EBENEZER IT · GUÍAS PASO A PASO

GUÍA 03 · EPISODIOS 8 Y 9

Docker, Docker Compose y Nginx como proxy inverso

Instala Docker desde su repositorio oficial, entiende cómo funcionan imágenes, contenedores y volúmenes, evita la trampa de Docker con el cortafuegos y publica tu primera aplicación en su propio subdominio con HTTPS.

~2,5 horas
la primera vez, sin prisa

Requisito
Guías 01 y 02 terminadas

Ubuntu 24.04 LTS
Docker CE · Compose v2 · Nginx

www.ebenezer.org.es · Alejandro Quevedo · Versión pública 1.0 · noviembre 2026

Índice

Antes de empezar

0 Qué vas a construir y valores de esta guía

Parte A · Docker

Paso 01 Qué es Docker y por qué lo usamos

Paso 02 Quitar versiones antiguas o no oficiales

Paso 03 Añadir el repositorio oficial de Docker

Paso 04 Instalar Docker y probarlo

Paso 05 Usar Docker sin sudo (y lo que implica)

Paso 06 Limitar el tamaño de los registros

Paso 07 La trampa de Docker y el cortafuegos

Paso 08 Los comandos que vas a usar a diario

Parte B · Docker Compose

Paso 09 Dónde viven tus aplicaciones

Paso 10 Tu primer compose.yaml

Paso 11 Fijar la versión de la imagen

Parte C · Nginx como proxy inverso

Paso 12 Qué es un proxy inverso

Paso 13 Ajustes comunes de proxy

Paso 14 Subdominio y certificado

Paso 15 Publicar la aplicación con HTTPS

Paso 16 Actualizar un contenedor

Paso 17 Comprobación final e instantánea

Anexos

A Problemas frecuentes

B Puertos abiertos al terminar esta guía

C Chuleta de Docker y Compose

Cómo usar esta guía

Tienes aquí **todos los pasos y comandos**, en orden. La explicación de cada paso (el porqué de cada línea) está en el **vídeo del episodio**.

Para copiar los comandos sin errores, usa la versión web: **https://www.ebenezer.org.es/guia-03-docker-proxy.html**. Al copiar desde un PDF se pueden romper los saltos de línea y las comillas.

ANTES DE EMPEZAR

Qué vas a construir

Al terminar tendrás Docker instalado y una aplicación real, **IT-Tools** (una caja de herramientas web para técnicos: generador de contraseñas, conversor de formatos, calculadora de subredes…), funcionando en un contenedor y publicada en `https://herramientas.TU_DOMINIO` a través del Nginx de la Guía 02.

Es el mismo esquema que usarás en el resto del curso para WordPress, Uptime Kuma, Grafana o la VPN:

Pieza	Papel
Contenedor	La aplicación, aislada, escuchando **solo dentro del servidor** (en `127.0.0.1`). Desde Internet no se puede llegar a ella directamente.
Nginx	La única puerta de entrada (puertos 80 y 443). Pone el HTTPS y reenvía cada subdominio al contenedor que le toca.
Docker Compose	Un archivo de texto (`compose.yaml`) que describe cada aplicación: qué imagen, qué puertos, qué datos guardar. Con él, montar o rehacer una aplicación es un solo comando.

El recorrido de una visita

1. El navegador pide `https://herramientas.TU_DOMINIO`. El DNS responde `TU_IP`.
2. Nginx recibe la conexión en el 443, presenta el certificado de ese subdominio y descifra la petición.
3. Nginx la reenvía **por dentro del servidor** a `http://127.0.0.1:8081`, donde escucha el contenedor.
4. El contenedor responde a Nginx, y Nginx se lo devuelve cifrado al navegador.

Valores de esta guía

Valor	Qué es
`TU_DOMINIO`	Tu dominio, como en las guías anteriores
`TU_IP`	La IP del VPS
`TU_USUARIO`	Tu usuario del servidor
`TU_VERSION`	La versión exacta de la imagen de IT-Tools (la buscarás en el paso 11)

Recordatorio

- Haz una instantánea en IONOS antes de empezar (`02-nginx-https`).
- Bloques **verdes** en el servidor (`ssh lab`), **azules** en PowerShell de tu PC, **morados** son contenido de archivos.
- `sudo nginx -t` siempre antes de recargar Nginx.

PARTE A

Docker

Instalamos Docker como recomienda el propio proyecto, entendemos sus piezas y dejamos resuelto desde el principio el fallo de seguridad más típico de los servidores con Docker. Corresponde al **episodio 8**.

Paso 1 Episodio 8

Qué es Docker y por qué lo usamos

Por qué: muchas aplicaciones necesitan versiones concretas de un lenguaje, librerías y ajustes que chocan entre sí.

Concepto	Qué es	Comparación
Imagen	Una plantilla de solo lectura con la aplicación y todo lo que necesita. La publica el autor (en Docker Hub, normalmente)	El instalador de un programa
Contenedor	Una imagen en marcha. Puedes tener varios contenedores de la misma imagen	El programa abierto
Volumen	Una carpeta que sobrevive al contenedor. Ahí van los datos (bases de datos, archivos subidos)	Tu carpeta de Documentos: si reinstalas el programa, no se borra
Puerto publicado	Conecta un puerto del servidor con un puerto del contenedor	Un tubo entre la calle y una habitación
Red de Docker	Una red privada interna por la que los contenedores hablan entre sí	La red de la oficina
Compose	Un archivo que describe uno o varios contenedores y cómo se conectan	La receta

Contenedor no es lo mismo que máquina virtual

Una máquina virtual lleva un sistema operativo completo dentro. Un contenedor comparte el núcleo (kernel) de Ubuntu y solo aísla la aplicación: arranca en segundos y gasta muy poca memoria. A cambio, el aislamiento es menor que el de una máquina virtual, y por eso importa cómo lo configures (pasos 5 y 7).

Comprueba antes de seguir

Sabes explicar con tus palabras la diferencia entre imagen, contenedor y volumen. Lo vas a necesitar en el vídeo.

Paso 2 Episodio 8

Quitar versiones antiguas o no oficiales

Por qué: Ubuntu tiene sus propios paquetes de Docker (`docker.io`), pero van con retraso y no incluyen Docker Compose v2.

EN EL SERVIDOR

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc;
do
    sudo apt-get remove -y $pkg
done
```

Comprueba antes de seguir

Termina sin errores. Los avisos de «not installed» son normales.

Paso 3 Episodio 8

Añadir el repositorio oficial de Docker

Por qué: vamos a instalar Docker desde los servidores de Docker, no desde los de Ubuntu.

EN EL SERVIDOR

```
sudo apt-get update
sudo apt-get install -y ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Ahora le decimos a `apt` dónde está el repositorio. **Es un solo comando**, partido en líneas con `\` al final para que se lea mejor:

EN EL SERVIDOR

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \
https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME")
stable" \
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
cat /etc/apt/sources.list.d/docker.list
```

SALIDA ESPERADA

```
deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu noble stable
```

Comprueba antes de seguir

El archivo contiene `arch=amd64` y `noble`. Si ves `arch=` o un hueco vacío donde debería ir `noble`, el comando se partió mal al copiarlo: bórralo con `sudo rm /etc/apt/sources.list.d/docker.list` y repítelo.

Paso 4 Episodio 8

Instalar Docker y probarlo

Por qué: con el repositorio añadido, `apt` ya puede instalar Docker y lo mantendrá actualizado como cualquier otro paquete.

EN EL SERVIDOR

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

Comprueba que el servicio funciona y arranca solo, y haz la prueba clásica:

EN EL SERVIDOR

```
systemctl is-enabled docker containerd
sudo docker run --rm hello-world
docker compose version
```

SALIDA ESPERADA

```
enabled
enabled
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
...
Hello from Docker!
This message shows that your installation appears to be working correctly.
...
Docker Compose version v2.x.x
```

Comprueba antes de seguir

Ves **Hello from Docker!** y la versión de Compose empieza por `v2`. Si `docker compose version` dice `command not found`, falta el paquete `docker-compose-plugin`.

Paso 5 Episodio 8

Usar Docker sin sudo (y lo que implica)

Por qué: escribir `sudo` delante de cada comando de Docker es incómodo.

Estar en el grupo docker equivale a ser root

Quien puede dar órdenes a Docker puede arrancar un contenedor que monte `/` (todo el disco del servidor) y modificar cualquier archivo, sin que le pidan contraseña. Por tanto, meter a un usuario en el grupo `docker` es darle permisos de administrador sin `sudo`.

En tu servidor eso es aceptable porque **tu usuario ya es administrador** y solo se puede entrar con tu clave SSH (Guía 01). Lo que nunca debes hacer es meter en el grupo `docker` a un usuario que no deba ser administrador.

EN EL SERVIDOR

```
sudo usermod -aG docker TU_USUARIO
exit
```

Vuelve a entrar con `ssh lab` y prueba **sin** sudo:

EN EL SERVIDOR

```
groups
docker ps
```

SALIDA ESPERADA

```
TU_USUARIO sudo users docker
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Comprueba antes de seguir

`groups` incluye `docker` y `docker ps` responde con la cabecera de la tabla (vacía) en vez de `permission denied`.

Paso 6 Episodio 8

Limitar el tamaño de los registros

Por qué: por defecto, Docker guarda todo lo que escribe cada contenedor en un registro que **crece sin límite**.

EN EL SERVIDOR

```
sudo tee /etc/docker/daemon.json > /dev/null <<'EOF'
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3"
  }
}
EOF
sudo systemctl restart docker
docker info --format '{{.LoggingDriver}}'
```

Solo afecta a los contenedores nuevos

Los contenedores que ya existían conservan su configuración de registros hasta que se vuelven a crear (`docker compose up -d` después de un cambio, o `docker compose down` y `up -d`). En un servidor recién instalado no hay ninguno, así que no te afecta.

Si Docker no arranca después de esto

Casi seguro es un error de sintaxis en el JSON. Mira el motivo con `sudo journalctl -u docker -n 20 --no-pager`, corrige el archivo y vuelve a reiniciar.

Comprueba antes de seguir

`docker info --format ...` responde `json-file` y `systemctl is-active docker` dice `active`.

Paso 7 Episodio 8

La trampa de Docker y el cortafuegos

Por qué: este es el error de seguridad más común en servidores con Docker, y conviene entenderlo bien.

Por qué pasa

UFW y Docker escriben reglas en el mismo cortafuegos del núcleo (iptables/nftables), pero en sitios distintos. El tráfico hacia un contenedor se desvía (se «reenvía») **antes** de llegar a las reglas de UFW, así que UFW nunca llega a verlo.

Compruébalo tú mismo (y por qué hoy no te pasa nada)

EN EL SERVIDOR

```
docker run -d --name prueba -p 8080:80 nginx:alpine
```

EN EL SERVIDOR

```
sudo ufw status | grep 8080 || echo "UFW no tiene ninguna regla para el 8080"
sudo ss -tlpn | grep 8080
```

UFW no tiene regla para el 8080, y sin embargo `ss` muestra que algo escucha en `0.0.0.0:8080`. Ahora, desde tu PC:

EN TU PC · POWERSHELL

```
curl.exe -I --max-time 5 http://TU_IP:8080
```

Fallará por tiempo de espera. **No es UFW quien te protege**: es el **firewall de IONOS**, que solo deja pasar 22, 80 y 443. Esta es exactamente la razón de tener dos capas (Guía 01, paso 16). Si un día abres por error el 8080 en el panel de IONOS, ese contenedor queda expuesto aunque UFW diga que no.

Borra el contenedor de prueba:

EN EL SERVIDOR

```
docker rm -f prueba
```

La regla del curso

Publica los puertos siempre en 127.0.0.1

En todos los contenedores del curso escribiremos el puerto así: `127.0.0.1:8081:80`. La primera parte es la dirección del servidor donde escucha. `127.0.0.1` es la dirección interna, la que solo es accesible desde el propio servidor. Así el contenedor **no es accesible desde fuera de ninguna manera**, pase lo que pase con los cortafuegos, y la única forma de llegar a él es a través de Nginx, que pone el HTTPS y los controles.

Cómo se escribe	Escucha en	Accesible desde Internet
`-p 8080:80`	`0.0.0.0:8080` (todas)	**Sí**, si el firewall de IONOS lo permite. UFW no lo frena
`-p 127.0.0.1:8080:80`	`127.0.0.1:8080`	**No**. Solo desde el propio servidor (Nginx)

La excepción: servicios que no son web

Algunos servicios no pueden ir detrás de Nginx porque no son HTTP, como la VPN WireGuard (UDP 51820). Esos se publican en todas las direcciones a propósito, y se abren de forma consciente en IONOS. Lo veremos en su episodio.

Comprueba antes de seguir

`docker ps -a` ya no muestra el contenedor `prueba`, y sabes explicar por qué UFW no protege los puertos de Docker.

Paso 8 Episodio 8

Los comandos que vas a usar a diario

Por qué: con estos comandos manejarás el 95 % de tu día a día con Docker.

EN EL SERVIDOR

```
docker run -d --name demo nginx:alpine
docker ps
docker logs demo
docker exec -it demo sh
```

El último comando te mete **dentro** del contenedor: el cursor cambia a `/ #`. Escribe `ls /usr/share/nginx/html` para ver sus archivos y `exit` para salir. Después:

EN EL SERVIDOR

```
docker stop demo
docker ps
docker ps -a
docker rm demo
docker images
docker image prune -a
```

Comando	Qué hace
`docker ps`	Lista los contenedores **en marcha**
`docker ps -a`	Lista **todos**, también los parados
`docker logs NOMBRE`	Lo que ha escrito el contenedor. Con `-f` lo sigue en directo; con `--tail 50`, solo las últimas 50 líneas
`docker exec -it NOMBRE sh`	Abre una terminal dentro del contenedor (`-i` interactivo, `-t` con terminal). Algunas imágenes tienen `bash`; las más ligeras, solo `sh`
`docker stop / start / restart NOMBRE`	Para, arranca o reinicia un contenedor
`docker rm NOMBRE`	Borra un contenedor parado (`-f` lo para y lo borra a la vez)
`docker images`	Las imágenes descargadas y lo que ocupan
`docker image prune -a`	Borra las imágenes que no usa ningún contenedor. Te pide confirmación: responde `y`

Comando	Qué hace
`docker system df`	Cuánto disco ocupa Docker en total (imágenes, contenedores, volúmenes)

Comprueba antes de seguir

`docker ps -a` está vacío y `docker images` ya no muestra `nginx:alpine` ni `hello-world`.

PARTE B

Docker Compose

En vez de recordar comandos `docker run` larguísimos, cada aplicación se describe en un archivo. Es más claro, se puede guardar en Git y rehacer la aplicación es un solo comando. Sigue siendo el **episodio 8**.

Paso 9 Episodio 8

Dónde viven tus aplicaciones

Por qué: es importante tener un orden fijo: cada aplicación en su carpeta, con su `compose.yaml` y sus datos al lado.

EN EL SERVIDOR

```
sudo mkdir -p /opt/apps
sudo chown TU_USUARIO:TU_USUARIO /opt/apps
mkdir -p /opt/apps/it-tools
cd /opt/apps/it-tools
pwd
```

La estructura que seguiremos en todo el curso:

SALIDA ESPERADA · ESTRUCTURA

```
/opt/apps/
├── it-tools/
│   └── compose.yaml
├── wordpress/          (episodio 10)
│   ├── compose.yaml
│   ├── .env            (contraseñas)
│   └── datos/          (los volúmenes)
└── uptime-kuma/        (episodio 23)
    ├── compose.yaml
    └── datos/
```

Comprueba antes de seguir

`pwd` responde `/opt/apps/it-tools`.

Paso 10 Episodio 8

Tu primer compose.yaml

Por qué: describimos IT-Tools en un archivo.

EN EL SERVIDOR

```
tee /opt/apps/it-tools/compose.yaml > /dev/null <<'EOF'
services:
  it-tools:
    image: corentinth/it-tools:latest
    container_name: it-tools
    restart: unless-stopped
    ports:
      - "127.0.0.1:8081:80"
EOF
cat /opt/apps/it-tools/compose.yaml
```

En YAML, los espacios importan

La **sangría** (los espacios al principio de cada línea) indica qué va dentro de qué. Usa siempre **espacios, nunca tabuladores**, y de dos en dos. Si un nivel está mal alineado, Compose da un error como `yaml: line 5: mapping values are not allowed`.

¿Y la línea version: '3.8' que sale en muchos tutoriales?

Ya no se usa. Compose v2 la ignora y avisa con `the attribute version is obsolete`. Si la ves en un tutorial, bórrala.

Arranca la aplicación:

EN EL SERVIDOR

```
cd /opt/apps/it-tools
docker compose up -d
docker compose ps
curl -I http://127.0.0.1:8081
```

SALIDA ESPERADA

```
NAME       IMAGE                          STATUS         PORTS
it-tools   corentinth/it-tools:latest     Up 5 seconds   127.0.0.1:8081→80/tcp

HTTP/1.1 200 OK
```

Comprueba antes de seguir

- `STATUS` dice `Up` y `PORTS` empieza por `127.0.0.1:8081`.
- `curl` responde `200 OK`.
- Desde tu PC, `curl.exe -I --max-time 5 http://TU_IP:8081` **falla**: la aplicación no es accesible desde fuera. Correcto.

Paso 11 Episodio 8

Fijar la versión de la imagen

Por qué: con `latest`, cada vez que actualices descargas «lo último», sea lo que sea.

11.1 · Busca la versión

1. Abre en el navegador `hub.docker.com/r/corentinth/it-tools/tags`.
2. Verás una lista de etiquetas. Ignora `latest` y fíjate en la más reciente con número, del estilo `2024.10.22-7ca5933` (fecha y un código).
3. Apúntala en la tabla como **TU_VERSION**.

> **En otras imágenes, el mismo método**
>
> Cada proyecto numera sus versiones a su manera (`1.23.4`, `v2`, `2024.10.22`…). Busca siempre en la pestaña **Tags** de Docker Hub o en la página de **Releases** del proyecto en GitHub, y elige una etiqueta concreta, no `latest` ni solo el número principal (`2`).

11.2 · Cámbiala en el archivo

EN EL SERVIDOR

```
cd /opt/apps/it-tools
sed -i 's|corentinth/it-tools:latest|corentinth/it-tools:TU_VERSION|' compose.yaml
grep image compose.yaml
docker compose up -d
docker compose ps
```

> **Comprueba antes de seguir**
>
> `docker compose ps` muestra la imagen con tu versión y el estado `Up`.

PARTE C

Nginx como proxy inverso

La aplicación ya funciona, pero solo dentro del servidor. Ahora Nginx la publica en su propio subdominio con HTTPS. Este mismo esquema lo repetirás con cada aplicación del curso. Corresponde al **episodio 9**.

Paso 12 Episodio 9

Qué es un proxy inverso

Por qué: un **proxy inverso** es un intermediario que recibe las peticiones de Internet y las reenvía a la aplicación que toca.

Ventaja	Por qué importa
Un solo punto de entrada	Solo los puertos 80 y 443 abiertos, para todas las aplicaciones. Cada una se distingue por su subdominio
HTTPS en un único sitio	Los certificados y el cifrado los gestiona Nginx. Las aplicaciones no tienen que saber nada de HTTPS
Las aplicaciones quedan ocultas	Escuchan solo en 127.0.0.1. Si una tiene un fallo de seguridad, no es accesible directamente
Control centralizado	Restringir por IP, limitar peticiones o añadir cabeceras se hace en Nginx, igual para todas

Proxy «inverso» frente a proxy normal

Un proxy normal está del lado del **cliente** (por ejemplo, el de una empresa que filtra la navegación de sus empleados). El inverso está del lado del **servidor** y representa a las aplicaciones ante el mundo.

Comprueba antes de seguir

Sabes explicar por qué el contenedor escucha en `127.0.0.1` y quién pone el HTTPS.

Paso 13 Episodio 9

Ajustes comunes de proxy

Por qué: cuando Nginx reenvía una petición, la aplicación ve que viene de `127.0.0.1` (de Nginx), no del visitante real, y cree que la conexión es HTTP aunque sea HTTPS.

13.1 · Soporte de WebSocket

Muchas aplicaciones modernas (Uptime Kuma, Grafana, chats...) mantienen una conexión abierta para actualizar la pantalla en tiempo real. Eso se llama **WebSocket** y necesita una pequeña regla global:

EN EL SERVIDOR

```
sudo tee /etc/nginx/conf.d/websocket.conf > /dev/null <<'EOF'
# Ebenezer: soporte de WebSocket para las aplicaciones detrás del proxy
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}
EOF
```

13.2 · El snippet de proxy

EN EL SERVIDOR

```
sudo tee /etc/nginx/snippets/proxy.conf > /dev/null <<'EOF'
# Ebenezer: ajustes comunes para reenviar a una aplicación
proxy_http_version 1.1;
proxy_set_header Host              $host;
proxy_set_header X-Real-IP         $remote_addr;
proxy_set_header X-Forwarded-For   $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header Upgrade           $http_upgrade;
proxy_set_header Connection        $connection_upgrade;
proxy_read_timeout 60s;
EOF
sudo nginx -t
```

Comprueba antes de seguir

`sudo nginx -t` dice `syntax is ok`. Todavía no lo usa ningún sitio.

Paso 14 Episodio 9

Subdominio y certificado

Por qué: cada aplicación tendrá su subdominio.

14.1 · Registro DNS

En el panel DNS de IONOS, añade un registro **A**: nombre de host `herramientas`, apunta a `TU_IP`, TTL 1 hora. Comprueba desde tu PC a los pocos minutos:

EN TU PC · POWERSHELL

```
Resolve-DnsName herramientas.TU_DOMINIO -Server 1.1.1.1
```

14.2 · Sitio HTTP mínimo

EN EL SERVIDOR

```
sudo tee /etc/nginx/sites-available/herramientas.TU_DOMINIO > /dev/null <<'EOF'
server {
    listen 80;
    listen [::]:80;
    server_name herramientas.TU_DOMINIO;

    location ^~ /.well-known/acme-challenge/ {
        root /var/www/letsencrypt;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}
EOF
sudo ln -s /etc/nginx/sites-available/herramientas.TU_DOMINIO /etc/nginx/sites-enabled/
sudo nginx -t && sudo systemctl reload nginx
```

14.3 · El certificado

EN EL SERVIDOR

```
sudo certbot certonly --webroot -w /var/www/letsencrypt -d herramientas.TU_DOMINIO --dry-run
sudo certbot certonly --webroot -w /var/www/letsencrypt -d herramientas.TU_DOMINIO --cert-name
herramientas.TU_DOMINIO
```

Primero el ensayo y, si sale `The dry run was successful`, el de verdad.

> **¿Un certificado por subdominio o uno para todos?**
> Podrías añadir `herramientas` al certificado del dominio principal (Anexo C de la Guía 02). Preferimos uno por aplicación: si un día quitas una aplicación, borras su certificado sin tocar los demás, y si falla la renovación de uno, no afecta al resto. El script que recarga Nginx (Guía 02, paso 14) sirve para todos.

> **Comprueba antes de seguir**
> `sudo certbot certificates` muestra dos certificados: TU_DOMINIO y `herramientas.TU_DOMINIO`.

Paso 15 Episodio 9

Publicar la aplicación con HTTPS

Por qué: ahora el sitio completo: el bloque HTTP del paso anterior más un bloque HTTPS que, en vez de servir archivos de una carpeta, **reenvía todo a la aplicación**.

EN EL SERVIDOR

```
sudo tee /etc/nginx/sites-available/herramientas.TU_DOMINIO > /dev/null <<'EOF'
# HTTP: Let's Encrypt y redirección a HTTPS
server {
    listen 80;
    listen [::]:80;
    server_name herramientas.TU_DOMINIO;

    location ^~ /.well-known/acme-challenge/ {
        root /var/www/letsencrypt;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

# HTTPS: reenvío a IT-Tools (contenedor en 127.0.0.1:8081)
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name herramientas.TU_DOMINIO;

    ssl_certificate     /etc/letsencrypt/live/herramientas.TU_DOMINIO/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/herramientas.TU_DOMINIO/privkey.pem;
    include snippets/tls.conf;

    add_header Strict-Transport-Security "max-age=86400" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;

    location / {
        proxy_pass http://127.0.0.1:8081;
        include snippets/proxy.conf;
    }

    access_log /var/log/nginx/herramientas.access.log;
    error_log  /var/log/nginx/herramientas.error.log;
}
EOF
sudo nginx -t && sudo systemctl reload nginx
```

Pruébalo desde tu PC:

EN TU PC · POWERSHELL

```
curl.exe -I http://herramientas.TU_DOMINIO
curl.exe -I https://herramientas.TU_DOMINIO
```

Comprueba antes de seguir

- HTTP da `301` hacia `https://herramientas.TU_DOMINIO/`.
- HTTPS da `200`.
- En el navegador, `https://herramientas.TU_DOMINIO` abre IT-Tools con el candado. Prueba el «Token generator» o el «IPv4 subnet calculator»: funcionan dentro de tu servidor.

Esta aplicación la puede usar cualquiera

IT-Tools no tiene usuarios ni guarda datos, así que publicarla no es un riesgo. Las aplicaciones de administración (paneles, Grafana, Uptime Kuma…) **no** se publican así a todo el mundo: en el episodio 12 aprenderás a dejarlas accesibles solo a través de la VPN, añadiendo dos líneas a este mismo bloque.

Paso 16 Episodio 9

Actualizar un contenedor

Por qué: las aplicaciones en contenedores **no** se actualizan con las actualizaciones automáticas de Ubuntu: lo haces tú, cuando decidas.

1. Mira en Docker Hub (o en las *Releases* de GitHub) si hay una versión nueva y lee qué cambia, sobre todo si avisa de cambios incompatibles (*breaking changes*).
2. En aplicaciones con datos, haz antes una copia de seguridad o una instantánea.
3. Cambia la versión en `compose.yaml` y aplica:

EN EL SERVIDOR

```
cd /opt/apps/it-tools
sed -i 's|it-tools:VERSION_ANTERIOR|it-tools:VERSION_NUEVA|' compose.yaml
docker compose pull
docker compose up -d
docker compose ps
docker image prune -f
```

Si la versión nueva falla

Vuelve a poner la versión anterior en `compose.yaml` y repite `docker compose up -d`. Si ya borraste la imagen vieja, Docker la descargará otra vez. Por esto `latest` es mala idea: con `latest` no sabrías ni a qué versión volver.

needrestart y Docker

Cuando actualices Ubuntu y salga la pantalla morada de **needrestart**, **no marques el servicio docker**: reiniciarlo para y arranca todos los contenedores. Las actualizaciones del propio Docker (paquetes `docker-ce`, `containerd.io`) ya reinician lo necesario por su cuenta.

Comprueba antes de seguir

Sabes cómo actualizar y cómo volver atrás. Con IT-Tools en su última versión no hace falta hacerlo ahora.

Paso 17 Episodio 9

Comprobación final e instantánea

Por qué: revisamos que todo está como debe, que sobrevive a un reinicio y guardamos el punto de partida de la siguiente guía.

EN EL SERVIDOR

```
systemctl is-enabled docker containerd
docker compose -f /opt/apps/it-tools/compose.yaml ps
sudo ss -tlpn | grep -E ':(80|443|8081) '
sudo nginx -t
sudo certbot renew --dry-run
```

Comprobación	Resultado correcto
Docker al arrancar	`enabled` y `enabled`
IT-Tools	`Up`, puerto `127.0.0.1:8081→80/tcp`
Puertos	80 y 443 en `0.0.0.0` y `[::]` (nginx); 8081 **solo** en `127.0.0.1` (docker-proxy)
Nginx	`syntax is ok` sin avisos
Renovación	Los **dos** certificados en `succeeded`
Desde tu PC	`https://herramientas.TU_DOMINIO` funciona; `http://TU_IP:8081` no

Reinicia el servidor y, al volver a entrar, comprueba que IT-Tools ha arrancado sola:

EN EL SERVIDOR

```
sudo reboot
# ...espera un minuto, ssh lab, y:
docker ps
```

Después, en IONOS, crea la instantánea `03-docker-proxy`.

Comprueba antes de seguir

Ya sabes publicar cualquier aplicación en contenedor con su subdominio y HTTPS. La siguiente guía (episodio 10) monta WordPress con Docker Compose detrás de este proxy, con base de datos, volúmenes y protección del formulario de acceso.

ANEXOS

Para cuando algo no sale

Con contenedores, el orden de búsqueda es: `docker compose ps` (¿está en marcha?), `docker compose logs` (¿qué dice?), `curl` a 127.0.0.1 (¿responde por dentro?) y, por último, el `error.log` de Nginx.

Anexo A · Problemas frecuentes

Docker

Lo que ves	Qué pasa	Qué hacer
`permission denied while trying to connect to the Docker daemon socket`	Tu usuario no está en el grupo `docker`, o no has vuelto a entrar desde que lo añadiste	`groups`; si falta `docker`, paso 5 y **cierra y abre la sesión**
`Cannot connect to the Docker daemon. Is the docker daemon running?`	El servicio de Docker está parado, a menudo por un error en `daemon.json`	`sudo journalctl -u docker -n 30 --no-pager`; revisa el JSON del paso 6
`docker compose`: `no configuration file provided: not found`	No estás en la carpeta de la aplicación	`cd /opt/apps/NOMBRE`, o usa `docker compose -f /ruta/compose.yaml ...`
`yaml: line N: ...`	Sangría mal hecha o tabuladores en el YAML	Revisa la línea N y la anterior; solo espacios
`Bind for 127.0.0.1:8081 failed: port is already allocated`	Otro contenedor ya usa ese puerto del servidor	`docker ps` para ver quién; elige otro puerto (8082…) y lleva la lista
`STATUS: Restarting (1) ...`	El contenedor arranca y se cae una y otra vez	`docker compose logs --tail 50`: el motivo está en las últimas líneas
`manifest unknown` / `not found` al descargar	La etiqueta de la imagen está mal escrita o no existe	Compruébala en la pestaña Tags de Docker Hub (paso 11)
El disco se llena	Imágenes viejas, contenedores parados o registros	`docker system df`; `docker image prune -a`; revisa el paso 6

Nginx como proxy

Lo que ves	Qué pasa	Qué hacer
502 Bad Gateway	Nginx no consigue hablar con la aplicación: el contenedor está parado o el puerto de `proxy_pass` no coincide	`docker compose ps` y `curl -I http://127.0.0.1:8081` en el servidor. El puerto de `proxy_pass` debe ser la **primera** cifra de `ports` (8081), no la del contenedor (80)
504 Gateway Timeout	La aplicación tarda más que `proxy_read_timeout`	Mira los registros de la aplicación; si es normal que tarde, sube el valor en ese sitio
`ERR_TOO_MANY_REDIRECTS`	La aplicación cree que está en HTTP y redirige a HTTPS, y Nginx le vuelve a mandar HTTP	Comprueba que el sitio incluye `snippets/proxy.conf` (cabecera `X-Forwarded-Proto`)

Lo que ves	Qué pasa	Qué hacer
La página carga pero no se actualiza en tiempo real, o dice «desconectado»	Falla el WebSocket	Revisa `conf.d/websocket.conf` y las líneas `Upgrade` / `Connection` del snippet
`unknown "connection_upgrade" variable` en `nginx -t`	Falta el `map` del paso 13.1	Crea `/etc/nginx/conf.d/websocket.conf`
Sale tu web principal en vez de la aplicación	El enlace del sitio no está en `sites-enabled`, o el `server_name` no coincide	`ls -l /etc/nginx/sites-enabled/` y revisa el nombre

Anexo B · Puertos abiertos al terminar esta guía

Puerto	Servicio	Firewall de IONOS	UFW	Qué lo protege
22/TCP	SSH	Abierto	Abierto	Solo clave, sin root, fail2ban (Guía 01)
80/TCP	HTTP (Nginx)	Abierto	Abierto	Solo Let's Encrypt y redirección (Guía 02)
443/TCP	HTTPS (Nginx)	Abierto	Abierto	TLS, cabeceras, rechazo de nombres ajenos (Guía 02)
8081/TCP	IT-Tools (contenedor)	Cerrado	—	Escucha **solo en 127.0.0.1**: inaccesible desde fuera aunque se abriera en un cortafuegos

Lista de puertos internos

Lleva esta lista para no repetir puertos. La iremos ampliando en cada guía:

Puerto interno	Aplicación	Subdominio
127.0.0.1:8081	IT-Tools	`herramientas.TU_DOMINIO`
127.0.0.1:8082	(WordPress, episodio 10)	
127.0.0.1:8083		

Anexo C · Chuleta de Docker y Compose

Para...	Comando
Arrancar o aplicar cambios de una aplicación	`cd /opt/apps/APP && docker compose up -d`
Estado de la aplicación	`docker compose ps`
Registros en directo	`docker compose logs -f` (sal con **Ctrl+C**)
Últimas líneas del registro	`docker compose logs --tail 50`
Reiniciar la aplicación	`docker compose restart`
Parar y quitar los contenedores (los datos en volúmenes se quedan)	`docker compose down`
Actualizar a la versión del archivo	`docker compose pull && docker compose up -d`
Ver la configuración final que usa Compose	`docker compose config`
Todos los contenedores del servidor	`docker ps -a`
Política de reinicio de todos	`docker inspect -f '{{.Name}} {{.HostConfig.RestartPolicy.Name}}' $(docker ps -q)`
Entrar en un contenedor	`docker exec -it NOMBRE sh`
Uso de CPU y memoria en directo	`docker stats` (sal con **Ctrl+C**)
Espacio que ocupa Docker	`docker system df`

Para...	Comando
Limpiar imágenes sin usar	`docker image prune -a`

Si te atascas

Pega: el `compose.yaml`, la salida de `docker compose ps`, las últimas líneas de `docker compose logs` y, si el problema es al entrar por el navegador, el sitio de Nginx y su `error.log`.