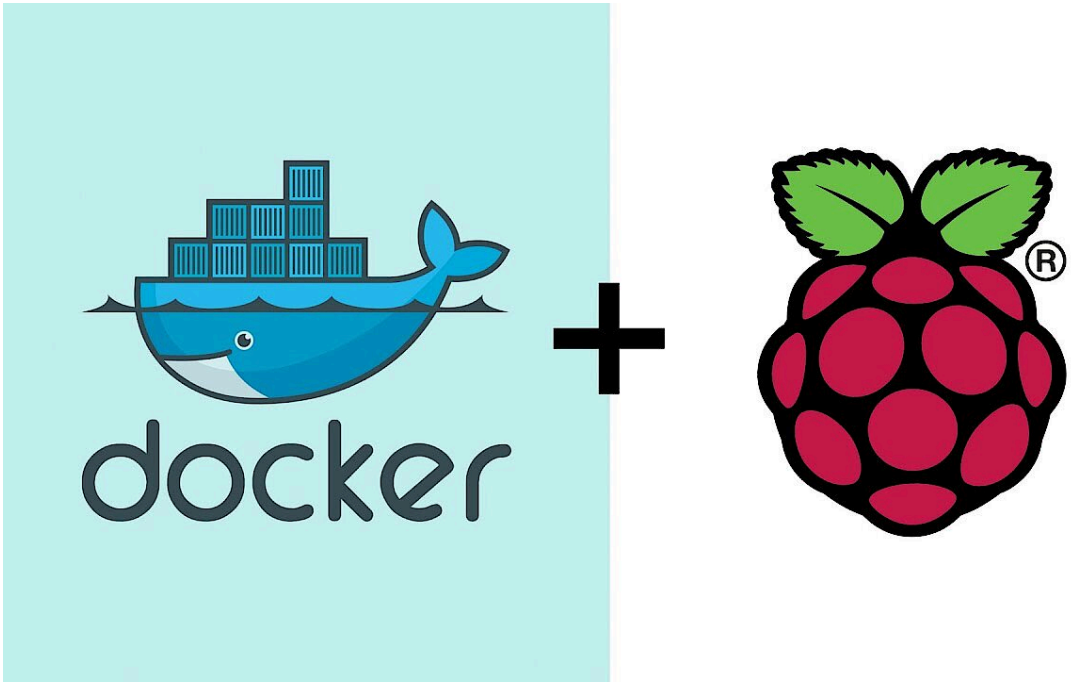
Docker bajo Raspberry Pi



Por: Carlos Manuel Gámez Pérez de Guzmán

1. Objetivos del proyecto

1.1 Contexto del proyecto

Para comenzar mencionare los inicios del proyecto, el cual pone en contexto el nacimiento y desarrollo del mismo.

Este proyecto se gesta desde el inicio del segundo cursos del grado de ASIR, en sus principios sin muchas pretensiones, simplemente ganas y curiosidad de como funcionan los servidores expuestos a la Red y como cualquiera puede crear y disponer de uno. En sus inicios esto solo sería un servidor de archivos el cual de manera muy rudimentaria compartiría archivos de forma local a través del protocolo SAMBA, una vez fui consciente de las posibilidades que podía tener no me detuve ahí, lo empresa a utilizar como mi propio servidor de pruebas reforzando lo aprendido en las clases y cómo dichas herramientas las podía aplicar a mi dia a dia en uso personal como mi propio servidor.

El servidor fue creciendo implementando tecnologías como docker, automatizaciones, copias de seguridad y demás tecnologías que descubriremos a lo largo de dicho documento. En definitiva el objetivo en sus inicios fue el de aprendizaje y formación a la vez que empieza a ver como dicho servidor podía ayudarme a mi y otros en el dia a dia.

1.2 Objetivos

El objetivo principal del proyecto es el de conseguir un servidor expuesto a la red totalmente funcional y accesible desde todo el mundo, que diera diferentes servicios a mi personalmente o diferentes usuarios, como amigos en los servicios de nube privada o servidor de video. Dicho servidor tiene que cumplir 3 principales características:

1. Ser fiable, ser accesible desde todo el mundo cuidando la seguridad del mismo, evitando ataques como intentos de conexiones remotas a través del protocolo SSH, el cual se encuentra expuesto en el router para poder ser administrado desde cualquier parte del mundo. Por otro lado deberá ser fiable a pruebas de casuísticas externas al mismo, como pueden ser los apagones, roturas, subidas de tensión, etc.

2. Ser escalable, encontrar un hardware económico pero versátil para poder correr simultáneamente varios contendores, que permitiera la conexión de más discos duros en el futuro por la necesidad de expansión en servicios como Nextcloud.

3. Duradero, estaría en la balda de una habitación, por lo que necesitaría algo que pudiera aguantar los cambios de temperatura, descartando piezas móviles como un ventilador, debido al gran número de horas seguidas que tendría que estar en funcionamiento. Para ello tuve que escoger un hardware que funcionará bajo la arquitectura ARM, ya que este permitiría la refrigeración pasiva.

Una vez se tuviera dichos requisitos, sólo se tenía que adaptar el servidor a la necesidades de los usuario que lo utilizarían, en este caso y sobre todo al principio, serían únicamente las mías.

1.3 Objetivos no alcanzados

Respecto a los objetivos a alcanzar mencionados en el punto anterior, cabe destacar que se lograron la mayoría, partiendo del punto que es un proyecto inacabable, ya que puedes estar en evolución constante, pero si caben destacar algunos punto que se pretenden lograr al inicio del proyecto y por diversos motivos no ha sido posible.

- Motivo económico: como mencionaremos más adelante el propio servidor solo tiene un aprovisionamiento de 500 GB en forma de disco duro sólido conectado a un puerto USB, la idea principal era la de comprar un docking stations con soporte para RAID físico (el cual no descartó integrar en un futuro). Por otro lado, el proyecto contaría con un SAI para proteger los datos y el equipo de posibles cortes de luz y subidas de tensión, ambas implementaciones escalaban bastante el precio del proyecto, por lo que no se pudo integrar.

- Rendimiento insuficiente: el hardware cuando se compró hace 2 años, era de los más potente en tecnología ARM, ya que contaba con el último modelo de Raspberry PI para ese entonces. Siendo suficiente para el uso de una única persona en varios servicios simultáneos, se queda corta cuando ese mismo uso lo hacen varias personas por lo que en el caso de ser necesario el uso de personas simultáneamente a varios servicios sería necesario tirar por algo más potente, incluso decantarse por arquitectura X86 ya que suele ser más potente en cuestión de multitarea.
  Sin embargo en la actualidad han salido diversas “imitaciones” a Raspberry PI de procedencia china, que pueden llegar a resultar más atractivas, para este tipo de proyecto, que la Raspberry.

2. Escenario

El escenario se compone de los siguientes elementos de hardware:

- Servidor Raspberry PI: este elemento es la pieza de hardware donde se apoya todo, en él se ubican todos los servicios que luego se ofrecerán.
  Se trata de una Raspberry Pi 5 con las siguientes características:

  - **CPU**: Broadcom BCM2712, 4 núcleos ARM Cortex-A76 @ 2.4 GHz
  - **GPU**: VideoCore VII, compatible con OpenGL ES 3.1 y Vulkan 1.2
  - **RAM**: 8 GB LPDDR4X-4267
  - **Almacenamiento**: microSD 64GB

- Disco duro de 500 GB: En dicho disco duro se guardaran todas las configuraciones de los servicio, recetas docker-compose, bases de datos creadas y archivos que se sirven o se guardan mediante nubes privadas. Al contrario que la tarjeta SD anteriormente mencionada, la cual solo tendrá el sistema operativo Debian 11 “Bullseye”, ya que si se tuvieran todos los archivos en la tarjeta SD esta se podría ver rápidamente dañada por el alto número de escrituras que se realizan al tener instalados numerosos servicios.

- Ordenador secundario: Este ordenador secundario se trata de una vieja torre la cual tiene instalado Linux MINT, con un servidor FTP, el propósito de este es poder realizar copias de seguridad en el mismo aprovechando la alta velocidad de la Red local.
  Esto se ideó de esta manera debido al no contar con la caja de discos mencionada anteriormente, esto provoca que al no tener RAID no habría respaldo de los archivos, por lo que se podrían perder. Más adelante se explicará qué contenedor se encarga de dicha tarea.

- Router: el router nos permitirá el acceso a internet, por lo que podremos acceder al servidor desde el exterior de la red local, y la interconexión de los diferente elementos, como puede ser la raspberry PI con el ordenador donde se realizan los respaldos.

3. Fundamentos previos a saber

Antes de explicar que se ha instalado y cómo funciona el servidor, hay que tener claro algunos términos sobre el funcionamiento de los servidores.

Cuando un usuario accede desde Internet a un servicio alojado en un servidor (como una web o una API), intervienen varios elementos clave en la red.

Primero, el usuario escribe un nombre de dominio (como `ejemplo.com`). Este nombre es resuelto por un servidor DNS (Domain Name System), que traduce el dominio a una dirección IP pública asociada al servidor donde está expuesto el servicio.

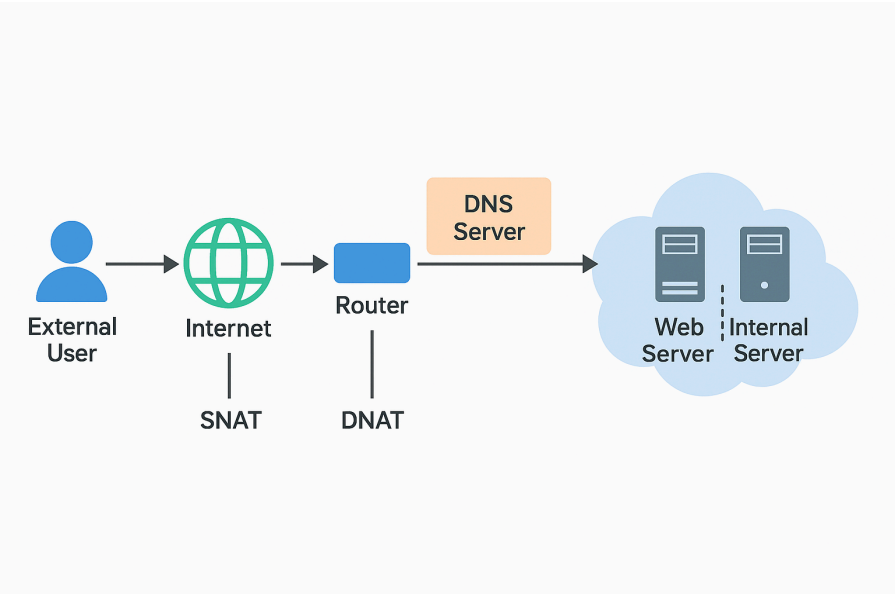
Una vez que se conoce la IP, el paquete de datos se dirige a esa dirección. Aquí entra en juego el NAT (Network Address Translation), especialmente cuando el servidor está en una red privada (como una LAN detrás de un router).

Tipos de NAT usados:

- DNAT (Destination NAT): se usa para redirigir el tráfico entrante desde la IP pública hacia una IP privada específica dentro de la red local, donde está el servidor real. Por ejemplo, el router recibe tráfico en el puerto 80 (HTTP) y lo redirige al puerto 80 del servidor local `192.168.1.100`, en mi servidor detrás de esto se encuentra un servidor proxy que redirige las peticiones al puesto donde se aloja el servicio concreto.

- SNAT (Source NAT): se aplica generalmente al tráfico saliente. Cambia la dirección IP de origen de los paquetes que salen desde la red privada a Internet, usando la IP pública del router para que las respuestas vuelvan correctamente.

Gracias a DNAT, los servicios internos pueden ser accedidos desde el exterior, y gracias a SNAT, los servidores pueden responder adecuadamente a los clientes en Internet.



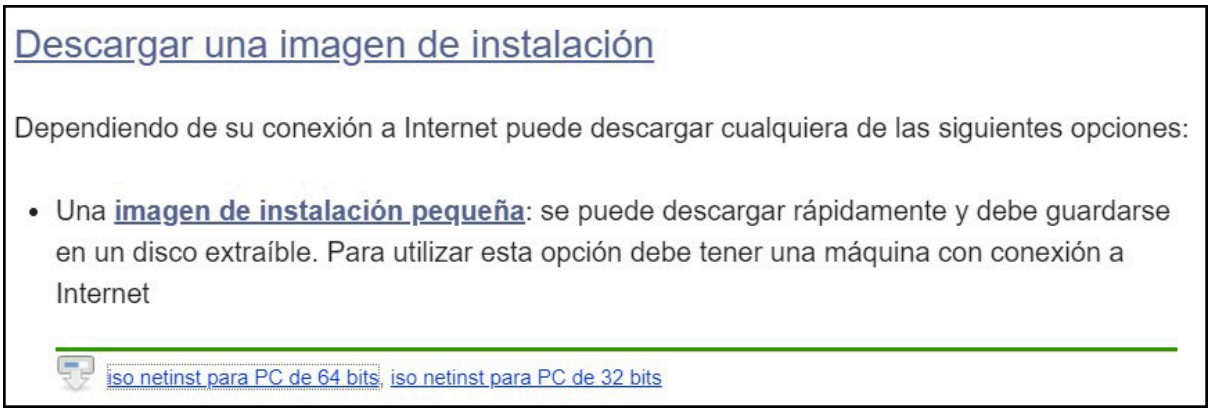
4. Aprovisionamiento del servidor

Lo primero que deberemos realizar para empezar con el proyecto, será la instalación de los recursos mínimos para que el servidor funcione correctamente.

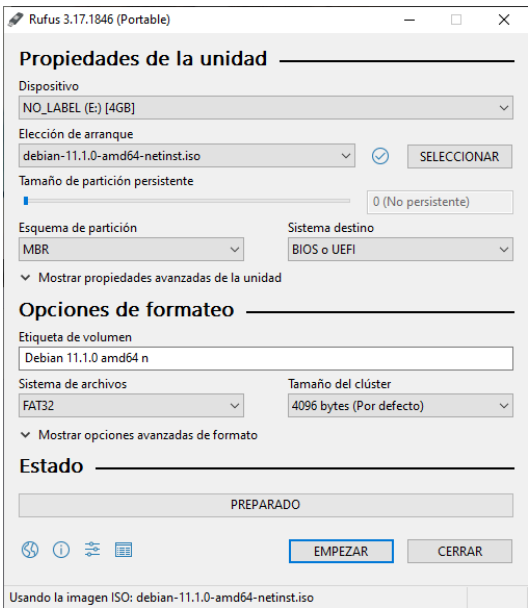
4.1 Sistema Operativo

Descargar y grabar la imagen:

Se va a optar por realizar una instalación con un USB de arranque que contenga la imagen de Debian 11. Lo primero es descargar la imagen y grabarla en la memoria USB usando Rufus. Desde la página para la obtención de Debian 11, seleccionamos la imagen para la arquitectura de nuestra máquina. En mi caso, PC de 64 bits.

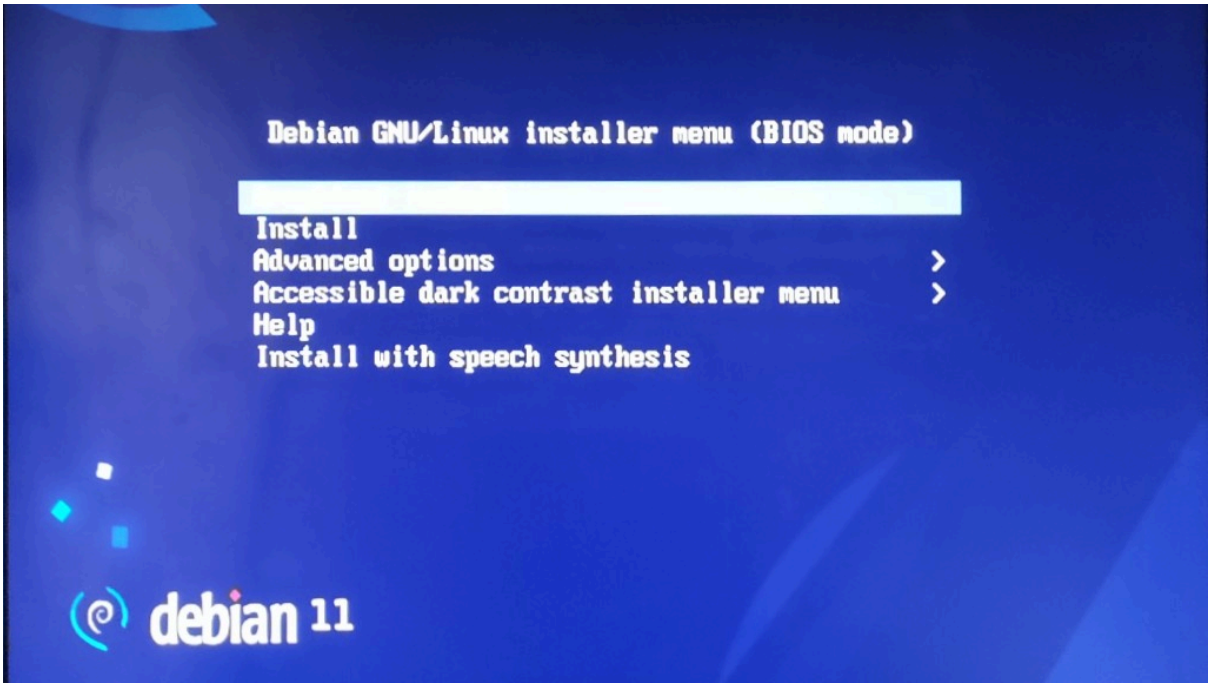


A continuación, descargamos Rufus desde su página oficial. También puede usarse cualquier otro software de creación de USBs de arranque.

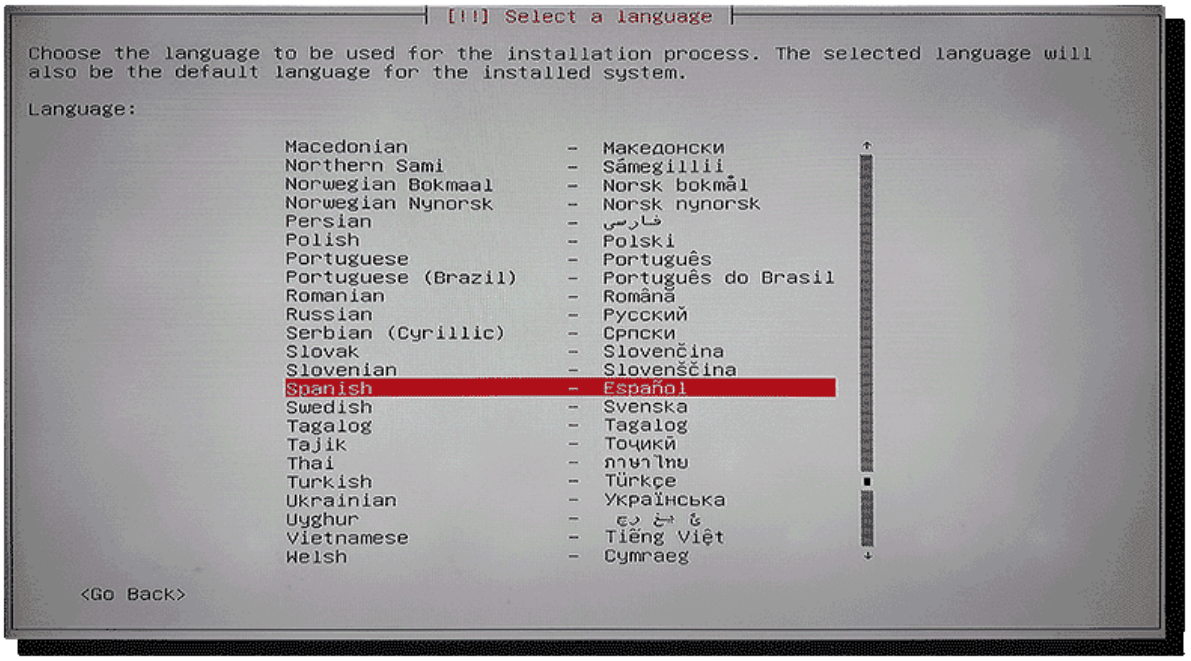


Insertamos la memoria USB en el servidor donde vayamos a instalar Debian 11, y lo encendemos.

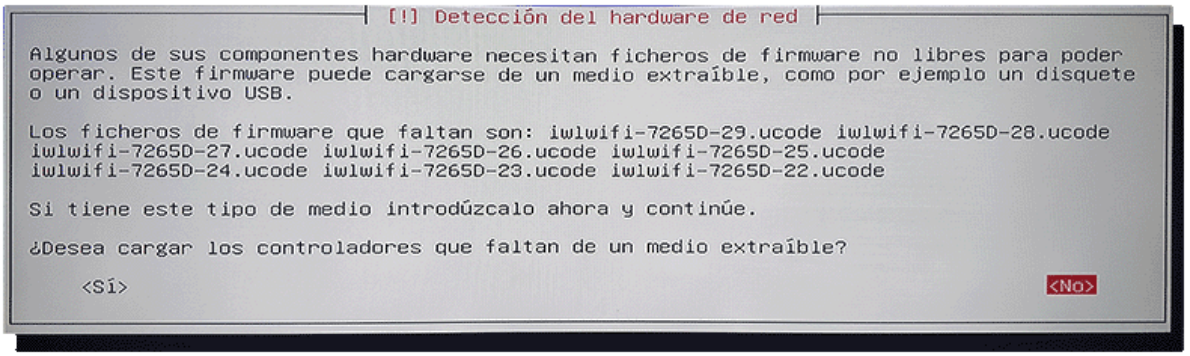
Aparecerá una pantalla con un menú de instalación. Haz clic en Install.



Selecciona el idioma, ubicación y mapa de teclado.

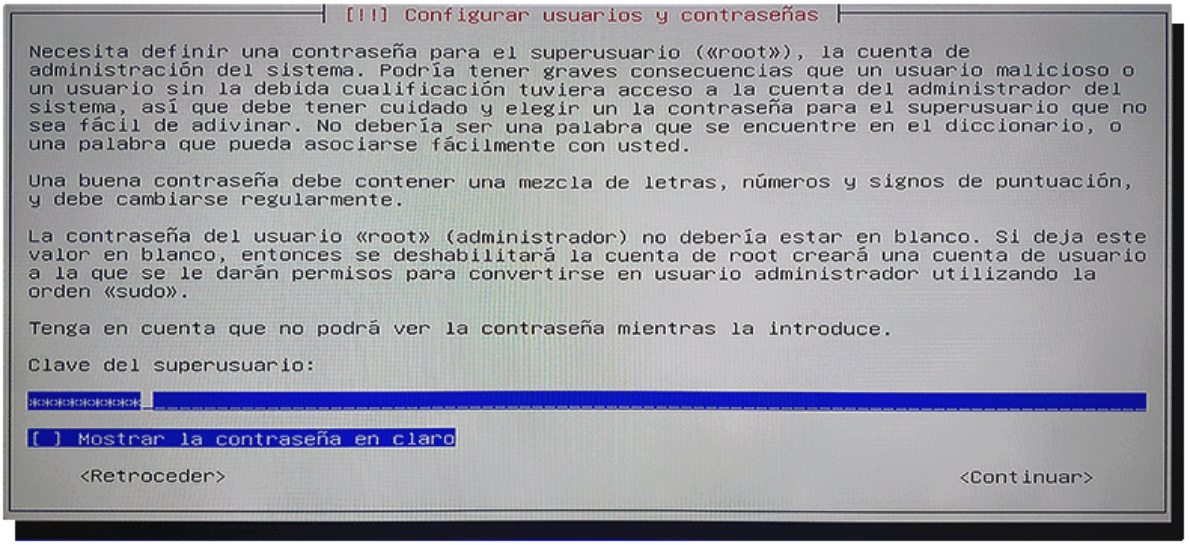


A continuación, se cargará la configuración de red para poder descargar los archivos necesarios durante la instalación. En mi caso, el ordenador dispone de conexión Wi-Fi y cable de red. El primero de ellos usa unos drivers propietarios, por lo que durante la instalación se hace difícil su configuración, opto por conectar el cable de red y configurar el dispositivo después.

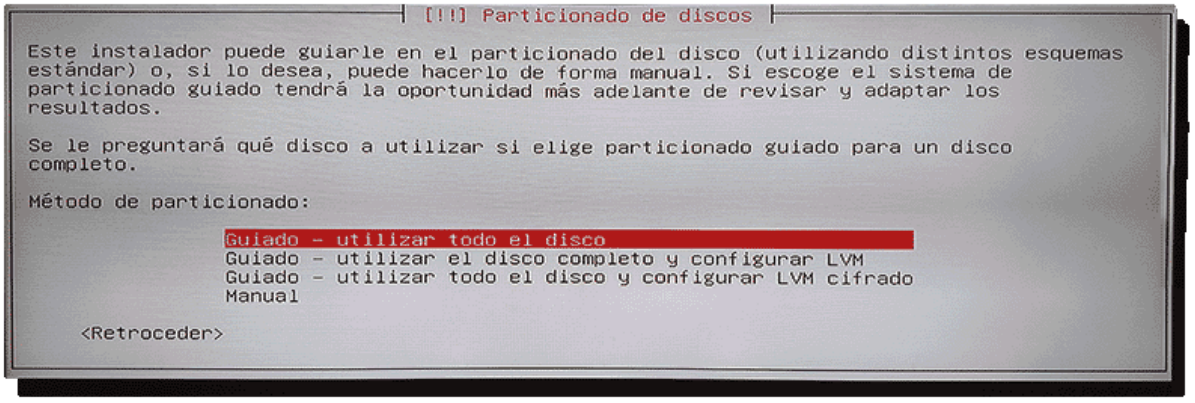


Inserta un nombre para la máquina y un nombre de dominio. Este último suele ser igual para todos los dispositivos de la casa. En mi caso lo dejo vacío al no usar esta opción en mi red.

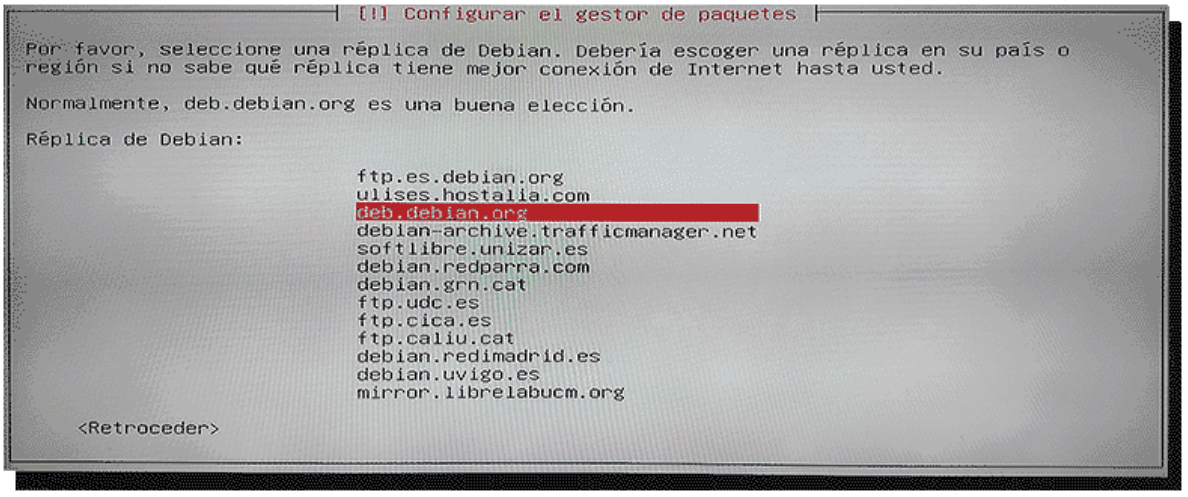
En la configuración de cuentas de usuario, define una contraseña para el usuario root (administrador), y para la cuenta de usuario que se va a usar habitualmente con menores privilegios: nombre completo, nombre de usuario y contraseña.



Como se va a usar la raspberry exclusivamente como servidor y casi todos los servicios van a estar encapsulados en contenedores Docker, se va a configurar ela SD para usar una única partición de disco, seleccionando el disco duro interno (tarjeta SD), e indicar que se van a guardar todos los ficheros en una única partición.

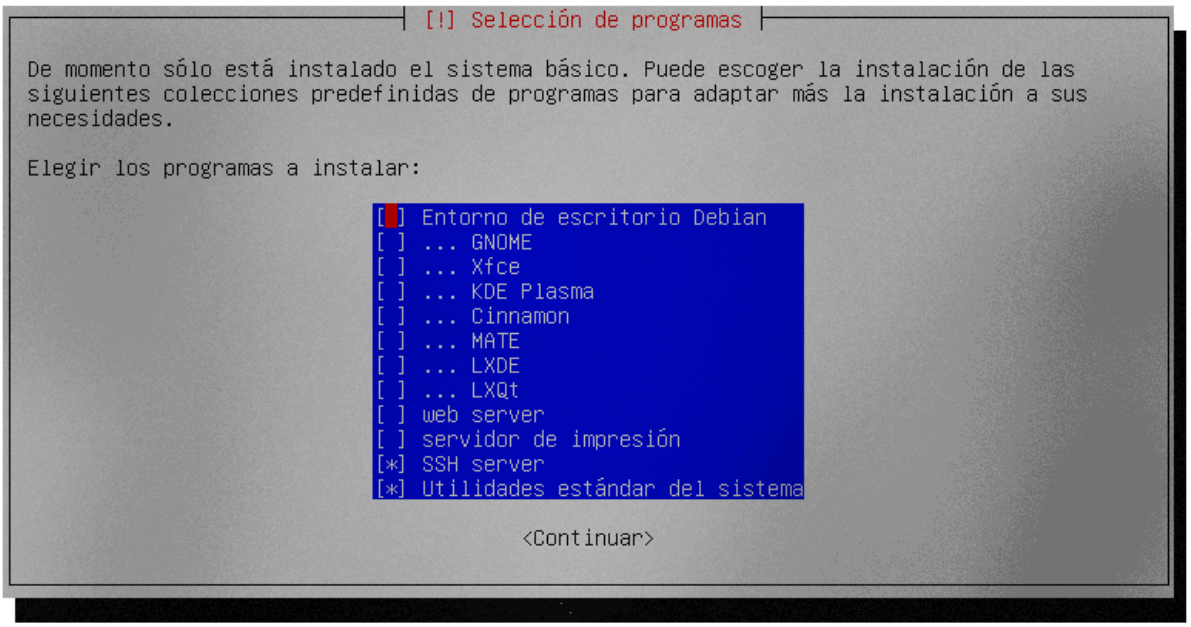


Selecciona el país de réplica de Debian para el gestor de paquetes, y una dirección Web al gusto según nuestro país. Si no conocemos la dirección Web de réplica más cercana, podemos elegir http://deb.debian.org/.

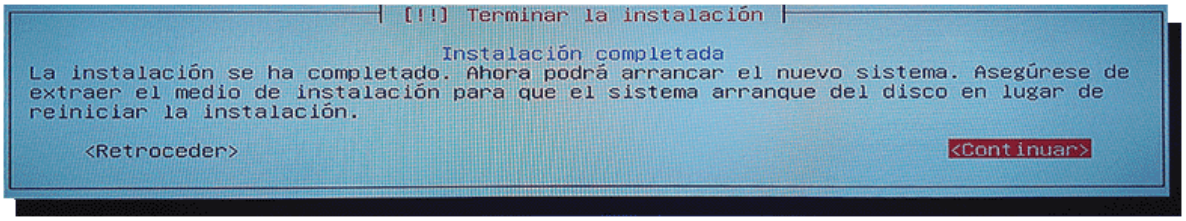


Ya casi hemos acabado. En la siguiente ventana, el instalador nos preguntará si queremos instalar automáticamente actualizaciones de seguridad en nuestro servidor. Aunque el sistema requiere un mantenimiento respecto a diversas actualizaciones, por defecto activa la opción de instalación automática para corregir vulnerabilidades graves sin tener que estar pendiente.

Por último, en la ventana de selección de software, dado que por lo menos en mi caso no quiero entorno de escritorio. Solo selecciono SSH Server y Utilidades del sistema.



Espera unos minutos hasta que se complete la instalación. Quita la memoria USB del ordenador y haz clic en continuar para reiniciar el equipo.

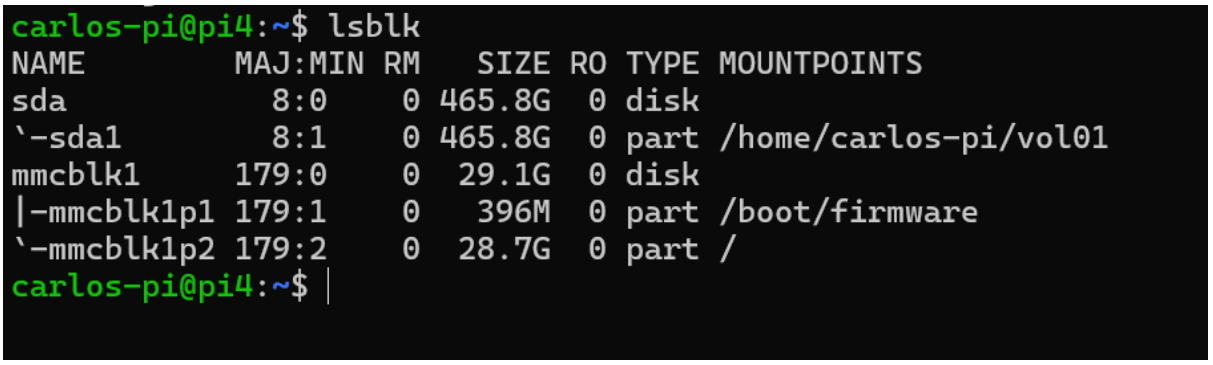


4.2 Montaje de almacenamiento externo

Como mencione en el apartado de escenario, los contenedores y sus respectivos ficheros irían instalados en un disco duro externo de 500 GB, este disco duro debe ser montado e incluido en el fichero fstab, de lo contrario cada vez que se arranque el sistema los contenedores no podrán funcionar al no encontrar sus ficheros.

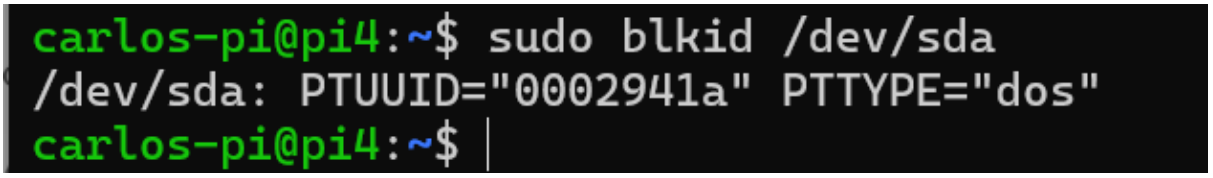
Para realizar esta acción deberemos conectar el disco duro a uno de los USB de nuestra raspberry y escribir los siguientes comandos.

lsblk

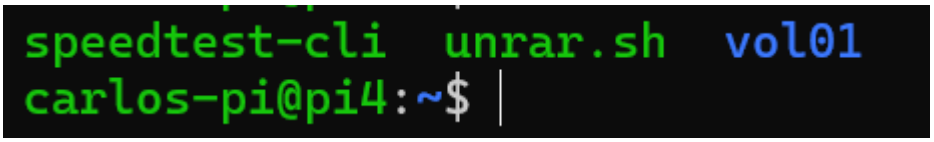


Digamos que tu USB es `/sda/sda1`.

Obtenemos el UUID del dispositivo

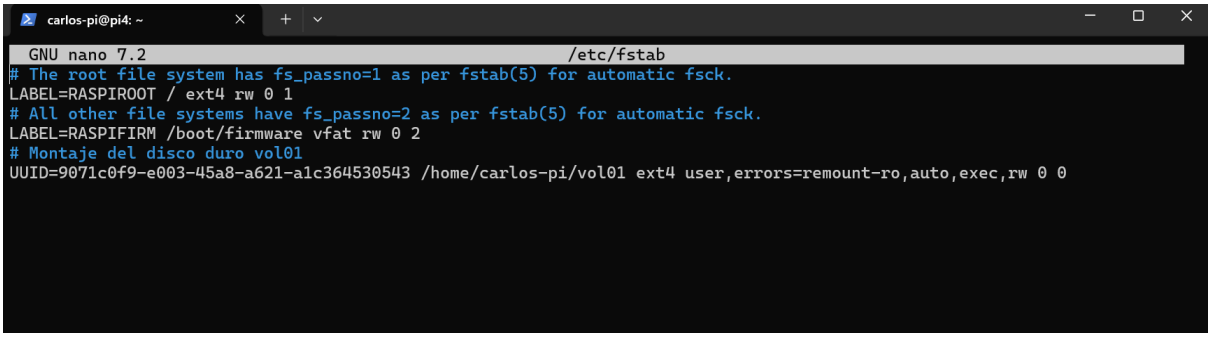


Creamos un punto de montaje, el mio será el directorio Vol01



Una vez hecho esto, ya podremos incluir la línea en el fichero /etc/fstab.

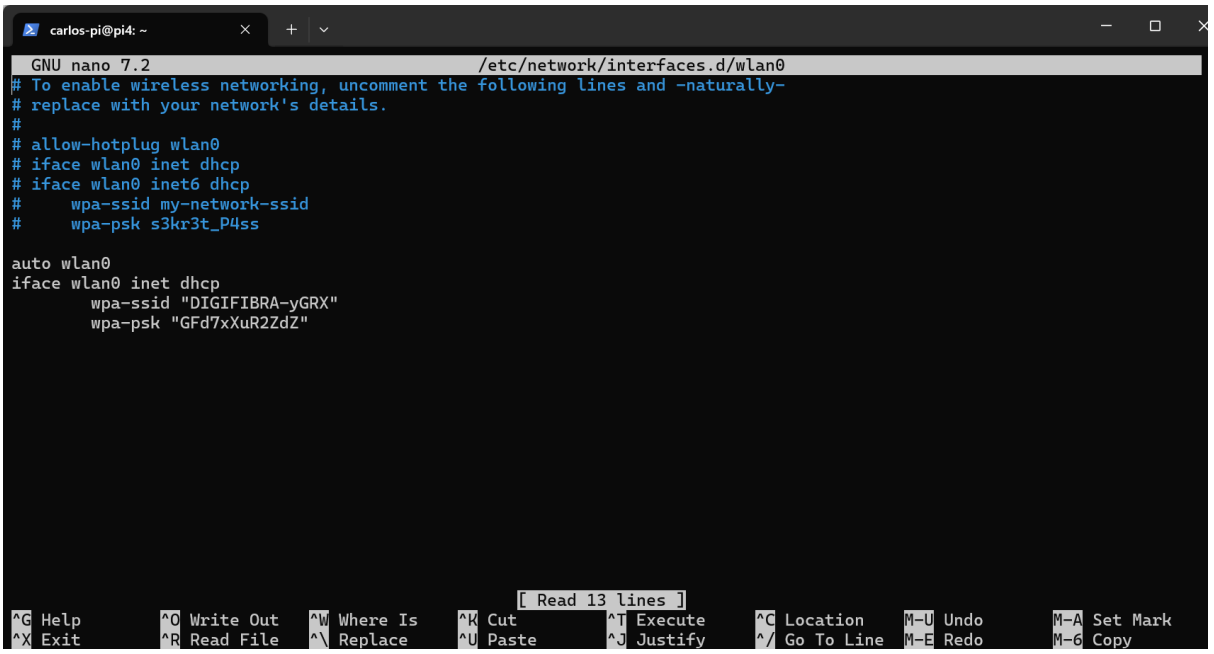
sudo nano /etc/fstab



4.3 Configuración de Red

En sus inicios el servidor estaba conectado a través de un cable de red tirado desde el salón de mi casa, hasta donde se ubica el servidor (mi habitación), por diversos problemas que se dieron posteriormente y que mencionare en la parte final de este proyecto, la conexión se vio forzada a ser a través de protocolo wifi, lo cual no es lo ideal para un servidor que ofrece servicios de video e imagen, ya que se puede ver ralentizado su funcionamiento a la hora de servir dichos archivos.

sudo nano /etc/network/interfaces.d/wlan0



4.4 Configuración protocolo ssh

Como ya sabemos los servidores no cuentan monitores ni interface grafica con la que poder interactuar para realizar las diferentes instalaciones y configuraciones, por lo que se realizará todo mediante conexión ssh.

SSH (Secure Shell) es un protocolo de red seguro que permite conectarte a otro ordenador o servidor de forma remota a través de una terminal.

Se utiliza para:

- Administrar servidores a distancia
- Transferir archivos de forma segura
- Túneles cifrados para redirigir tráfico de red

Por otro lado, no podemos permitir que la conexión se realice mediante contraseña, ya que este servidor estará expuesto a la red. Los servidores expuestos a la red por protocolo ssh reciben cientos de ataques diarios desde todas partes del mundo, ya que hay muchos bots intentando autentificarse en los mismo, para ellos haremos que los inicios de sesión sean con clave pública y privada.

Este método de autenticación es más seguro y automático que usar usuario y contraseña. Se basa en criptografía de clave pública (asimétrica).

Funcionará de la siguiente manera.

Generas un par de claves en tu ordenador:

- Clave privada: se guarda en tu equipo y no se comparte nunca.
- Clave pública: se copia al servidor remoto.

Cuando te conectas al servidor, este:

- Usa la clave pública que tiene almacenada.
- Verifica si tu clave privada local coincide.
- Si coinciden, te deja entrar sin necesidad de contraseña.

Por defecto las claves se guardan en las siguientes rutas.

Por defecto:

- Clave privada: `~/.ssh/id_rsa`
- Clave pública: `~/.ssh/id_rsa.pub`

Y en el servidor, la clave pública se guarda en:

- `~/.ssh/authorized_keys` (del usuario remoto)

Para poder utilizar este método, tendremos que realizar los siguientes pasos desde nuestro portátil desde el que realizaremos las conexiones.

1. Generar claves en tu máquina local

   ssh-keygen -t rsa -b 4096

Presiona Enter para aceptar la ruta por defecto. Puedes añadir una frase de seguridad si quieres.

2. Copiar la clave pública al servidor

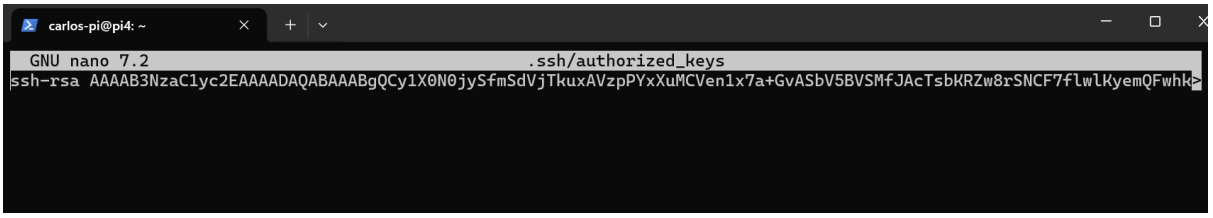
   ssh-copy-id usuario@ip_servidor

Esto añade tu clave a `~/.ssh/authorized_keys` del servidor.

3. Conectarte sin contraseña

   ssh usuario@ip_servidor

El servidor comprobará tu clave privada y, si es válida, te dejará entrar sin pedir contraseña.



5. Servicios y instalaciones

5.1 Docker

Como he mencionado todos los servicios que este servidor ofrecerá, estarán ejecutados bajo docker, para ello lo primero que deberemos hacer es instalar los paquete necesarios.

Pero antes, haremos una breve explicación de que es docker y para que se utiliza.

Docker es una plataforma que permite crear, ejecutar y gestionar contenedores, que son entornos ligeros, aislados y portables donde puedes ejecutar aplicaciones.

Piensa en un contenedor como una "caja" que lleva dentro todo lo necesario para que una aplicación funcione: código, librerías, sistema base, configuración, etc., sin importar dónde se ejecute.

Docker se usa principalmente para:

- Aislar servicios: cada contenedor puede correr un servicio (como un servidor web, base de datos, etc.).
- Facilitar despliegues: puedes mover contenedores fácilmente entre servidores.
- Evitar conflictos de dependencias: cada contenedor tiene su propio entorno.
- Automatizar entornos: útil para desarrollo, pruebas, producción.

Docker es ideal para montar un servidor con múltiples servicios expuestos en red, como:

- Un servidor web con Nginx o Apache
- Una base de datos como PostgreSQL o MySQL
- Aplicaciones como Nextcloud, WordPress, o GitLab
- Un proxy inverso que gestione certificados HTTPS automáticamente (con Traefik o Nginx)
- Redes privadas virtuales con WireGuard o OpenVPN

Cada uno va en su propio contenedor y se comunican entre sí mediante una red Docker interna. Puedes exponer puertos al exterior para que los servicios estén disponibles desde Internet.

Para facilitarnos la tarea utilizaremos Docker-compose.

Docker Compose es una herramienta que permite definir y gestionar múltiples contenedores Docker desde un solo archivo de configuración: docker-compose.yml.

En lugar de arrancar contenedores uno por uno con comandos docker run, puedes describir todo un entorno o "stack" de servicios en un solo archivo, y levantarlo con un simple comando:

```
docker-compose up -d
```

Docker Compose es una capa sobre Docker que usa el propio motor de Docker para ejecutar los contenedores. Todo lo que hace Compose se traduce en comandos docker normales, pero de forma automática, coordinada y repetible.

Al definir todos estos ficheros podremos realizar lo siguiente:

- Definir múltiples servicios (por ejemplo, web, base de datos, proxy).
- Crear redes y volúmenes persistentes.
- Configurar variables de entorno.
- Orquestar dependencias (ej: esperar a que la base de datos arranque antes de iniciar una app).

Instalación:

Docker

- Actualiza el sistema:

  ```
  sudo apt update

  sudo apt upgrade -y
  ```

- Instala paquetes necesarios:

  ```
  sudo apt install \
      ca-certificates \
      curl \
      gnupg \
      lsb-release -y
  ```

- Añade la clave GPG oficial de Docker:

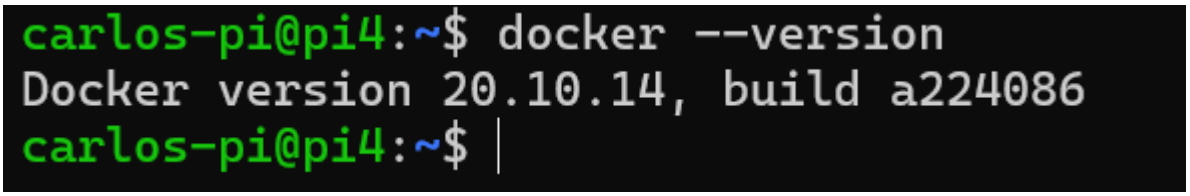
  ```
  sudo mkdir -p /etc/apt/keyrings
  curl -fsSL https://download.docker.com/linux/debian/gpg | \
  sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
  ```

- Añade el repositorio oficial de Docker:

  ```
  echo \
    "deb [arch=$(dpkg --print-architecture) \
    signed-by=/etc/apt/keyrings/docker.gpg] \
    https://download.docker.com/linux/debian \
    $(lsb_release -cs) stable" | \
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
  ```

- Instala Docker:

  ```
  sudo apt update
  sudo apt install docker-ce docker-ce-cli containerd.io -y
  ```



Docker-compose

- Instalar la versión oficial de Docker Compose:

  ```
  DOCKER_CONFIG=${DOCKER_CONFIG:-$HOME/.docker}
  mkdir -p $DOCKER_CONFIG/cli-plugins
  ```

```
curl -SL
https://github.com/docker/compose/releases/download/v2.24.2/docker-compose-linux
-$(uname -m) \

    -o $DOCKER_CONFIG/cli-plugins/docker-compose


chmod +x $DOCKER_CONFIG/cli-plugins/docker-compose
```

```
carlos-pi@pi4:~$docker-compose --version
docker-compose version 1.29.2, build unknown
carlos-pi@pi4:~$
```

5.2 Servicios a instalar

Una vez teníamos la plataforma sobre la que instalar los diferentes servicios que aloja el servidor, solo faltaria saber a que enfocarlo. Existen muchos tipos de servidores: video, datos, correo, etc. Para nuestro caso particular y al tratarse de un servidor casero tenía que cubrir las necesidades del usuario el cual lo iba a utilizar, en este caso sería yo, por lo que los servicios son una recopilación de aquellos servicio que utilizo o alguna vez he necesitado, para este proyecto nos centraremos en los más curiosos o importantes.

Cada servicio alojado dentro del disco duro, cuenta con su propio directorio y su propio contenedor. Este es el esquema general que sigue, donde cada uno tiene su propio fichero docker-compose desde el cual se genera el escenario.

```
carlos-pi@pi4:~$ tree -L 2 vol01/docker/
vol01/docker/
|-- bitwarden
|   |-- bitwardenrs
|   `-- docker-compose.yaml
|-- duckdns
|   |-- appdata
|   `-- docker-compose.yml
|-- duplicati
|   |-- backups
|   |-- config
|   `-- docker-compose.yml
|-- nextcloud
|   |-- datos
|   |-- db
|   `-- docker-compose.yml
|-- npm
|   |-- data
|   |-- data-db
|   |-- docker-compose.yml
|   `-- letsencrypt
|-- pepermerge
|   |-- config
|   |-- data
|   `-- docker-compose.yml
|-- pi-hole
|   |-- docker-compose.yml
|   |-- etc-dnsmasq.d
|   `-- etc-pihole
|-- plex
|   |-- config
|   `-- docker-compose.yml
|-- prestashop
|   |-- docker-compose.yml
|   |-- prestashop-db
|   `-- prestashop-php
|-- samba
|   `-- docker-compose.yml
|-- transmission
|   |-- config
|   |-- docker-compose.yml
|   `-- watch
|-- webdav
|   `-- docker-compose.yaml
```

Lo primero antes de seguir, es explicar cómo se crean contenedores con docker-compose, el uso de esta herramientas se puede simplificar de esta manera

1. Crear un archivo docker-compose.yml

   Este archivo define los servicios (contenedores), sus imágenes, puertos, volúmenes, redes, etc.

   Ejemplo básico con un servidor web Nginx:

```
version: '3'

services:

  web:

    image: nginx:latest

    ports:

      - "80:80"
```

   Como veremos más adelante en los contendores de mi servidor, es habitual que haya muchos mas parametros especificados para un correcto funcionamiento del mismo y un correcto orden de los archivos que se generan.

2. Levantar los contenedores

   En la misma carpeta donde está el archivo docker-compose.yml, ejecuta:

   docker compose up -d

   - up: inicia los contenedores definidos
   - -d: modo detached, se ejecutan en segundo plano

3. Ver contenedores en ejecución
   docker compose ps

4. Detener los contenedores

   docker compose down

   Esto detiene y elimina los contenedores (pero no las imágenes o volúmenes, salvo que se especifique).

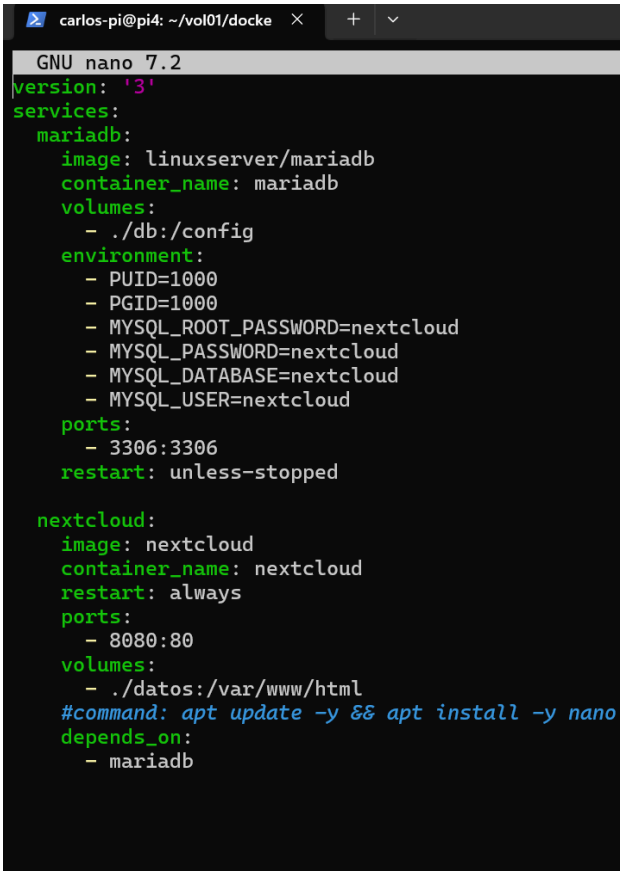
*Una imagen de Docker es una plantilla que contiene todo lo necesario para ejecutar una aplicación: sistema base, código, dependencias y configuraciones. A partir de ella se crean contenedores, que son instancias en ejecución. Las imágenes se pueden descargar, personalizar o construir con un Dockerfile, y permiten desplegar servicios de forma rápida, aislada y portable.

Una vez explicado el funcionamiento básico, veremos los servicios instalados en el servidor.

5.2.1 Nextcloud

Nextcloud es una plataforma de software libre para crear tu propia nube privada. Permite almacenar, sincronizar y compartir archivos, calendarios, contactos y más, de forma segura y controlada, alojándola en tu propio servidor o en uno de confianza. Es una alternativa a servicios como Google Drive o Dropbox, pero con control total sobre tus datos.

Fichero de configuración



En este fichero se especifica la creación de dos servicios services:.

Servicio mariadb (base de datos):

- image: linuxserver/mariadb
  La imagen oficial de MariaDB mantenida por LinuxServer, que es un sistema de gestión de bases de datos.

- container_name: mariadb
  Nombre asignado al contenedor para identificarlo fácilmente.

- volumes:
  ./db:/config

Mapea el directorio local ./db al directorio /config dentro del contenedor. Esto permite persistir los datos de la base de datos fuera del contenedor para que no se pierdan al detenerlo o eliminarlo.

- environment:
  Variables de entorno para configurar la base de datos:
    - PUID=1000 y PGID=1000: IDs de usuario y grupo para asignar permisos dentro del contenedor.
    - MYSQL_ROOT_PASSWORD=nextcloud: contraseña para el usuario root de la base de datos.
    - MYSQL_PASSWORD=nextcloud: contraseña para el usuario nextcloud.
    - MYSQL_DATABASE=nextcloud: nombre de la base de datos que se crea para Nextcloud.
    - MYSQL_USER=nextcloud: nombre del usuario que tendrá acceso a la base de datos.

- ports:
  3306:3306
  Expone el puerto 3306 del contenedor (puerto por defecto de MariaDB) al puerto 3306 del host.

- restart: unless-stopped
  Configura el contenedor para que se reinicie automáticamente a menos que se detenga manualmente.

Servicio nextcloud (aplicación Nextcloud):

- image: nextcloud
  Imagen oficial de Nextcloud, que es el software de nube privada.

- container_name: nextcloud
  Nombre asignado al contenedor para identificarlo fácilmente.

- restart: always
  Siempre reinicia el contenedor si se detiene, para mantener el servicio activo.

- ports:
  8080:80
  Expone el puerto 80 del contenedor (puerto web estándar HTTP) en el puerto 8080 del host. Esto permite acceder a Nextcloud desde el navegador en

http://localhost:8080 (o IP del servidor).

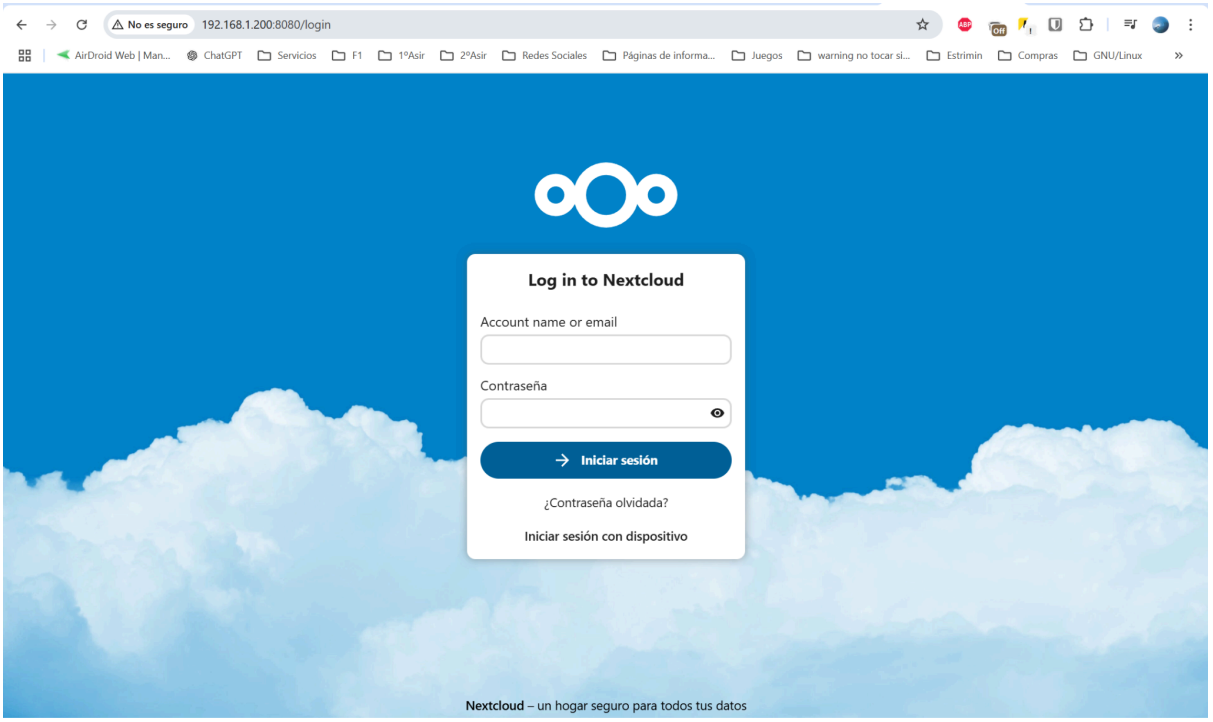
- volumes:
  ./datos:/var/www/html
  Mapea el directorio local ./datos al directorio /var/www/html dentro del contenedor, donde Nextcloud almacena sus archivos y configuración web. Esto permite persistir datos.

- depends_on:
  Indica que el servicio nextcloud depende del servicio mariadb. Docker Compose se asegurará de que mariadb arranque antes que nextcloud

Una vez creado el fichero, solo tendremos que realizar el comando mencionado anteriormente para ejecutarlos.

docker-compose up d

Ahora podremos acceder al servicio de forma local poniendo en el buscador del navegador ip del servidor : puerto expuesto de la máquina anfitriona.



Una vez desplegado el contenedor y accedes por navegador (por ejemplo, http://localhost:8080), Nextcloud inicia un asistente de configuración web donde debes completar:

1. Crear cuenta de administrador
   Introduces el nombre de usuario y contraseña del administrador que gestionará Nextcloud.

2. Ruta de datos
   Especificas la ruta donde se almacenarán los archivos de los usuarios (por defecto suele ser /var/www/html/data si no se ha montado otra).

3. Conexión con la base de datos

    - Tipo: MySQL/MariaDB

    - Usuario: (ej. nextcloud)

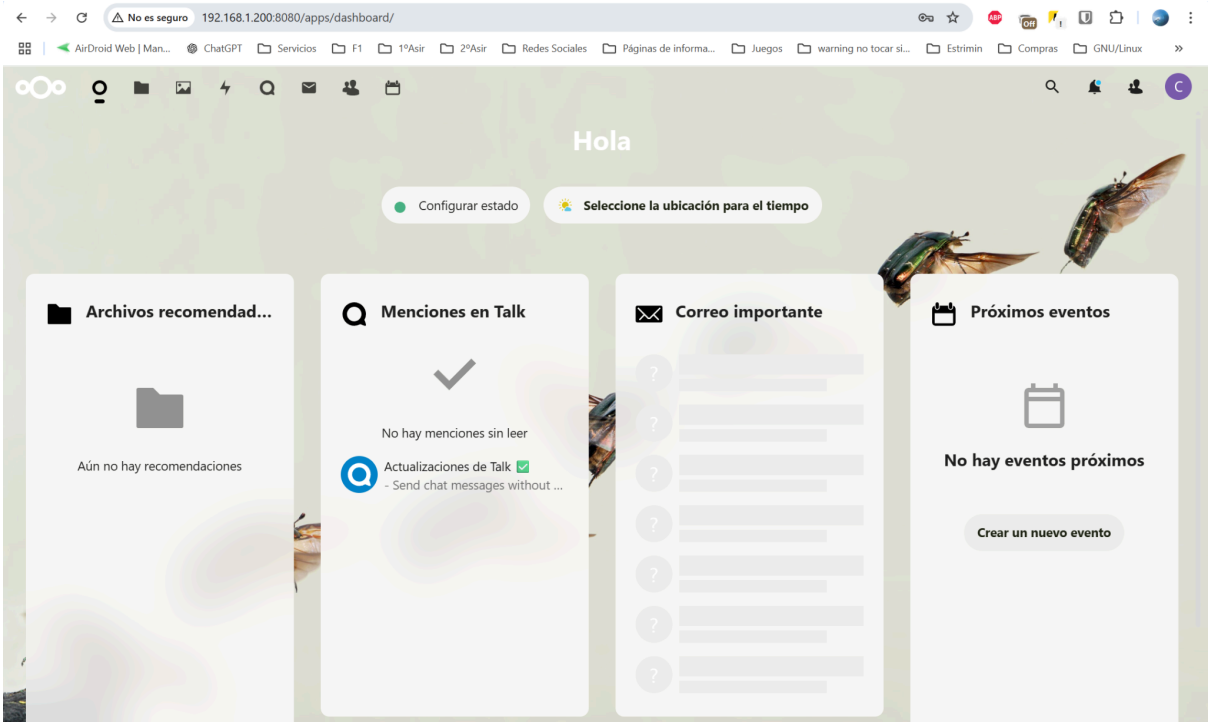
    - Contraseña: (ej. nextcloud)

    - Nombre de la base de datos: (ej. nextcloud)

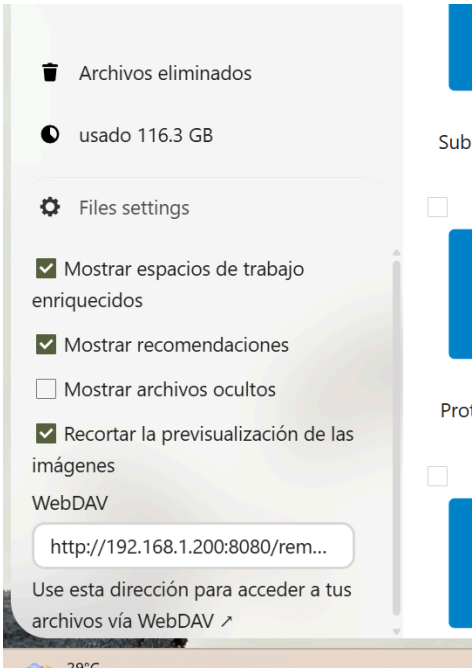
    - Servidor de la base de datos: (ej. mariadb si se llama así el contenedor en Docker Compose)

4. Instalación opcional de aplicaciones recomendadas
   Puedes marcar para que se instalen apps como calendario, contactos o correo.

Tras completar estos pasos, Nextcloud se instala y podrás acceder al panel principal para gestionar usuarios, sincronizar archivos, instalar más apps, configurar notificaciones, cifrado, etc.



Como datos curiosos cabe destacar que Nextcloud implementa la opción de poder añadir tu directorio de archivos en la nube, a tu directorio de archivos locales de tu ordenador personal a través de WebDav.



WebDAV (Web Distributed Authoring and Versioning) es una extensión del protocolo HTTP que permite gestionar archivos de forma remota a través de la web. Con WebDAV, puedes subir, descargar, mover, copiar o eliminar archivos en un servidor como si fuera una carpeta local en tu sistema.

5.2.2 Duplicati

El contenedor de Duplicati ejecuta una aplicación de copias de seguridad cifradas y automáticas, diseñada para hacer backups seguros, incrementales y programados de archivos y carpetas hacia distintos destinos, como:

- Discos locales o unidades externas
- Servidores remotos vía FTP, SFTP, WebDAV
- Nubes como Google Drive, Dropbox, OneDrive, Amazon S3, etc.

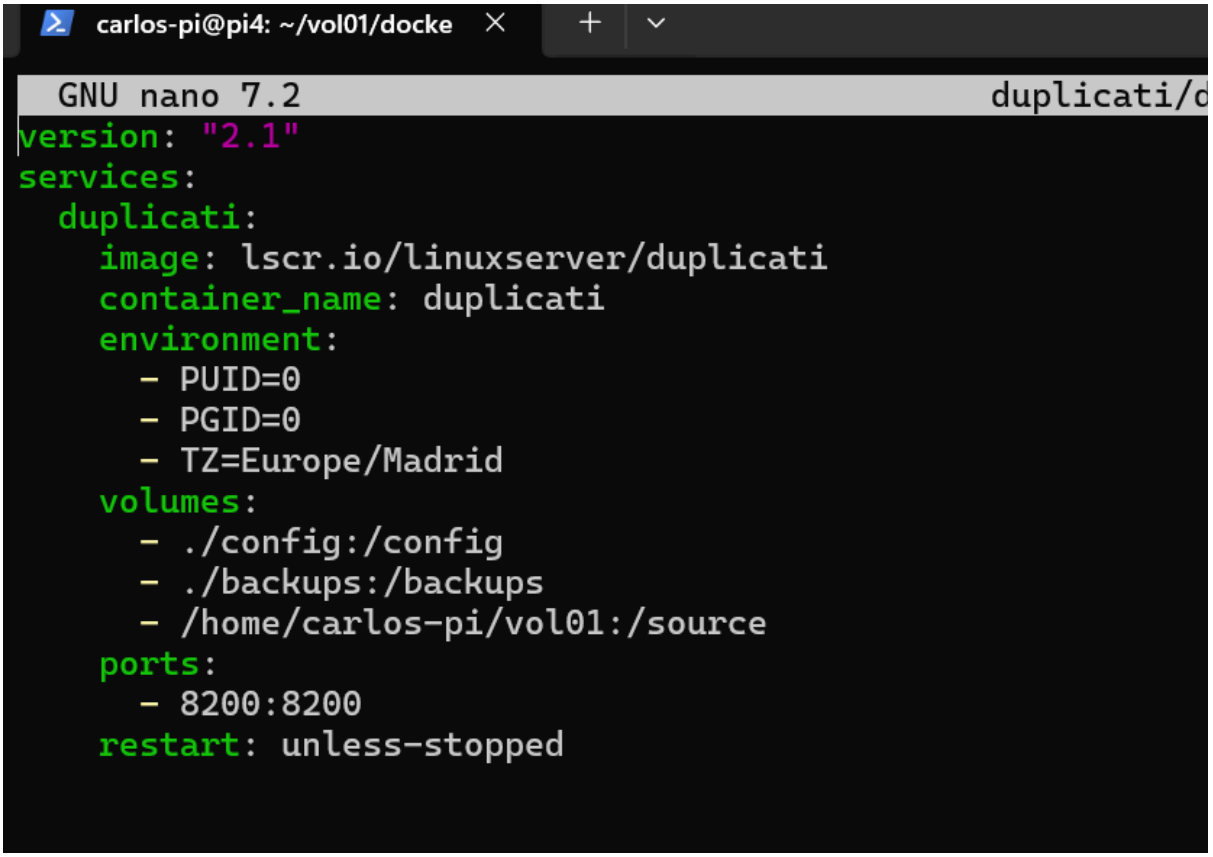
En un entorno Docker, Duplicati se ejecuta en un contenedor con interfaz web (normalmente en el puerto 8200), desde la cual puedes:

- Configurar trabajos de copia de seguridad
- Elegir qué carpetas incluir o excluir
- Establecer la frecuencia y destino del backup
- Cifrar los datos con contraseña
- Restaurar archivos fácilmente cuando lo necesites

Es especialmente útil para hacer backups automáticos de volúmenes de otros contenedores, como por ejemplo los datos de Nextcloud.

Este contenedor está configurado para que realice las copias de seguridad a un servidor local ftp de su misma red, pero podría realizarse a un dispositivo fuera de su red para una mayor seguridad.

Fichero de configuración.



- version: "2.1": Indica el formato del archivo Docker Compose utilizado.
- services:: Define los servicios que se van a ejecutar. En este caso, solo hay uno: duplicati.
- image: lscr.io/linuxserver/duplicati: Es la imagen Docker que contiene Duplicati ya preparado para su uso.
- container_name: duplicati: Nombre del contenedor, útil para identificarlo fácilmente.
- environment:
    - PUID=0 y PGID=0: Definen el usuario y grupo con los que se ejecuta el contenedor (en este caso, root).
    - TZ=Europe/Madrid: Establece la zona horaria del contenedor.
- volumes:
    - ./config:/config: Guarda la configuración de Duplicati.

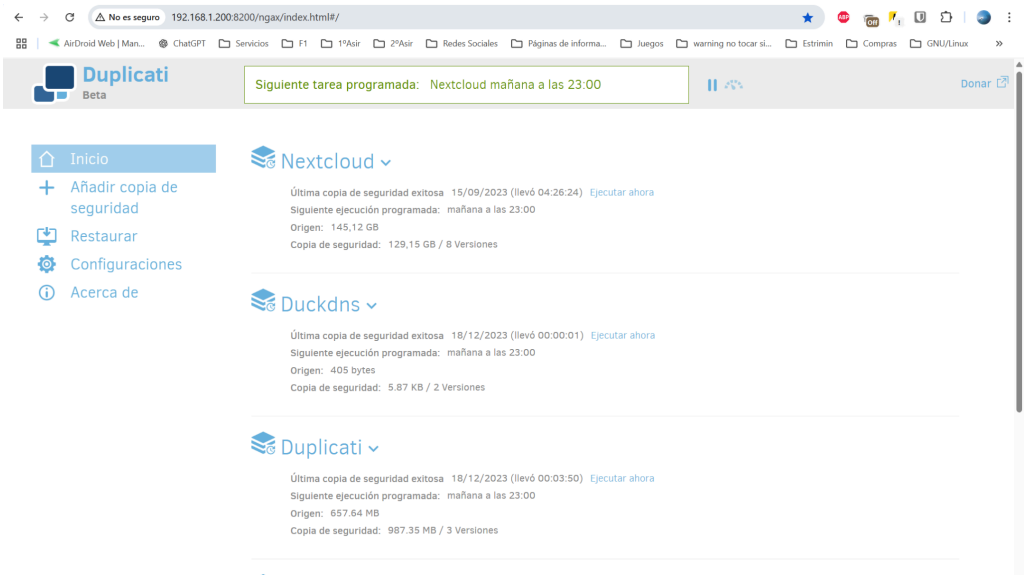
- ./backups:/backups: Carpeta donde se guardan las copias de seguridad generadas.

- /home/carlos-pi/vol01:/source: Carpeta local que se va a respaldar, accesible dentro del contenedor como /source.

- ports: 8200:8200: Expone la interfaz web en el puerto 8200 del host.

- restart: unless-stopped: El contenedor se reinicia automáticamente, excepto si se detuvo manualmente.

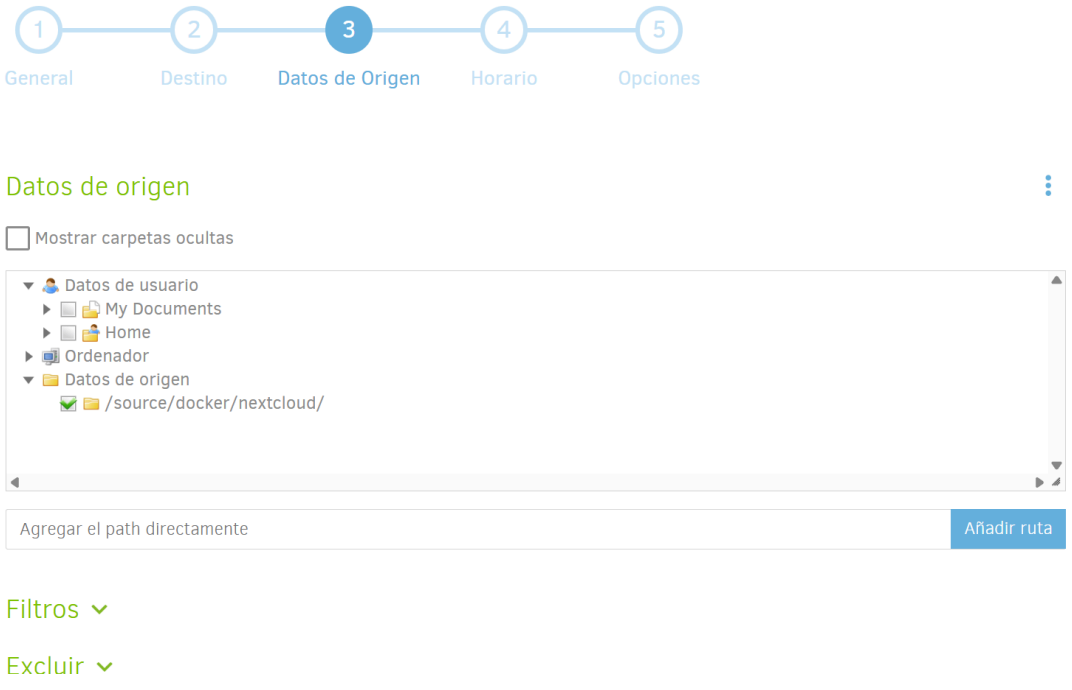
Una vez desplegado con docker compose up -d, se accede a la interfaz desde http://localhost:8200 (o desde la IP del servidor), donde se pueden configurar los trabajos de backup, definir los archivos a guardar y seleccionar el destino (disco local, nube, FTP, etc.).



En esta imagen podemos apreciar que de manera predeterminada las copias de seguridad se realizan a las 23:00 diariamente, así como las últimas copias realizadas.

Si nos vamos a la configuración de cada copia podremos ver los datos necesarios para la realización de la misma en ellos podremos especificar la configuración del servidor donde se realizan .

En origen podremos ver los ficheros de los cuales se realizará la copia:



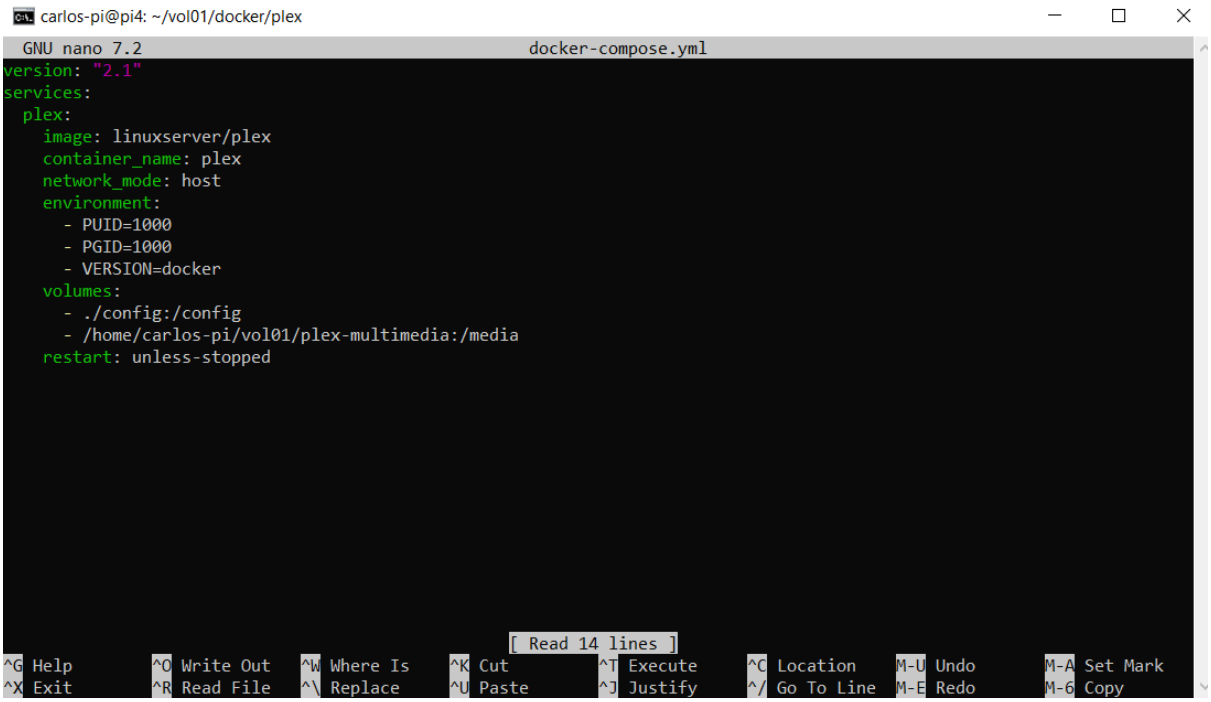
5.2.3 Plex

Un servidor Plex es una solución para centralizar, organizar y reproducir archivos multimedia personales (películas, series, música, fotos) desde cualquier lugar y en múltiples dispositivos. Actúa como tu propio "Netflix casero", permitiéndote hacer streaming de tus contenidos locales, tanto dentro como fuera de casa.

Características principales:

- Escanea tus carpetas multimedia y organiza automáticamente el contenido con metadatos, carátulas, descripciones y subtítulos.
- Permite acceder al contenido desde móviles, tablets, Smart TVs, consolas o navegadores web.
- Soporta múltiples usuarios y permite compartir tu biblioteca con familiares o amigos.
- Transcodifica el contenido en tiempo real si es necesario, para adaptarse al dispositivo que está reproduciendo.

Fichero de configuración:



- plex:
  Nombre del servicio. Puedes cambiarlo, pero lo lógico es usar “plex”.

- image: linuxserver/plex
  Imagen de Docker utilizada para el contenedor. Es una imagen oficial mantenida por LinuxServer.io, optimizada para Plex.

- container_name: plex
  Nombre personalizado del contenedor. Permite gestionarlo fácilmente con comandos como docker start plex.

- network_mode: host
  El contenedor comparte la red del host. Es decir, Plex se comporta como si estuviera instalado directamente en el sistema. Esto es necesario para que pueda detectar otros dispositivos locales como Smart TVs, Chromecast, etc.

- environment:
  Define variables de entorno necesarias para la configuración:

  - PUID=1000: ID del usuario en el host. Se usa para que los archivos creados/modificados tengan los permisos correctos.

  - PGID=1000: ID del grupo del usuario.

  - VERSION=docker: Indica a la imagen que se está ejecutando bajo Docker.

- volumes:
  Define carpetas compartidas entre el host y el contenedor:

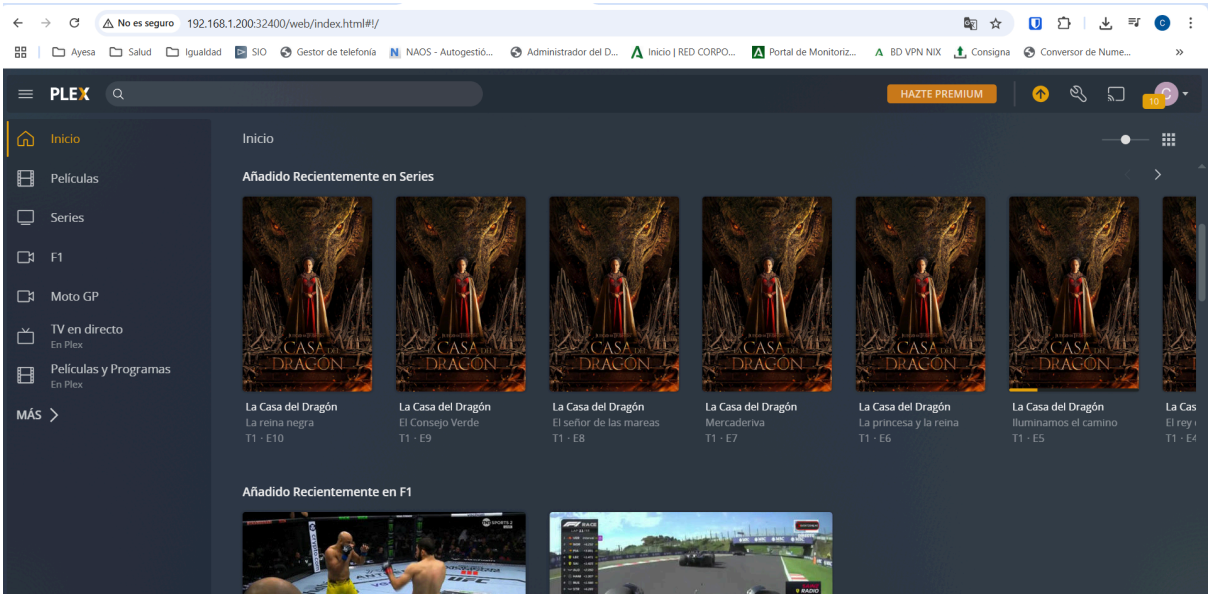
  - ./config:/config: Carpeta local donde se almacenarán la configuración y la base de datos de Plex.

  - /home/carlos-pi/vol01/plex-multimedia:/media: Carpeta del host que contiene los archivos multimedia que Plex va a indexar y servir.

- restart: unless-stopped
  Política de reinicio automático. El contenedor se reiniciará siempre, excepto si fue detenido manualmente.

Dado que se está usando network_mode: host, el contenedor no necesita mapeo de puertos. Puedes acceder al servidor Plex desde tu navegador con la siguiente dirección:

http://localhost:32400/web



Una vez se está ejecutando el docker, tendremos que realizar las configuraciones básicas del servicio:

- Iniciar sesión o crear cuenta

Plex te pedirá que inicies sesión con tu cuenta. Si no tienes una, puedes crearla gratuitamente en:

https://plex.tv

- Asignar un nombre al servidor

Introduce un nombre personalizado para tu servidor (ejemplo: MiServidorPlex o Plex-Raspberry).
Marca la opción “Permitir el acceso a mi servidor desde fuera de mi red” solo si planeas usar Plex desde fuera de casa (como es nuestro caso).

- Añadir bibliotecas multimedia

Ahora debes añadir las carpetas que contienen tus archivos de vídeo, música o fotos. En tu archivo docker-compose.yml, la ruta /media apunta a esta carpeta local:

/home/carlos-pi/vol01/plex-multimedia

Pasos para añadir una biblioteca:

- Haz clic en “Agregar biblioteca”.
- Selecciona el tipo de contenido: Películas, Series, Música, etc.

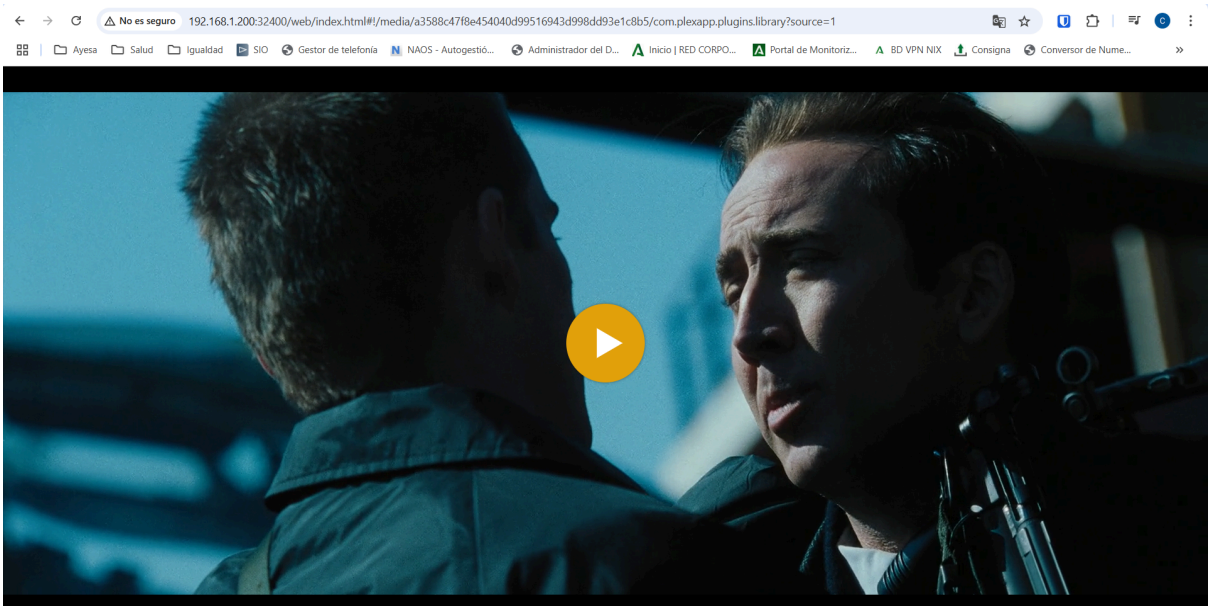
- Plex mostrará el sistema de archivos interno. Navega hasta /media y selecciona la subcarpeta que corresponda.

- Repite este paso para cada tipo de contenido que quieras añadir.

- Finalizar configuración

Una vez añadidas las bibliotecas, Plex comenzará a escanear los archivos y descargará metadatos automáticamente (carátulas, sinopsis, información de episodios, etc.).

Este proceso puede tardar unos minutos según la cantidad de archivos que tengas.



5.2.4 Transmission

El Docker de Transmission ejecuta una instancia del cliente Transmission, que es un programa de descarga de archivos vía BitTorrent. Al usarlo en un contenedor Docker, puedes tenerlo aislado, fácilmente configurable y accesible desde una interfaz web.

Brevemente:

- Transmission es un cliente BitTorrent ligero.
- El Docker de Transmission te permite:
    - Descargar torrents automáticamente.
    - Acceder a la interfaz web para gestionarlos.
    - Configurar carpetas de descarga, velocidad, puertos, etc.
    - Mantenerlo corriendo en segundo plano.

Es útil si quieres automatizar y controlar descargas sin ocupar recursos de tu sistema principal. Por esto, es el mejor acompañante para el docker de plex, ya que al configurar el volumen de descarga en la misma ruta donde plex accede a los archivos que se pueden reproducir, podremos descargar directamente cualquier torrent que queramos visualizar y se servirá una vez descargado sin necesidad de hacer nada en nuestro plex, disponible inmediatamente para todo el mundo que queramos.

Fichero de configuración:

carlos-pi@pi4: ~/vol01/docker/plex

```
 GNU nano 7.2                                  ../transmission/docker-compose.yml
version: "2.1"
services:
  transmission:
    image: lscr.io/linuxserver/transmission
    container_name: transmission
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Madrid
    volumes:
      - ./config:/config
      - /home/carlos-pi/vol01/plex-multimedia:/downloads
      - ./watch/folder:/watch
    ports:
      - 9091:9091
      - 51413:51413
      - 51413:51413/udp
    restart: unless-stopped
```

- transmission:
  Nombre del servicio.

- image: lscr.io/linuxserver/transmission
  Imagen del contenedor de Transmission mantenida por LinuxServer.io.

- container_name: transmission
  Nombre fijo para el contenedor, en lugar de uno aleatorio.

- environment:
  Variables de entorno que configuran el contenedor:

    - PUID=1000: ID del usuario del host para los permisos.

    - PGID=1000: ID del grupo del host.

    - TZ=Europe/Madrid: Zona horaria del contenedor.

- volumes:
  Directorios compartidos entre el host y el contenedor:

    - ./config:/config: Carpeta local donde se guardará la configuración de Transmission.

    - /home/carlos-pi/vol01/plex-multimedia:/downloads: Carpeta donde se guardarán las descargas.

    - ./watch/folder:/watch: Carpeta para monitorear archivos .torrent y añadirlos automáticamente.

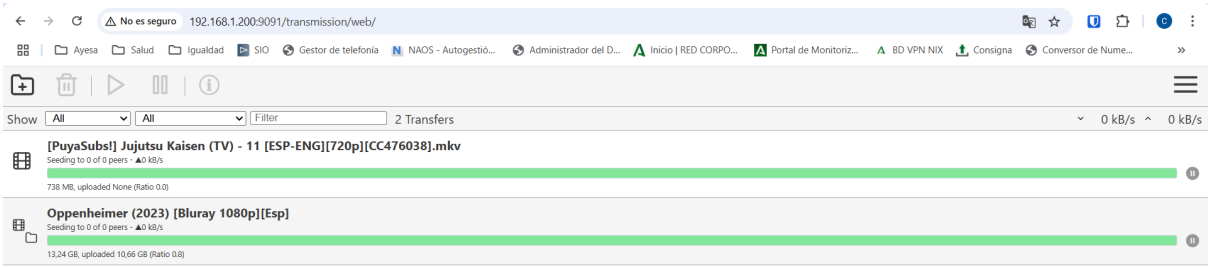
- ports:
  Puertos expuestos del contenedor al host:

    - 9091:9091: Acceso a la interfaz web de Transmission.

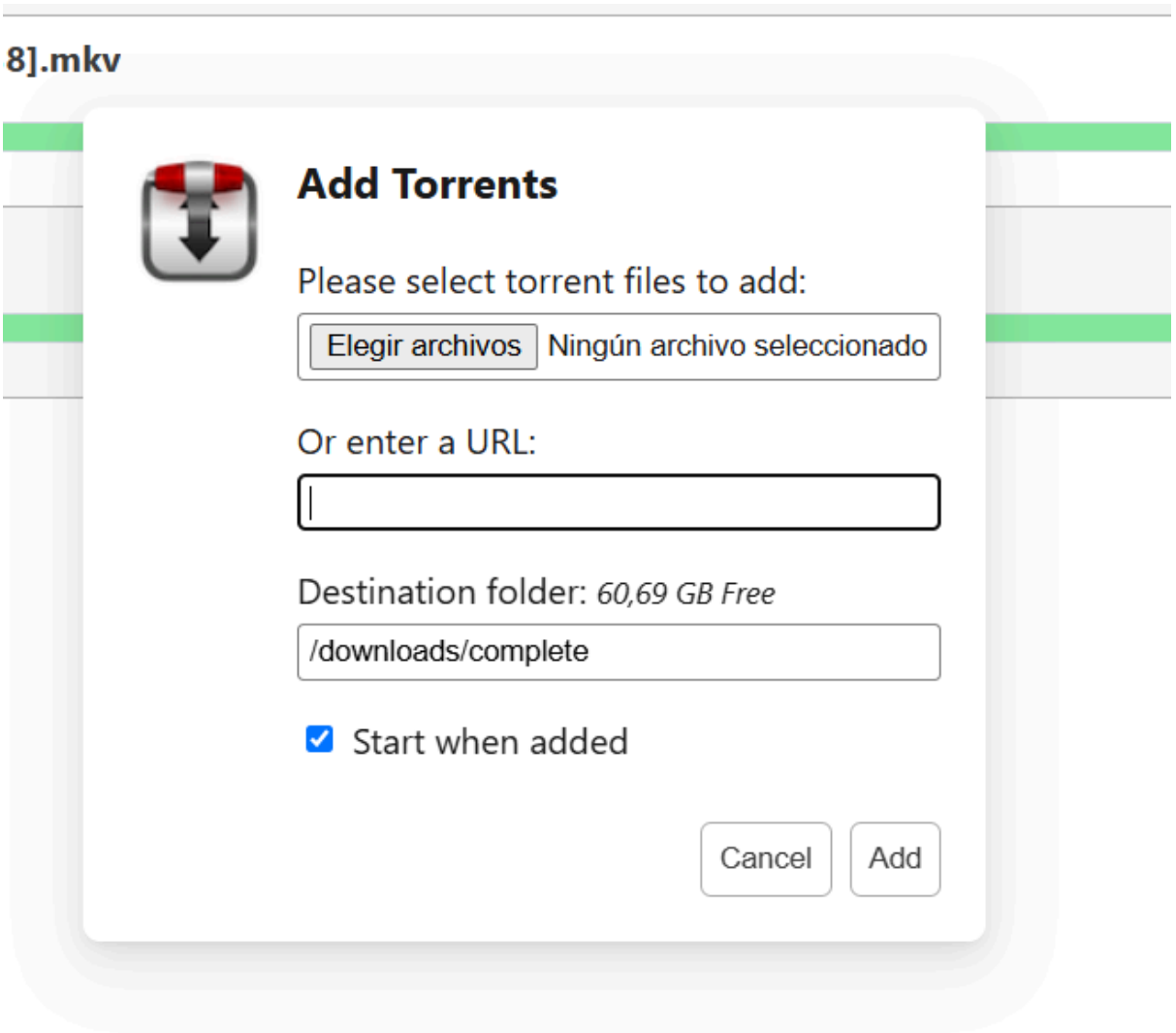
    - 51413:51413: Puerto TCP para conexiones BitTorrent.

    - 51413:51413/udp: Puerto UDP para conexiones BitTorrent.

- restart: unless-stopped
  El contenedor se reiniciará automáticamente si falla, excepto si se detiene manualmente.



Para iniciar la descarga de los archivos, sería tan sencillo como especificar la ruta de internet donde se encuentra o por el contrario subir el archivo torrent y lo tenemos en local.



5.2.5 Automatización

Anteriormente mencioné que los archivos descargados en transmission, se servirán directamente en plex, pero esto no es del todo así.

Los archivos descargados en plex en no se descarguen en formato comprimido, se servirán directamente en el servidor plex, por el contrario aquellos que vengan comprimidos no se podrán servir hasta que se descompriman.

Para solucionar esto cree el siguiente script.

Este script busca todos los archivos .rar dentro del directorio especificado y sus subcarpetas, y los extrae automáticamente en su mismo directorio, sin mostrar ninguna salida por pantalla.

Funcionaría de la siguiente manera:

- Ruta de búsqueda:
  El comando comienza con find /home/carlos-pi/vol01/plex-multimedia/complete/, lo que indica que se buscarán archivos dentro de este directorio y todas sus subcarpetas.

- Filtro por nombre:
  La opción -name '*.rar' filtra los resultados para que solo se incluyan archivos que terminen con la extensión .rar.

- Acción a ejecutar:
  La parte -execdir unrar e -r {} \; indica que por cada archivo .rar encontrado, se ejecutará el comando unrar en el mismo directorio donde se encuentra el archivo (gracias a -execdir, en lugar de -exec, que ejecutaría el comando desde el directorio donde se lanzó el script).
  El comando unrar e -r {} hace lo siguiente:

- e: extrae los archivos directamente en el directorio actual, sin conservar subcarpetas.

- -r: permite que la extracción sea recursiva, útil si hay múltiples volúmenes o archivos dentro del .rar.

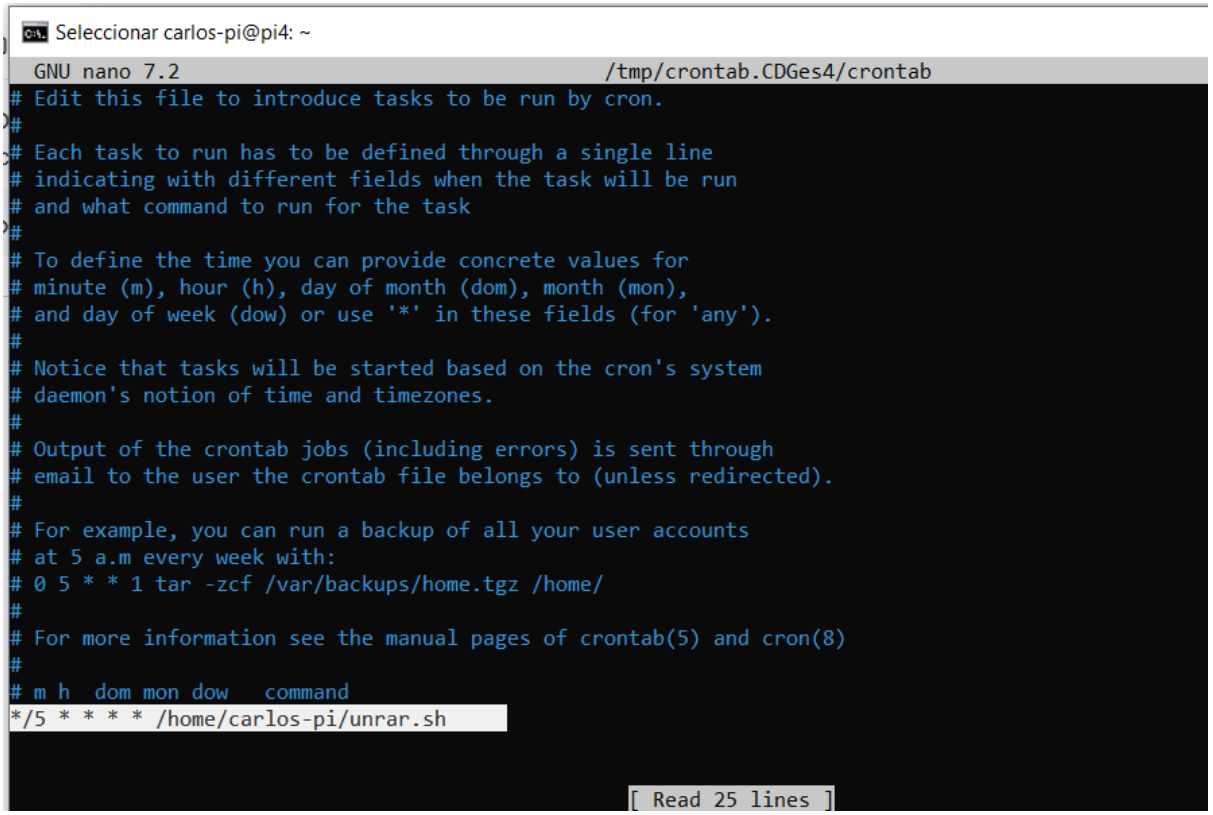
- {}: representa el nombre del archivo .rar encontrado.

- Silenciar la salida:
  La redirección > /dev/null hace que cualquier salida estándar del comando (mensajes normales, información de progreso, etc.) sea descartada. De esta forma, el script se ejecuta de manera silenciosa, sin mostrar mensajes por pantalla.

Para que este script se ejecute de manera automática cada 5 minutos, tendremos que realizar el siguiente comando:

crontab -e

Añadiendo la siguiente línea al fichero.



6. Exponer servidor a Internet

Para exponer todos estos servicios a la red y que sean accesibles desde cualquier parte del mundo, deberemos realizar los siguientes procesos. Hasta ahora todo esto funciona muy bien de forma local y aunque puede resultar muy útil, el atractivo principal de un servidor es que sea accesible desde cualquier parte del mundo.

6.1 Configuración del router

Para exponer el servidor a internet deberemos abrir los puertos del router, abrir los puertos del router significa permitir que conexiones externas desde Internet puedan acceder a un dispositivo o servicio específico dentro de tu red local (como un servidor, una cámara IP o un contenedor Docker).

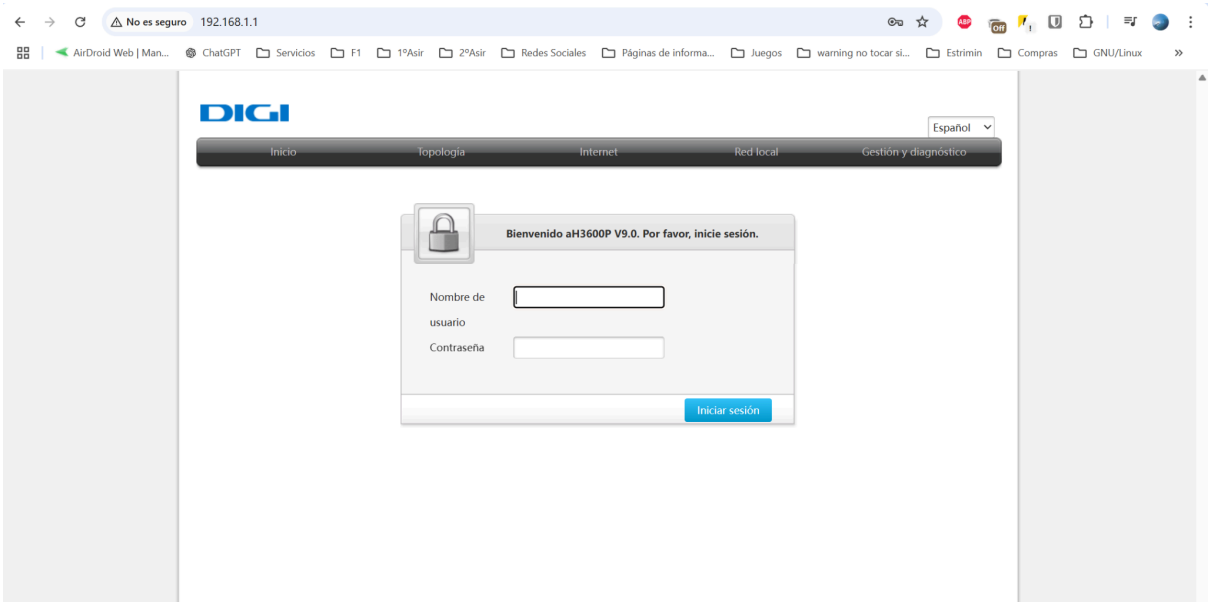
Por otro lado deberemos asignar un ip fija a nuestro servidor, Asignar una IP fija a un dispositivo en el router (como un servidor o Raspberry Pi) permite que siempre tenga la misma dirección IP dentro de la red local. Esto es útil porque:

- Permite acceder siempre al dispositivo en la misma dirección, sin cambios.
- Es necesario para redirigir puertos correctamente desde el router.
- Facilita configurar servicios en red (como Nextcloud, Duplicati, o impresoras).

En resumen, una IP fija evita que el router le asigne una dirección distinta cada vez que se reinicia, garantizando estabilidad en la conexión.

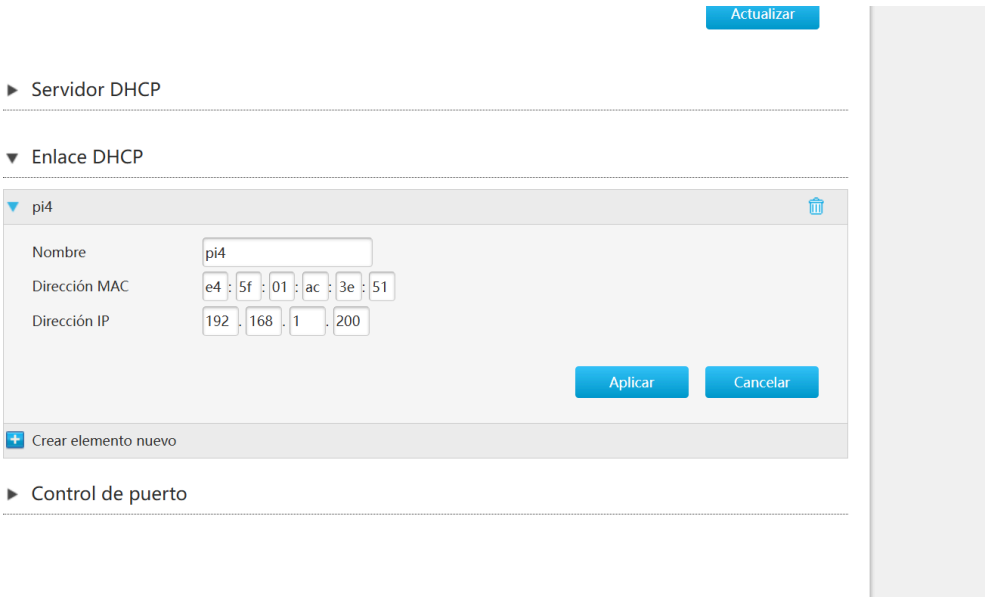
6.1.1 Asignar IP fija al servidor

Este procedimiento será diferente dependiendo del router de tu compañía o si por el contrario el router no es de tu compañía, en mi caso me dirigiré a la ip del router en la barra de navegación “192.168.1.1”.



Una vez nos identifiquemos (este procedimiento nos servirá para cualquier gestión del router), deberemos dirigirnos a “red local > LAN > Enlace DHCP”.

Una vez en dicho apartado deberemos introducir la MAC de nuestro servidor y la IP que queramos asignarle.



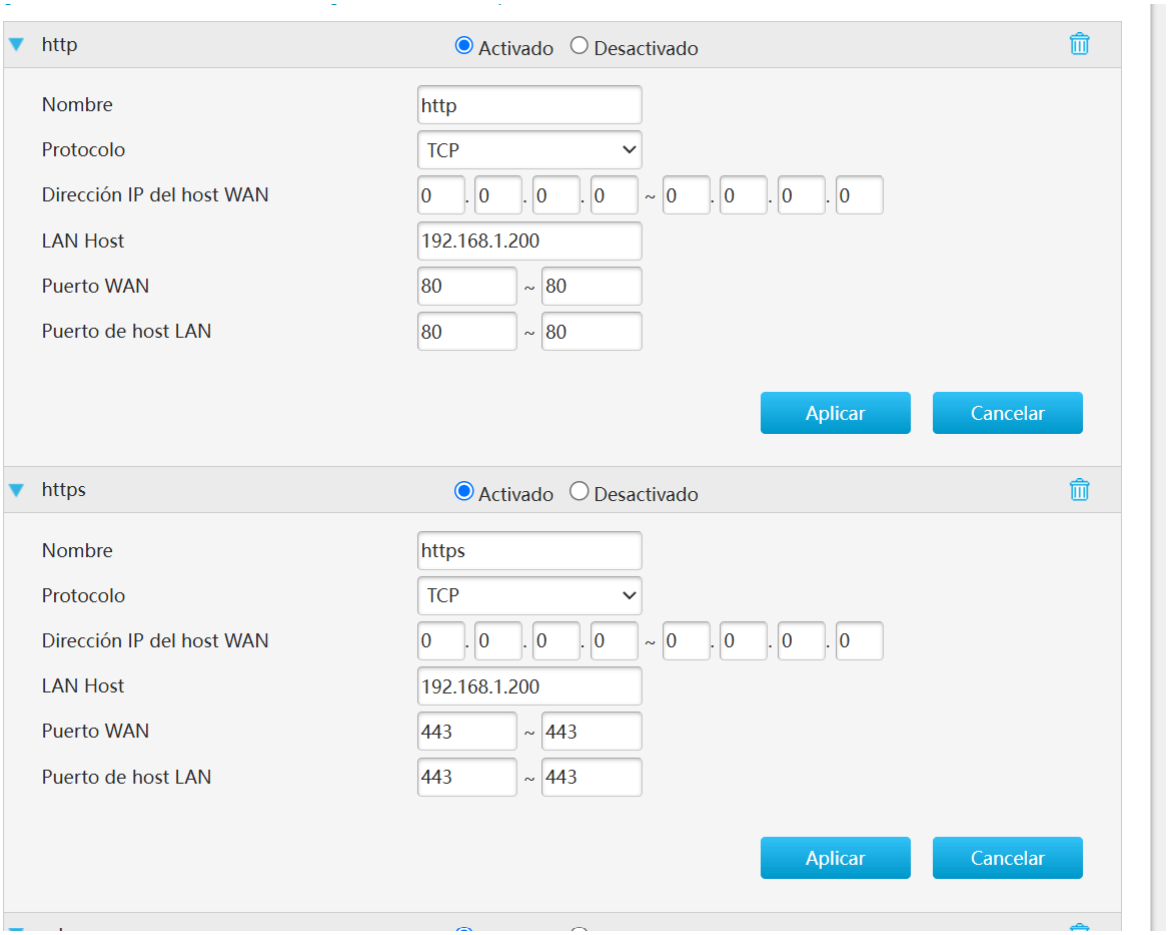
Con esto ya tendría nuestro servidor una IP fija siempre asociada.

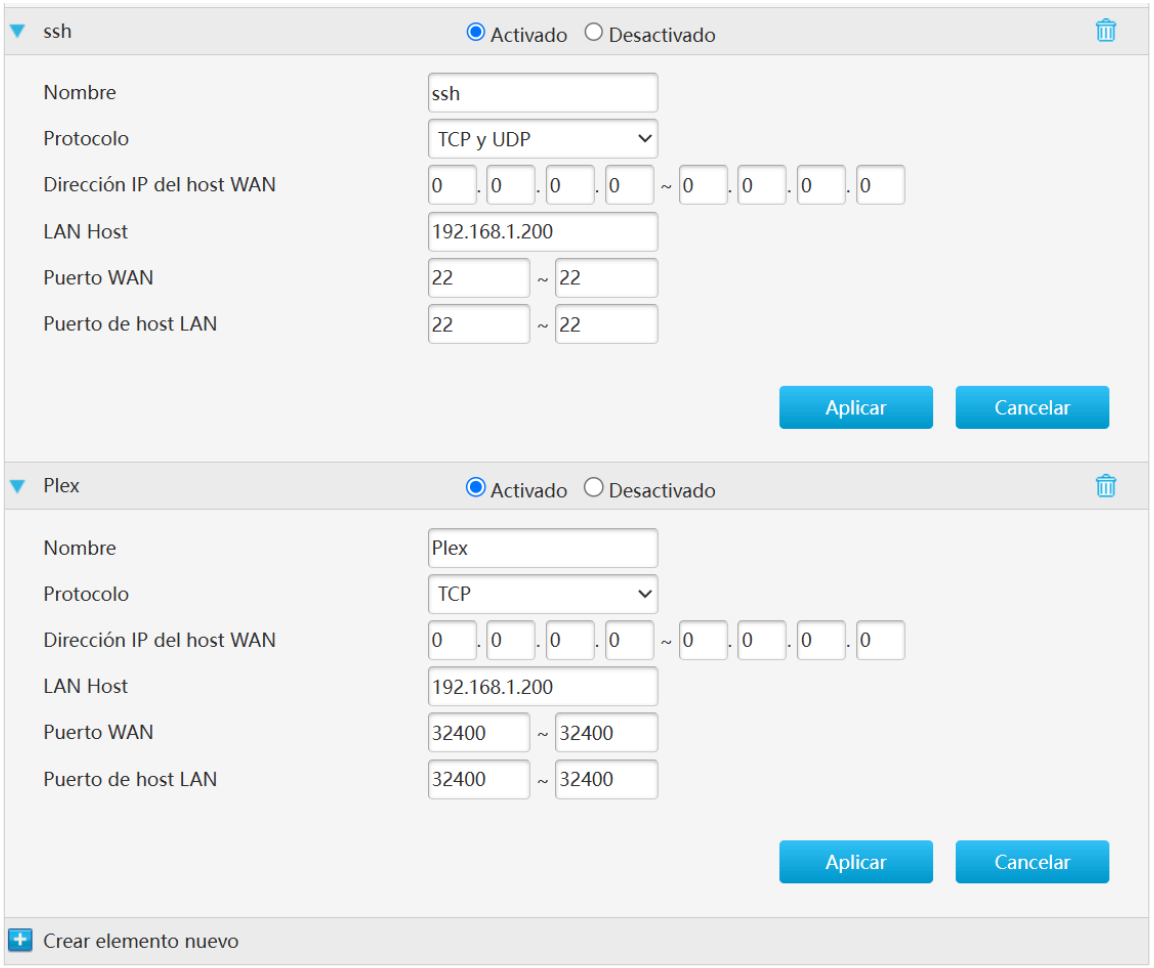
6.1.2 Abrir puertos

Para abrir los puertos deseados, en el router Digi nos dirigiremos al apartado “internet > seguridad > port forwarding”.

En mi servidor la redirección a los servicios se realiza desde el propio servidor utilizando un proxy que veremos más adelante, por lo que para el correcto funcionamiento solo tendremos que abrir los puertos 80 y 443, correspondiente a los protocolos http y https. También abriremos el 22 para permitir la conexión remota de gestión por ssh y el 32400 que es el puerto especial de plex.

Para la apertura de puertos solo tendremos que especificar la ip del servidor donde están los servicios, al haberlo configurado anteriormente siempre será la misma, y por otro lado el número del puerto con el protocolo.





Una vez hecho esto los puertos donde se alojan los servicios ya son accesibles desde el exterior.

6.2 Proxy y contenedores necesario

Una vez contamos con la configuración necesaria en el router, al tener un número tan grande de servicios con diferentes puertos, utilizaremos un proxy, más específicamente un contenedor de Nginx que viene preparado para esta tarea.

NGINX se utiliza como proxy inverso en un servidor con servicios expuestos a Internet para gestionar y redirigir el tráfico web de forma eficiente y segura. En este contexto, su función principal es:

- Recibir las peticiones externas (por ejemplo, a midominio.com)
- Redirigirlas internamente al servicio correspondiente (por ejemplo, a Nextcloud, Duplicati, etc.) según la URL o el puerto
- Añadir capa de seguridad (por ejemplo, certificados HTTPS con Let's Encrypt)
- Permitir usar un solo puerto (como el 80 o 443) para múltiples servicios internos

Esto simplifica el acceso, mejora el rendimiento y centraliza la gestión del tráfico web hacia los contenedores o aplicaciones alojadas en el servidor.

Por otro lado necesitaremos que la IP pública de nuestro router siempre se encuentre actualizada en los servidores DNS, para esto utilizaremos duckdns.

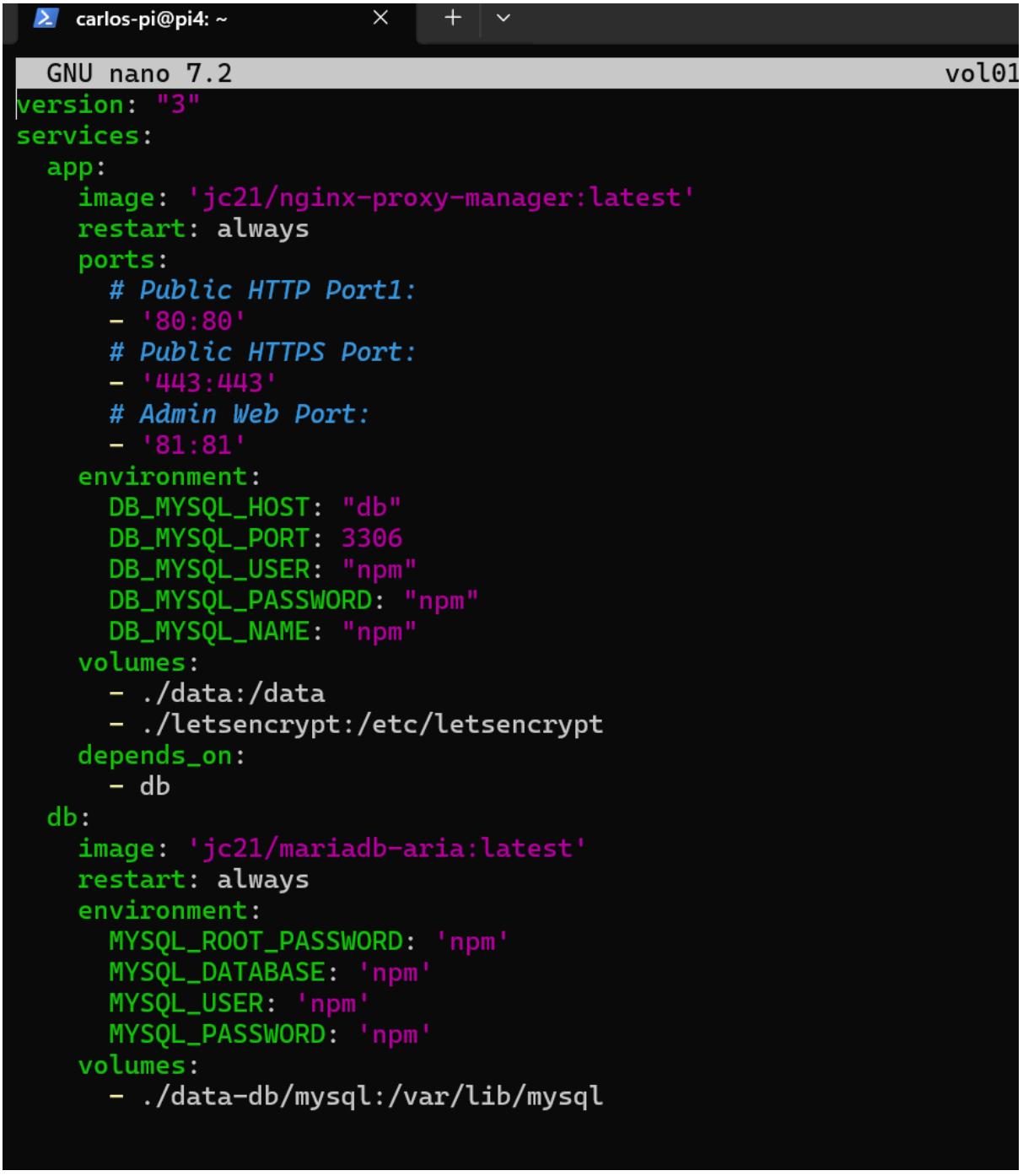
DuckDNS es un servicio gratuito de DNS dinámico (DDNS) que te permite asignar un nombre de dominio (como miservidor.duckdns.org) a tu IP pública, incluso si esta cambia, como ocurre en la mayoría de conexiones domésticas.

Permite acceder a tu servidor desde Internet usando un nombre fijo, aunque tu IP cambie, como suele ocurrir en conexiones sin IP fija.

Se usa un contenedor Docker que se conecta periódicamente a DuckDNS y actualiza automáticamente la IP pública asociada a tu dominio.

6.2.1 Nginx como proxy inverso

Para poder tener nuestro proxy funcionando, lo primero será crear el documento docker-compose que generará el servicio.



Este archivo docker-compose.yml levanta dos servicios:

1. app → Nginx Proxy Manager
2. db → MariaDB (base de datos para la app)

Servicio: app (Nginx Proxy Manager)

Imagen:

- jc21/nginx-proxy-manager:latest
  Imagen oficial del proxy inverso con interfaz web.

Reinicio automático:

- restart: always
  El contenedor se reinicia si se detiene por error o reinicio del sistema.

Puertos expuestos:

- 80:80 → Acceso público por HTTP
- 443:443 → Acceso público por HTTPS
- 81:81 → Acceso al panel de administración web

Variables de entorno:

- DB_MYSQL_HOST=db → Nombre del contenedor de base de datos
- DB_MYSQL_PORT=3306 → Puerto estándar de MySQL
- DB_MYSQL_USER=npm → Usuario de conexión a la base de datos
- DB_MYSQL_PASSWORD=npm → Contraseña del usuario
- DB_MYSQL_NAME=npm → Nombre de la base de datos que se usará

Volúmenes:

- ./data:/data → Guarda la configuración de la aplicación

- ./letsencrypt:/etc/letsencrypt → Guarda certificados SSL generados

Dependencias:

- depends_on: db
  El servicio app espera a que db esté listo antes de iniciarse.

Servicio: db (Base de datos MariaDB)

Imagen:

- jc21/mariadb-aria:latest
  Versión optimizada de MariaDB para funcionar con Nginx Proxy Manager.

Reinicio automático:

- restart: always
  Se reinicia automáticamente en caso de error o apagado.

Variables de entorno:

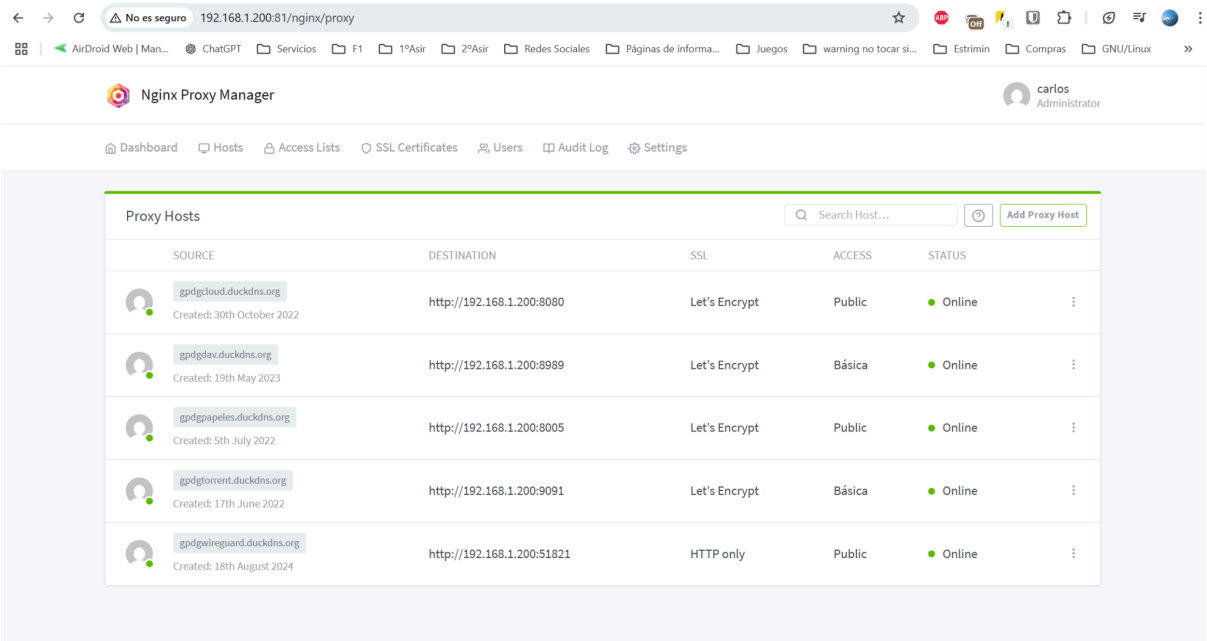
- MYSQL_ROOT_PASSWORD=npm → Contraseña del usuario root
- MYSQL_DATABASE=npm → Base de datos que se creará al inicio
- MYSQL_USER=npm → Usuario con acceso a la base
- MYSQL_PASSWORD=npm → Contraseña de ese usuario

Volúmenes:

- ./data-db/mysql:/var/lib/mysql
  Guarda de forma persistente los datos de la base de datos.

Este docker-compose.yml despliega Nginx Proxy Manager con su base de datos, permitiendo:

- Redirigir tráfico HTTP/HTTPS a servicios internos
- Administrar todo desde un panel web en el puerto 81
- Obtener certificados SSL automáticos con Let's Encrypt
  Ideal para servidores caseros con múltiples contenedores o aplicaciones expuestas a Internet.



1. Acceso inicial

- Abre tu navegador y ve a:
  http://TU_IP_DEL_SERVIDOR:81
- Usuario y contraseña por defecto:
  - Email: admin@example.com
  - Contraseña: changeme

Se recomienda cambiar estos datos en cuanto accedas.

2. Cambiar usuario administrador

- Una vez dentro, te pedirá cambiar el correo y la contraseña del administrador por motivos de seguridad.

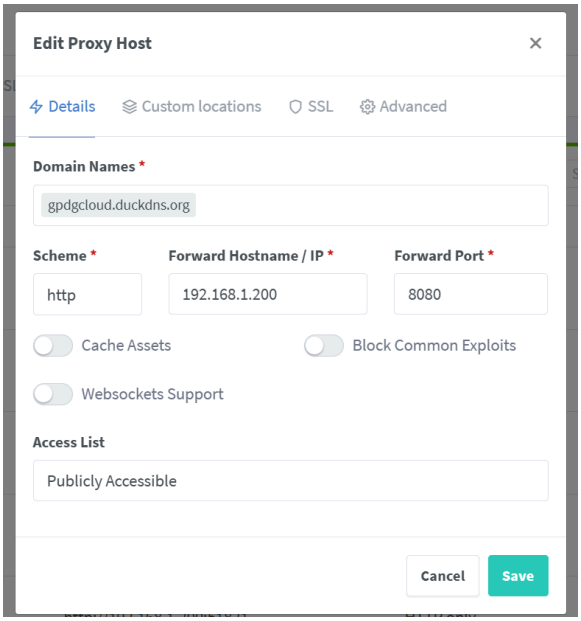
3. Añadir un "Proxy Host"

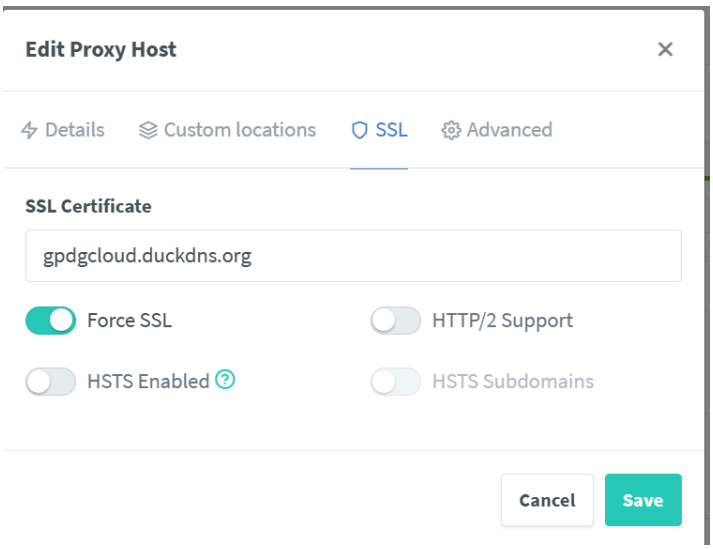
Para redirigir tráfico a uno de tus servicios internos:

1. Ve a "Proxy Hosts" y haz clic en "Add Proxy Host".
2. Rellena:
    - Domain Names: el dominio que usarás (ej. nextcloud.midominio.duckdns.org)
    - Forward Hostname / IP: la IP o nombre del contenedor al que redirigir (ej. nextcloud-app)
    - Forward Port: el puerto del servicio (ej. 80 o 443)
    - Marca "Block Common Exploits"
    - Si tienes dominio con HTTPS, activa SSL, selecciona "Request a new SSL certificate", y marca "Force SSL"

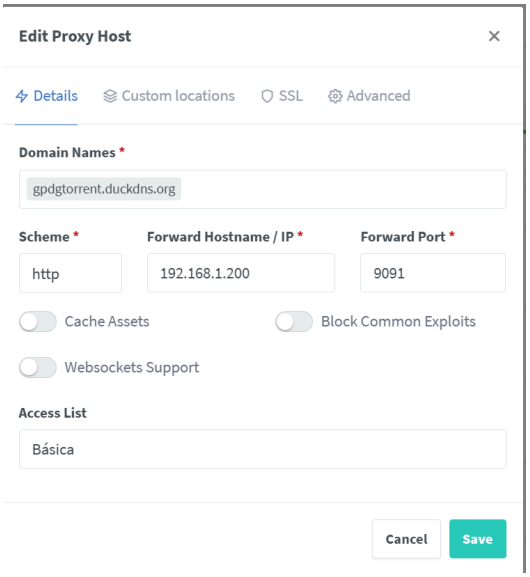
4. Resultado

- Cuando termines, NPM creará automáticamente las reglas de redirección y configurará el certificado SSL con Let's Encrypt si lo solicitaste.
- Ya puedes acceder al servicio desde Internet con tu dominio DuckDNS (por ejemplo).

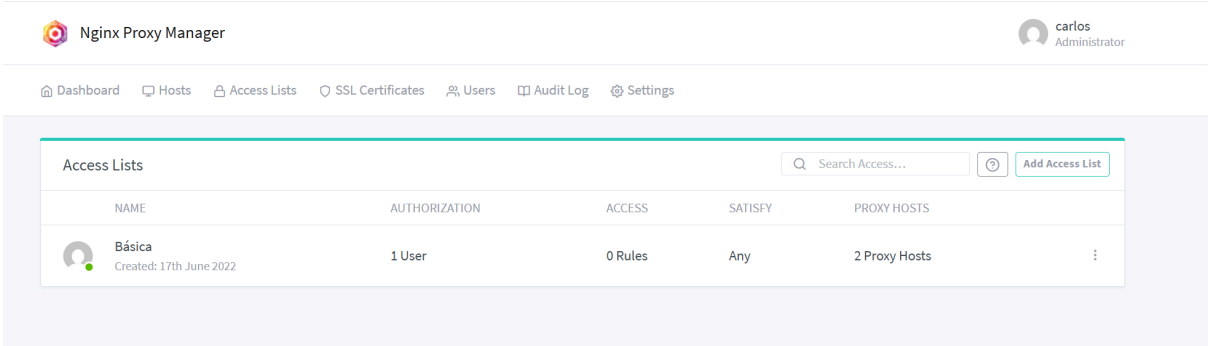


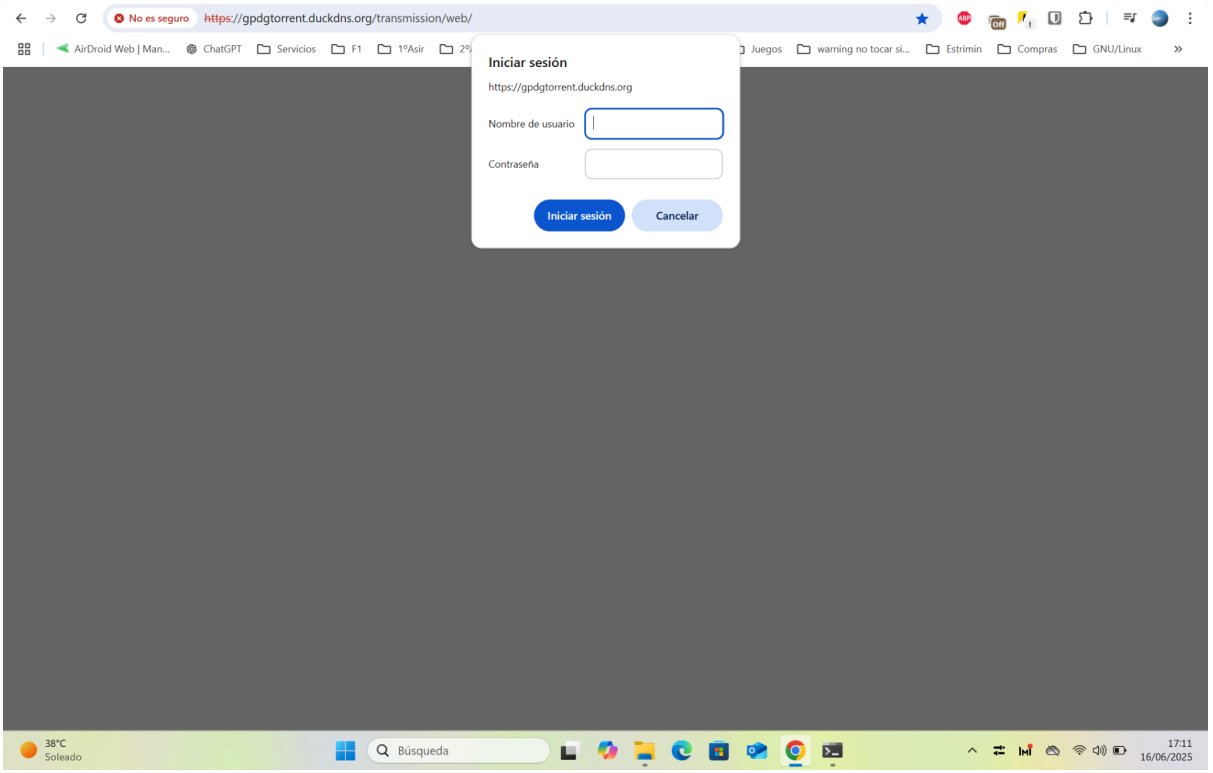


En el caso de la configuración de transmission, también hemos creado un control de acceso básico http ya que esta plataforma no tiene autentificación.



Las credenciales se configuran desde el apartado acces list y ya las puedes incluir en todos los dominios que desees.





6.2.2 Duckdns

```
carlos-pi@pi4: ~
  GNU nano 7.2                                  vol01/docker/duckdns/docker-c
version: "2.1"
services:
  duckdns:
    image:  linuxserver/duckdns:arm32v7-version-e60b35ff
    container_name: duckdns
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Madrid
      - SUBDOMAINS=gpdg,gpdgcloud,gpdgtorrent,gpdgpapeles
      - TOKEN=d2ddfad5-2e19-40e1-9fbc-d99acbd92a1c
      - LOG_FILE=false
    volumes:
      - ./appdata/config:/config
    restart: unless-stopped
```

Imagen utilizada:
linuxserver/duckdns:arm32v7-version-e60b35ff
(versión para arquitecturas ARM 32 bits, como Raspberry Pi)

Configuración del contenedor:

- Nombre: duckdns
- Reinicio automático: restart: unless-stopped (reinicia salvo parada manual)

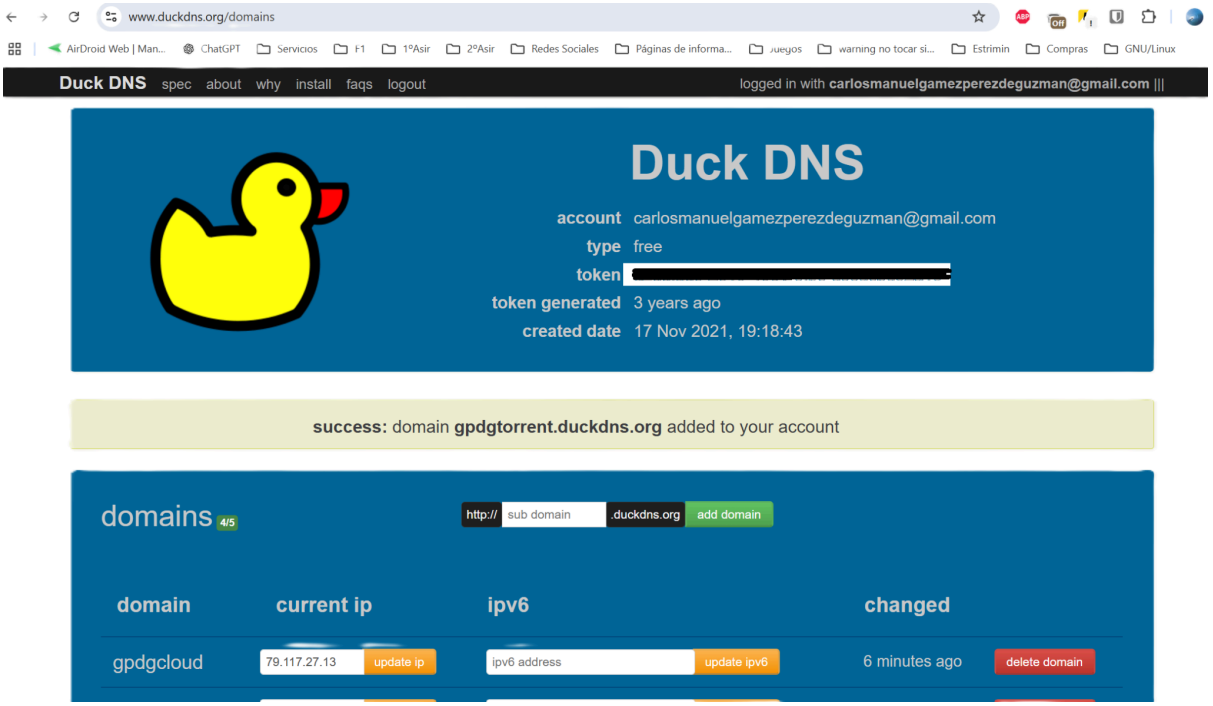
Variables de entorno (environment):

- PUID=1000 → ID usuario del host
- PGID=1000 → ID grupo del host
- TZ=Europe/Madrid → Zona horaria
- SUBDOMAINS=gpdg, gpdgcloud, gpdgtorrent, gpdgpapeles → Subdominios DuckDNS a actualizar
- TOKEN → Token de autenticación DuckDNS
- LOG_FILE=false → Desactiva escritura de logs en fichero

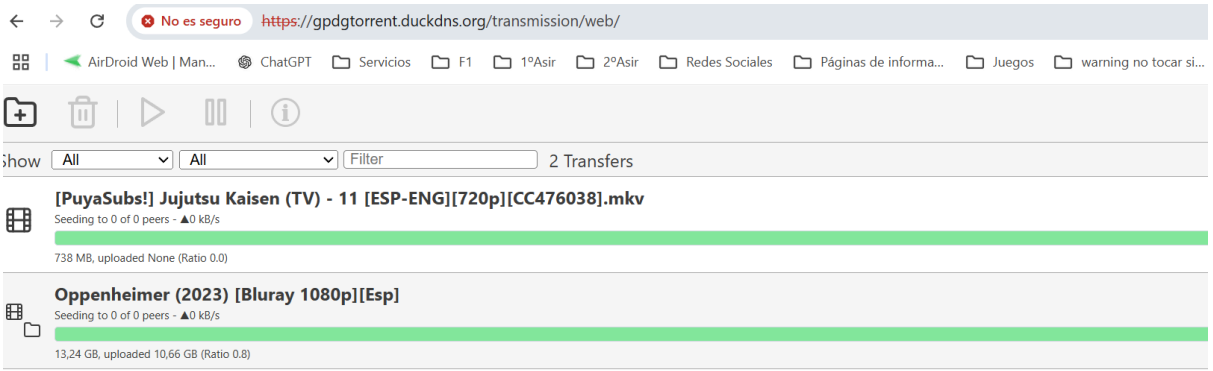
Volumen montado:

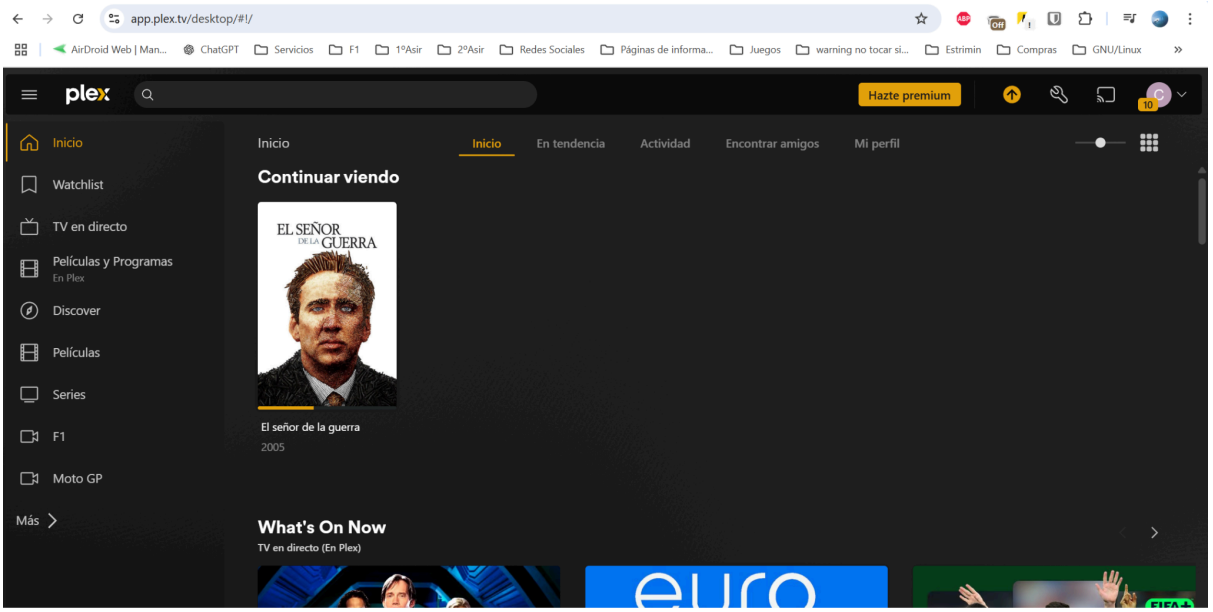
- Host: ./appdata/config
- Contenedor: /config
- Propósito: almacenar configuración persistente para conservar ajustes tras reinicios o recreación

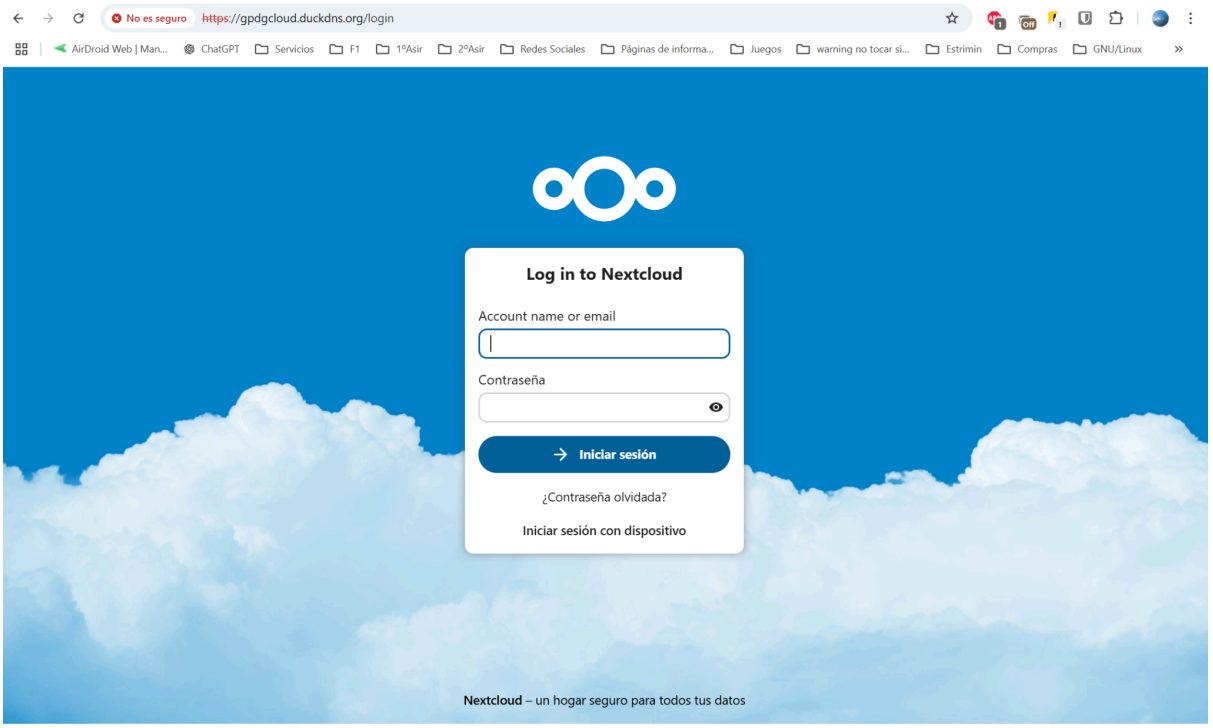
Los parámetros como el token de autenticación se te proporciona una vez te registras en su web, desde la misma también podrás elegir el nombre de los dominios.



Una vez hecho esto los servicios ya serán accesibles desde el exterior.




app.plex.tv/desktop/#!/
plex
Hazte premium
Inicio
Watchlist
TV en directo
Películas y Programas
En Plex
Discover
Películas
Series
F1
Moto GP
Más
Inicio
Inicio
En tendencia
Actividad
Encontrar amigos
Mi perfil
Continuar viendo
EL SEÑOR DE LA GUERRA
El señor de la guerra
2005
What's On Now
TV en directo (En Plex)


No es seguro
https://gpdgcloud.duckdns.org/login
Log in to Nextcloud
Account name or email
Contraseña
Iniciar sesión
¿Contraseña olvidada?
Iniciar sesión con dispositivo
Nextcloud – un hogar seguro para todos tus datos

7. Inconvenientes encontrados CGNAT

El principal inconveniente encontrado en este proyecto ha sido el CGNAT.

CGNAT (Carrier-Grade NAT) es una técnica que usan las compañías de telecomunicaciones para compartir una única dirección IP pública entre varios clientes.

El principal motivo es la escasez de direcciones IPv4. Como no hay suficientes IPs públicas para todos los usuarios, las operadoras usan CGNAT para que muchos clientes puedan navegar por Internet usando una misma IP pública, funciona de la siguiente manera:

- Cada cliente tiene una IP privada en su red doméstica (como 192.168.x.x).
- Esa IP privada es traducida por el router del cliente a otra IP privada de la red del operador.
- Luego, el operador usa CGNAT para traducir esa IP a una IP pública compartida.
- Esto permite que decenas o cientos de usuarios naveguen con la misma IP pública, gestionando cada conexión mediante distintos puertos.

Este problema no dio la cara hasta el día que tuve que cambiar de compañía por el elevado precio de la anterior, en la anterior compañía contaba con una IP pública asociada, pero al cambiar a la nueva compañía en mi cabeza solo tendría que volver a configurar el router para que todo funcionara. Esto no fue así debido a que esta compañía utilizaba CGNAT.

Esta misma compañía me solucionaba el problema y me daba una IP pública con un sobre coste en el pago, a lo cual acepté al tratarse de un bajo coste. Una vez realizado esto, todo volvió a funcionar correctamente, ya que el mayor gasto de tiempo es en descubrir que estaba pasando.

8. Conclusión

Como conclusión recomendaría a todo el mundo que empiece en este grado y le apasiona, realizar este proyecto ya que no solo aprenderás a medida que crear un proyecto que está en constante evolución, si no que también ahorraras en servicios como netflix, nubes privadas y todo aquello que cada día suben más los precios y nos dejan más a merced de sus condiciones una vez los hemos incluido en nuestra vida, a la vez que no dejamos nuestros datos en manos de ninguna empresa privada.