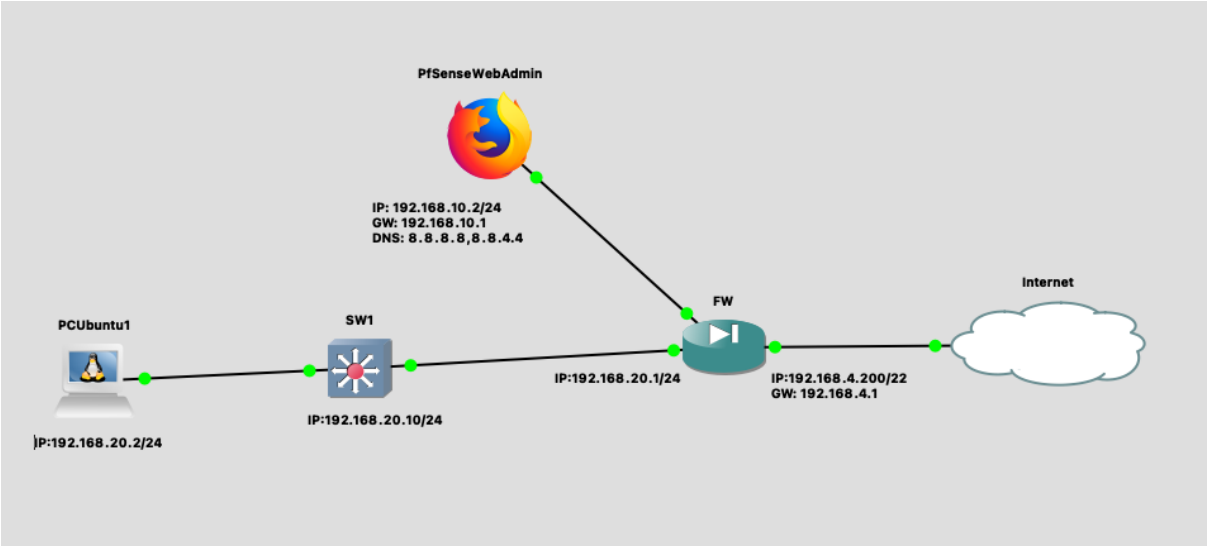


Figura 21.3.1-31 Guía instalación Pfsense. Escenario final conexionado Firewall

Para que las máquinas tengan conexión a internet, así como acceso desde PfSenseWebAdmin al Web de gestión del firewall, se deben de habilitar las siguientes reglas en el Firewall:

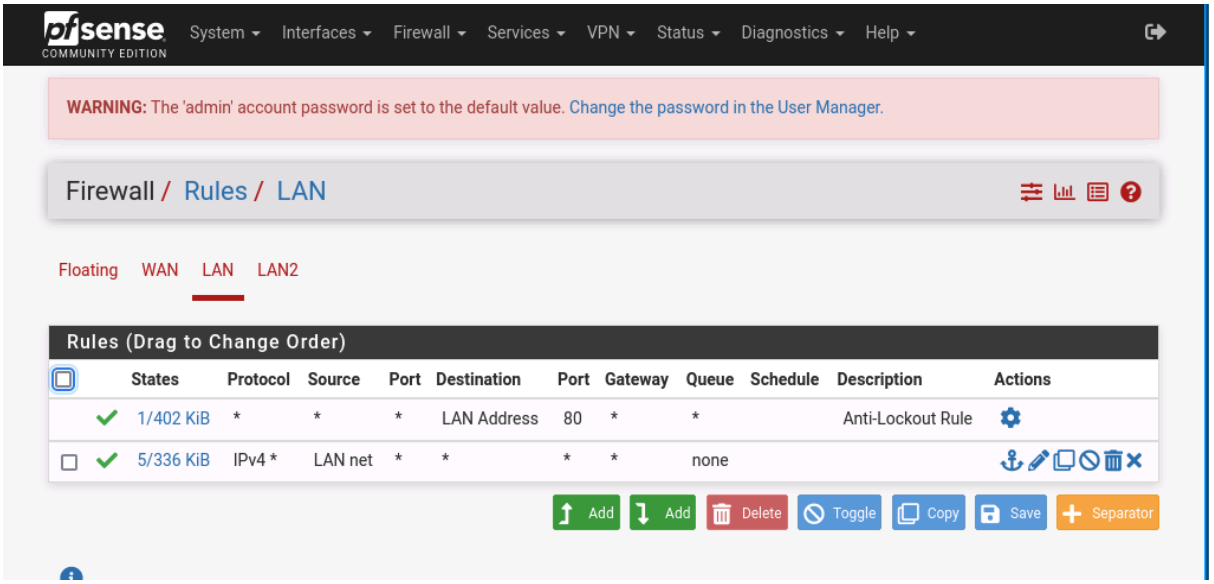


Figura 21.3.1-32 Guía instalación Pfsense. Configuración reglas Firewall (1)

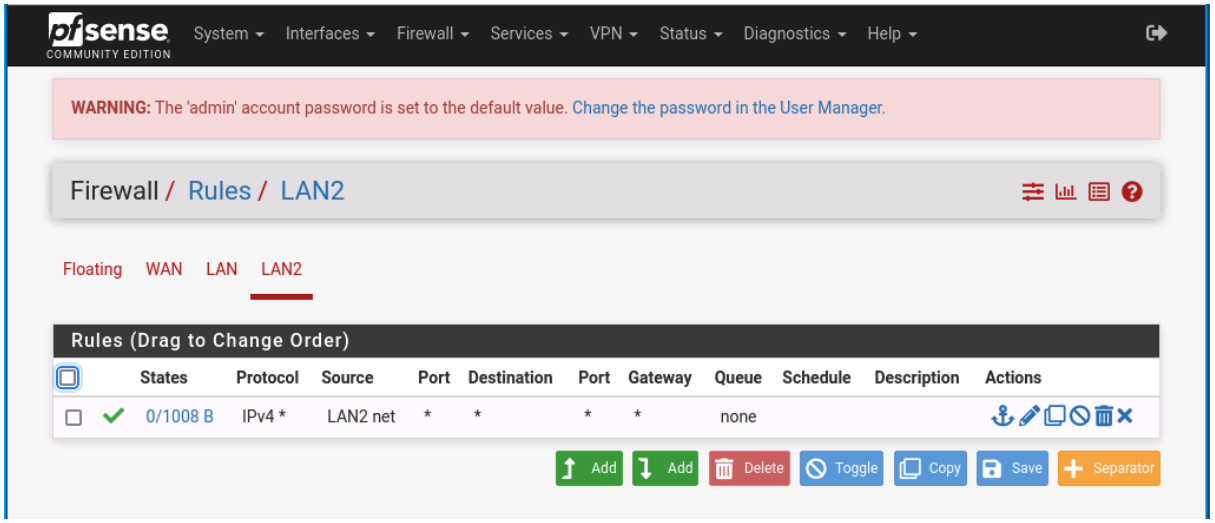


Figura 21.3.1-33 Guía instalación Pfsense. Configuración reglas Firewall (2)

21.3.2. Instalacion de servidor Ubuntu

Descargar la imagen de Ubuntu server 22.04 LTS en el siguiente link
https://releases.ubuntu.com/jammy/ubuntu-22.04.3-live-server-amd64.iso

Crear una nueva máquina virtual usando Vmware Fusion o Vmware Workstation

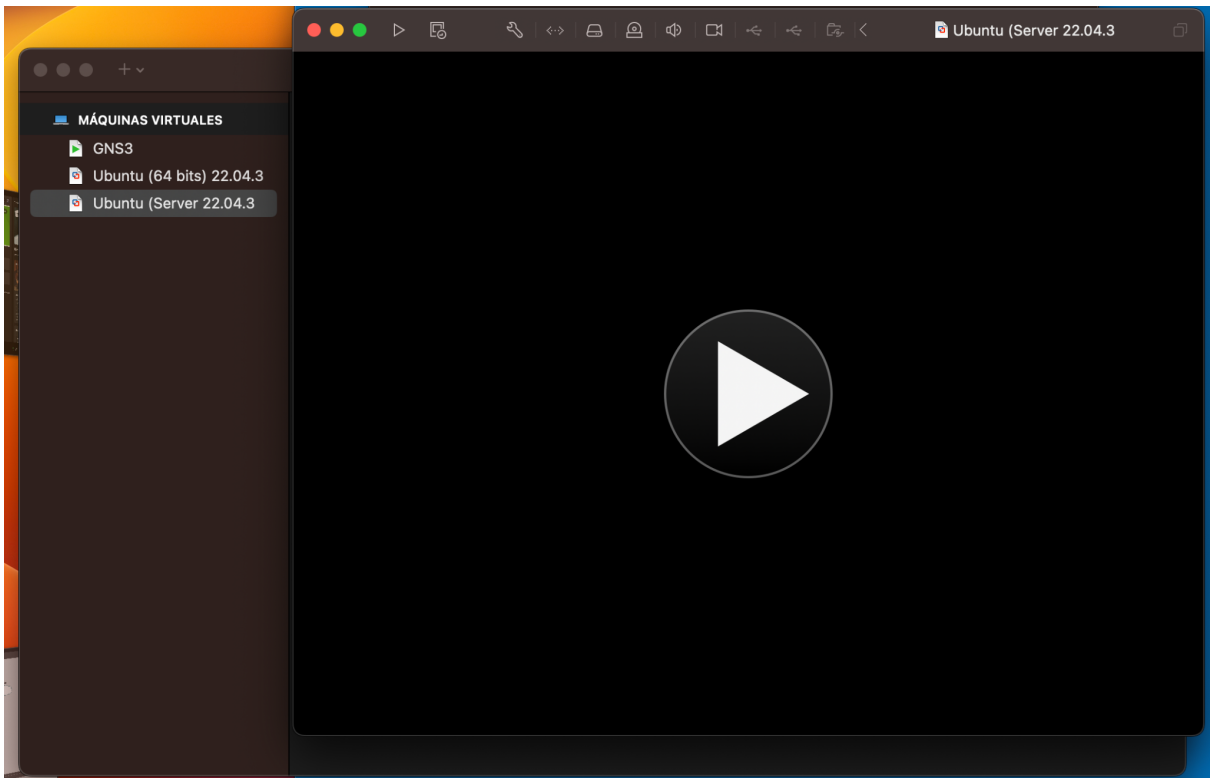


Figura 21.3.2-1 Creación de una maquina virtual en vmware

Una vez creada la máquina virtual, realizar una exportación en formato .OVF

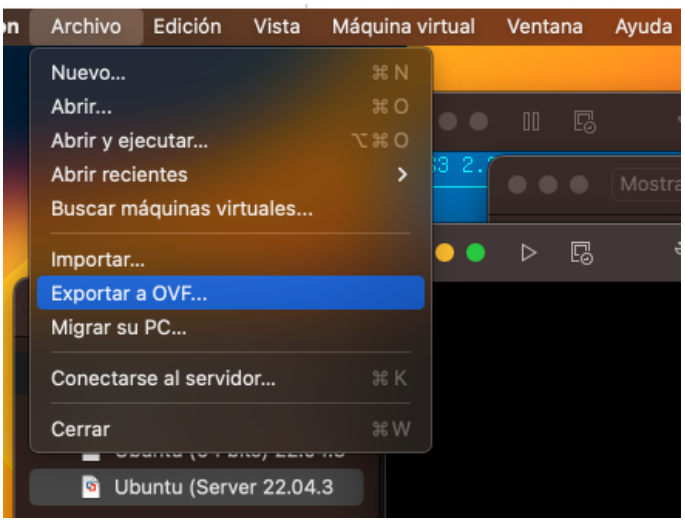


Figura 21.3.2-2 Exportación a OVF de máquina virtual

Los ficheros obtenidos de la exportación son los siguientes:

Figura 21.3.2-3 Archivos que conforman la máquina virtual exportada

Ahora generamos una nueva template en GNS3, seleccionando la opción “Manually create a new template”

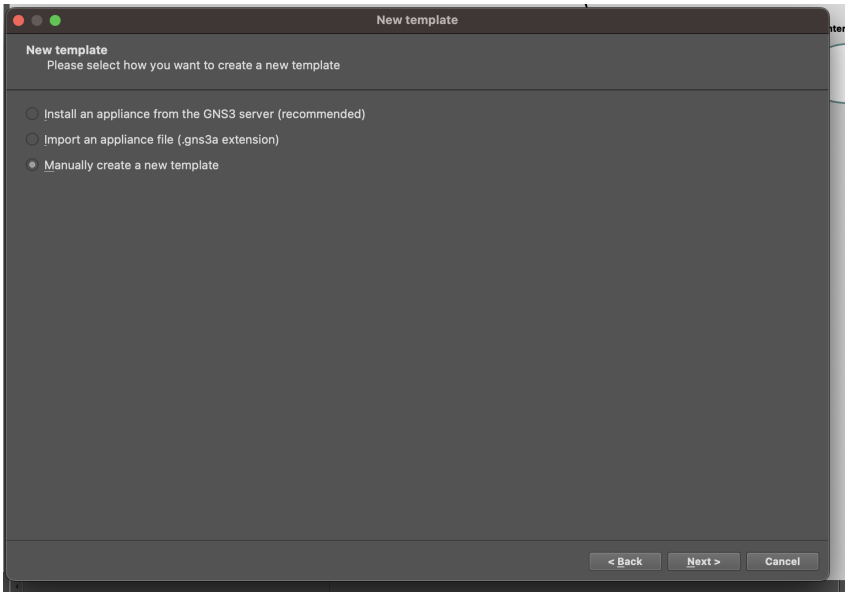


Figura 21.3.2-4 Crear nuevo template manual en GNS3

Nos dirigimos a la sección de “Qemu VMs”, seleccionamos la opción “new”. Server type, “Run this Qemu VM on the GNS3 VM”.

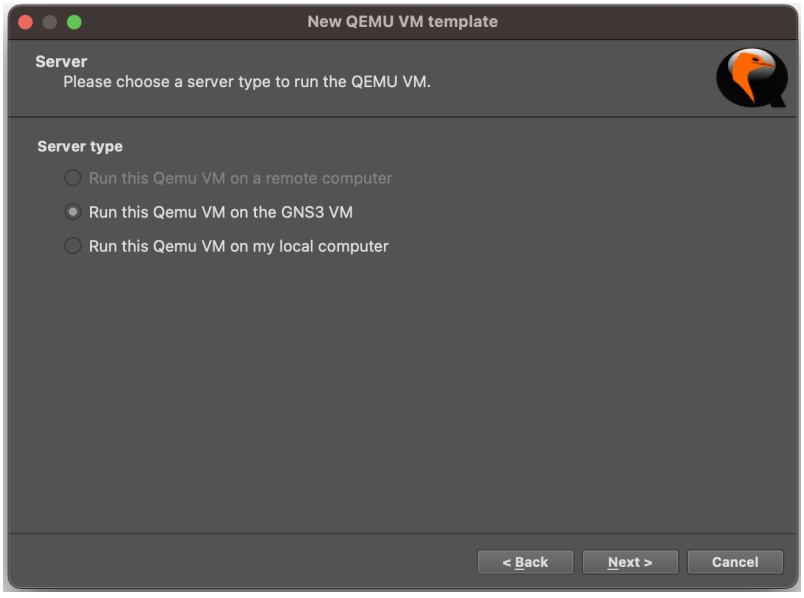


Figura 21.3.2-5 Asignar nombre a VM en GNS3 (1)

Le damos un nombre

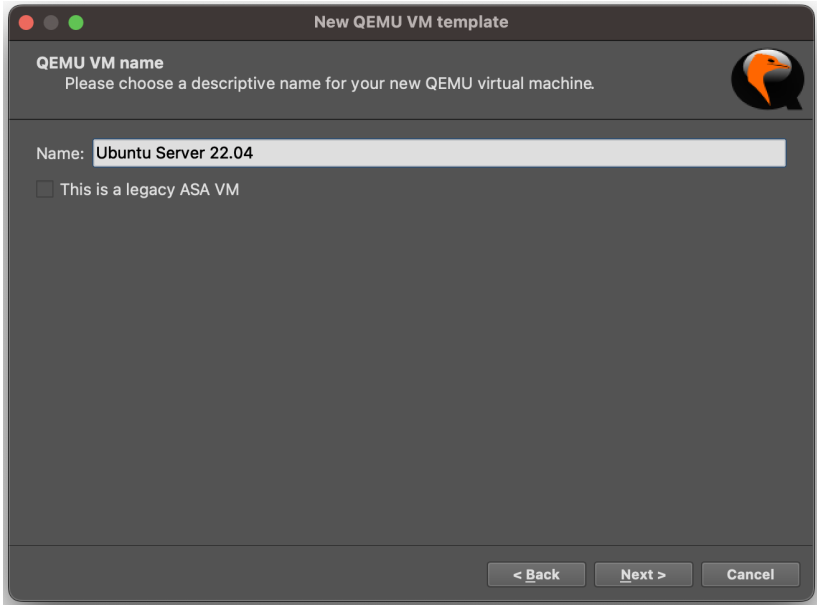


Figura 21.3.2-6 Asignar nombre a VM en GNS3 (2)

Lee asignamos memoria RAM

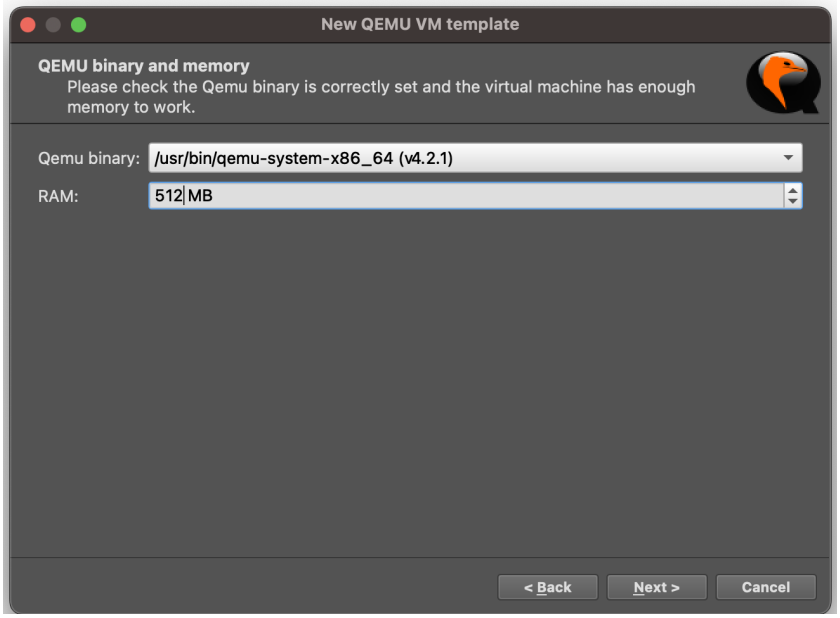


Figura 21.3.2-7 Asignar nombre a VM en GNS3 (2)

Seleccionamos VNC como tipo de consola

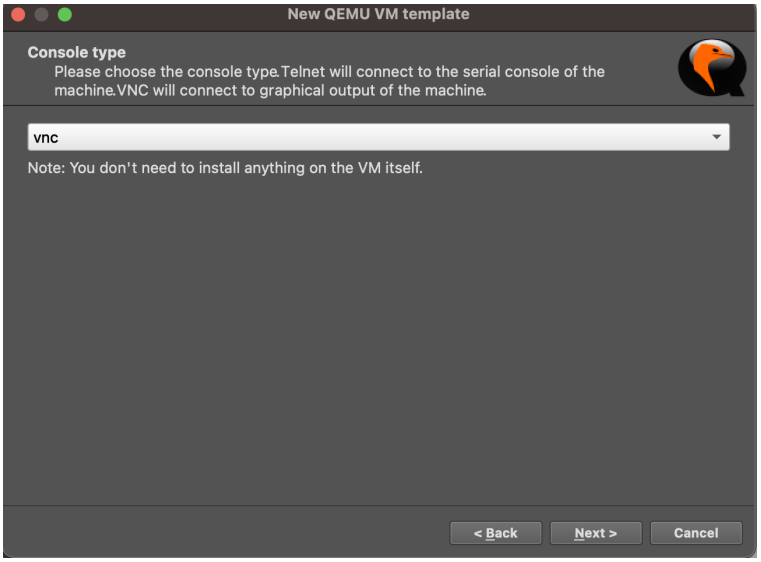


Figura 21.3.2-8 Asignar nombre a VM en GNS3 (2)

Ahora, seleccionamos la opción “New Image” → “browse”, y de la imagen exportada de Vmware como .OVF, seleccionamos el archivo con extensión .vmdk

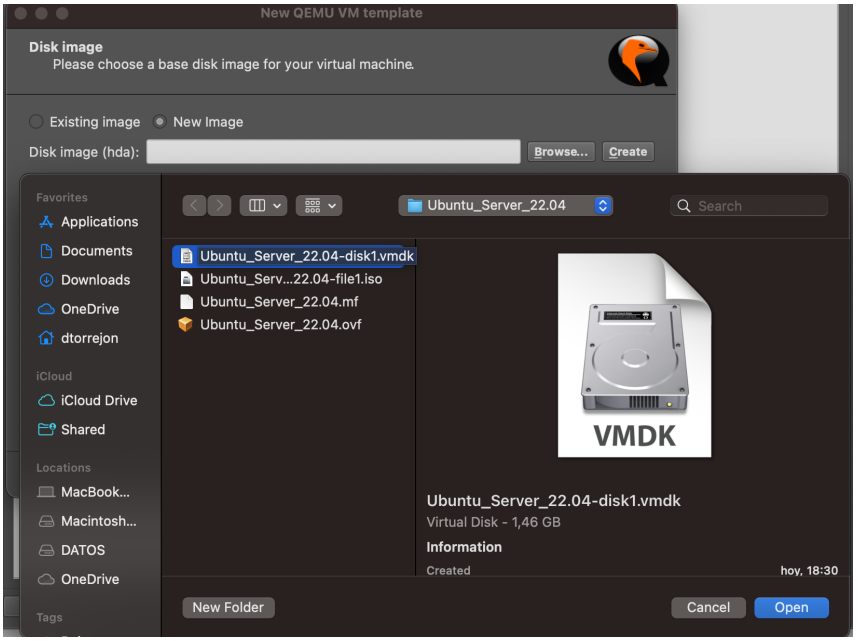


Figura 21.3.2-9 Seleccionar imagen OVA a importar

Ahora ya tenemos la imagen generada.

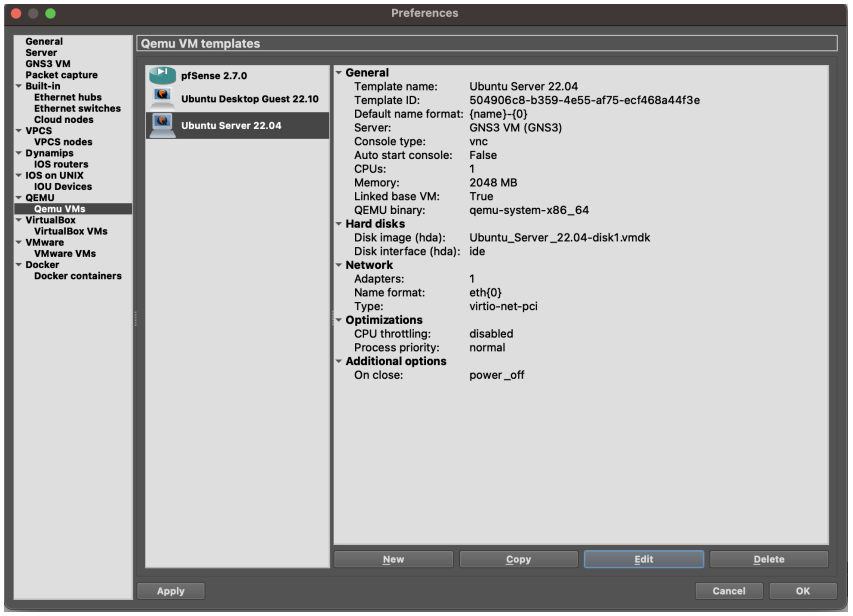


Figura 21.3.2-10 Template generado disponible en GNS3

Como último paso, para asegurar la compatibilidad de la tarjeta de red, vamos a la opción “edit”, pestaña “Network”, y en el campo “type” seleccionamos “Paravirtualized Network I/O (virtio-net-pci)”

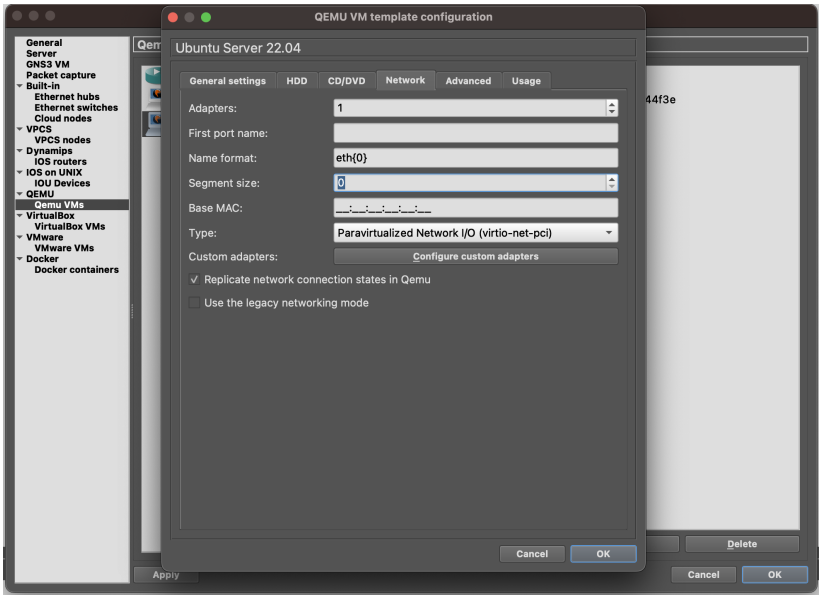


Figura 21.3.2-11 Configuración de la interfaz de red en el template de GNS3

Con el template ya preparado, el siguiente paso es el de crear una máquina en nuestro proyecto, y conectarla a un interfaz del firewall.

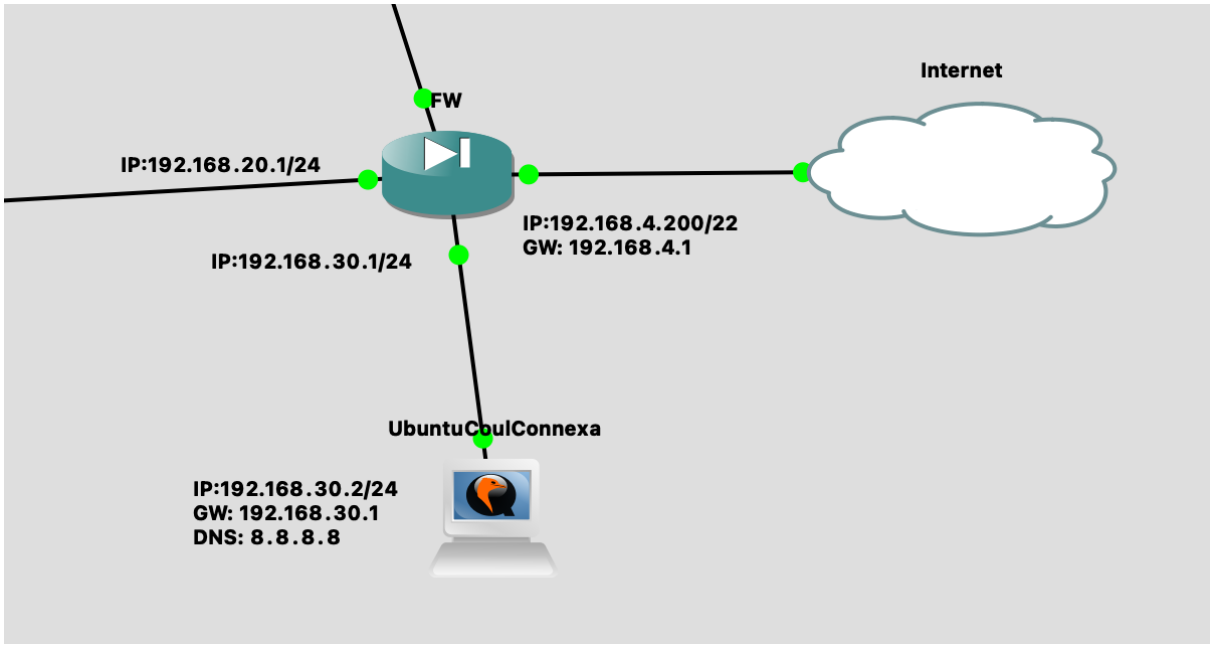


Figura 21.3.2-12 Conexión de la máquina generada con un template, con el Firewall

Arrancamos la máquina, y configuramos el adaptador de red. Para la versión 22.04 se modifica como se indica a continuación.
Comprobamos el nombre de la interfaz de red

```
ip a
```

Figura 21.3.2-13 Configuración del adaptador de red en Ubuntu 22.04 LTS (1)

Editamos el fichero de configuración tal cual se indica en la captura

```
dtorrejon@ubserver:~$ sudo vim /etc/netplan/00-installer-config.yaml
```

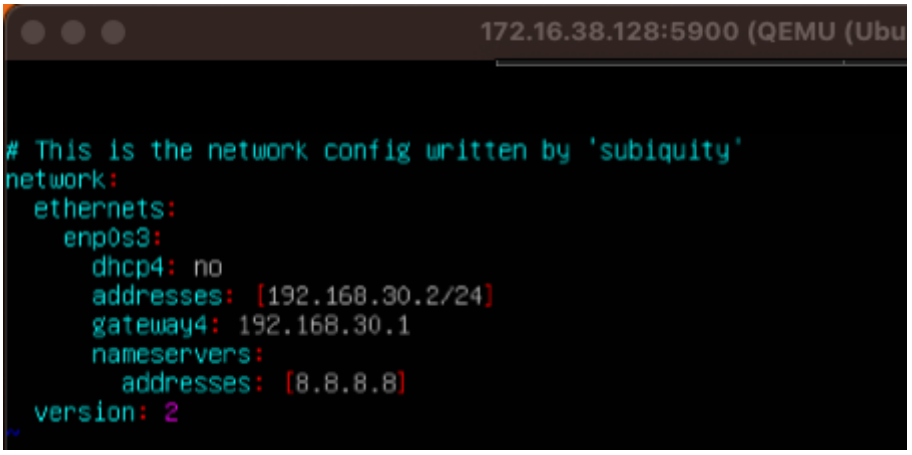


Figura 21.3.2-14 Configuración del adaptador de red en Ubuntu 22.04 LTS (2)

Importante asegurarse de que coincida el nombre de la interfaz, en este caso “enp0s3”

A continuación, guardamos el fichero y cargamos la configuración

```
dtorrejon@ubserver:~$ sudo netplan apply
```

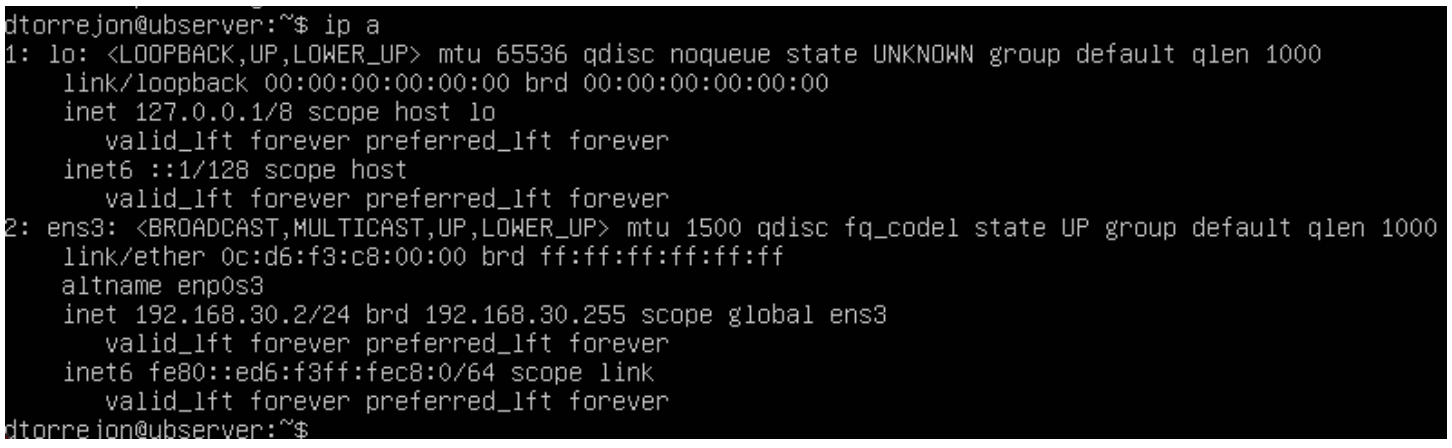


Figura 21.3.2-15 Configuración del adaptador de red en Ubuntu 22.04 LTS (3)

21.4. Configuración del servicio MongoDB Atlas

El primer pasó a realizar, es registrarse en la Web del servicio en el siguiente enlace: https://www.mongodb.com/es/cloud/atlas/register

Se rellenan los datos solicitados y se valida el registro.

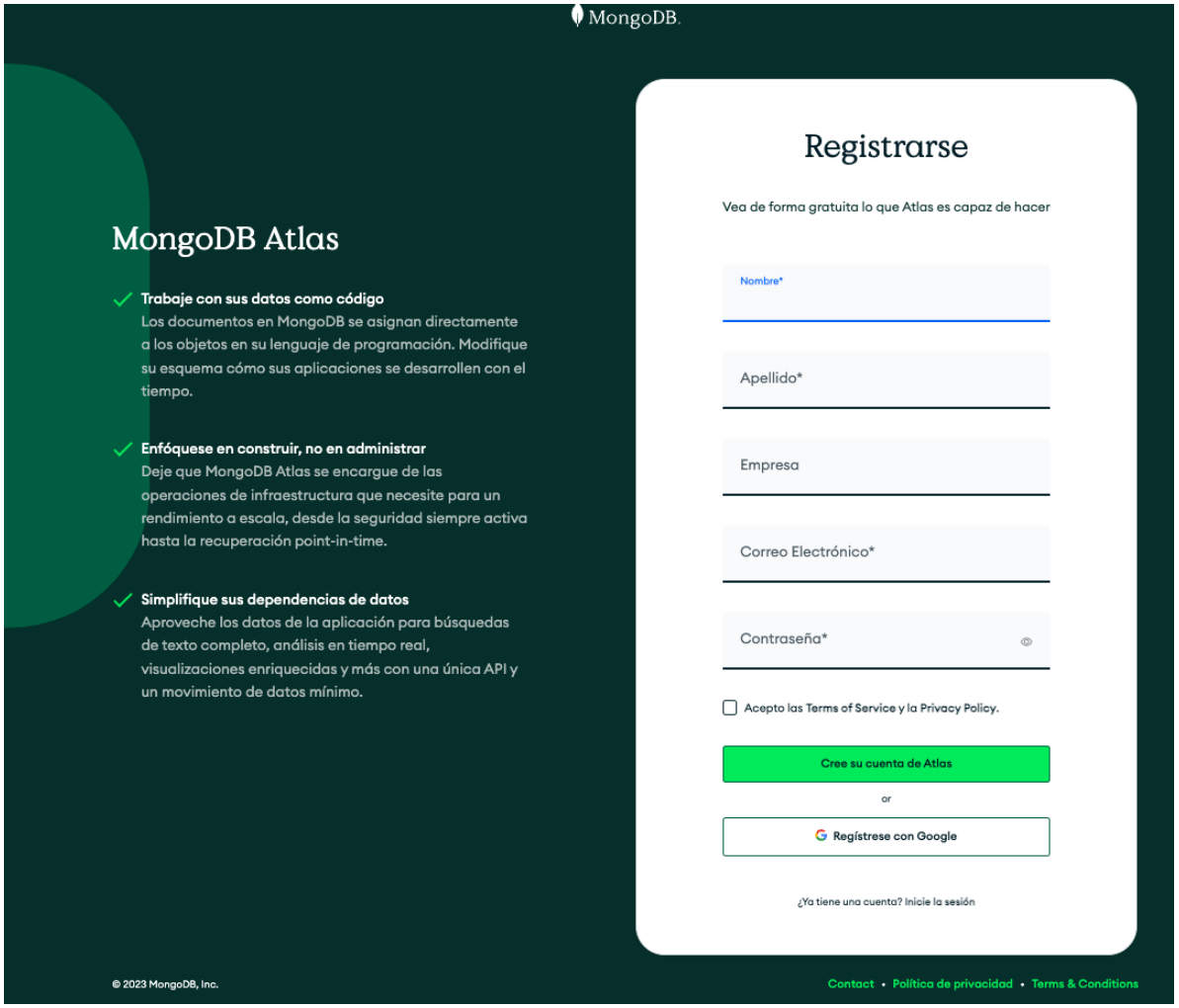


Figura 21.4-1 Registro MongoDB Atlas

A continuación se accede a la vista general

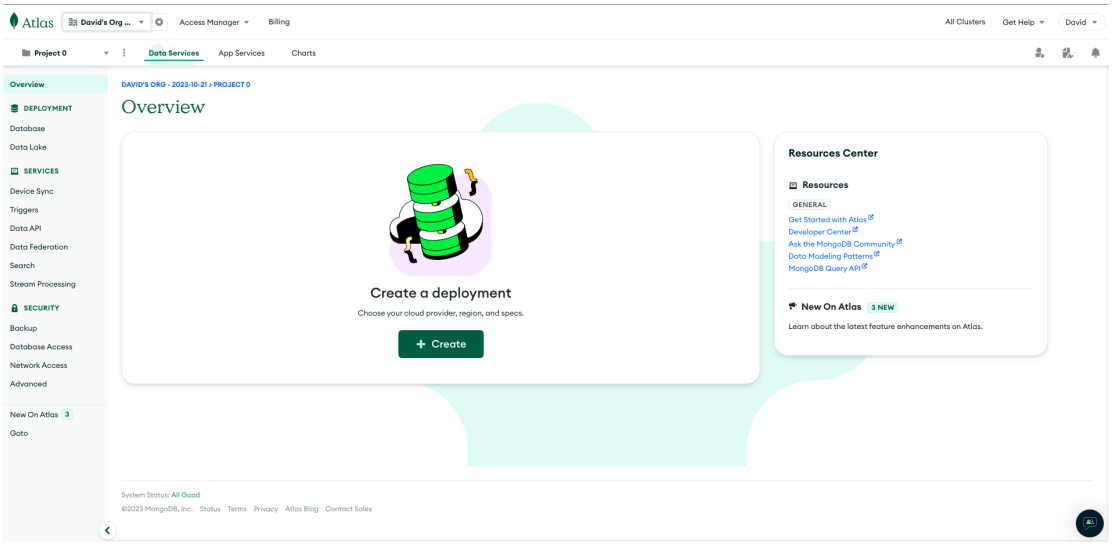


Figura 21.4-2 Vista general MongoDB Atlas

Una vez dentro, vamos a la opción “Build a database”

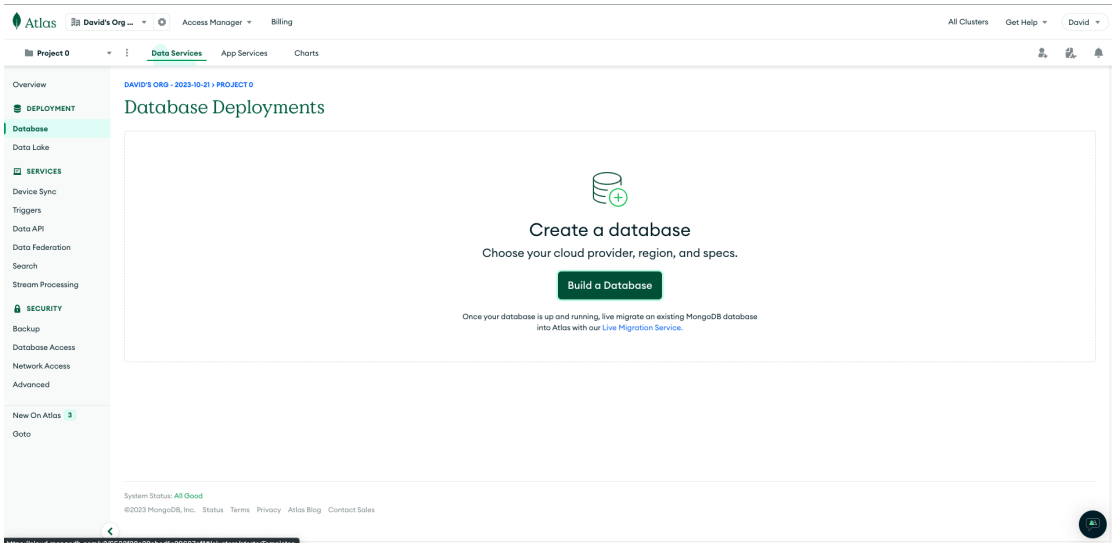


Figura 21.4-3 Desplegar una Base de Datos en mongoDB Atlas

Se crea un clúster gratuito de tipo M0 en una región lo más cercana, en este caso Bélgica.

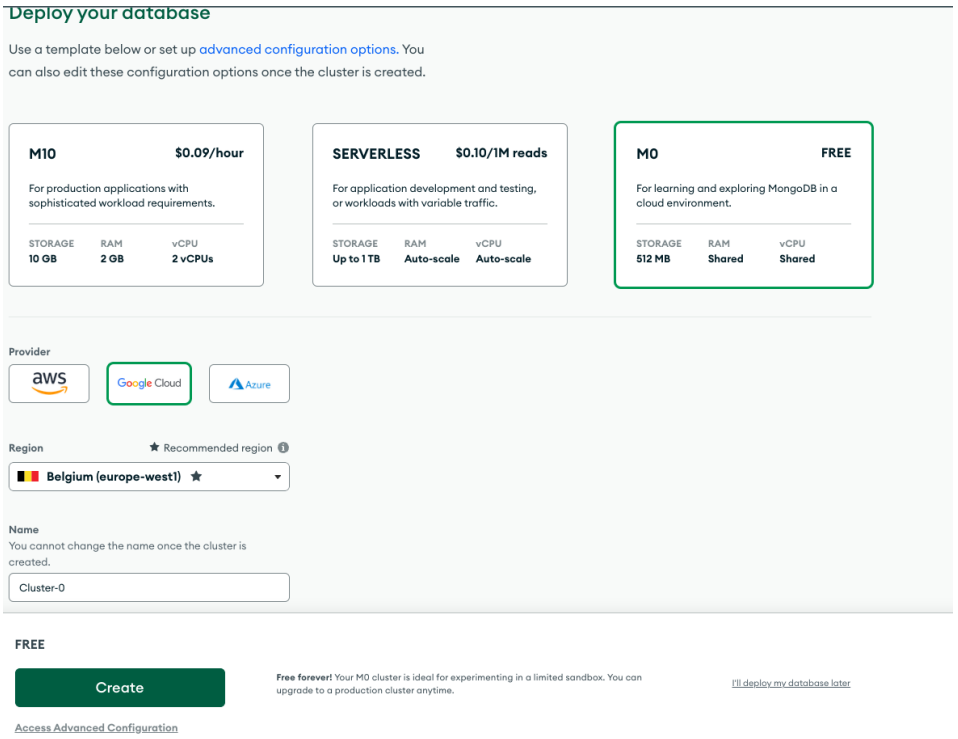


Figura 21.4-4 Elección de instancia y región para desplegar una Base de Datos en MongoDB Atlas

Se genera un usuario y password

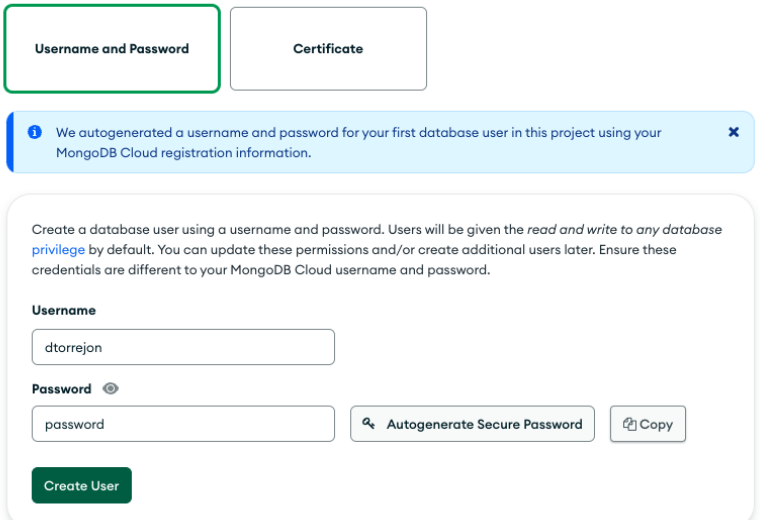


Figura 21.4-5 Generar credenciales de acceso a Base de Datos en MongoDB Atlas

Se especifica las IPs a las que se permite el acceso a la base de datos

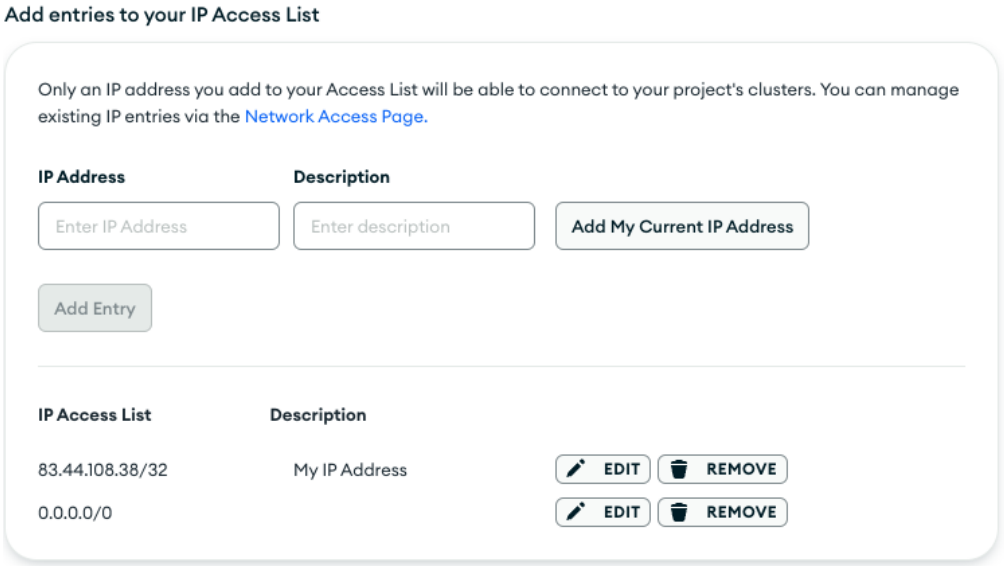


Figura 21.4-6 Lista de IPs con acceso a la Base de Datos en MongoDB Atlas

Ahora se debe acceder a la opción “connect”

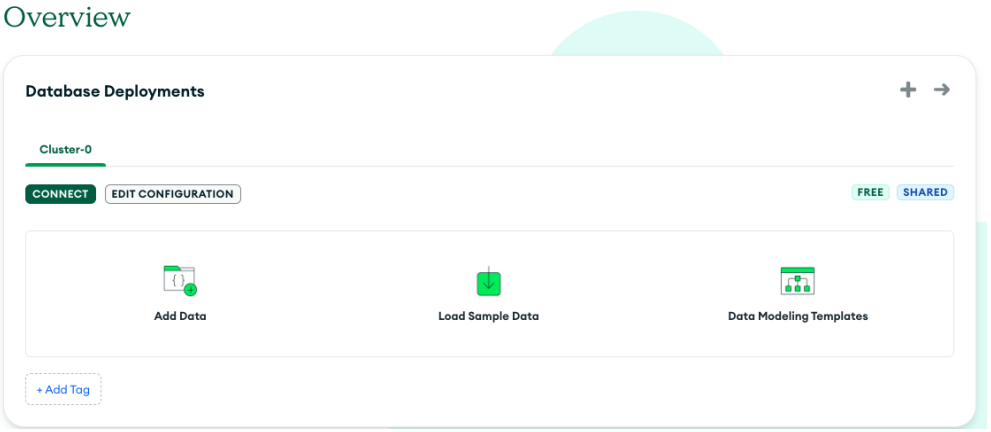


Figura 21.4-7 Información de conexión a Base de Datos en MongoDB Atlas

Se selecciona la opcion “Drivers”

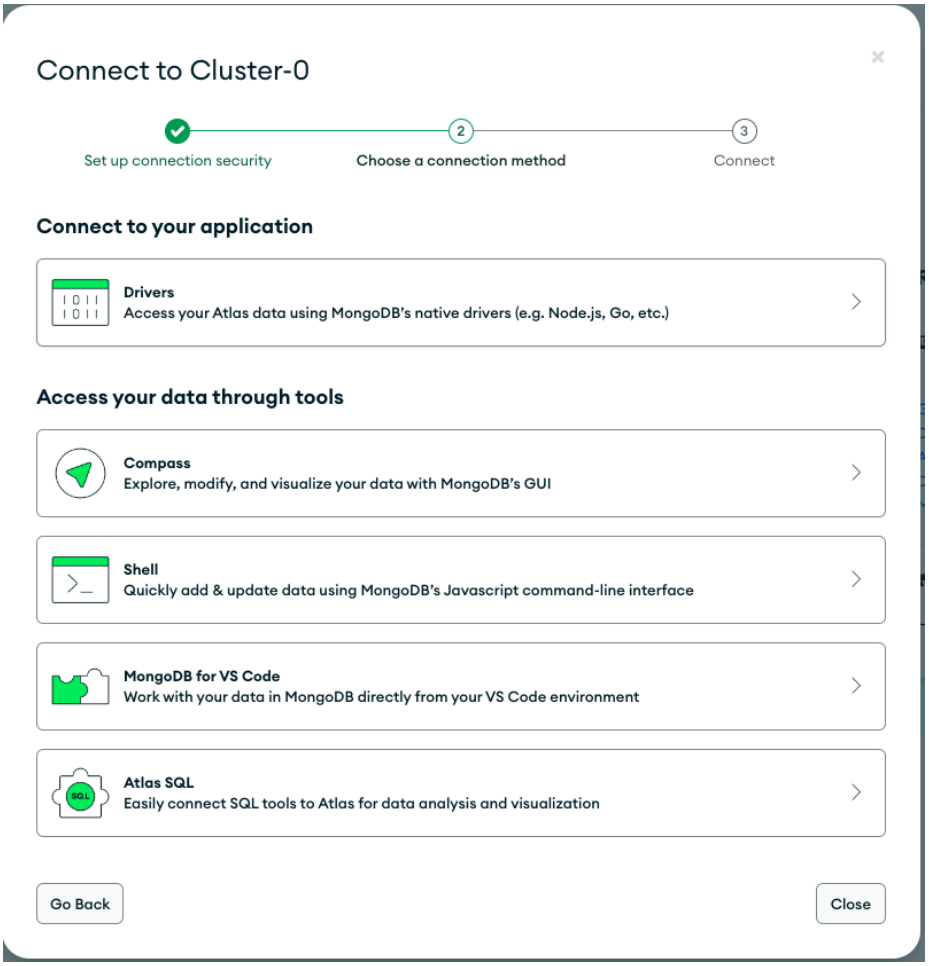


Figura 21.4-8 Formas de conexión a Base de Datos en MongoDB Atlas

Finalmente, se obtiene la url y la forma de instalación de los drivers para Python 3.11

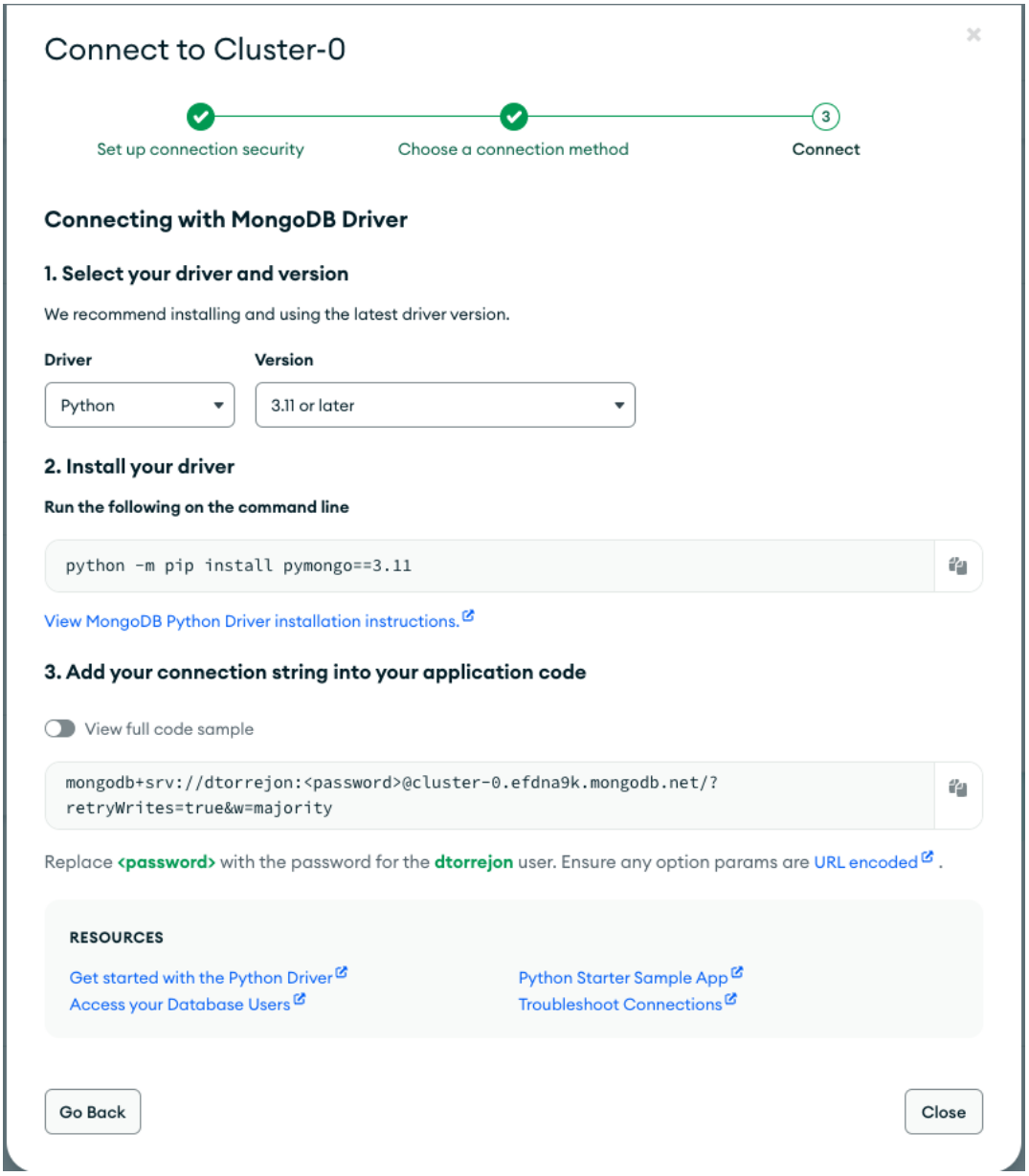


Figura 21.4-9 Datos de conexión a Base de Datos en MongoDB Atlas usando Python 3.11

A partir de este punto, ya se dispone de toda la información necesaria para conectarse a la base de datos desde la aplicación.

21.5. Instalación y configuración del servicio Cloud Connexa

A continuación, se describen los pasos para configurar Cloud Connexa en una máquina Linux. En este caso en particular, realizaremos un ejemplo de configuración, para un servidor sobre el que correrá un servicio aislado para un cliente

Accedemos a cloud connexa y generamos la nueva red

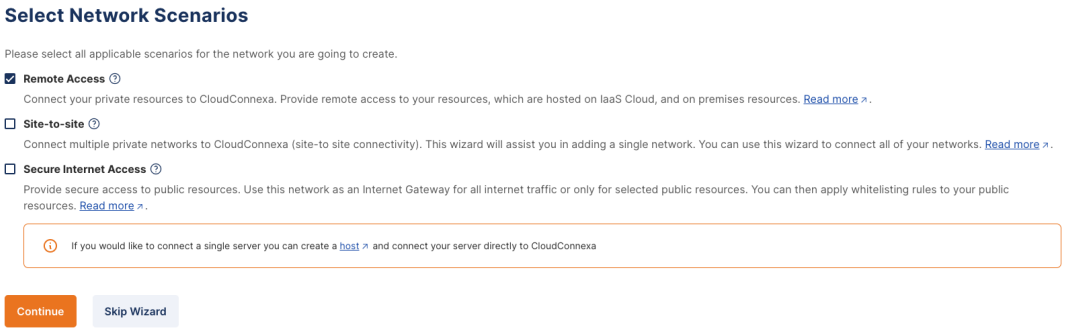


Figura 21.5-1 Configuración Cloud Connexa. Selección de tipo de red

Definimos el nombre y la región

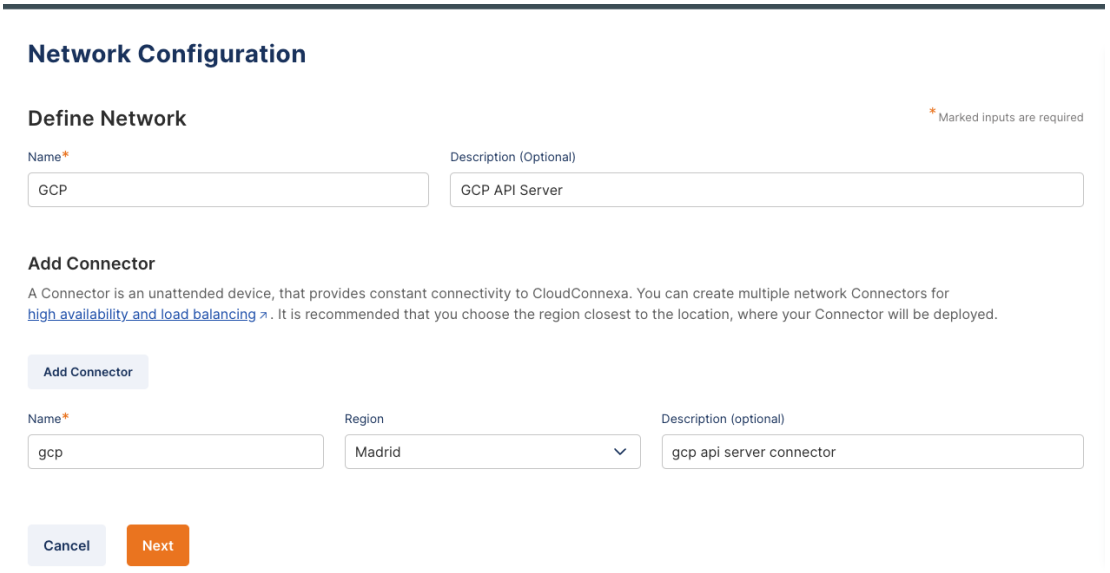


Figura 21.5-2 Configuración Cloud Connexa. Nombre y región

Configuramos el conector, en esta caso para una máquina Ubuntu 22.04 LTS, lanzando el *script* indicado en la captura.

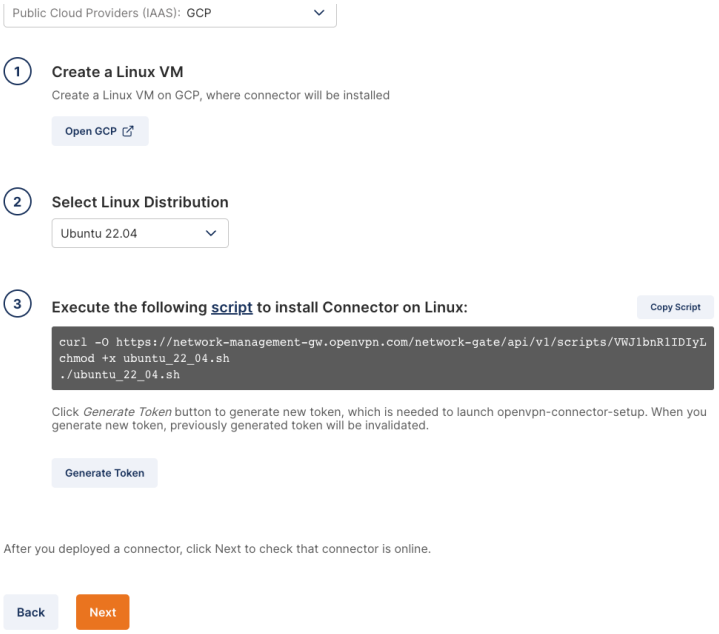


Figura 21.5-3 Configuración Cloud Connexa. *script* de configuración en e servidor

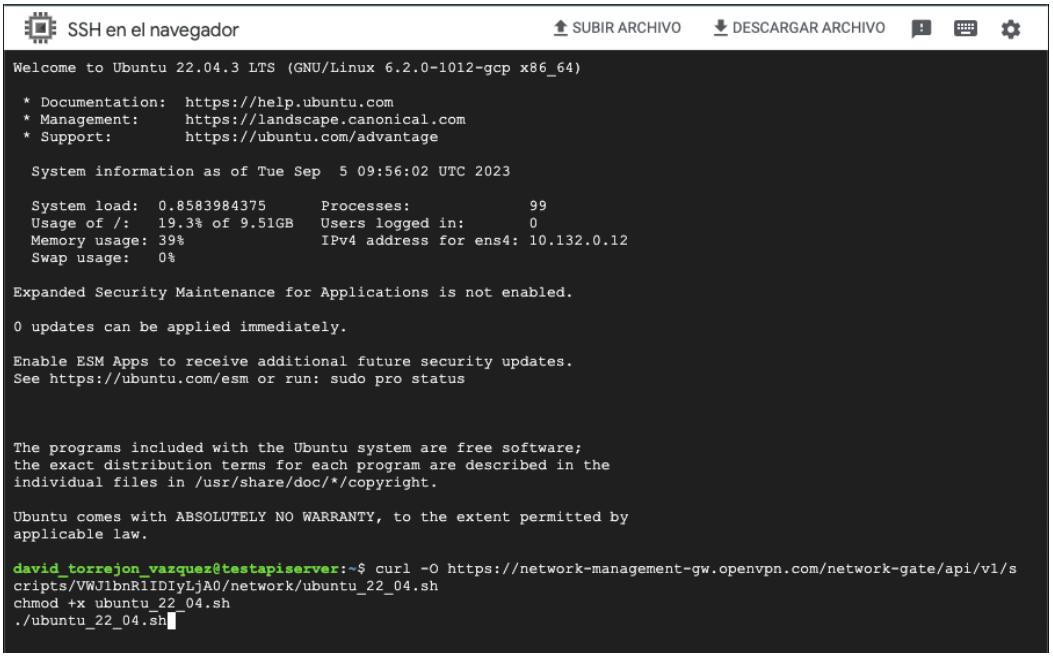


Figura 21.5-4 Configuración Cloud Connexa. Ejecución del script en el servidor

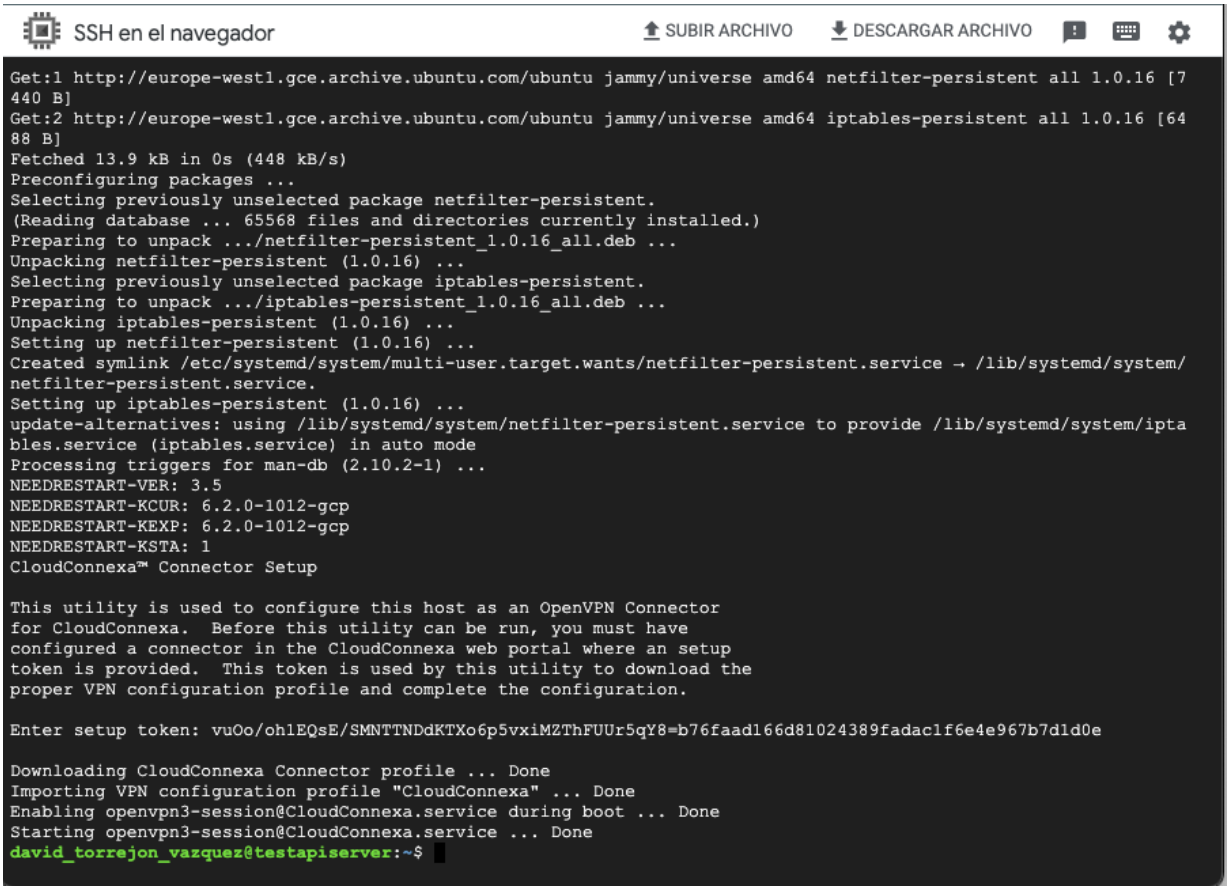


Figura 21.5-5 Configuración Cloud Connexa. Ejecución del script en el servidor (2)

Una vez ejecutado el script, podemos observar que el estado es “connected”

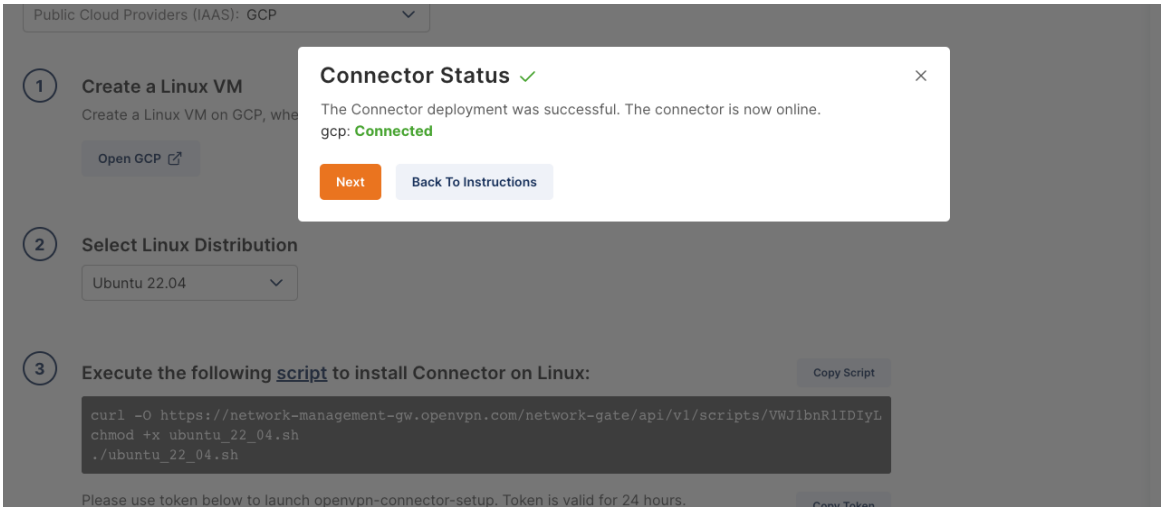


Figura 21.5-6 Configuración Cloud Connexa. Estado del conector en lado servidor

A continuación, pasamos a configurar las redes. En "*IP Services*" añadimos la IP del propio servidor, y en el apartado "*Routes*", añadimos la red en notación CIDR.

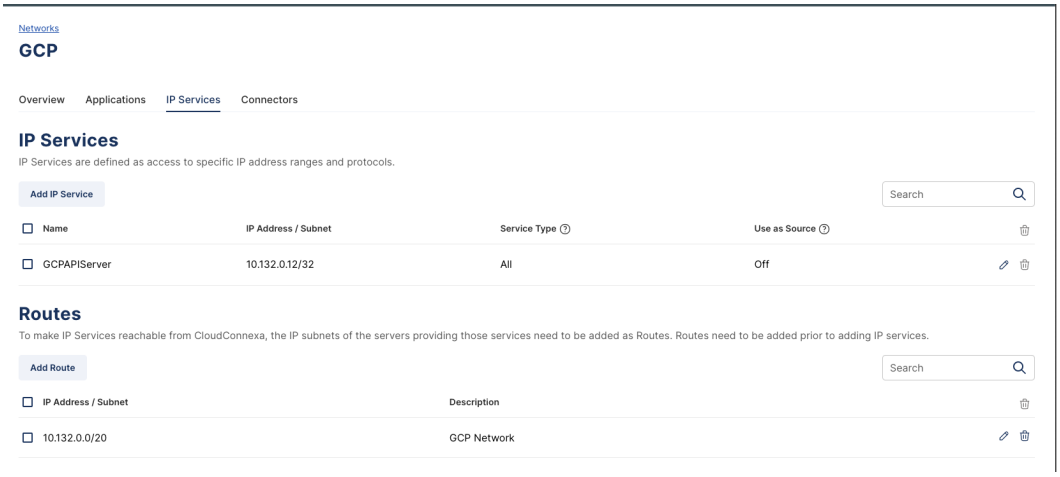


Figura 21.5-7 Configuración Cloud Connexa. Configuración redes lado servidor

Verificamos el estado final de la configuración

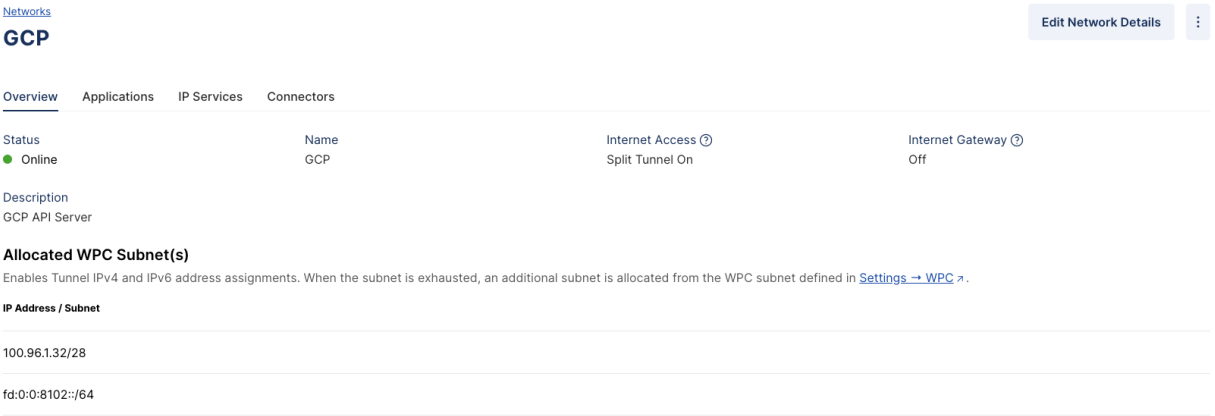


Figura 21.5-8 Configuración Cloud Connexa. Estado configuración final en lado servidor

Ahora hay que habilitar PORT FORWARDING y NAT para llegar al resto de redes desde el conector ubicado en el lado cliente

Para habilitar PORT FORWARINDG en Linux:

```
sudo sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/g' /etc/sysctl.conf
sudo sysctl -p
```

Para Habilita NAT en Linux

```
sudo apt install iptables-persistent
IF=`ip route | grep default | awk '{print $5}'`
sudo iptables -t nat -A POSTROUTING -o $IF -j MASQUERADE
sudo iptables-save | sudo tee /etc/iptables/rules.v4
sudo ip6tables -t nat -A POSTROUTING -o $IF -j MASQUERADE
sudo iptables-save | sudo tee /etc/iptables/rules.v6
```

21.6. Guia para el despliegue y configuración de los servidores en Google Cloud Platform

Se dispone de las siguientes plantillas en formato json, para desplegar servidores a través del envío de una operación POST HTTP con una aplicación, como puede ser curl.

A continuación, se detalla el proceso a seguir para la configuración de los servidores necesarios para desplegar la solución

Templates-Server

```
POST
https://www.googleapis.com/compute/v1/projects/networkmanager-397512/zones/europe-west4-a/instances
```

```
{
  "canIpForward": false,
  "confidentialInstanceConfig": {
    "enableConfidentialCompute": false
  },
  "deletionProtection": false,
  "description": "",
  "disks": [
    {
      "autoDelete": true,
```

```
      "boot": true,
      "deviceName": "templates",
      "initializeParams": {
        "diskSizeGb": "30",
        "diskType":
"projects/networkmanager-397512/zones/europe-west4-a/diskTypes/pd-balanced",
        "labels": {},
        "sourceImage":
"projects/ubuntu-os-cloud/global/images/ubuntu-2204-jammy-v20230908"
      },
      "mode": "READ_WRITE",
      "type": "PERSISTENT"
    }
  ],
  "displayDevice": {
    "enableDisplay": false
  },
  "guestAccelerators": [],
  "instanceEncryptionKey": {},
  "keyRevocationActionType": "NONE",
  "labels": {
    "goog-ec-src": "vm_add-rest"
  },
  "machineType":
"projects/networkmanager-397512/zones/europe-west4-a/machineTypes/e2-highcpu-4"
,
  "metadata": {
    "items": []
  },
  "name": "templates-server-1",
  "networkInterfaces": [
    {
      "accessConfigs": [
        {
          "name": "External NAT",
          "networkTier": "PREMIUM"
        }
      ],
```

```
      "stackType": "IPV4_ONLY",
      "subnetwork":
"projects/networkmanager-397512/regions/europe-west4/subnetworks/default"
    }
  ],
  "params": {
    "resourceManagerTags": {}
  },
  "reservationAffinity": {
    "consumeReservationType": "ANY_RESERVATION"
  },
  "scheduling": {
    "automaticRestart": true,
    "onHostMaintenance": "MIGRATE",
    "provisioningModel": "STANDARD"
  },
  "serviceAccounts": [
    {
      "email": "60698791164-compute@developer.gserviceaccount.com",
      "scopes": [
        "https://www.googleapis.com/auth/devstorage.read_only",
        "https://www.googleapis.com/auth/logging.write",
        "https://www.googleapis.com/auth/monitoring.write",
        "https://www.googleapis.com/auth/servicecontrol",
        "https://www.googleapis.com/auth/service.management.readonly",
        "https://www.googleapis.com/auth/trace.append"
      ]
    }
  ],
  "shieldedInstanceConfig": {
    "enableIntegrityMonitoring": true,
    "enableSecureBoot": false,
    "enableVtpm": true
  },
  "tags": {
    "items": []
  },
  "zone": "projects/networkmanager-397512/zones/europe-west4-a"
```

```
}
```

Una vez desplegado este servidor, con una imagen del SO Ubuntu 22.04 LTS, se debe proceder a la instalación del entorno de virtualización, la herramienta de automatización Ansible, y un servidor Apache + PHP escuchando en el puerto 80 para descargar las imágenes de las VM's generadas. También se debe albergar una copia de la imagen base que dará lugar, una vez configurada, al conector.

Para configurar el servidor Apache + PHP, se debe realizar la siguiente configuración:

```
sudo apt install apache2
sudo systemctl status apache2
sudo apt install php libapache2-mod-php
sudo a2enmod php8.1
sudo systemctl restart apache2
```

El código correspondiente, se deberá copiar en la ruta /var/www/html/

En cuanto a la configuración de Ansible, el entorno de virtualización y la imagen preconfigurada, se debe ejecutar el procedimiento que se describe a continuación:

Lo primero que hay que hacer es instalar ansible y el sistema de virtualización

```
sudo apt-get update
sudo apt-get install ansible
sudo apt-get install openssh-serversudo apt-get install qemu-kvm
libvirt-daemon-system libvirt-clients bridge-utils virtinst libvirt-clients
```

Hay que asegurarse de que libvirt esté configurado correctamente, incluyendo la definición de la red predeterminada. Se puede verificar y configurar la red predeterminada utilizando el comando virsh.

```
sudo virsh net-list
```

Si la red predeterminada no está activa, puedes activarla con:

```
sudo virsh net-start default
```

Después, se debe configurar el fichero de sudoers añadiendo el usuario correspondiente según la imagen base de Ubuntu que se está usando. En este caso el usuario es connexa

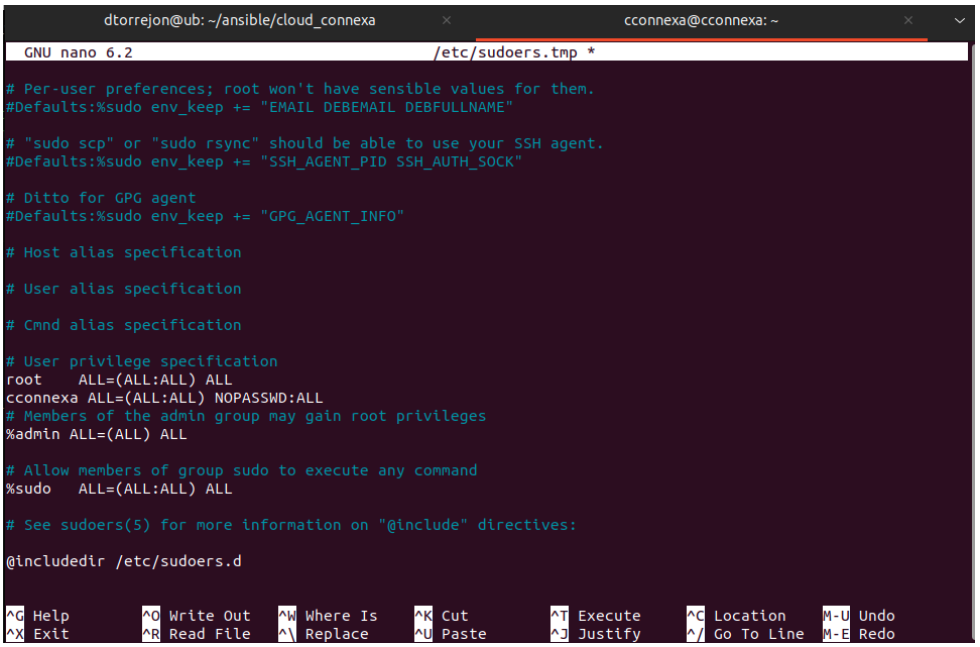


Figura 21.6-1 Configuración fichero sudoers

A continuación, se debe crear el siguiente archivo y añadir el contenido indicado

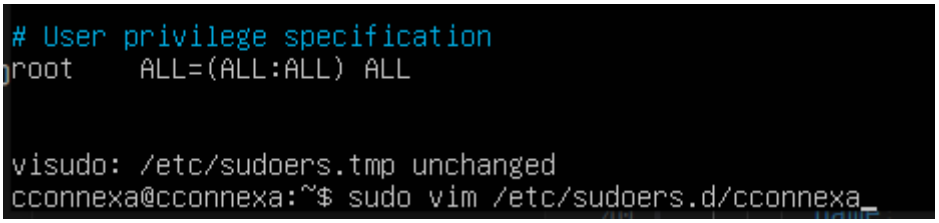


Figura 21.6-2 Creación fichero para asignar privilegios sudo a usuario

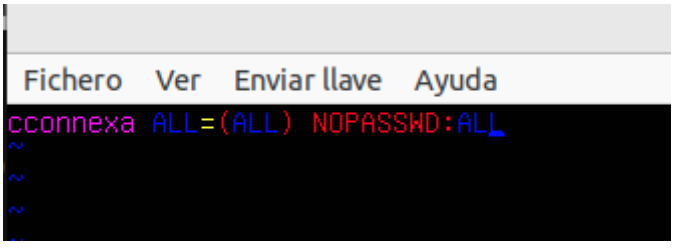


Figura 21.6-3 Contenido fichero para asignar privilegios sudo a usuario

El último paso, es configurar el acceso por SSH a la imagen que se ejecuta sobre VIRTIO.Para ello, levantamos la maquina virtual.

```
sudo virt-install --name=cloud_connexa_client       --ram=2048       --vcpus=1
--disk path=/var/local/images/cloud_connexa_client.qcow2,size=20       --network
network=default,model=virtio       --os-type=linux        --os-variant=ubuntu22.04
--boot hd
```

Se comprueba que la maquina ha levantado con los siguientes comandos. En este caso la imagen que estamos usando tiene configurada por defecto la IP 192.168.122.10 (confirmamos que responde a PING)

```
sudo virsh list
ping 192.168.122.10
```

Se genera una pareja de claves y, se habilita a la máquina virtual para que acepte conexiones ssh desde el servidor a través de clave (necesario para comunicarse con ansible)

```
ssh-keygen
ssh-copy-id ansible@192.168.122.10
```

Tras ello, podemos apagar la VM que está ejecutándose, y dar por finalizada tanto la configuración del servidor como de la imagen a usar como plantilla.

Deploy

```
POST
https://www.googleapis.com/compute/v1/projects/networkmanager-397512/zones/euro
pe-west1-b/instances
```

```
{
  "canIpForward": false,
  "confidentialInstanceConfig": {
    "enableConfidentialCompute": false
  },
  "deletionProtection": false,
  "description": "",
```

```
  "disks": [
    {
      "autoDelete": true,
      "boot": true,
      "deviceName": "deploy",
      "initializeParams": {
        "diskSizeGb": "10",
        "diskType":
"projects/networkmanager-397512/zones/europe-west1-b/diskTypes/pd-balanced",
        "labels": {},
        "sourceImage":
"projects/ubuntu-os-cloud/global/images/ubuntu-2204-jammy-v20230919"
      },
      "mode": "READ_WRITE",
      "type": "PERSISTENT"
    }
  ],
  "displayDevice": {
    "enableDisplay": false
  },
  "guestAccelerators": [],
  "instanceEncryptionKey": {},
  "keyRevocationActionType": "NONE",
  "labels": {
    "goog-ec-src": "vm_add-rest"
  },
  "machineType":
"projects/networkmanager-397512/zones/europe-west1-b/machineTypes/f1-micro",
  "metadata": {
    "items": []
  },
  "name": "deploy-1",
  "networkInterfaces": [
    {
      "accessConfigs": [
        {
          "name": "External NAT",
          "networkTier": "PREMIUM"
```

```
        }
      ],
      "stackType": "IPV4_ONLY",
      "subnetwork":
"projects/networkmanager-397512/regions/europe-west1/subnetworks/default"
    }
  ],
  "params": {
    "resourceManagerTags": {}
  },
  "reservationAffinity": {
    "consumeReservationType": "ANY_RESERVATION"
  },
  "scheduling": {
    "automaticRestart": true,
    "onHostMaintenance": "MIGRATE",
    "provisioningModel": "STANDARD"
  },
  "serviceAccounts": [
    {
      "email": "606987911 64-compute@developer.gserviceaccount.com",
      "scopes": [
        "https://www.googleapis.com/auth/devstorage.read_only",
        "https://www.googleapis.com/auth/logging.write",
        "https://www.googleapis.com/auth/monitoring.write",
        "https://www.googleapis.com/auth/servicecontrol",
        "https://www.googleapis.com/auth/service.management.readonly",
        "https://www.googleapis.com/auth/trace.append"
      ]
    }
  ],
  "shieldedInstanceConfig": {
    "enab
leIntegrityMonitoring": true,
    "enableSecureBoot": false,
    "enableVtpm": true
  },
  "tags": {
```

```
    "items": []
  },
  "zone": "projects/networkmanager-397512/zones/europe-west1-b"
}
```

En cuanto a la configuración de este servidor es relativamente sencilla. Se debe dar acceso al proyecto de github privado que contiene los ficheros de configuración con las credenciales, crear el entorno virtual de python con venv, instalar los requerimientos, y configurar una línea en el cron que tras el inicio del servidor, levante levante un servicio de *uvicorn* en el puerto 8000 para que se pueda consumir la API para el despliegue de tenants.

Para ello, simplemente se debe ejecutar el siguiente script, añadiendo el token de github correspondiente.

```
sudo apt-get update -y
sudo apt-get install -y git
sudo apt-get install -y python3.11 pip
sudo apt install -y python3.11-venv
sudo mkdir /var/www/
sudo chmod 777 /var/www/
git config --global credential.helper store
echo https://<token>:x-oauth-basic@github.com > ~/.git-credential
git clone https://github.com/dtorrejon/networkmanager-deploy.git
/var/www/networkmanager-deploy/
sudo chmod 777 -R /var/www/
echo '@reboot /var/www/networkmanager-deploy/run_server.sh' | crontab -
pip3 install --upgrade pip
python3.11 -m venv /var/www/networkmanager-deploy/deploy
source /var/www/networkmanager-deploy/deploy/bin/activate
pip install -r /var/www/networkmanager-deploy/requirements.txt
```

Tenant-Server

Dado que la implementación de estos servidores se encuentra automatizada dentro del servidor Deploy, se explicarán los detalles en el apartado 12.1. módulo de despliegue.

En lo que respecta a la plantilla de despliegue en formato JSON, para desplegar las correspondientes máquinas en “Compute Engine”, se emplea la que se muestra a continuación.

```
POST
https://www.googleapis.com/compute/v1/projects/networkmanager-397512/zones/europe-west1-b/instances
```

```
{
  "canIpForward": false,
  "confidentialInstanceConfig": {
    "enableConfidentialCompute": false
  },
  "deletionProtection": false,
  "description": "",
  "disks": [
    {
      "autoDelete": true,
      "boot": true,
      "deviceName": "tenant-name",
      "diskEncryptionKey": {},
      "initializeParams": {
        "diskSizeGb": "10",
        "diskType":
"projects/networkmanager-397512/zones/europe-west4-a/diskTypes/pd-balanced",
        "labels": {},
        "sourceImage":
"projects/ubuntu-os-cloud/global/images/ubuntu-2204-jammy-v20230919"
      },
      "mode": "READ_WRITE",
      "type": "PERSISTENT"
```

```
    }
  ],
  "displayDevice": {
    "enableDisplay": false
  },
  "guestAccelerators": [],
  "instanceEncryptionKey": {},
  "keyRevocationActionType": "NONE",
  "labels": {
    "goog-ec-src": "vm_add-rest"
  },
  "machineType":
"projects/networkmanager-397512/zones/europe-west4-a/machineTypes/f1-micro",
  "metadata": {
    "items": []
  },
  "name": "tenant-name",
  "networkInterfaces": [
    {
      "accessConfigs": [
        {
          "name": "External NAT",
          "networkTier": "PREMIUM"
        }
      ],
      "stackType": "IPV4_ONLY",
      "subnetwork":
"projects/networkmanager-397512/regions/europe-west4/subnetworks/default"
    }
  ],
  "params": {
    "resourceManagerTags": {}
  },
  "reservationAffinity": {
    "consumeReservationType": "ANY_RESERVATION"
  },
  "scheduling": {
    "automaticRestart": true,
```

```
    "onHostMaintenance": "MIGRATE",
    "provisioningModel": "STANDARD"
  },
  "serviceAccounts": [
    {
      "email": "60698791164-compute@developer.gserviceaccount.com",
      "scopes": [
        "https://www.googleapis.com/auth/devstorage.read_only",
        "https://www.googleapis.com/auth/logging.write",
        "https://www.googleapis.com/auth/monitoring.write",
        "https://www.googleapis.com/auth/servicecontrol",
        "https://www.googleapis.com/auth/service.management.readonly",
        "https://www.googleapis.com/auth/trace.append"
      ]
    }
  ],
  "shieldedInstanceConfig": {
    "enableIntegrityMonitoring": true,
    "enableSecureBoot": false,
    "enableVtpm": true
  },
  "tags": {
    "items": []
  },
  "zone": "projects/networkmanager-397512/zones/europe-west4-a"
}
```