

Proyecto Integrado

Admón. de Sistemas Informáticos en Red

NETWORKING AUTOMATIZADO

Solucionando los problemas de la infraestructura clásica

AUTOR:

Javier Sánchez Páez

TUTOR/ES:

Víctor Montero Malagón

Javier Pastor Cascales

IES Zaidín-Vergeles (Granada), curso 2021-2023

# Agradecimientos

A mi familia, porque sin ellos no estaría donde estoy.

A Carlos, por ser mi mentor desde que tengo uso de razón.

A Vicky, por darme la paz y la guerra que necesito para seguir adelante.

A los buenos profesores que me han acompañado, por enseñarme y mantenerme motivado.

A mis amigos y amigas, por ser quienes son y por estar siempre ahí.

# Abstract

Al montar una infraestructura de red en un entorno siempre se hace difícil o pesado tener que configurar todos los dispositivos manualmente. Este método implica pérdida de tiempo, tener que aprender múltiples comandos de varios sistemas operativos distintos y no tener ninguna seguridad de si funcionará y si, en caso de que no funcionara, poder volver a una versión estable, entre otros.

La idea de este proyecto es demostrar que hoy en día podemos crear un entorno de desarrollo seguro también para nuestra infraestructura, con todos los puntos positivos que esto conlleva (testing, rollbacks en caso de fallo, notificaciones, visualización de recursos…)

# Palabras clave

(GitHub, AWX, Grafana, Kubernetes, Python, Telegram)

Contenido

[Agradecimientos 2](#_Toc132906395)

[Abstract 3](#_Toc132906396)

[Palabras clave 3](#_Toc132906397)

[Introducción 5](#_Toc132906398)

[Objetivos 6](#_Toc132906399)

[Requisitos previos 7](#_Toc132906400)

[Configuración e instalación de Rancher K3s 8](#_Toc132906401)

[Instalación de AWX 10](#_Toc132906402)

[Instalación de Netbox 15](#_Toc132906403)

# Introducción

Cada vez que tenemos que pensar en hacer cambios en una red nos echamos las manos a la cabeza. Es normal que, al momento, nos vengan pensamientos de:

* ¿Cómo voy a configurar mis dispositivos de uno en uno?
* ¿Cómo voy a aprenderme todos los comandos que tiene cada uno de los sistemas operativos para dispositivos de red?
* ¿Y si no funciona lo que quiero hacer?

Es por ello que nació la idea de Infraestructura como código (**IaC**). De esta manera, podíamos hacer cambios en nuestra infraestructura de forma simple y fácil, pero todavía se nos escapa un posible escenario: ¿Y si no funciona la configuración que quiero desplegar?. Para dar una solución a este problema, tenemos que hablar de **GitOps**.

En resumidas cuentas, lo que GitOps nos permite es tener un mejor control del software que vayamos escribiendo, usando nuestro repositorio (GitHub, Gitlab, Bitbucket) como **fuente de la verdad**. Básicamente la fuente de la verdad significa que todo el código fuente con el que automaticemos la infraestructura vendrá del repositorio. Es decir, todo el código que apliquemos en nuestro entorno de trabajo previamente habrá de pasar por el repositorio. De esta manera tendremos una infraestructura, podríamos decir, "portable". Así, si usamos GitOps en conjunción con herramientas CI/CD (AWX, GitHub Actions) tendremos un flujo de trabajo automatizado, estable y que puede reaccionar a errores haciendo rollbacks.

En este proyecto concretamente utilizaremos Netbox junto a GitHub como fuentes de la verdad (Netbox para configuración de dispositivos de red y GitHub para almacenar playbooks), AWX (Ansible Tower) y GitHub Actions para aprovechar el CI/CD, pyTest/pyATS para hacer pruebas sintácticas/de configuración y por último, Grafana para monitorización.

# Objetivos

El objetivo de este proyecto es montar y configurar un entorno de trabajo local con Kubernetes y ProxMox con todas las partes necesarias con las que el equipo de SRE/DevOps trabajará. También se harán algunas pruebas para comprobar el correcto funcionamiento del “workflow”.

# Requisitos previos

Para poder desarrollar este proyecto se han utilizado los siguientes hosts:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Hosts** | **Provider** | **CPUs** | **RAM** | **Versión S.O.** | **Info extra** |
| ASUS M3500QC | - | 16 | 16GB | Windows 11 | Cliente |
| K8s-rancher | VirtualBox | 4 | 8GB | Ubuntu22.04 | AWX |
| Ubuntu SRV | VirtualBox | 4 | 4GB | Ubuntu22.04 | NetBox |

Para poder montar toda la infraestructura necesaria podemos hacerlo de dos formas: utilizando Linux como sistema operativo principal o utilizando el Subsistema de Windows para Linux (WSL). En mi caso utilizaré la primera opción por:

* Facilidad de uso.
* Evitar concurrencia de datos al hacer dual boot.
* Mucha documentación disponible.
* Evitar problemas de networking que tiene WSL.

Al no utilizar Linux nativamente desde mi máquina principal, he configurado dos servidores con Ubuntu Server 22.04: Uno de ellos para el clúster de Rancher K3s y otro para Netbox y otras plataformas que podamos llegar a utilizar.

Las razones de utilizar K3s en lugar de un clúster “típico” de K8s son:

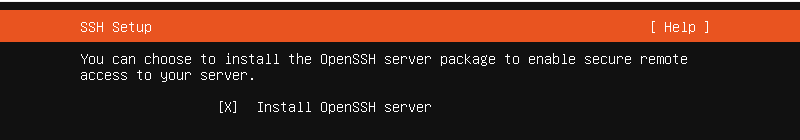
* Facilidad de uso.
* Ahorro de recursos del sistema.
* Mismo resultado.
* Simplicidad.

Además, al contrario que otras soluciones locales basadas en Kubernetes, Rancher nos permite abrir los puertos para nuestras aplicaciones de forma muy simple con Ingress (otras alternativas como Minikube necesitan cierta configuración extra específica).

## Configuración e instalación de Rancher K3s

La razón por la que aumenté los valores de CPU y RAM de nuestro servidor de Rancher K3s es para asegurar que la infraestructura no se quedara colgada cuando montáramos varios “pods”. El almacenamiento también se vio modificado para poder almacenar todos los datos necesarios y la red fue modificada a adaptador puente para que pudiéramos acceder a los servicios web de AWX y que este, a su vez, pudiera acceder a la API de NetBox. Podríamos haber utilizado una red interna, pero con el adaptador puente nos permite acceder a los dispositivos de los que luego queramos modificar la configuración.

Más allá de estos cambios, la instalación del sistema operativo que utilizará de base (Ubuntu Server) será “vanilla”, salvo porque instalaremos un servidor OpenSSH para poder acceder cómodamente desde nuestro cliente. Llamaré a mi usuario “jsp” y el nombre del equipo será “pi”.



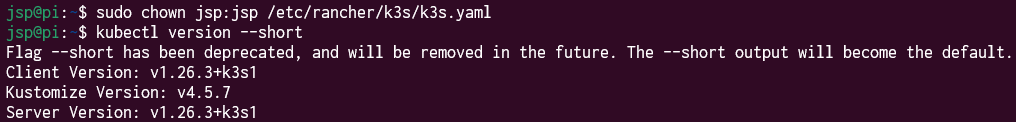
Una vez tengamos instalado nuestro Ubuntu Server, podemos proceder con la instalación de Rancher. Estando en la terminal de nuestro servidor, ejecutaremos el siguiente comando:



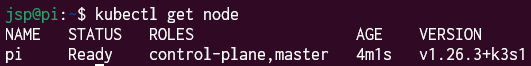
Lo que este comando hará es descargarnos el script de instalación y pasárselo como entrada a una terminal de sh. Cuando la instalación haya finalizado, debería salirnos un output como este:



Si intentamos hacer un “kubectl version –short” nos dará error, puesto que no tenemos los permisos para poder ejecutar el comando. Para arreglarlo, cambiaremos el dueño del archivo “k3s.yaml” con chown:



Como vemos, ahora sí tenemos un output sin errores. Vamos a comprobar que nuestro nodo se esté ejecutando:



Un pequeño detalle antes de empezar: En nuestro archivo “.bashrc” vamos a escribir dos nuevas líneas:

* alias k=”kubectl”
* export KUBECONFIG=/etc/rancher/k3s/k3s.yaml

La primera línea nos servirá para llamar al comando kubectl únicamente escribiendo k, lo cual nos ayudará en términos de eficiencia. La segunda nos evitará unos warnings muy molestos cuando hagamos un “kubectl get nodes” o cualquier comando en el que accedamos a recursos de nuestro clúster.

## Instalación de AWX

Para poder seguir adelante, vamos a hacer un pequeño repaso a qué es Ansible, de dónde salen y en qué se diferencian AWX y Ansible Tower.

Ansible es una herramienta de código abierto que permite la gestión de configuración y la orquestación de tareas en entornos distribuidos, lo que la convierte en una solución eficiente y muy potente para automatizar la implementación y gestión de infraestructuras IT.

Su funcionamiento se divide en tres elementos:

* **Playbooks**:

Un playbook es, esencialmente, el “set” de instrucciones que queremos ejecutar contra un determinado host. Un ejemplo de playbook que utilizaremos en este proyecto es, por ejemplo, asignar una dirección IP a un puerto de un router y comprobar que tenga conexión con otro host de esa misma red mandando un ping.

* **Inventarios**:

En un archivo de inventario escribiremos en qué hosts vamos a realizar las tareas de automatización. Dentro de un inventario de Ansible podemos establecer variables, como por ejemplo con qué usuario y qué clave entraremos al modo de configuración de un router.

Un inventario puede ser de tres maneras distintas: Estático (los hosts se escriben directamente en el archivo), dinámico (los hosts se reciben a través de una API de otro servicio, como por ejemplo Netbox) o inteligente

* **Roles**:

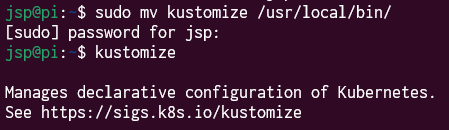
Los roles en Ansible sirven para agrupar los hosts en diferentes nombres. Esto puede ser muy útil si, por ejemplo, queremos hacer un cambio en todos los hosts, ya que de otra manera tendríamos que ir uno por uno. También es muy útil para agrupar los hosts útiles (en producción) de los inútiles (desconectados, averiados, en almacén…)

Teniendo esto en cuenta, AWX y Tower no son más que interfaces gráficas para Ansible con ciertas herramientas para hacer CI/CD. La diferencia entre AWX y Tower es semejante a la que hay entre Fedora y Red Hat: esencialmente son lo mismo, pero AWX es gratuito y libre de usar; sin embargo, tú eres el responsable de que todo funcione. Ansible Tower es de pago pero tenemos la garantía de que, si algo no funciona, tenemos al equipo de Red Hat para ayudarnos.

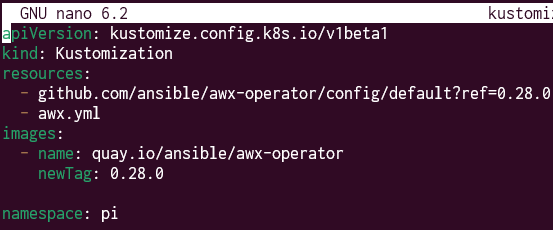
Una vez explicado esto vamos a instalar Kustomize, que es necesario para poder instalar el operador de AWX.



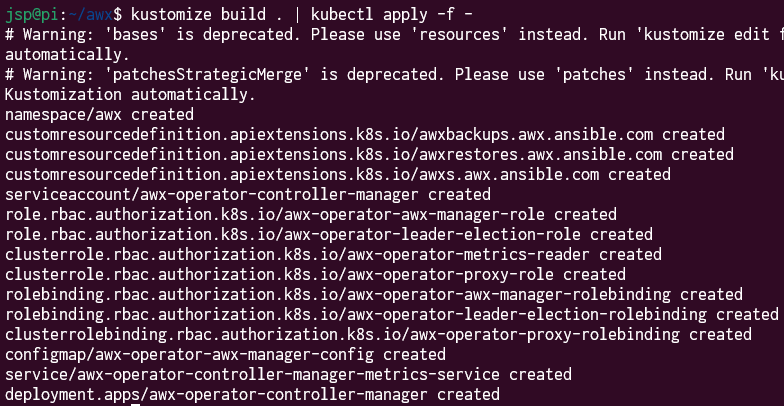
Por defecto se nos guarda el ejecutable de Kustomize en el directorio de trabajo en el que estemos. Vamos a moverlo a una ruta que esté dentro del PATH:



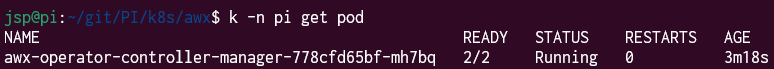
Ahora vamos a lo importante: Levantar el operador de AWX. El operador es el “arquitecto”, es decir, a él le mandaremos la tarea de instalar AWX con todas las dependencias necesarias (Postgres, servidor web, etc). Para ello, crearemos un archivo “kustomization.yml” con el siguiente contenido:



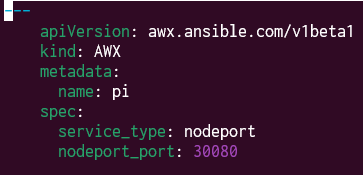
Una vez creado el archivo, levantaremos el operador con el siguiente comando:



Vamos a comprobar que el operador haya sido levantado (importante ahora ejecutar kubectl con “-n pi” para referirnos al espacio de nombres que acabamos de crear):



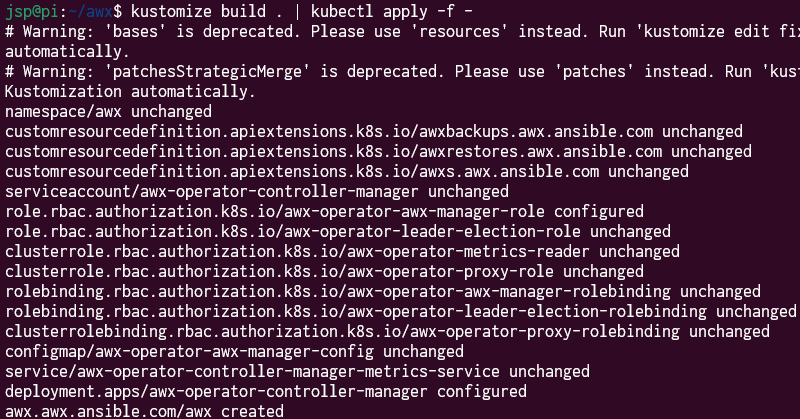
Una vez esté ejecutado, vamos a crear un segundo archivo llamado “awx.yml”:



Ahora volvemos al archivo “kustomization.yml” y descomentaremos la línea de awx.yml en metadata:



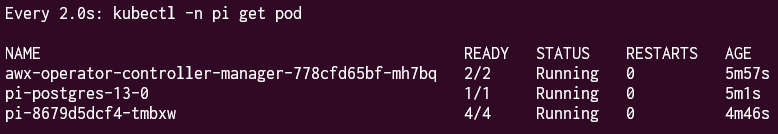
Volvemos a ejecutar el comando de antes:

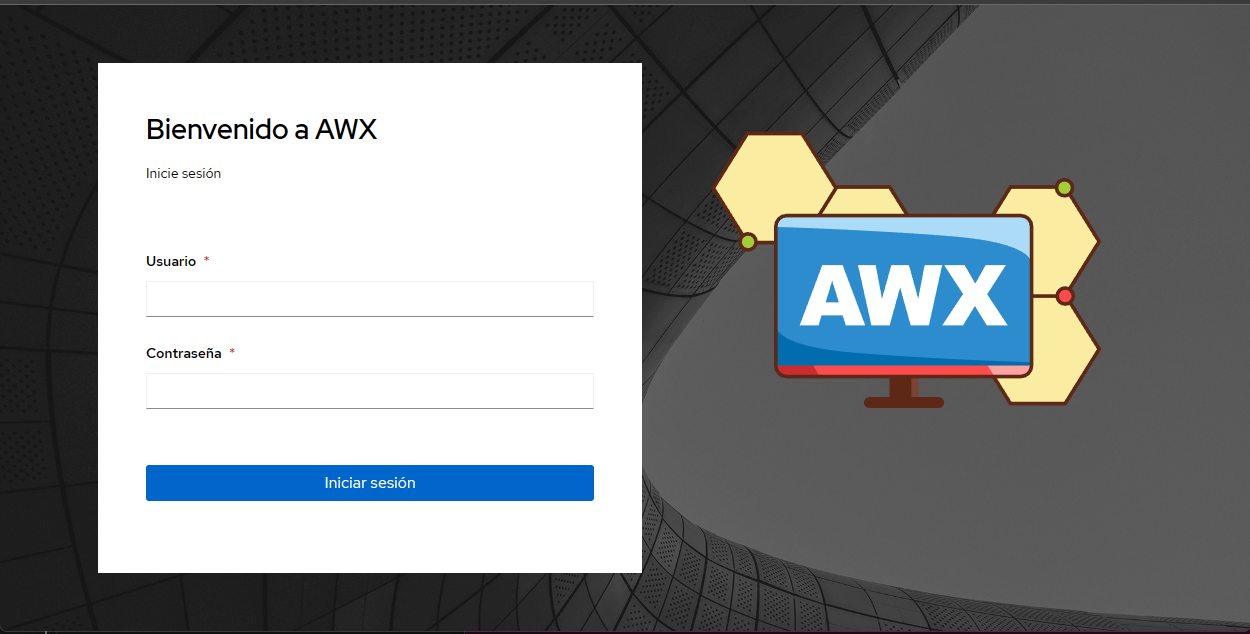


Para monitorear que todo se va ejecutando, ejecutaremos el siguiente comando:



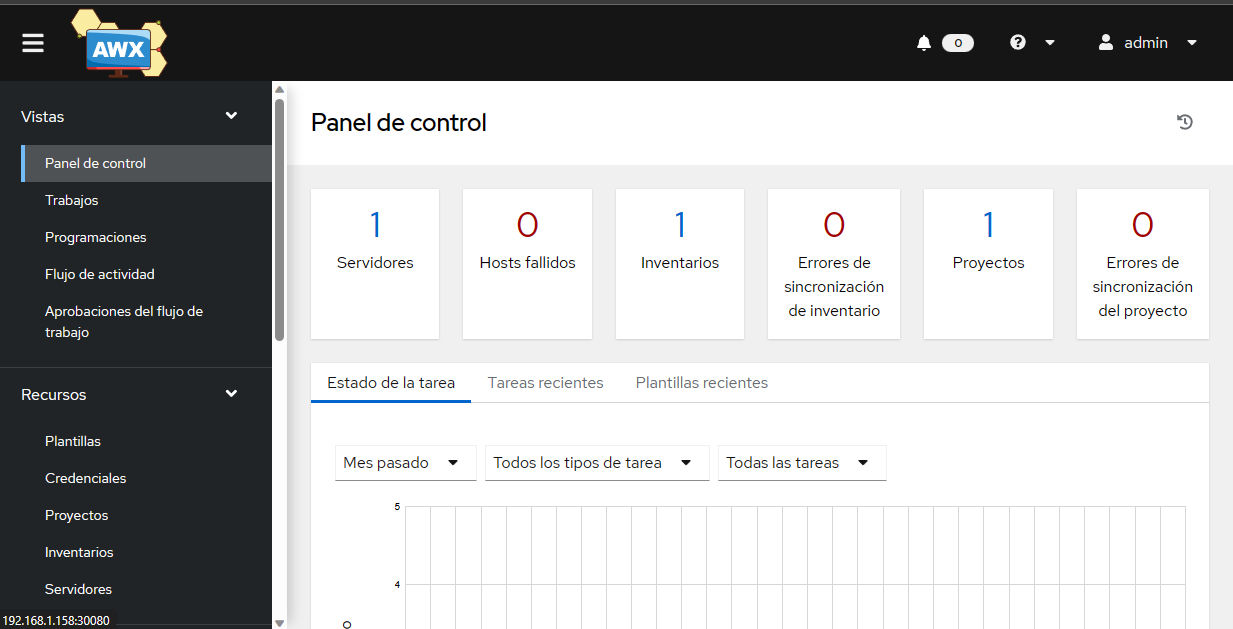
Después de aprox. 5-7 minutos (dependiendo de dónde hayamos montado el clúster) veremos que están ejecutándose los siguientes pods y podremos acceder a la interfaz web:





Ahora nos falta obtener la contraseña para entrar como usuario “admin”. Para ello, ejecutaremos la siguiente línea y copiaremos el output:





Vamos a cambiar la contraseña yendo a admin > detalles de usuario > editar:



## Instalación de Netbox

Como mencionamos anteriormente, esta máquina virtual que utilizaremos con Netbox está basada en Ubuntu 22.04 y, al igual que el clúster de Rancher K3s, tiene una configuración de red en puente para que podamos poblar el servidor fácilmente. También fueron modificadas las características de CPU y RAM pero en menor medida que en el clúster.

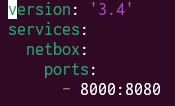
Para empezar nos conectaremos por SSH y clonaremos el repositorio de Netbox. Utilizaremos la imagen de Docker en lugar de la instalación “legacy” por simplicidad, velocidad y estabilidad; si queremos modificar algo es muy sencillo.



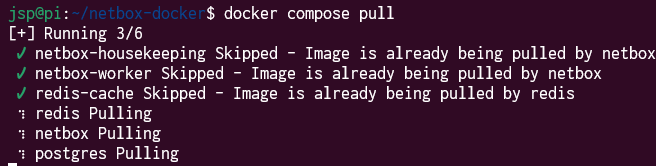
Una vez clonado, instalaremos Docker. Seguiremos los pasos de la guía oficial y comprobaremos que funciona haciendo un “Docker run hello-world”:



Ahora sí podemos continuar. Entraremos en la carpeta de netbox y crearemos un archivo “Docker-compose.override.yml” con el siguiente contenido:

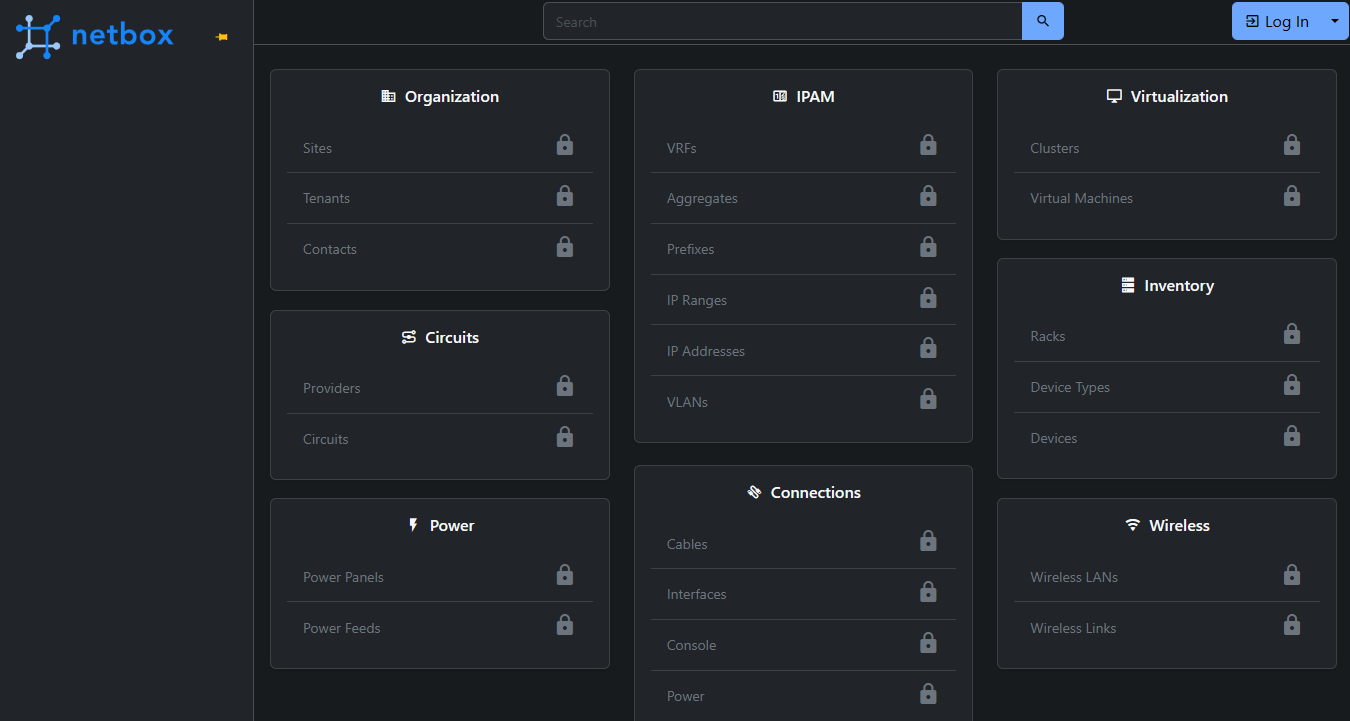


Con este archivo podemos cambiar el puerto en el que se descubrirá el servicio de Netbox. En este caso lo podemos dejar por defecto ya que no tenemos nada que utilice el puerto 8000. A continuación vamos a hacer un “Docker compose pull” para descargar todas las imágenes que vamos a necesitar en el Docker compose y a continuación un Docker compose up:

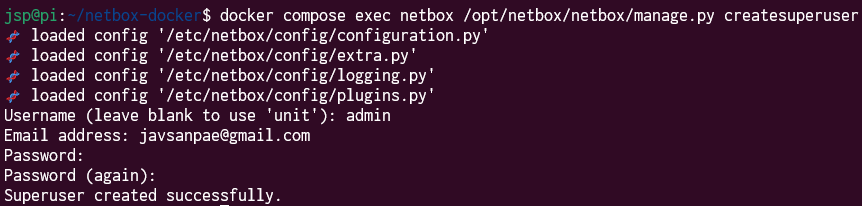




Al cabo de un momento si entramos en nuestra IP:8000 veremos la interfaz de Netbox:



Para poder acceder necesitamos crear un superusuario. Para ello, ejecutamos este comando:



## Configuración del repositorio en GitHub

En construcción…!