🛠️Configurar un servidor VPS desde 0 paso a paso💾

Montar un servidor Ubuntu con Dockers, Portainer y Nginx Proxy Manager

Realizado por Jose Rodríguez.

LinkedIn: www.linkedin.com/in/joseperfil

GitHub: jose-giithub

Portfolio: https://portfolio.jose-rodriguez-blanco.es

🚧Requisitos previos

1. ✅Tener un **servidor** basado en **Linux**. En este tutorial se usa un **Ubuntu 24.04.2 LTS (Noble Numbat)**.
2. ✅Tener un **dominio** y **subdominios**
3. ✅**Conocer** la **IP pública** de nuestro **servido**.

4. ✅Tener **acceso** al **usuario root** y su **contraseña**

- 👮**Crear el usuario `jose`**

Crear un usuario propio que no sera root pero con derechos a usar sudo para modificar cosas en el servidor

✅ **1. Crear un nuevo usuario**

```
adduser jose
```

✅ **2. Añadir jose al grupo sudo**

```
usermod -aG sudo jose
```

⚠️ *Solo los usuarios en el grupo `sudo` pueden usar el comando `sudo`.*

✅ **3. Verifica que se añadió bien al grupo `sudo`**

```
groups jose
```

Deberías ver algo como:

```
root@vmi2595990:~# groups jose
jose : jose sudo users
```

✅ **4. Iniciar sesión como el nuevo usuario**

su - jose

Ahora estarás usando el sistema como el usuario `jose`.

✅ **5. Actualiza sistema e instalar los nuevos paquetes**

sudo apt update && sudo apt upgrade -y

✅ **6. Instalar 7zip, nano y tree**

sudo apt install nano

sudo apt install tree

Todo junto:

sudo apt install p7zip-full nano tree -y

- 🛡️**Seguridad**

✅ **Paso previo. 🟥Actualizar sistema**

sudo apt update

✅**1 fail2ban**

Protege contra intentos de acceso por fuerza bruta.

sudo ufw allow OpenSSH

sudo apt install fail2ban -y

sudo systemctl enable fail2ban

sudo systemctl start fail2ban

Para ver su estado:

sudo systemctl status fail2ban

Y para ver qué está protegiendo:

sudo fail2ban-client status

✅**2 ufw**

Firewall básico. Ejemplo para activar UFW y permitir solo SSH (puerto 22):

sudo ufw enable

✅3 Configurar un nuevo sistema Linux o al prepararse para instalar otro software que depende de estas utilidades para descargas seguras y verificación de paquetes

1. Instala **curl**, una herramienta práctica para obtener datos de internet.
2. Instala **GnuPG**, que es importante para verificar la autenticidad de los paquetes de software.
3. Instala **lsb-release**, que ayuda a identificar tu distribución de Linux.

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

✅ 4. Instalar apache2-utils para crear htpasswd

sudo apt install apache2-utils -y

🧼Escanear tu VPS en busca de archivos maliciosos (y limpiar basura)

✅ Paso 5.1: Instalar rkhunter y chkrootkit

Estas dos herramientas son básicas para detectar rootkits, troyanos y modificaciones sospechosas.

sudo apt update
sudo apt install rkhunter chkrootkit -y

✅ Paso 5.2: Ejecutar el escaneo manualmente de rkhunter

sudo rkhunter --update

sudo rkhunter --propupd *#* Solo la primera vez (guarda estado actual del sistema)

sudo rkhunter --check *#* Iniciar escaneo, esto puede tardar un rato,

🔍**Para ver el resumen del escaneo:**

sudo cat /var/log/rkhunter.log | grep Warning:

✅ **Paso 5.3: Ejecutar el escaneo manualmente de chkrootkit**

sudo chkrootkit

✅ **Paso 5.5.🗑️ Borrar archivos temporales y limpieza general**

sudo apt autoremove -y

sudo apt autoclean

sudo journalctl --vacuum-time=7d

🧹**Eso borrará los logs del sistema de más de 7 días (¡sin romper nada!).**

- 🛡️🤖📄**Crear un Script para que el sistema se limpie automáticamente**🧼

❗Con el usuario jose (no root)
⚠️Para realizar esta parte tienes que haber instalado **rkhunter y chkrootkit**

ℹ️Crearemos un archivo .sh que se ejecutará todos los días a las 4:20 de la madrugada (esta hora está libre y nada más se estará ejecutando con cron) para hacer un escaneo y limpieza de cosas malas. Cuando termine creará un .log en (***/home/jose/scripts/logs).*** Creará un .log diario, por ejemplo ***limpieza_seguridad_diaria_2025-06-04.log*** y se guardarán solo los de los 7 días posteriores. A partir del 7 se borrará el más viejo

✅ **1. 📁Creamos un directorio *scripts* (Si aún no existe) dentro del directorio de**

nuestro usuario (*/home/jose*)

mkdir scripts

✅ **2.** 📄 **Entramos al directorio *scripts.* Creamos el archivo *limpieza_seguridad_diaria***

nano limpieza_seguridad_diaria

✅ **3.** 📝 **Contenido del archivo *limpieza_seguridad_diaria.sh***

```
#!/bin/bash Script para limpieza y seguridad diaria a las 4:20 AM
# Rotación de logs, solo se guardan 7 días de logs

BASE_DIR="/home/jose/scripts"
LOG_DIR="$BASE_DIR/logs"
FECHA_HOY=$(date +%F)  # formato: 2025-06-04
LOG_FILE="$LOG_DIR/limpieza_seguridad_diaria_$FECHA_HOY.log" # formato:
limpieza_seguridad_diaria_2025-06-04.log
mkdir -p "$LOG_DIR"

# Limitar a los últimos 7 logs (los más recientes)
find "$LOG_DIR" -type f -name 'limpieza_seguridad_diaria*.log' | sort | head -n -7 | xargs
-r rm

# Comenzar a escribir el .log de hoy
echo "🕒 Fecha: $(date)" > "$LOG_FILE"
echo "🔐 Iniciando escaneo de seguridad..." >> "$LOG_FILE"

# RKHunter update
echo "📥 Actualizando RKHunter..." >> "$LOG_FILE"
sudo rkhunter --update >> /dev/null 2>&1

# Propiedades del sistema
echo "📦 Actualizando base de archivos del sistema..." >> "$LOG_FILE"
sudo rkhunter --propupd -q
```

```
# RKHunter check
echo "🔎 Analizando con RKHunter (solo advertencias)..." >> "$LOG_FILE"
sudo rkhunter --check --sk --nocolors > /tmp/rkhunter_check.txt 2>&1
grep -E "Warning|Possible rootkits" /tmp/rkhunter_check.txt >> "$LOG_FILE" || echo
"✔️ Sin advertencias de RKHunter." >> "$LOG_FILE"


# chkrootkit (solo lo importante)
echo "🔍 Ejecutando chkrootkit (resumen)..." >> "$LOG_FILE"
sudo chkrootkit | grep -v "not found" | grep -v "not infected" >> "$LOG_FILE"

# Limpieza
echo "🧹 Limpiando sistema (paquetes y cachés)..." >> "$LOG_FILE"
sudo apt autoremove -y >> /dev/null
sudo apt autoclean >> /dev/null

# Logs viejos
echo "🧻 Borrando logs antiguos (más de 7 días)..." >> "$LOG_FILE"
sudo journalctl --vacuum-time=7d >> "$LOG_FILE"

# Final
echo "✅ Escaneo completado correctamente." >> "$LOG_FILE"
echo "---Fin" >> "$LOG_FILE"
```

✅ 4. ↩️ Hacerlo ejecutable

chmod +x /home/jose/scripts/limpieza_seguridad_diaria.sh

✅ 5.🚀 Validar que funciona ejecutándolo manualmente

👀⚠️ Si estás en el mismo directorio que el script será:

bash limpieza_seguridad_diaria.sh

👀 ⚠️Si estás en otro directorio que el script será

bash /ruta/completa/limpieza_seguridad_diaria.sh

❗ Esto tardará un rato.

✅ 🔎Ver logs

Para ver el resultado en:

cat /home/jose/scripts/logs/limpieza_seguridad_diaria_2025-06-04.log

---

✅ 6.🔄 Automatizar la ejecución del archivo, *limpieza_seguridad_diaria.sh* para que se ejecute cada día las 4:20 de la mañana

crontab -e

Se abrirá un archivo, tienes que añadir el código:

20 4 * * * /ruta/del/archivo/limpieza_seguridad_diaria.sh

🔹 Eso lo ejecutará cada lunes a las 4:20 de la madrugada y guardará el resultado en

👁️ Así es como debe quedar:

20 4 * * * /home/jose/scripts/limpieza_seguridad_diaria.sh

🚨**Superimportante**: la ruta se tiene que especificar donde está el archivo.

- ### 🌐 Red y diagnóstico

✅ **Paso previo. Actualizar sistema**

```
sudo apt update
```

✅**1 net-tools**

Comando `ifconfig`, `netstat` y otros.

sudo apt install net-tools

✅ **2 nmap**

Escaneo de puertos (para saber qué está abierto o cerrado).

sudo apt install nmap

✅ **3 lsof**

Ver qué procesos están usando archivos o puertos.

sudo apt install lsof

- 🤖📄**Crear un Script para que el sistema se actualice automáticamente**

Crear un archivo ejecutable que se ejecutará automáticamente todas las semanas y se actualice todo el sistema.

✅ **1.** 📁**Creamos un directorio *scripts* (Si aún no existe) dentro del directorio de nuestro usuario (*/home/jose*)**

✅ **2.** 📄 **Creamos el archivo con los comandos**

nano actualizarSistema.sh

✅ **3.** 📝 **Contenido del archivo *actualizarSistema.sh***

```
#!/bin/bash

echo "📦 Actualizando lista de paquetes..."
sudo apt update

echo "⬆️ Actualizando paquetes instalados..."
sudo apt upgrade -y

echo "🧹 Limpiando paquetes innecesarios..."
sudo apt autoremove -y
sudo apt autoclean

echo "✅ Sistema actualizado correctamente el $(date)"
```

✅ **4.** ↩️**Convertir el archivo *actualizarSistema.sh* en ejecutable**

chmod +x actualizarSistema.sh

✅ **5.**🚀 **Validar que funciona ejecutándolo manualmente**

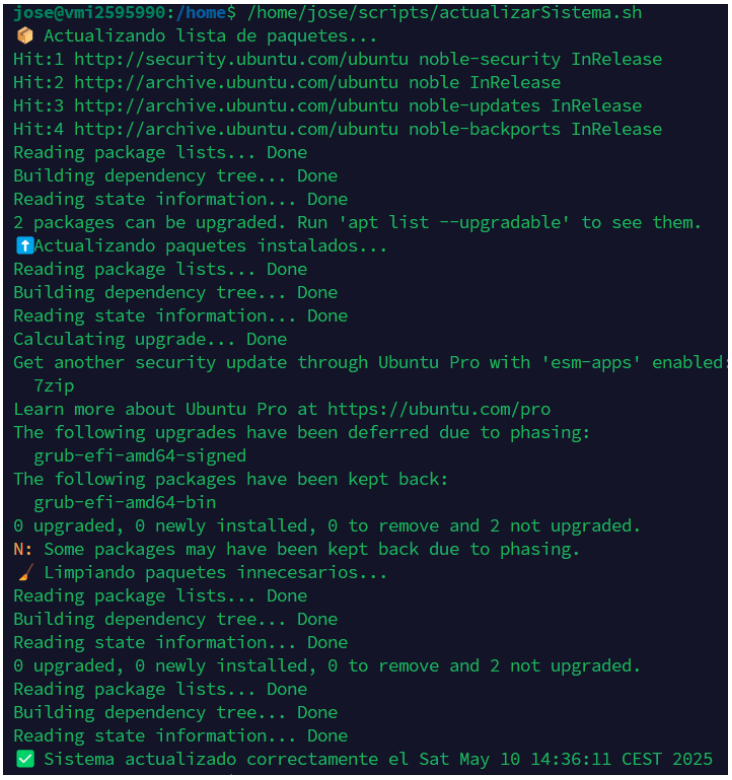
👀⚠️ Si estás en el mismo directorio que el script será:

bash actualizarSistema.sh

👀 ⚠️Si estás en otro directorio que el script será

bash /ruta/completa/actualizarSistema.sh

Resultado:



✅ 6.🔄 Automatizar la ejecución del archivo, *actualizarSistema.sh* para que se ejecute cada lunes a las 3 de la mañana

crontab -e

Se abrirá un archivo, tienes que añadir el código:

0 3 * * 1 /ruta/del/archivo/actualizarSistema.sh >> /var/log/actualizarSistema.log 2>&1

- Eso lo ejecutará cada lunes a las 3:00 de la madrugada y guardará el resultado en

/var/log/***actualizarSistema***.log.

✅ Así es como debe quedar:

0 3 * * 1 /home/jose/scripts/actualizarSistema.sh >> /var/log/actualizarSistema.log 2>&1

🚨**Superimportante**: la ruta se tiene que especificar desde home hasta el archivo.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 3 * * 1 /home/jose/scripts/actualizarSistema.sh >> /var/log/actualizarSistema.log 2>&1
```

✅ 7 🚀Validar que funciona ejecutable de crontab funciona como se espera

Simula que eres el crontab y lanza la ejecución

sudo bash -c '/home/jose/scripts/actualizarSistema.sh >> /var/log/actualizarSistema.log 2>&1'

🔎 Luego puedes revisar el .log con:

cat /var/log/actualizarSistema.log

- ### 🤖✉️Correos automáticos

❗ Con el usuario jose (no root) pero usando sudo

❗ El correo que vas a usar para mandar correos tiene que estar habilitado para permitir el uso de aplicaciones menos seguras, sigue los pasos de este tutorial:

🛠️ **¿Cómo crear una contraseña de aplicaciones en Gmail? - YouTube**

🔗Enlace para crear contraseña para aplicaciones:

Contraseñas de aplicación

ℹ️ Instalaremos todo el entorno para enviar correos electrónicos de forma automática cada vez que se ejecute el script de actualización automática o cuando queramos.

✅1.🛠️ Instalar msmtp y herramientas necesarias

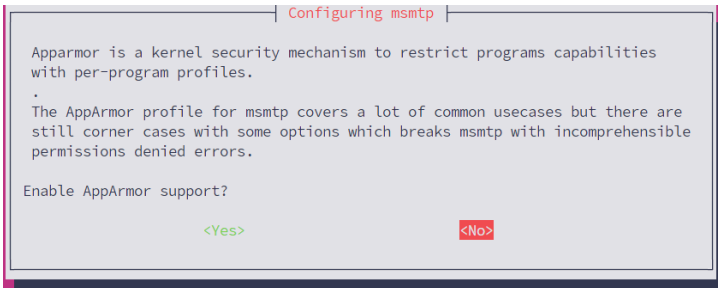
sudo apt update

sudo apt install -y msmtp msmtp-mta

ℹ️msmtp: el cliente SMTP.

ℹ️msmtp-mta: hace que msmtp se comporte como un agente de correo.

ℹ️ ❗ Te mostrará esta ventana:
"Quieres que `msmtp` se instale con su perfil de AppArmor activado" es una capa de seguridad, en mi caso le daré a no, pues puede dar problemas difíciles de depurar en el futuro y en este caso es un servicio muy sencillo



✅ 2. 🗂️Crear archivo de configuración .msmtprc

Este archivo contiene los datos para enviar el correo. Vamos a crearlo en tu carpeta personal (**/home/user .msmtprc**).

nano ~/.msmtprc

❗ El archivo ***.msmtprc*** debe estar en el directorio home del usuario que ejecuta ***msmtp***), si no fuera así daría problemas

---

✅3. ✏️Contenido típico (ejemplo con Gmail, puedes adaptarlo a tu proveedor o poste.io si lo usas más adelante):

```
defaults
auth           on
tls            on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
logfile        ~/.msmtp.log

account        default
host           smtp.gmail.com
port           587
from           ejemplo@gmail.com
user           ejemplo@gmail.com
password       TU_CONTRASEÑA_DE_APLICACIÓN_AQUI
```

ℹ️ control + x para salir y guardar

✅ 4. 🔐Proteger el archivo *.msmtprc*

Este archivo contiene tu contraseña, así que ponle permisos seguros:

chmod 600 ~/.msmtprc

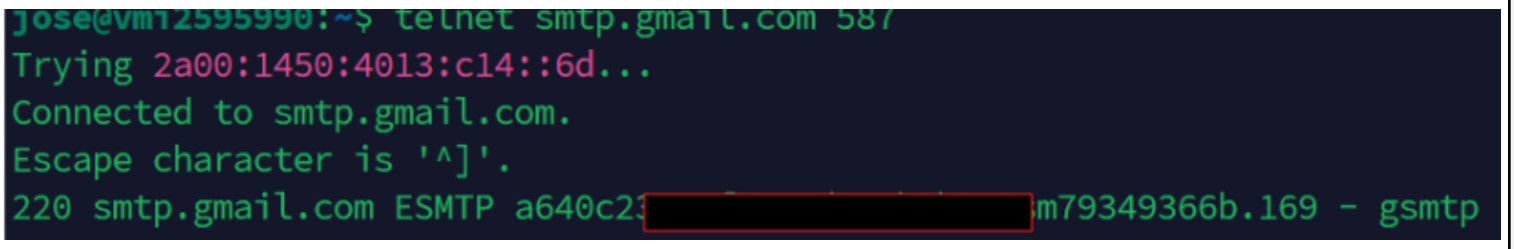
✅ 5.🔓 Validar que el puerto esté abierto en tu servidor

Para Gmail se usa **puerto 587** con TLS.

Verifica con telnet o nc

telnet smtp.gmail.com 587

Deberías ver algo así:



❗Te quedarás atrapado en una especie de terminal, pero salir tienes que:

🔚 Para salir de Telnet presiona:

Ctrl +]

Después, ahí escribes:

quit

Y saldrás del modo Telnet.

✅6. 🚀Probar manualmente el envío de correo

Usa el comando `mail` para probarlo:

```
echo "Este es un mensaje de prueba" | mail -s "Prueba desde msmtp" ejemplo@gmail.com
```

- Si todo va bien, recibirás el mensaje en tu bandeja de entrada.
- Si falla, revisa el archivo de .log en `~/.msmtp.log`.

✅7. ✍️📄Automatizar el envío con tu script

Objetivo: Modificar el script ***actualizarSistema.sh*** para automatizar las actualizaciones del servidor, enviar notificaciones por correo (éxito o error), y gestionar los logs de forma rotativa.

El script ahora guardará los logs de cada ejecución en la ruta ***/home/jose/scripts/logs/*** con un formato de nombre diario (ej. ***actualizar_sistema_2025-06-05.log***), manteniendo solo los últimos 7 días.

```
#!/bin/bash

# Script para actualizar el servidor automáticamente
# Notifica por correo si hay errores o si se completa correctamente
# Rotación de logs, solo se guardan 7 días de logs

DESTINATARIO="ejemplo@gmail.com"
#Ruta del .msmtprc explícitamente, esto obliga a msmtp a usar tu configuración aunque el script lo ejecute root,
MSMTP_CONFIG="/home/jose/.msmtprc"
LOG_DIR="/home/jose/scripts/logs" # Ubicación de los archivos .log
FECHA_HOY=$(date +%F)  # formato: 2025-06-05
LOG_FILE="$LOG_DIR/actualizar_sistema_$FECHA_HOY.log"  # formato: del .log
actualizar_sistema_2025-06-05.log

# Asegurarse de que el directorio de logs existe ANTES de hacer el mkdir para el .log del día
mkdir -p "$LOG_DIR"

# Limitar a los últimos 7 logs (los más recientes)
# Nota: La lógica de rotación debe ejecutarse ANTES de crear o escribir en el LOG_FILE del día actual
find "$LOG_DIR" -type f -name 'actualizar_sistema*.log' | sort | head -n -7 | xargs -r rm

# --- Iniciar la captura de la salida a un archivo de .log y a la consola ---
# Todas las líneas siguientes se enviarán tanto al LOG_FILE como a la salida estándar
# `exec > >(tee -a "$LOG_FILE") 2>&1` redirige stdout y stderr al tee, que a su vez lo envía al archivo y a la consola.# Esto debe ir al principio del script, después de definir LOG_FILE, para capturar toda la salida.
exec > >(tee -a "$LOG_FILE") 2>&1

echo "📦 Actualizando lista de paquetes..."
echo "🕒 Fecha de inicio: $(date)" # Muestra la fecha de inicio en el .log y en la consola

if ! apt update; then
    echo "❌ ERROR: Fallo actualización (apt update)."
```

```
    # Envía el correo con el error, incluyendo el contenido del .log hasta ese punto
    echo -e "Subject: ❌ ERROR: Fallo actualización (apt update)\n\nFalló la actualización de paquetes. Ver log adjunto." | msmtp --file="$MSMTP_CONFIG" "$DESTINATARIO"
    exit 1
fi

echo "⬆️ Actualizando paquetes instalados..."
if ! apt upgrade -y; then
    echo "❌ ERROR: Fallo actualización (apt upgrade)."
    # Envía el correo con el error, incluyendo el contenido del .log hasta ese punto
    echo -e "Subject: ❌ ERROR: Fallo actualización (apt upgrade)\n\nFalló la actualización de paquetes instalados. Ver log adjunto." | msmtp --file="$MSMTP_CONFIG" "$DESTINATARIO"
    exit 1
fi

echo "🧹 Limpiando paquetes innecesarios..."
apt autoremove -y
apt autoclean

echo "---Fin de la actualización del sistema---"

# Notificación de éxito (esta notificación se envía por correo, el contenido de la salida de apt ya está en el log)
echo -e "Subject: ✅ Sistema actualizado\n\nEl sistema se actualizó correctamente el $(date)" | msmtp --file="$MSMTP_CONFIG" "$DESTINATARIO"

echo "🔎 Resumen de advertencias (Warnings) del día:"
# Asegúrate de que el grep busque en el .log actual, y que la salida también vaya al .log (gracias a exec > >(tee...))
grep -i "Warning" "$LOG_FILE" | tail -n 5 || echo "✔ Sin advertencias importantes."

exit 0 # Asegura una salida exitosa si todo va bien
```

✅ 7.🔄 Modifica el comando de con y quítale la segunda parte

crontab -e

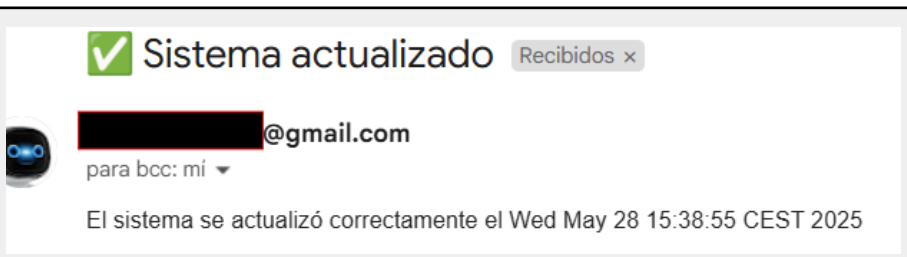
Se abrirá un archivo, tienes que añadir el código:

0 3 * * 1 /home/jose/scripts/actualizarSistema.sh

✅**8. 🧐Válida, si todo funciono, ejecutamos manualmente el archivo "*actualizarSistema.sh*"**

sudo bash /home/jose/scripts/actualizarSistema.sh

Tendrías que recibir un correo como este



👀Podrás ver los logs diarios en (***/home/jose/scripts/logs***)

cat /home/jose/scripts/logs/actualizar_sistema_$(date +%F).log

✅**9. 🗑️Limpia los logs viejos "*actualizarSistema.log*" situados en (*/var/log*)**

Entra en el directorio log:

cd /var/log

Válida si existe:

ls

Si ves "*actualizarSistema.log*" bórralo

sudo rm -r actualizarSistema.log

- **🐳Instalar Docker**

Instalar Docker en Ubuntu

⚠️Asegúrate de haber instalado los paquetes especificados en**:**

🔗 • 🛡️**Seguridad**

✅1. Añadir la clave GPG oficial de Docker:

1. Actualizo.

sudo apt update

2. Instalación de software de terceros como Docker

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

3. Creo el directorio keyrings

sudo mkdir -p /etc/apt/keyrings

4. Validamos que se creó correctamente en la ruta correcta

ls -d /etc/apt/keyrings

Si existe mostrará algo como esto

```
jose@vmi2595990:~$ ls -d /etc/apt/keyrings
/etc/apt/keyrings
```

5. Descarga la clave pública GPG de Docker de su sitio web de forma silenciosa yguarda esa clave binaria en el archivo ***/etc/apt/keyrings/docker.gpg***

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/d

---

✅2. Añadir el repositorio oficial de Docker:

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | \
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

✅3. Instalar Docker Engine:

sudo apt update

sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

---

✅4. Añadir tu usuario jose al grupo docker

sudo usermod -aG docker jose

🔴 Muy importante: debes cerrar sesión y volver a entrar (o reiniciar) para que el grupo se aplique correctamente.

sudo reboot

Comprobar que estás en el grupo con:

groups

Debe aparecer algo como:

```
jose@vmi2595990:~$ groups
jose sudo users docker
```

✅5. Validar instalación de Docker

docker run hello-world

Si ves un mensaje de bienvenida, ¡Docker está funcionando!

```
jose@vmi2595990:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:dd01f97f252193ae3210da231b1dca0c[redacted]
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

- ⚓ **Instalar Portainer**

❗ Con el usuario jose (no root) creamos.

✅**1. 📂Entramos al directorio services, si no existe lo creamos (*/home/jose/servers*)**

mkdir services

✅**2. 📂Crear el directorio 'portainer'**

mkdir portainer

✅**3.📝Entramos en el nuevo directorio "*portainer*" y creamos el manifiesto**

nano docker-compose.yaml

Contenido:

```
services:
  portainer:
    image: portainer/portainer-ce:latest
    ports:
      - 9443:9443   # HTTPS versión nueva
#     - 9000:9000 # Versión vieja
```

```
#     - 8000_8000 # HTTPS vieja
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data
    restart: unless-stopped

volumes:
  portainer_data:
```

ℹ️ control + x para salir y guardar

✅4. 🚀Dentro del directorio "*portainer*" levantamos el contenedor

❗ Estar en el directorio portainer

docker compose up -d

✅5.👀Validar que funciona

ℹ️Vamos a nuestro navegador y ponemos https://ipServidor:9443

Ejemplo:

🔗 https://123.123.123.123:9443

- ### 👮Nginx proxy manager, controlador de puertos y tráfico

❗ Con el usuario jose (no root).

⚠️ Si no has creado un subdominio que apunte la IP de tu servidor para esta parte, hazlo ahora.

ℹ️ Entramos al directorio 'services' **(*/home/jose/servers*)**

✅1.📂 Crear el directorio nginx

mkdir nginx

✅2.📝**Entramos en el directorio "*nginx*" y creamos manifiesto *"docker-compose.yaml"***

nano docker-compose.yaml

Contenido

```
services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    container_name: nginxproxymanager
    restart: unless-stopped
    ports:
      - '80:80' #🌐 HTTP normal (sin cifrar)
      - '81:81' #🔒 HTTPS
      - '443:443' # 🧠 Interfaz web de Nginx Proxy Manager (panel de control)
    volumes:
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt
    networks: # ✅ Aquí le dices a qué red debe conectarse este contenedor
      - proxiable

networks: # 🔁 redes disponibles
  proxiable:
    name: proxiable
```

ℹ️ control + x para salir y guardar

✅3.🚀**Dentro del directorio "*nginx*" levantamos el contenedor**

docker compose up -d

✅5.🌐Entramos a su interfaz web

ℹ️Vamos a nuestro navegador y ponemos ***http://ip.de.tu.servidor:81/login***

Ejemplo:

🔗 **http://123.123.123.1231:81/login**

✅6.🔒Entramos por primera vez y modificamos el user y la contraseña

⚠️El usuario y la contraseña al iniciar sesión por primera vez es

Email: admin@example.com

Password: changeme

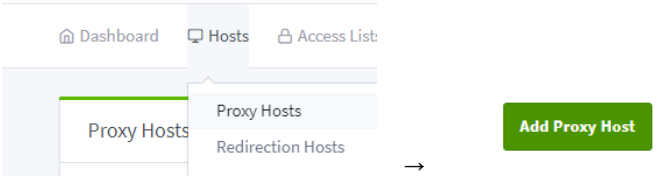
✅7. 🔎Validar que funciona correctamente

ℹ️Ahora tenemos la interfaz web por http y vamos a ponerla por https junto con un certificado.

✅7.1.🔎 Validamos que nuestro subdominio apunte a la IP de nuestro servidor

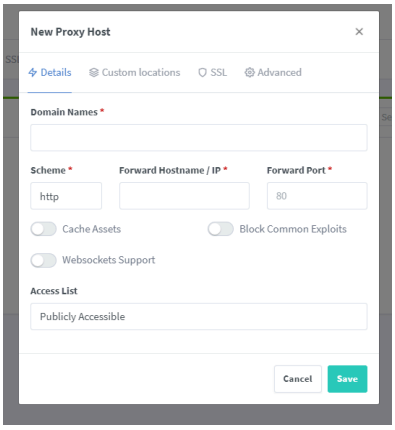
---

✅7.2.🌐 Entramos en la interfaz web de nginx y añadimos un nuevo host



→

Nos saldrá esto



✅7.3. El 'Domain Names' será el subdominio que hemos creado antes

Domain Names * → subdominio-creado**.es**

Ejemplo: nginx.jose.es

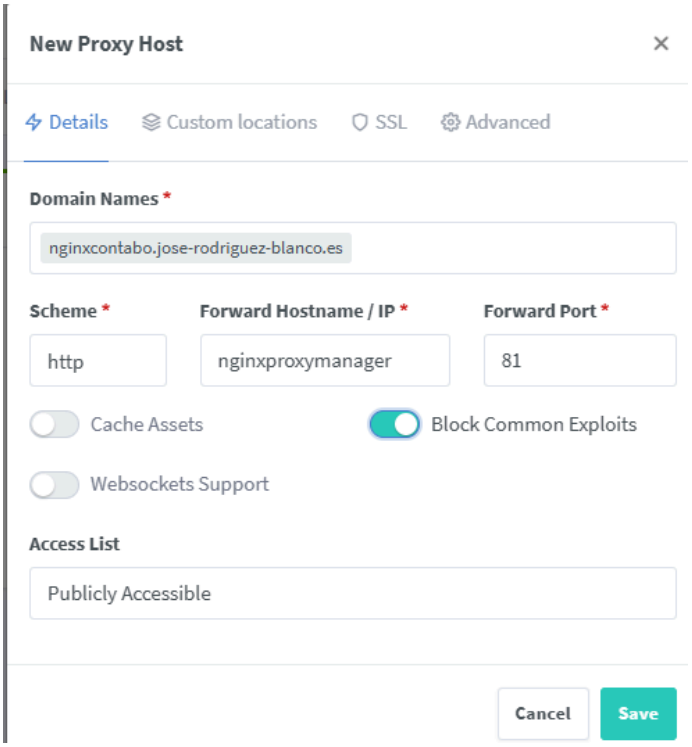
✅7.4. Forward Hostmane /I P será el nombre definido en container_name que tenemos en el archivo *"docker-compose.yaml"*

Forward Hostname / IP* → **nginxproxymanager**

✅7.5. Forward Port será el puerto https por donde entrara

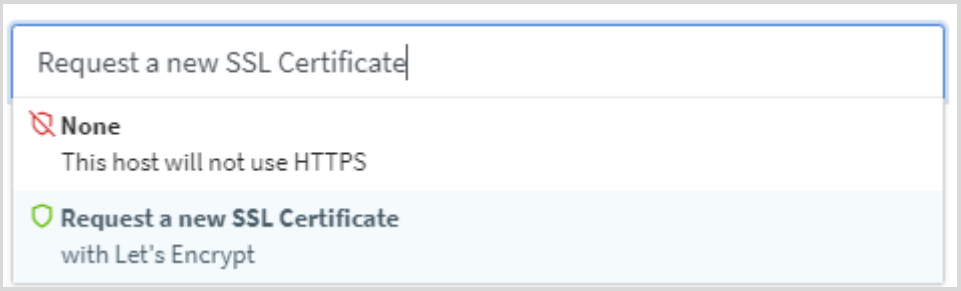
Forward Port * → 81

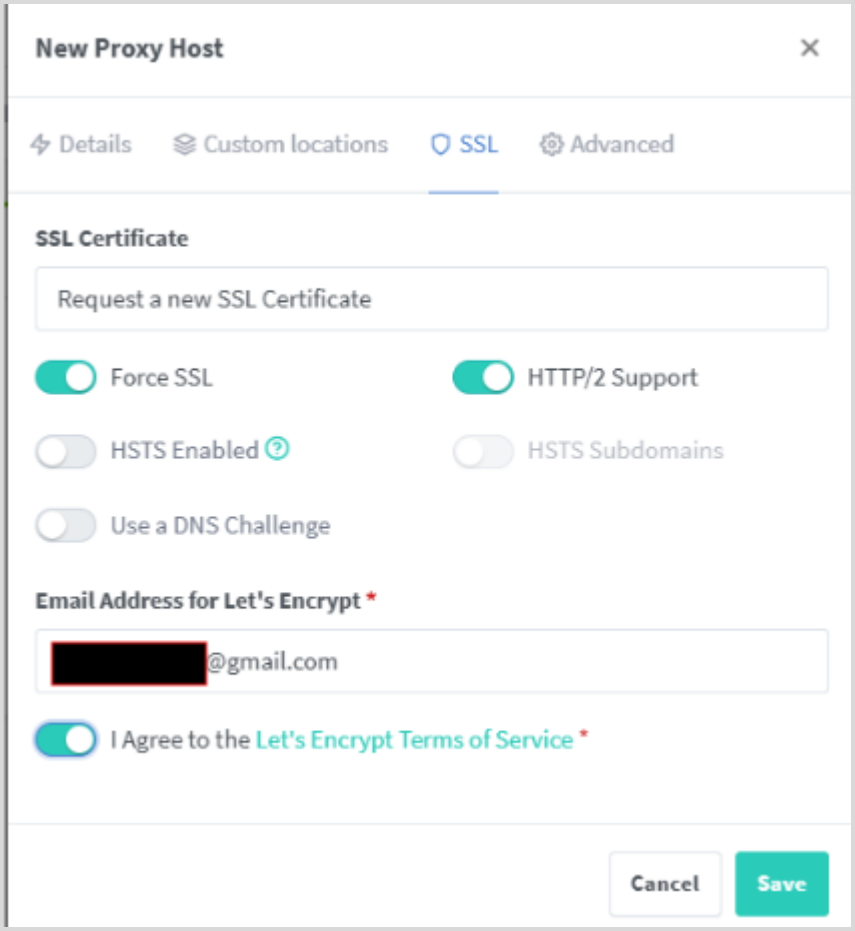
clicar las siguientes pestañas



✅7.6. Ahora presionamos el aparato SSL

En la ventana superior presionaremos sobre la verde → request a new ssl certificate y clicas todas las pestañas que se muestran en la foto





🥳🥳**Ya tenemos nuestros servicios por HTTP y HTPPS**🥳🥳

- ☁️**Subir un contenedor test para una web hola mundo**

❗Con el usuario jose (no root)

⚠️ Si no has creado un subdominio que apunte la IP de tu servidor, hazlo ahora.

ℹ️ Entramos al directorio 'services' (***/home/jose/servers***)

✅**1.** 🗂️**Crear el directorio "*testweb*"**

mkdir testweb

✅**2.** 📝**Entramos en nuevo directorio "*testweb*" y dentro creamos manifiesto**

"docker-compose.yaml"

nano docker-compose.yaml

Contenido;

```
services:
  testweb:
    image: nginx:alpine
    container_name: testweb
# Si el contenedor falla o se reinicia el servidor, se vuelve a levantar automáticamente
    restart: unless-stopped
    volumes:
# Monta la carpeta 'www' en el contenedor como carpeta web
      - ./www:/usr/share/nginx/html
      - ./images:/usr/share/nginx/html/img
    expose:
      - "80"
    networks:
# Se conecta a una red externa llamada 'proxiable' (donde está también NPM)
      - proxiable

networks:
  proxiable:
# Indica que esta red ya existe (no la creamos aquí)
    external: true
```

✅3. 🗂️Creamos otro directorio llamado www

mkdir www

✅4. 📝Entramos en nuevo directorio "*www*" y creamos un html de pruebas

Contenido:

```
<!DOCTYPE html>
  <html>
   <head>
     <title>Test Web</title>
   </head>
   <body>
    <h1>Hola mundo desde Docker y Nginx</h1>
   </body>
</html>
```

✅ 5. 🚀Lanza el contenedor

ℹ️Nos situamos en el directorio anterior " *testweb"* situado en (*/home/jose/servers/testweb)*

cd /home/jose/servers/testweb

Levantamos

docker compose up -d

Si todo ha ido bien, tendría que aparecer algo así:

```
jose@vmi2595990:~/servers/testweb$ docker compose up -d
[+] Running 1/1
 ✓ Container testweb  Started
```

💡**Información extra:** Comprueba si el contenedor ***testweb*** se comunica

🔴**(comando + explicación):**

docker network inspect proxiable

Luego, **busca que aparezcan los contenedores esperados**, por ejemplo:

"Name": "testweb"

"Name": "nginxproxymanager"

Esto **confirma que ambos están en la misma red Docker** y se pueden comunicar por nombre de contenedor.

✅ 6. 🔓 Abre los puertos 80 y 443 desde tu servidor

comprobar que el firewall no lo esté bloqueando:

sudo ufw status

Te muestra algo como esto:

```
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
8200/tcp                   ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
8200/tcp (v6)              ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)
```

Si no lo tienes abierto puedes abrirlo con:

sudo ufw allow 80/tcp

💡 Información extra: 🌍Verifica si tu dominio está accesible desde dentro de tu servidor y desde fuera

Desde la terminal de tu servidor

ping tu-dominio.es

Te muestra algo así:

```
jose@vmi2595990:~/servers/testweb$ ping test.jose-rodriguez-blanco.es
PING test.jose-rodriguez-blanco.es (62.171.171.137) 56(84) bytes of data.
64 bytes from vmi2595990.contaboserver.net (62.171.171.137): icmp_seq=1 ttl=64 time=0.143 ms
^C
--- test.jose-rodriguez-blanco.es ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.143/0.143/0.143/0.000 ms
```

Desde la terminal de tu computadora (fuera del servidor)

ping tu-dominio.es

Te muestra algo así:

```
PS C:\Users\osoho> ping test.jose-rodriguez-blanco.es

Haciendo ping a test.jose-rodriguez-blanco.es [62.171.171.137] con 32 bytes de datos:
Respuesta desde 62.171.171.137: bytes=32 tiempo=42ms TTL=52
Respuesta desde 62.171.171.137: bytes=32 tiempo=43ms TTL=52
Respuesta desde 62.171.171.137: bytes=32 tiempo=44ms TTL=52
Respuesta desde 62.171.171.137: bytes=32 tiempo=43ms TTL=52

Estadísticas de ping para 62.171.171.137:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 42ms, Máximo = 44ms, Media = 43ms
```

- 🤖 💾**Backups automáticos Duplicati**

Contenedor para gestionar el backup automático del servidor o partes de ella

ℹ️Info del contenedor:

🔗 **linuxserver/duplicati - Docker Image | Docker Hub**

❗ Con el usuario jose (no root)

ℹ️ Entramos al directorio 'services' **(*/home/jose/servers*)**

✅**1.📂 Crear el directorio "*duplicati"***

mkdir duplicati

✅**2.📝Entramos en el directorio"*duplicati"* y dentro creamos manifiesto *"docker-compose.yaml"***

ℹ️ Entramos al directorio ***"duplicati"***

nano docker-compose.yaml

Contenido

```yaml
services:
  duplicati:
    image: duplicati/duplicati:latest
    container_name: duplicati
    volumes:
      - ./duplicati-data:/data   # Configuración de duplicati
      - /home/jose:/source    # Carpeta origen (lo que quieres guardar)
      - ./backups:/backups    # Carpeta destino de backups  en el mismo servidor si así lo deseamos
    environment:
      # - PUID=?¿         # ID de tu usuario  `id -u`
      # - PGID=?¿         # ID de grupo: `id -g`
      # - TZ=Europe/Madrid
      - SETTINGS_ENCRYPTION_KEY=CLAVE_DE_32_DIGITOS # ❗ clave de cifrado para su base de datos de configuración tiene que tener 32 de longitud
      - DUPLICATI_WEBSERVICE_PASSWORD=tuContraseñaAqui #contraseña de la interfaz web
    ports:
      - 8200:8200                    # 🌐Interfaz web
    restart: unless-stopped
```

ℹ️ control + x para salir y guardar

✅3.🚀Levantamos el contenedor

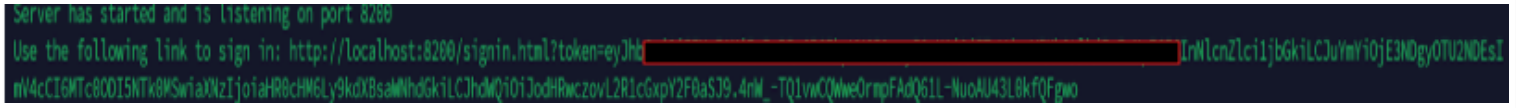
❗ Estar en el directorio ***duplicati*** (***/home/jose/servers/duplicati***)

docker compose up -d

✅4.🔑Vemos los logs para guardar el token del primer acceso

docker logs duplicati

Saldrá algo del estilo:



Guarda todo el token despues de token=....

✅5.🌐Entramos a su interfaz web

ℹ️Vamos a nuestro navegador y ponemos ***http://ip.de.tu.servidor:8200***

Ejemplo:

🔗http://123.123.123.123***:8200/signin.html?token=***

Nos pedirá el token, pegamos el token guardado en el paso 4.
Al entrar por primera vez nos pedirá una nueva contraseña, la ponemos y la guardamos bien que no se pierda.

✅6.🔐Una vez validado que funciona todo, vamos a proteger el token y la contraseña para que esté menos expuesta.

Crearemos un archivo oculto, lo protegeremos para que solo se pueda leer y modificar con "sudo" y desde el .yml leeremos ese archivo

Creamos un archivo ***.env***

nano .en

✅7.🔒Añadimos las environment que queremos ocultar

```
# ID de tu usuario  `id -u`
PUID=?¿
# ID de grupo: `id -g`
PGID=?¿
TZ=Europe/Madrid
 # ❗ clave de cifrado para su base de datos de configuración tiene que tener 32 de longitud
SETTINGS_ENCRYPTION_KEY=tu_key_de_32_longitud
#contraseña de la interfaz web
DUPLICATI_WEBSERVICE_PASSWORD=tuContraseñaAqui
```

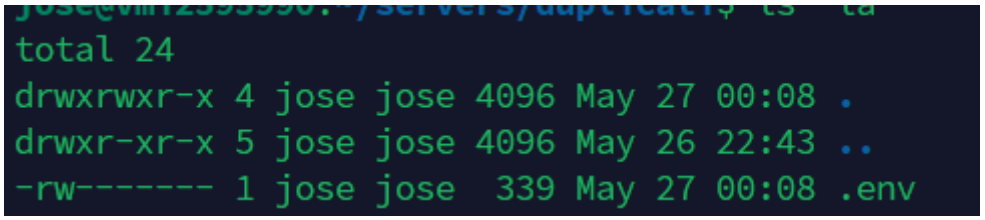
ℹ️ control + x para salir y guardar

✅7. 🔒Modificamos los permisos del archivo .env

chmod 600 .env

Al validar tendría que salir algo del estilo

ls -la



✅8.📝Modificamos el .yml para que llame al .env y saque la información protegida

Para buscar esta info el .env tiene que estar en el mismo nivel que el .yml. Dentro del .yml usaremos las variables ${nombre_en_el_.env}

ℹ️Nos situamos en el directorio **"*duplicati" (/home/jose/servers/duplicati)***

Bajamos el contenedor

docker compose down

Ejemplo:

```
environment:
 # Referenciamos las variables del .env con ${}
   - SETTINGS_ENCRYPTION_KEY=${SETTINGS_ENCRYPTION_KEY} # ❗ clave de cifrado
   - DUPLICATI_WEBSERVICE_PASSWORD=${DUPLICATI_WEBSERVICE_PASSWORD}
```

En el directorio **"*duplicati" (/home/jose/servers/duplicati).* Subimos el contenedor**

docker compose up -d

✅9. Validamos que siga funcionando todo en la interfaz web

Ejemplo:

🔗http://ip.de.tu.servidor:8200/ngax/index.html

- **ℹ️Como crear una copia de backup automático con Duplicati**

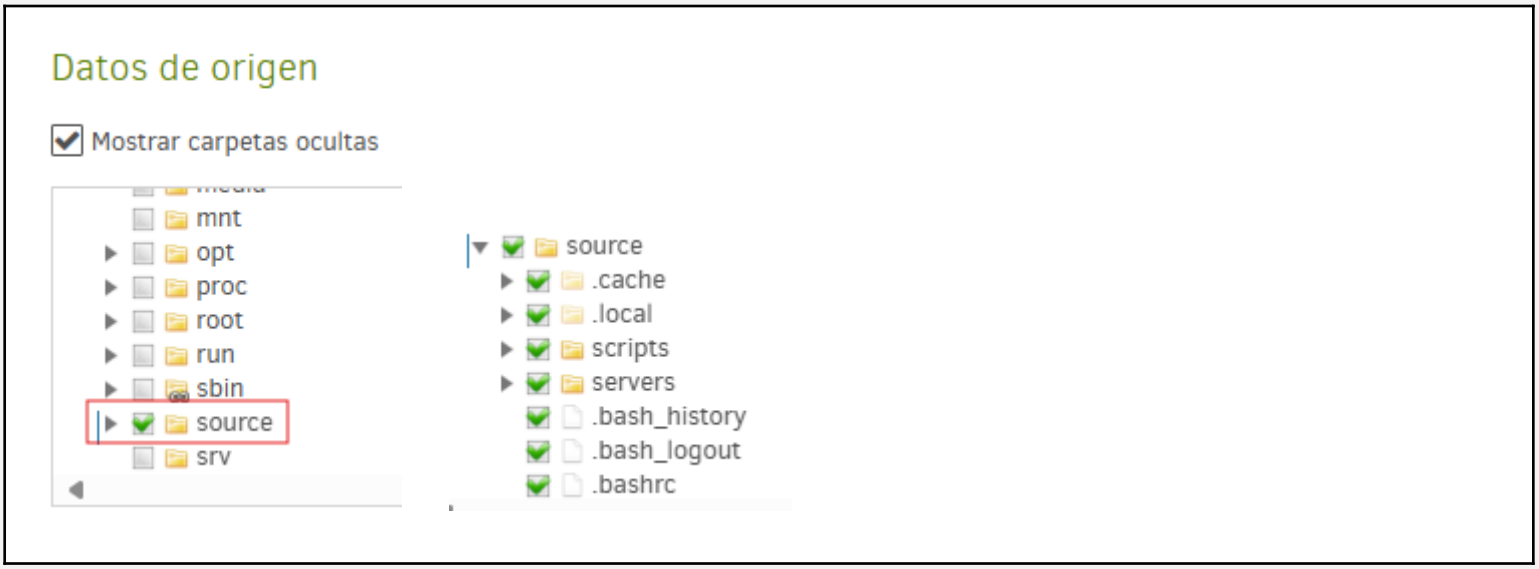
❗ ℹ️ Tutorial a partir del minuto 5:

🔗 **[Backup FÁCIL de Docker con Duplicati: Home Assistant y Node Red SEGUROS]**

✅1. ❗ En el paso 3 no encuentro el directorio que añadí en mi .yml que es el directorio que quiero hacer el backup. ¿Dónde está?

Ves esta pantalla:

ℹ️Para encontrar el directorio que marcaste en tú .yml tienes que desplegar "ordenador" y buscar el directorio ***source***, al desplegarlo verás todo lo que tendría que haber en tu servidor.



- 🕵️**GoAccess analizador de registros del servidor web**

Puedes segur el tutorial paso a paso de como montar este contenedor en:

🔗🕵️**GoAccess Dockerizado**🐋

- 📚Bibliografía

- YouTube
- GitHub
- Docker Docs
- Docker Hub Container Image Library | App Containerization
- Solvetic - Solución a los problemas informáticos
- Nginx Proxy Manager
- Duplicati
- GoAccess - Visual Web Log Analyzer
- 🐢 EmojiTerra 🌍 | Emojis Copiar & Pegar

- IA utilizadas

- ChatGPT: Preguntas y razonamiento profundo.
- Gemini: Preguntas genéricas y fuerza bruta.

- Claude: Mejora de la documentación.