Montando mi BLOG con Wordpress.

Hecho por: Cristian Jiménez – Estudiante de Ciberseguridad

- El motivo por el cual estoy haciendo esto es simplemente la necesidad que me surgió del poder mostrarle a los demás lo que hago, puede que haya tardado un poco en hacer esto ya que estoy en casi tercer año de mi carrera de Ciberseguridad y aún no había creado un portafolio para evidenciar todo lo que hago, pero nunca es tarde.

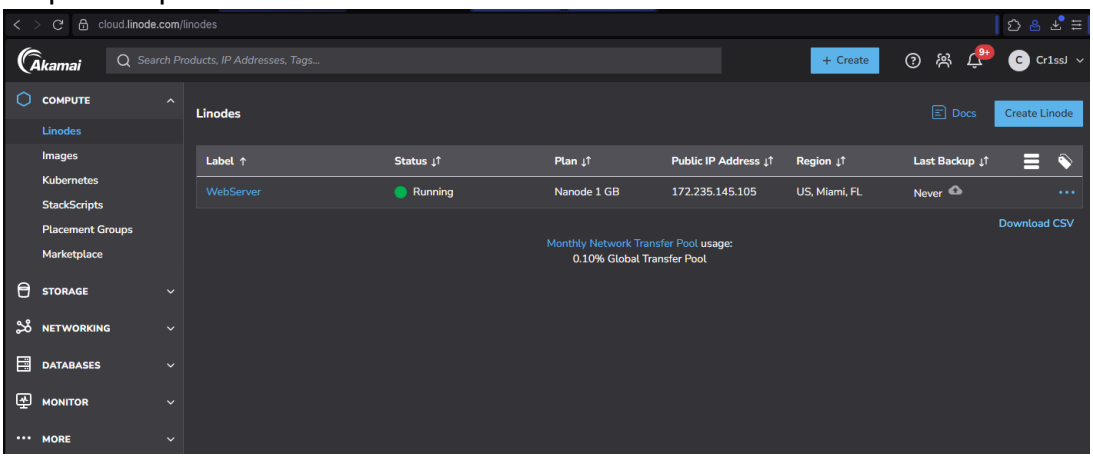
- Primeramente he decir que este primer “blog” lo estoy escribiendo en un archivo de Word, así que no sé como haré luego para subir este archivo o subir el texto al blog.

- Pero si, básicamente lo que voy a estar explicando (eso intentaré) será como hice mi propio blog, diferente al de muchos. Aquí lo que voy a estar haciendo es crear un wordpress con herramientas o frameworks como lo son Docker, Portainer y seguramente alguna otra que seguro mencione mas adelante. Todo esto está siendo hosteado en una instancia de Linode.

Creando la instancia

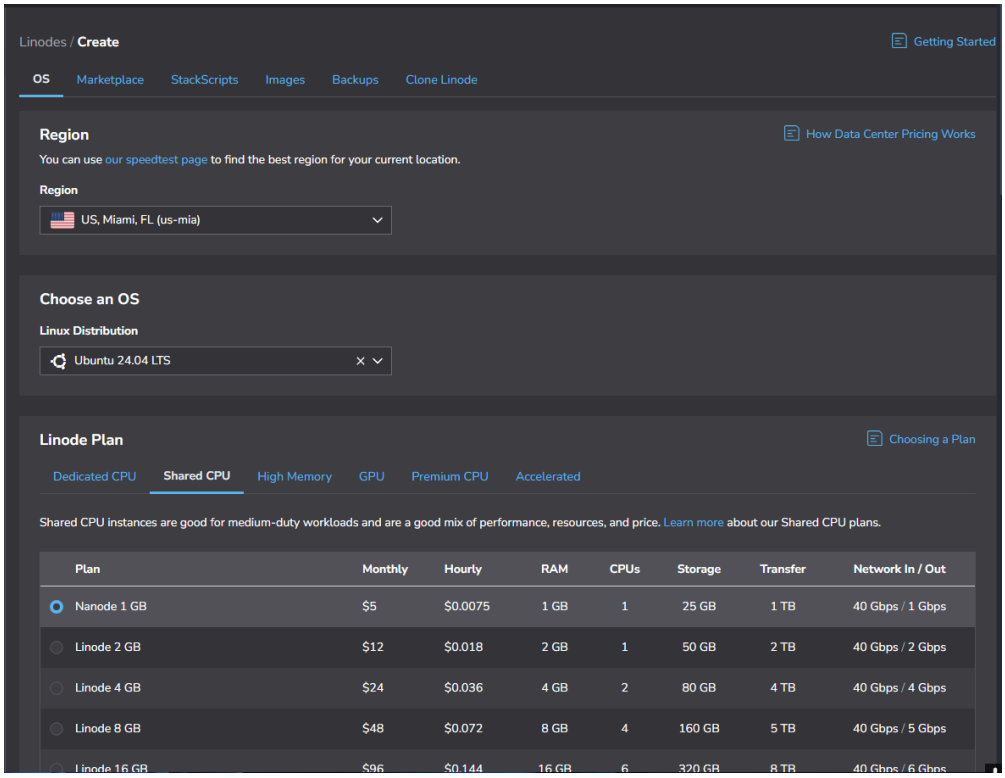
Para empezar toda esta trayectoria lo que primero hice fue crear una instancia en Linode. ¿Porqué Linode?, simplemente porque lo vi en un video de networkchuck, que al registrarme en linode con su código obtenia $100 de créditos para usar, asi que digamos que ese fue el motivo por el cual escogí este proveedor de nube en lugar de AWS o Azure o algún otro proveedor.

Una vez dentro de linode, debemos ubicar la opción de “Create Linode” en la esquina superior derecha

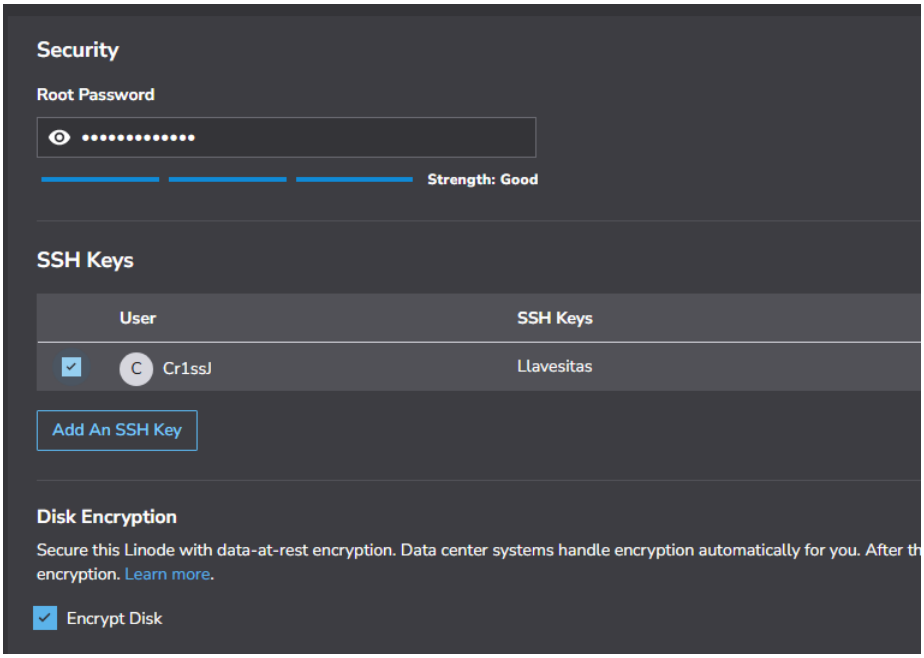


Una vez estamos creando la instancia, seleccionamos la región. (PD: Para los que estamos en Panamá por lo menos, los servidores con mejor latencia para nosotros siempre serán los de miami, Atlanta y Texas ;)). Luego de seleccionar la región, elegimos el OS, en mi caso utilicé Ubuntu 24.04 LTS.

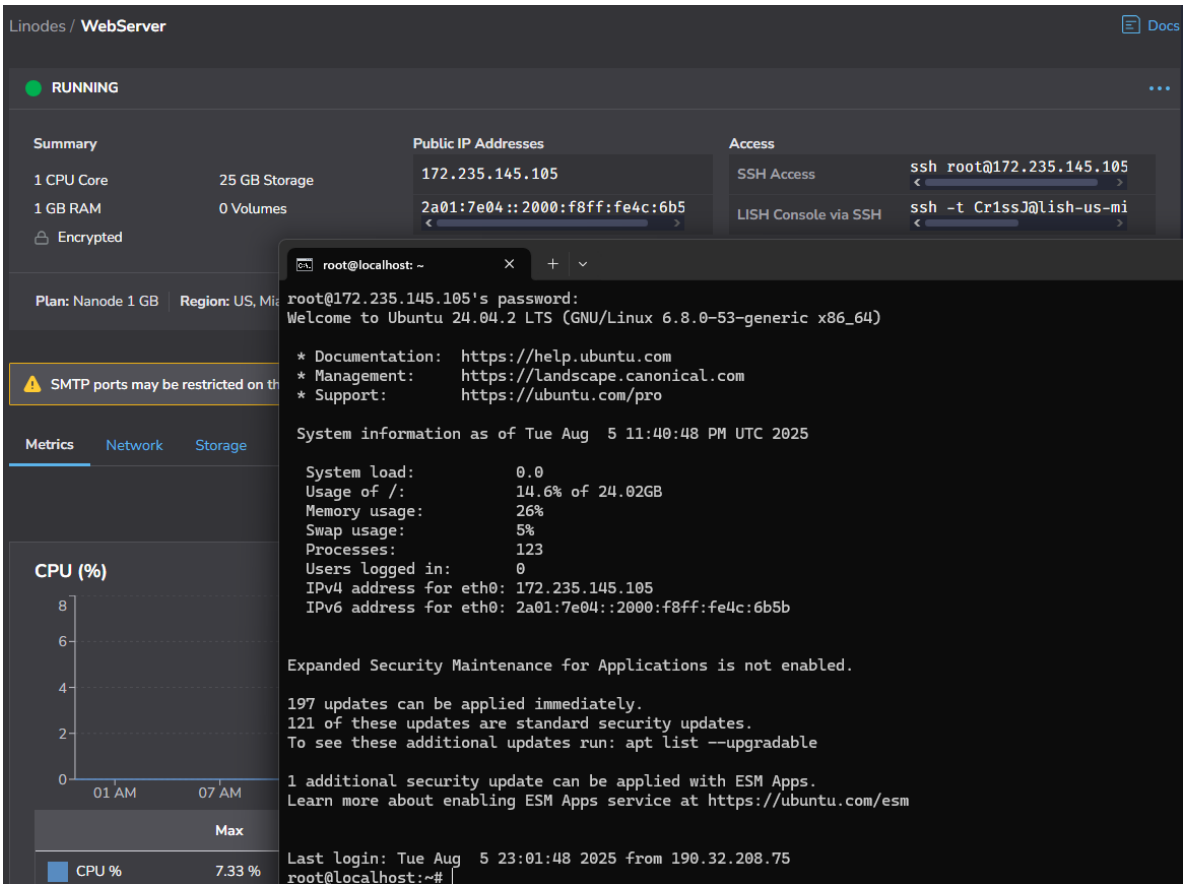
Luego seleccioné mi plan, que para mi casi propio utilizo el plan mas barato que hay que es de $5 al mes (con un hourly rate de $0.0075) en cual nos da 1GB de RAM, 1 CPU Core y 25GB de almacenamiento. Basicamente esto me sale “gratis” gracias al crédito de $100 que tengo por haberme registrado usando el código de networkchuck :p.



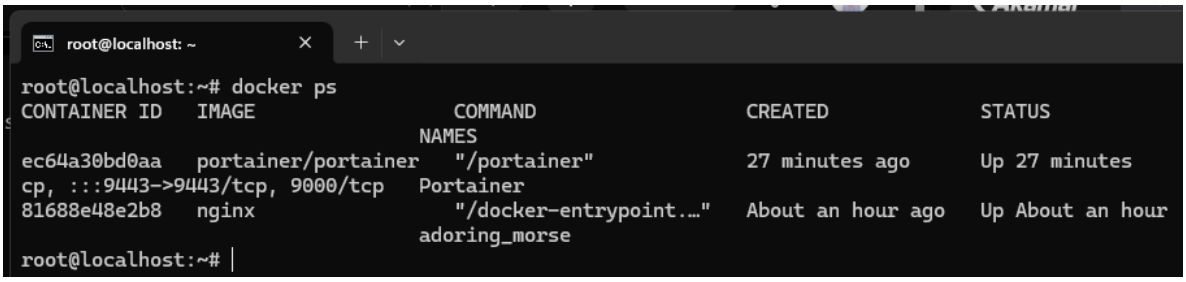
Luego creamos el par de llaves para poder conectarnos por SSH mas tarde, no sin antes configurar una contraseña segura para conectarnos a la instancia.



Una vez creada la instancia, el mismo dashboard nos deja copiar el comando ssh para conectarnos a la instancia directamente, sin necesidad de descargar las llaves. Asi que sería cuestión de abrir la terminal y copiar y pegar ;)



Una vez ya dentro de la instancia, corremos “sudo apt update” y ya ahora si instalaremos Docker corriendo el comando “sudo apt install docker.io -y” , en este paso yo ya tenia el Docker instalado y ya tenia 2 contenedores corriendo.



una vez instalado el Docker, procedo con montar una imagen de nginx, que luego esperemos que nos rediriga todo el trafico que va hacia el puerto 80 lo lleve al 443 o al puerto que nosotros asignemos. Para el nginx usamos “Docker un -itd -p 80:80 nginx”.

Luego de verificar de que la imagen se haya creado correctamente procedemos a instalar el Portainer, que si no lo conocen, es una herramienta que nos facilitaría mucho a la hora de trabajar con Docker.

Basicamente lo que haremos es crear un volumen que es en donde estará nuestro portainer, asi que para crear ese volumen corremos el comando “docker volume créate portainer_stuff”, posteriormente procedemos a crear la imagen del portainer con el siguiente comando:

```
root@localhost:~# docker run -d -p 9443:9443 -p 8000:8000 --name portainer --restart=alway
s -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer
```

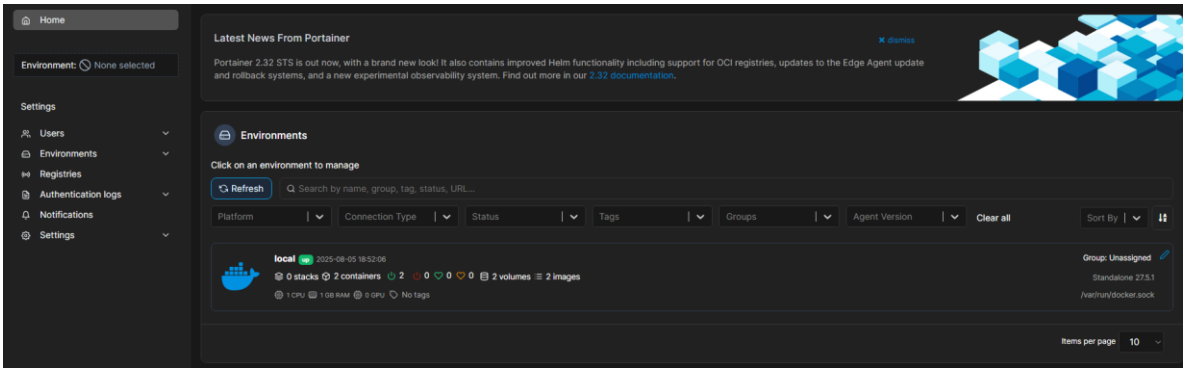
Luego de chequear el estado de los contenedores, solo para verificar que todo está en orden.

```
root@localhost:~# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED             STATUS             PORTS
                                NAMES
ec64a30bd0aa   portainer/portainer   "/portainer"             27 minutes ago      Up 27 minutes      0.0.0.0:8000->8000/tcp, :::8000->8000/tcp, 0.0.0.0:9443->9443/t
cp, :::9443->9443/tcp, 9000/tcp   Portainer
81688e48e2b8   nginx                "/docker-entrypoint.…"   About an hour ago   Up About an hour   0.0.0.0:80->80/tcp, :::80->80/tcp
                                adoring_morse
```

Ahora vamos a entrar a nuestro dashboard del Portainer abriendo una pestaña en el navegador y pegar la dirección IP de la instancia através del puerto 9443.

Quedaría tal que asi: https://direccionip:9443

Una vez dentro de la pagina, creamos nuestro usuario y contraseña del portainer y ya estaríamos dentro del Portainer.



A partir de ahora solo seleccionamos nuestro contenedor y podemos hacer lo que queramos con él.

En mi caso lo que hice fue eliminar el contenedor que tenia la imagen del NGINX desde el mismo dashboard del portainer, para luego crear otro contenedor con la imagen de Nginx Proxy Manager-

- Primero que nada lo que hice fue crear una nueva network bridge, desde el portainer colocandole el nombre de “proxy_net” y dejando en “true” la opción de “attachable” me preguntaba ¿Por qué no usar la red bridge que ya venia con la instancia de linode por defecto?
- Docker crea una red bridge por defecto llamada bridge, pero:
- No se puede modificar fácilmente (por ejemplo, para hacerla attachable)
- No es una red externa "visible" en Portainer (por defecto)
- No es recomendable usarla para múltiples servicios como NGINX Proxy Manager y WordPress

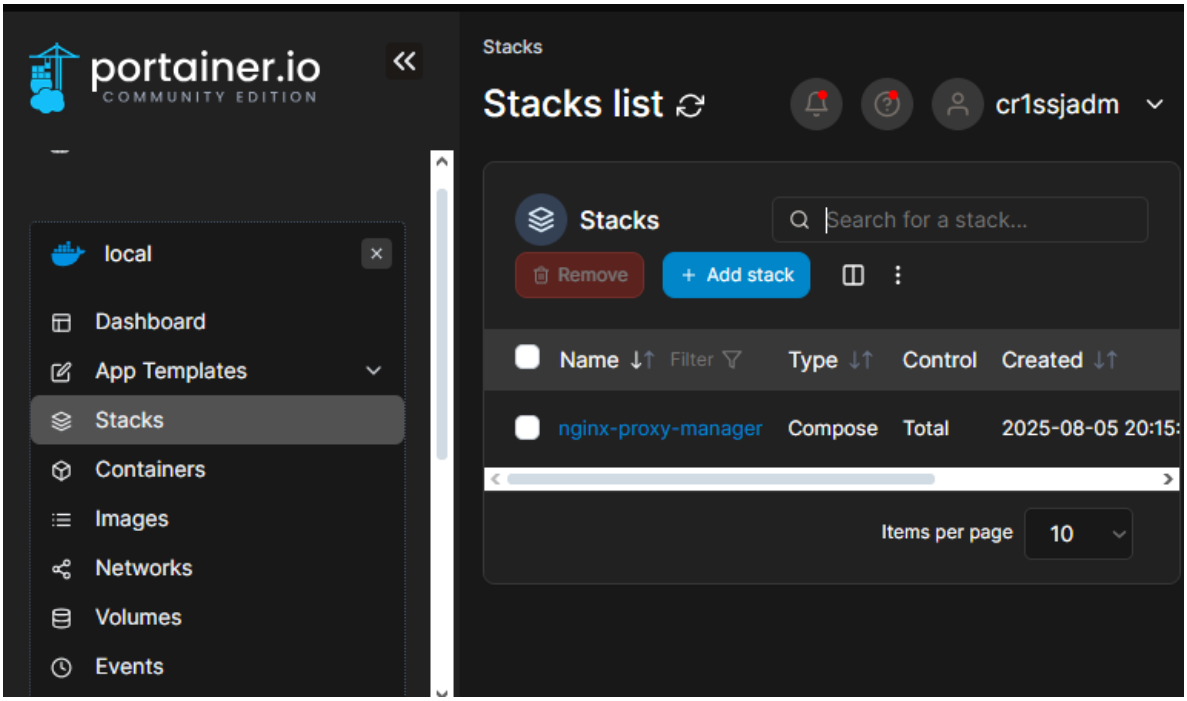
Al crear la nueva red externa, nos permitirá conectar fácilmente contenedores entre sí (como Wordpress y NGINX PROXY MANAGER) y usar esta red desde distintos stacks y contenedores en portainer.

Asi que luego de crear la nueva red, fui a desplegar Nginx Proxy Manager como stack, con el nombre de archivo “nginx-proxy-manager" y utilizando este codigo YAML:

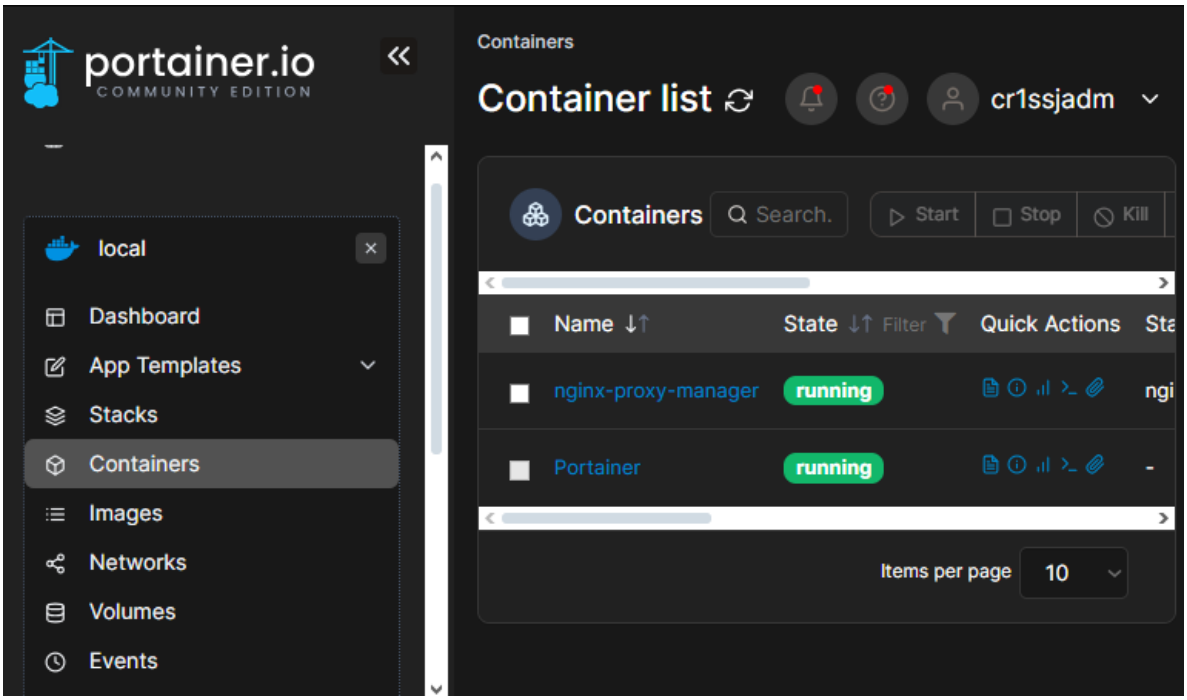
```
version: "3"

services:

  npm:

    image: jc21/nginx-proxy-manager:latest

    container_name: nginx-proxy-manager

    restart: unless-stopped

    ports:

      - "80:80"      # HTTP

      - "443:443"    # HTTPS

      - "81:81"      # Panel de administración

    volumes:

      - npm_data:/data

      - npm_letsencrypt:/etc/letsencrypt
```

```
    networks:
      - proxy_net


volumes:
  npm_data:
  npm_letsencrypt:


networks:
  proxy_net:
    external: true
```

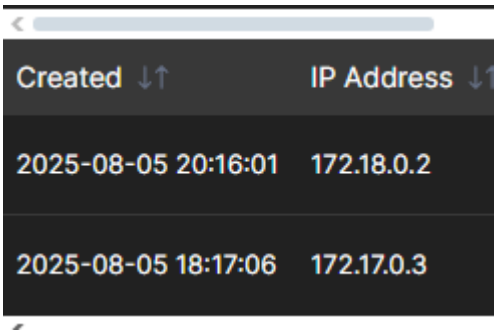
Y posteriormente darle click a Deploy Stack y nos quedaría así desde el dashboard “stacks”



Asi quedaría en la sección de containers:



Y si notamos en las direcciones IP (la de arriba es la nueva imagen de NGINX PROXY MANAGER) podemos darnos cuenta de que ambas pertenecen a diferentes redes.



Luego de esto vamos a conectarnos a nuestro panel de control del nginx proxy manager copiando la dirección IP y conectándome a través del puerto 81 que fue el que asigné para conectarme a la interfaz de administración.

Luego de varios intentos fallidos, y de preguntas a chatGPT decidí optar por utilizar la herramienta Ngrok, que basicamente es una aplicación multiplataforma que crea túneles seguros desde un punto final público a un servicio web local.

Lo instalamos en linux:

```
root@localhost:~/Docker-Wordpress-Proyect# curl -s https://ngrok-agent.s3.amazonaws.com/ngrok.asc | sudo t
ee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null
echo "deb https://ngrok-agent.s3.amazonaws.com buster main" | sudo tee /etc/apt/sources.list.d/ngrok.list
sudo apt update && sudo apt install ngrok
deb https://ngrok-agent.s3.amazonaws.com buster main
Hit:1 http://mirrors.linode.com/ubuntu noble InRelease
Hit:2 http://mirrors.linode.com/ubuntu noble-updates InRelease
Hit:3 http://mirrors.linode.com/ubuntu noble-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://ngrok-agent.s3.amazonaws.com buster InRelease [20.3 kB]
Get:6 https://ngrok-agent.s3.amazonaws.com buster/main amd64 Packages [8,626 B]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.6 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.3 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 229 kB in 1s (222 kB/s)
Reading package lists... Done
```

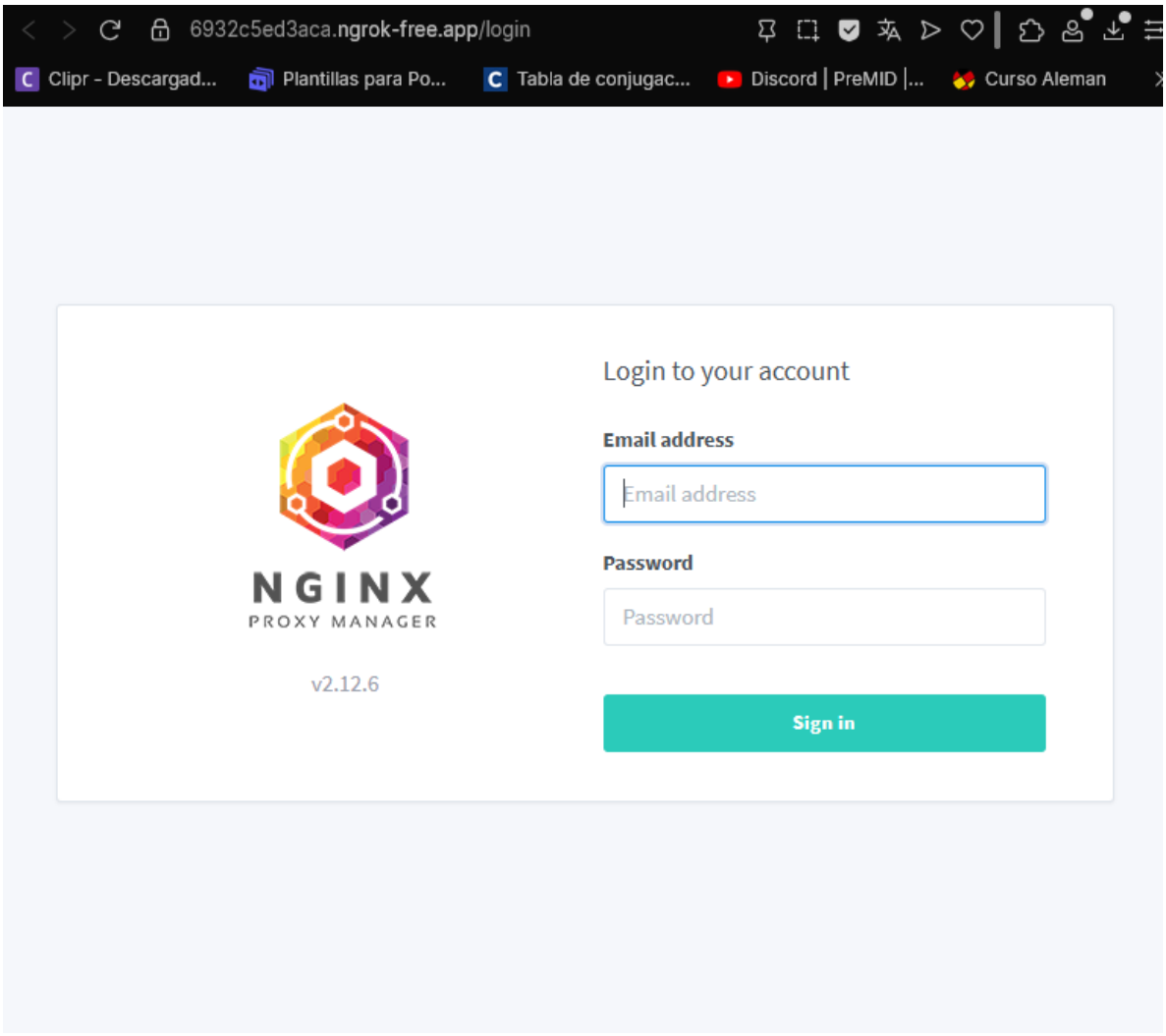
Aqui me autentico en ngrok y guardo mi toquen

```
root@localhost:~/Docker-Wordpress-Proyect# ngrok config add-authtoken
Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml
root@localhost:~/Docker-Wordpress-Proyect# n
```

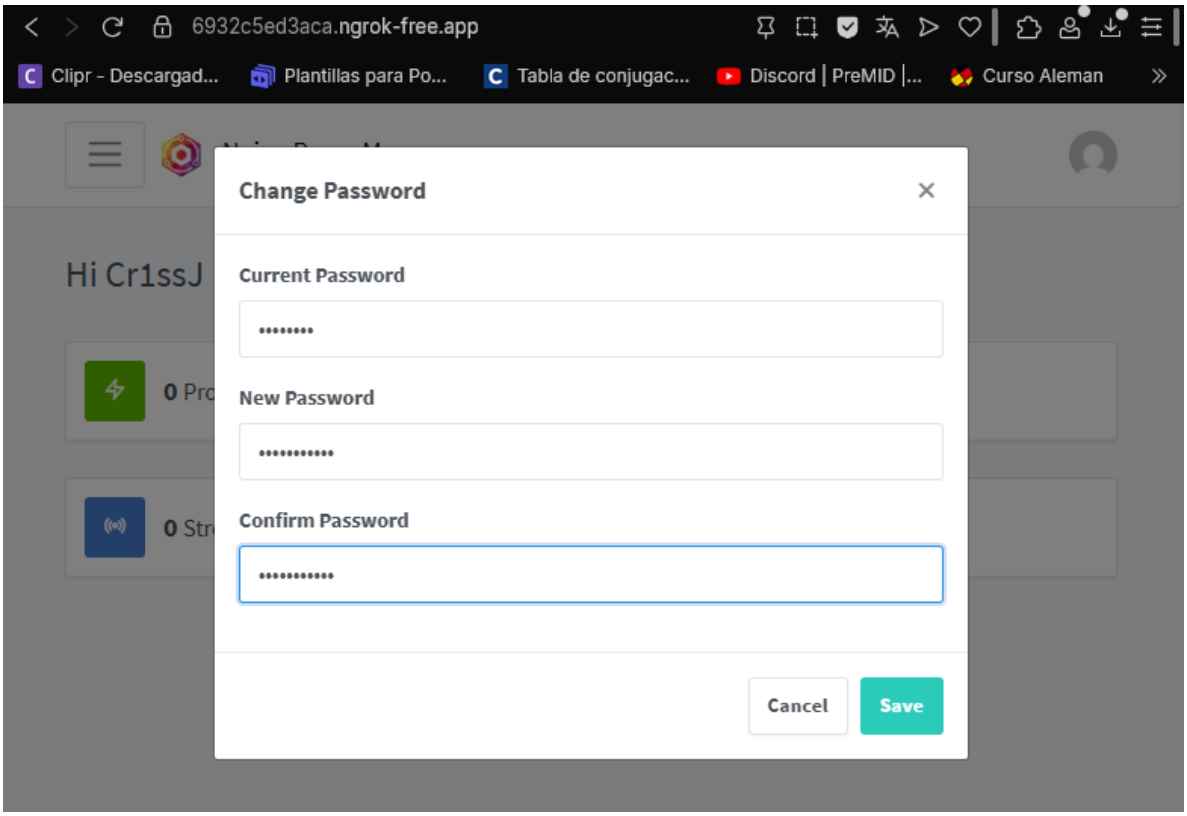
Ahora si voy a exponer el puerto 8881 utilizando ngrok http 8881 y obtengo la siguiente salida

```
ngrok                                                                    (Ctrl+C to quit)

Take our ngrok in production survey! https://forms.gle/aXiBFWzEA36DudFn6

Session Status                online
Account                       cristianjime2026@gmail.com (Plan: Free)
Version                       3.25.1
Region                        United States (us)
Latency                       41ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://6932c5ed3aca.ngrok-free.app -> http://localhost:8881

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       2       0.00    0.00    0.00    0.00

HTTP Requests
-------------

02:02:36.398 UTC GET /assets/source-sans-pro-v14-latin-ext_latin-700.woff2     200 OK
02:02:36.100 UTC GET /login                                                    200 OK
02:02:36.235 UTC GET /js/login.bundle.js                                       200 OK
02:02:36.133 UTC GET /images/favicons/apple-touch-icon.png                     200 OK
02:02:36.385 UTC GET /images/logo-text-vertical-grey.png                       200 OK
02:02:35.751 UTC GET /assets/source-sans-pro-v14-latin-ext_latin-regular.woff2 200 OK
02:02:35.774 UTC GET /api/                                                     200 OK
02:02:35.019 UTC GET /js/main.bundle.js                                        200 OK
02:02:35.883 UTC GET /images/favicons/favicon.ico                              200 OK
02:02:35.956 UTC GET /api/tokens                                               401 Unauthorized
```

Luego abro el enlace https y me lleva a la pagina de proxy manager

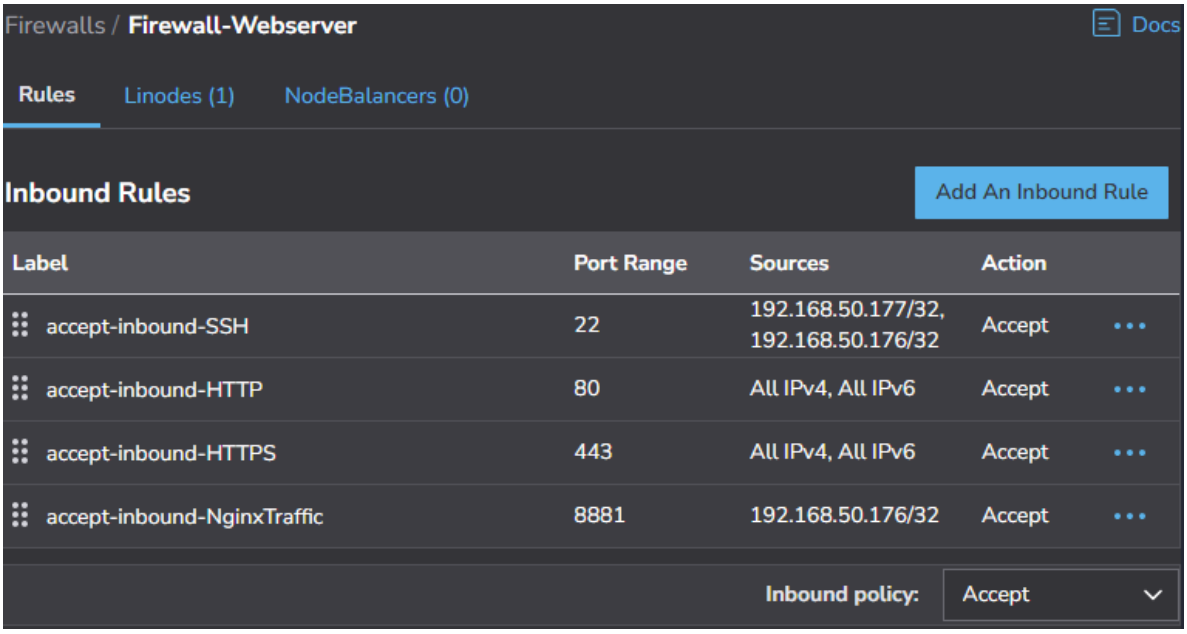


Aqui ingresamos con credenciales default y una vez entramos nos pide directamente que cambiemos el correo y contraseña, cosa que obviamente hacemos



Todo esto lo hice con tal de verificar de que el contenedor está bien desplegado y funciona correctamente, ya que me extraña que no pudiera ingresar directamente usando la IP. Asi que con esto descarto muchas cosas, como ya mencioné veo que todo está funcionando bien, entonces parece que el problema tiene que ver con el acceso externo directo que sería la IP:8881 que aparentemente está bloqueado. Muy posiblemente pueda deberse a un bloqueo automatico de parte de Linode como proveedor de servicios, asi que en mi caso iré a la pagina de linode y crearé un firewall con reglas explicitas permitiendo el trafico externo a los puertos 22. 80, 443 y 8881

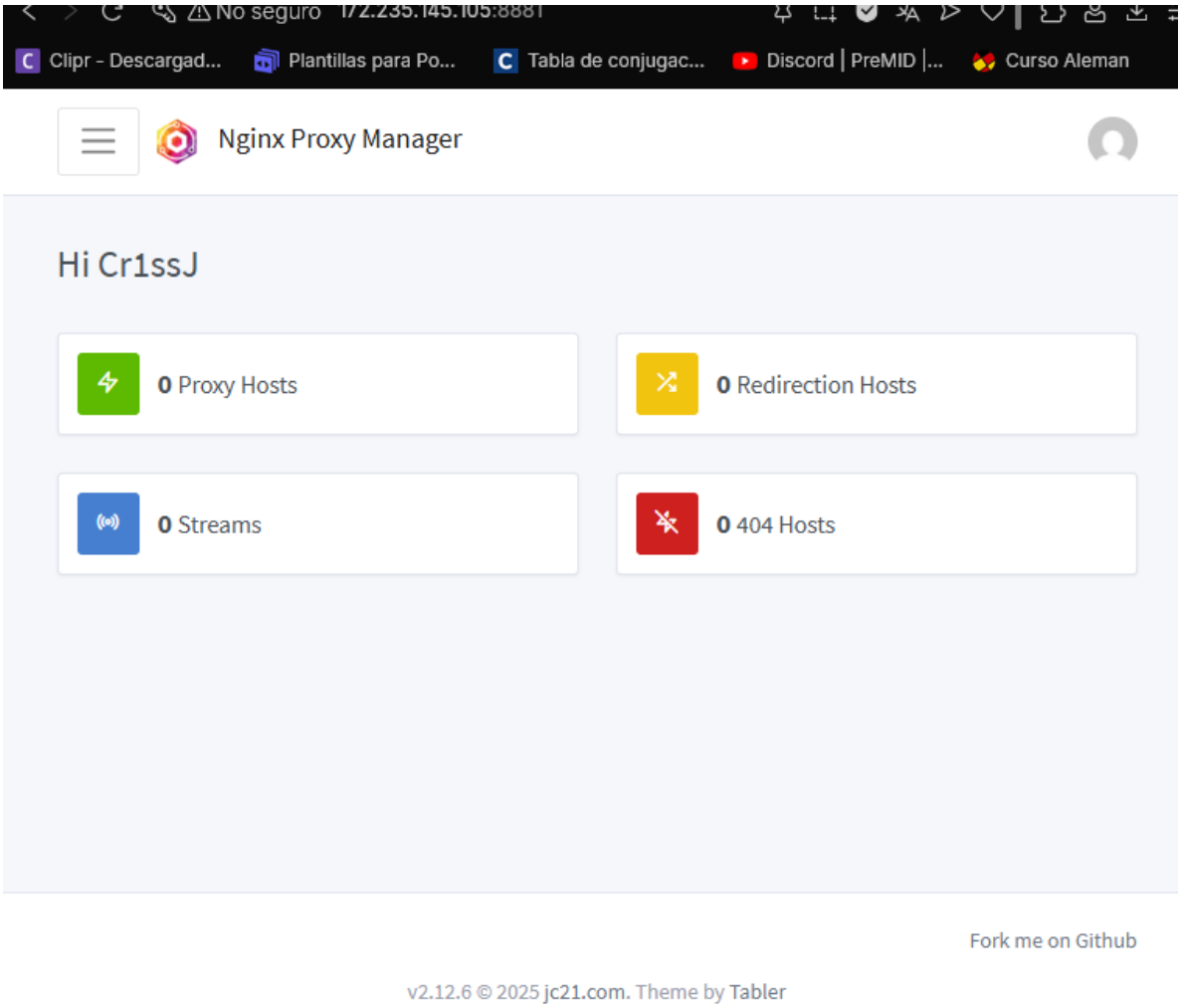
Que alfinal me queda asi



Pero es casi por el gusto porque el problema sigue, no puedo entrar a la pagina por el puerto 8881. Cuando intenté hacer ping a la IP obtenia respuesta, pero cuando hacia ping hacia ese puerto en especifico obtenia un connection refused. Asi que lo que haré será chequear el firewall del ubuntu.

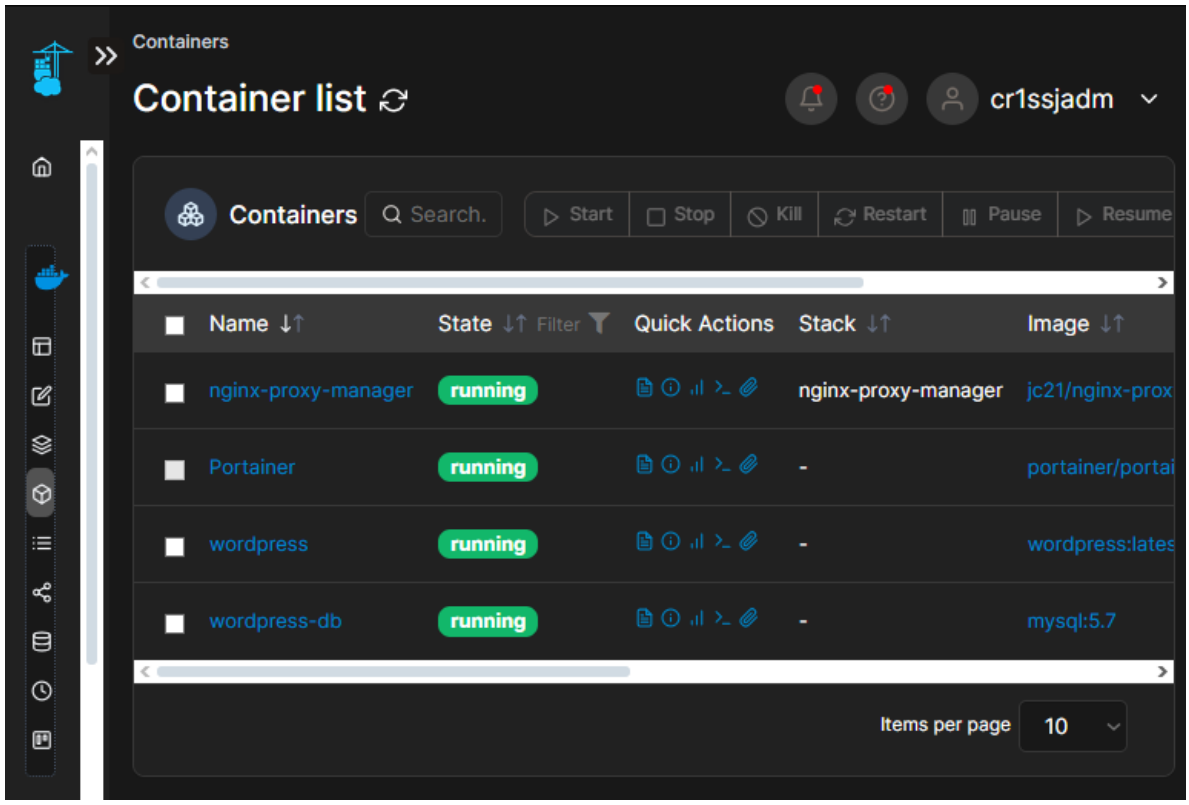
Luego de varios chequeos y vueltas, el error fue lo más tonto posible y siempre estuvo bajo mis narices XD, mi problema era de que no estaba utilizando la IP publica de la instancia de linode y estaba usando la IP del proxy_net que habia creado casi al inicio.

Entonces ahora si puedo sin problemas entrar a mi panel de control de Nginx proxy Manager.

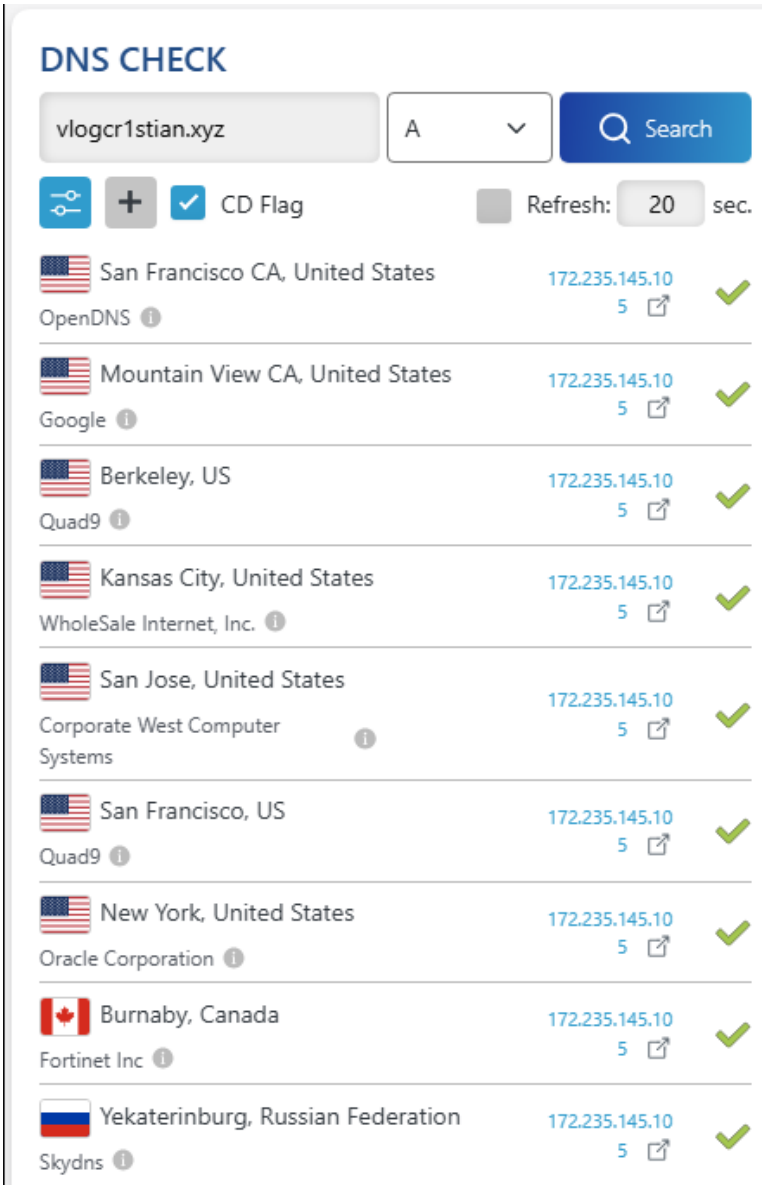


Entonces AHORA SI se puede proceder con la configuracion de los contenedores de WordPress, se debe tener en cuenta que voy a utilizar la misma network “proxy_net” para que WordPress y NGINX Proxy Manager estén en la misma red y puedan comunicarse entre sí. Pero primero vamos a crear un contenedor para la base de datos, en mi caso voy a utilizar MySQL que es con lo que estoy un poco más familiarizado y luego de la BD proceso a crear el contenedor que tiene wordpress..

Quedando todo así:

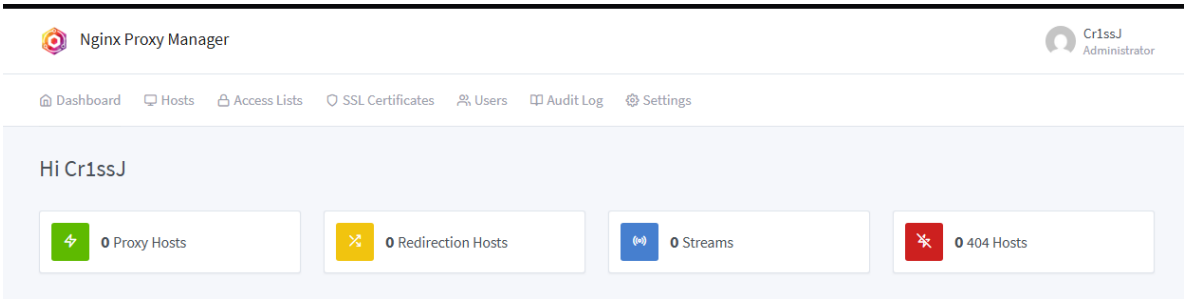


Ahora para configurar nuestro Nginx Proxy Manager, necesitamos tener nuestro propio dominio. Yo en mi caso encontré un dominio en porkboun por $2.06 el primer año, asi que lo compré, y ahora ya con el dominio comprado, solo tenia que esperar unos minutos para que los servidores DNS se actualizaran, y alfinal parece que tenemos dominio que es el que seguro están viendo en sus navegadores.

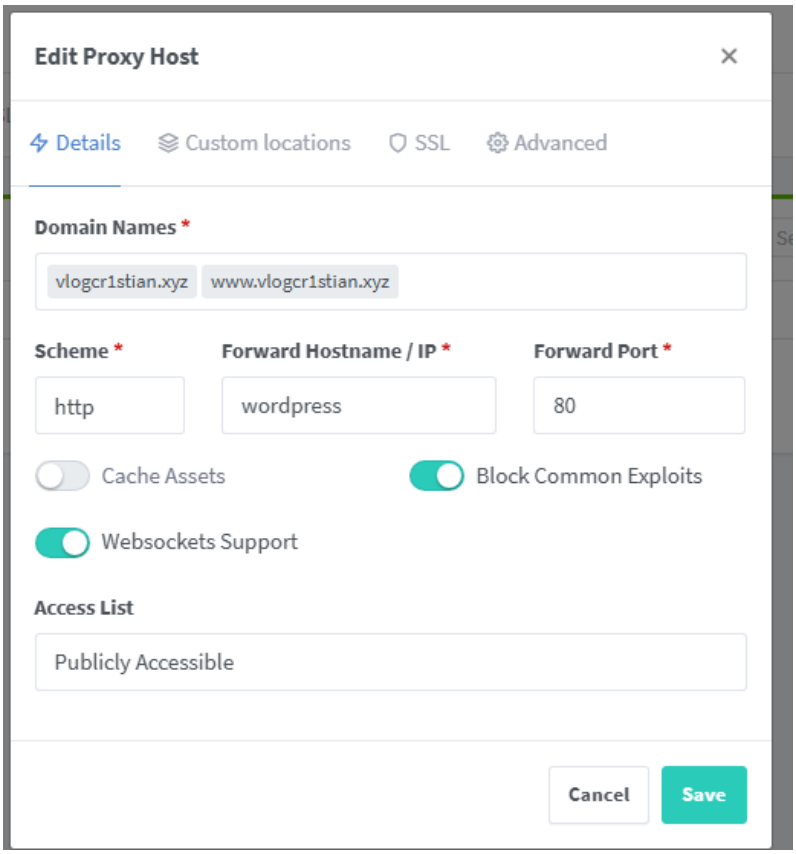


Ahora que los servidores DNS se actualizaron con mi dominio nuevo ya puedo ir a NGINX Proxy Manager y configurar el servidor de la forma que quiero.

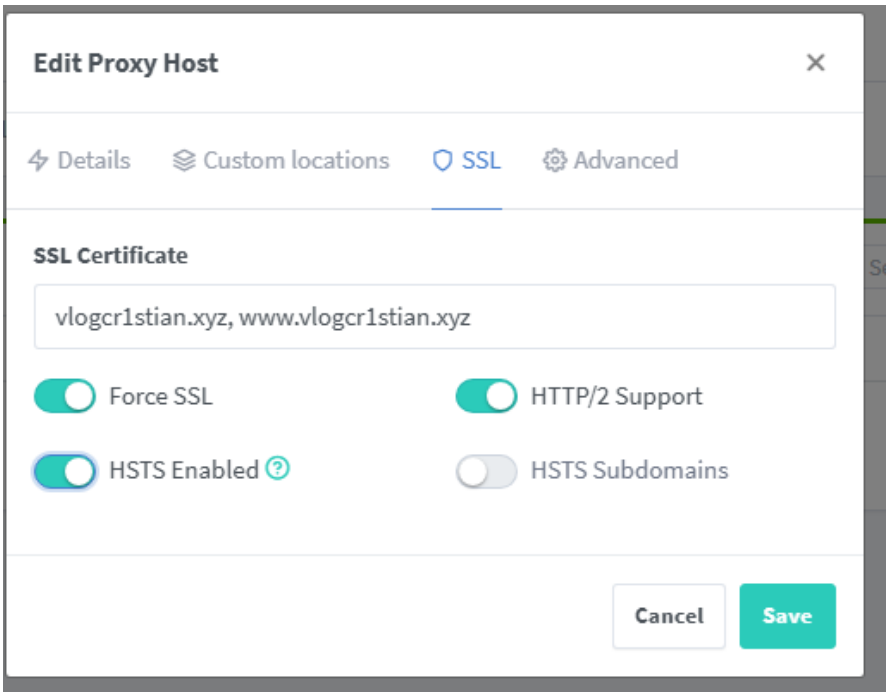
Dentro del panel de control de NPM voy a la sección de Proxy Host



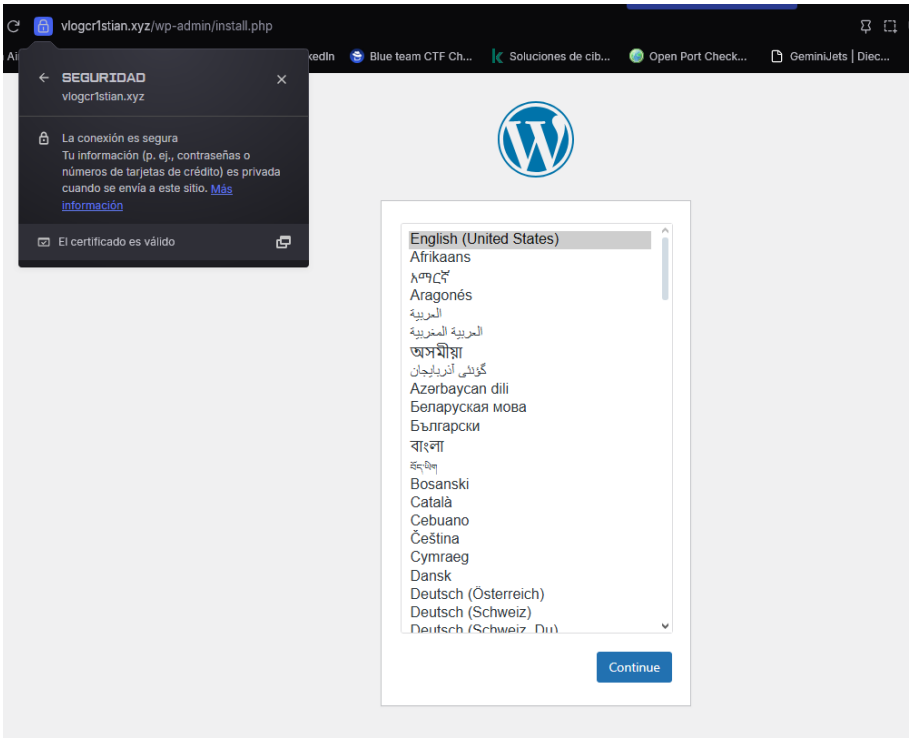
Y ahí agrego un nuevo host, agregando mi dominio, el nombre del contenedor de wordpress y agregamos el puerto de destino



Y ya como último paso tocaría ir a la sección de SSL y activar el forzado SSL y Soportar HTTP/2



Y luego de un buen par de horas de darle vueltas a las cosas y leer un poco y de ayuda de chatGPT obtengo lo que todo el mundo espera a la hora de montar un WordPress con certificado válido.



Si es verdad que es posible que haya tenido otras formas mas sencillas de montar este WordPress, pero la verdad quise mantenerme firme en montar este metodo simplemente para superarme a mi mismo y aprender un poco sobre Portainer y un poco sobre como se montan estos webservers cuando no eres programador como es mi caso.

Veremos si en algún futuro cercano me acuerdo de documentar mas cositas para entonces poder subirlas al blog.

Nisiquiera sé como subiré este archivo, literalmente es un archivo Word a la hora de escribir esto.

Gracias por llegar hasta aquí :)