

# Docker

Documentación relativa a los containers creados.

- [Índice](#)
- [Portainer](#)
- [Cloudflare Tunnel](#)
- [Excalidraw](#)
- [MariaDB](#)
- [PostgreSQL](#)

# Índice

Para todos estos servicios que aquí se muestran, se da por supuesto que ya está instalado en el servidor, [Docker](#). Si no lo está, se pueden seguir las instrucciones que se encuentran en su web:

- [Get Docker](#)
- [Install Docker Engine](#)

Herramienta	Descripción
<a href="#">Portainer</a>	Gestor gráfico para contenedores Docker
<a href="#">Cloudflare Tunnel</a>	Acceso a tus servicios desde Internet
<a href="#">Excalidraw</a>	Pizarra virtual
<a href="#">MariaDB</a>	Base de datos MariaDB
<a href="#">PostgreSQL</a>	Base de datos PostgreSQL

# Portainer

Más información en [Portainer](#).

Este es el script que uso para generar y actualizar Portainer (aunque últimamente se puede actualizar desde la interfaz):

```
#!/bin/bash
container="portainer-ee"
echo "Actualizamos la imagen de $container"
docker pull portainer/portainer-ee
echo "Paramos $container"
docker stop $container
echo "Borramos el dock de $container"
docker rm $container
echo "Creamos el dock de $container"
docker run -d -p 9000:9000 -p 9001:9001 \
    --restart always \
    --name="$container" \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v ./portainer-ee:/data \
    portainer/portainer-ee:latest
echo "Hecho."
```

De esta manera siempre que ejecutemos este archivo nos bajará la última versión estable por defecto, parará el contenedor de **Portainer**, lo borrará (no perderemos datos) y lo volverá a levantar.

El directorio donde se guardan los datos persistentemente y que se ha de hacer copia de seguridad, en este caso es el directorio donde se ejecute el script dentro de `portainer-ee`.

El archivo generado se le ha de dar permisos de ejecución. Por ejemplo:

```
chmod 700 portainer-ee.sh
```

Recordar que si queremos eliminar las imágenes antiguas para liberar espacio en disco, podemos ejecutar:

```
docker image prune -a
```

En el caso que aquí se expone, se llama a la versión *Enterprise* que permite tener hasta 5 licencias gratuitamente. Para la versión *Community* se puede utilizar `docker pull portainer/portainer-ce`.

[Vover a HomeLab](#) ☐☐

# Cloudflare Tunnel

Más información: [Cloudflare Tunnel](#).

Cloudflare Tunnel es un servicio completamente gratuito y seguro que permite acceder a los servicios de nuestra red sin importar que estemos detrás de un CG-NAT, sin tener que abrir ningún puerto de nuestro router, instalar un proxy inverso como [Traefik](#), [Caddy](#), [NPM](#)... o montar una VPN, con el añadido de la generación automática de certificado [Let's Encrypt](#). Podemos incluso *securizar* más el acceso añadiendo credenciales para que sólo puedan acceder aquellos usuarios que estén validados.

Se puede crear facilmente el Stack en [Portainer](#) con el siguiente YAML:

```
---
services:
  cloudflaretunnel:
    container_name: cloudflaretunnel
    image: cloudflare/cloudflared:latest
    restart: unless-stopped
    environment:
      - TUNNEL_TOKEN=$TUNNEL_TOKEN
    command: tunnel --no-autoupdate run
```

## Variables

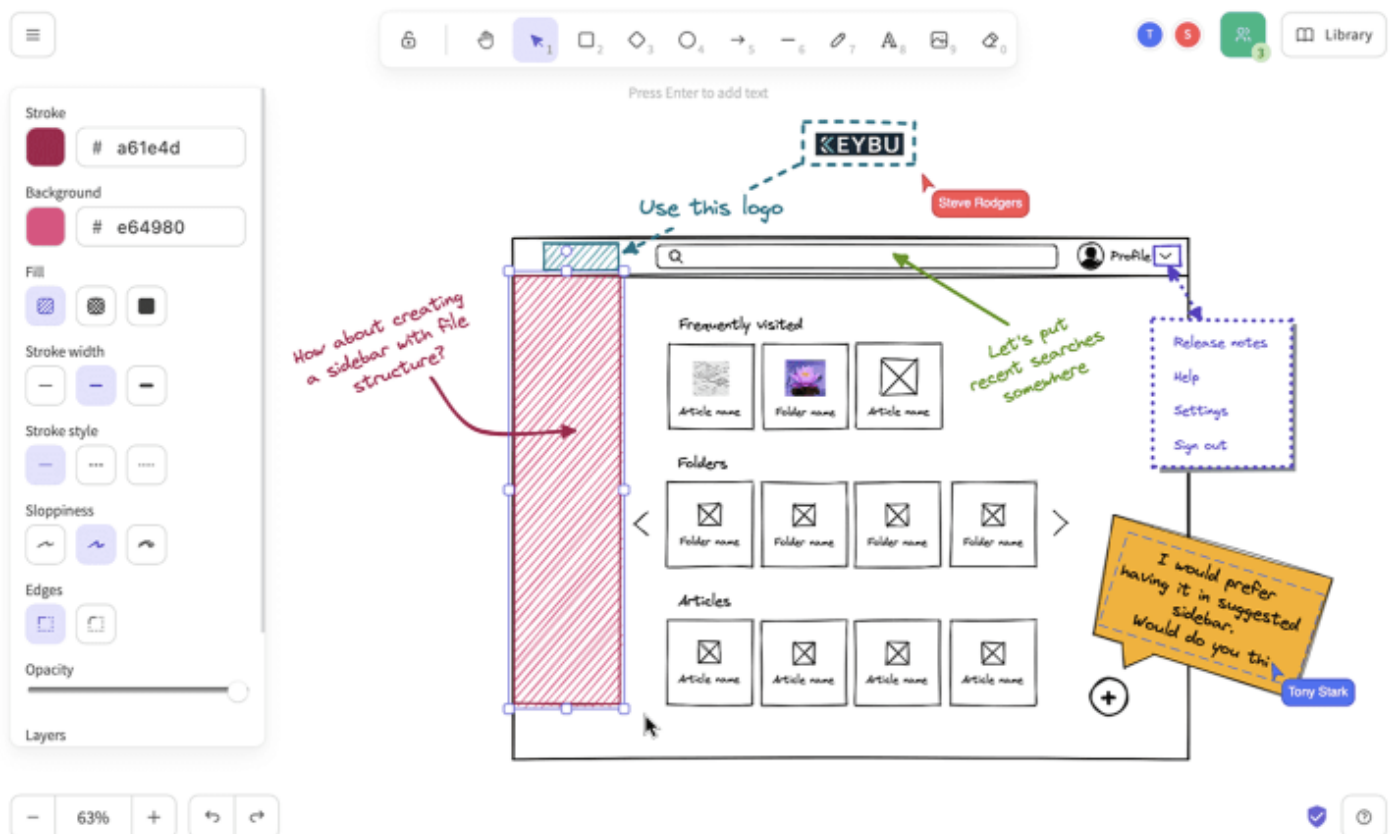
- `TUNNEL_TOKEN`: corresponde al token que nos generará Cloudflare.

[Vover a HomeLab](#) ☐☐

# Excalidraw

Código fuente: <https://github.com/excalidraw/excalidraw>.

Cuando tienes que realizar diagramas esquemáticos, representaciones gráficas de tus ideas, fluxogramas (diagramas de flujo)... Excalidraw te ayuda a plasmar todas tus ideas o representaciones con facilidad.



Inicialmente las herramientas básicas son las típicas: cuadrados, círculos, líneas... Con todo esto puedes llegar a dibujar todo lo que necesitas, pero si no tienes tiempo, puedes explorar la biblioteca donde la gente ya se ha tomado la molestia de dibujar desde cualquier tipo de hardware a todo tipo de símbolos variopintos.

Se puede crear fácilmente el Stack en [Portainer](#) con el siguiente YAML:

```
---
services:
  excalidraw:
    container_name: excalidraw
    image: excalidraw/excalidraw:latest
```

```
restart: always
```

```
ports:
```

```
- 80:80
```

[Volver a HomeLab](#) 

# MariaDB

Fuente: [MariaDB](#) y [phpMyAdmin](#)

Uno de los elementos importantes de un entorno son las bases de datos (BBDD). Es donde se almacenan los datos necesarios para que las aplicaciones puedan guardar y gestionar persistentemente en el disco.

Habitualmente, cuando creo un Docker de motor de BBDD, añado también un gestor gráfico o tipo web para agilizar la gestión de la misma. Así que en este caso he añadido [phpMyAdmin](#).

También prefiero centralizarlo todo en un contenedor y que el resto se conecte a este, para así no duplicar las instalaciones de motores de BBDD y ser más conservador en los recursos de memoria, centralizar copias de seguridad...

```
---
services:
  mariadb:
    container_name: mariadb
    image: mariadb
    restart: always
    networks:
      - mariadbnet
    volumes:
      - ./mariadb:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: ${DATABASE_PWD}

  phpmyadmin:
    container_name: phpmyadmin
    image: phpmyadmin
    restart: always
    depends_on:
      - mariadb
    networks:
      - mariadbnet
    ports:
      - "80:80"
```



```
environment:
  - PMA_HOST=mariadb

networks:
  mariadbnet:
    external: true
```

## Variables

- `DATABASE_PWD`: Contraseña del usuario `root` de la BD.

## Notas

- `image`: Al no indicar nada, siempre escogerá la etiqueta `latest`.
- `networks`: Todos los contenedores que creemos a partir de ahora que necesiten conectarse a MariaDB, se han de añadir a la red `mariadbnet`.

## Copias de seguridad

Hacer copias de seguridad es una obligación si no quieres perder tus datos. Con este script puedes hacer copias separadas por BD.

```
#!/bin/bash

# Eliminamos los archivos más antiguos de 20 días
find -L ./mariadb_bck -mtime +20 -type f -delete

# Realizamos copia de seguridad individual de cada BD
docker exec mariadb mariadb -u root --password="ContraseñaMolonaDeroot" --batch --skip-column-names -e 'show databases;' | grep -E -v 'information_schema|performance_schema|mysql' | while read db; do docker exec mariadb mariadb-dump -u root --password="ContraseñaMolonaDeroot" --comments --dump-date --extended-insert --routines --system=user "$db" | gzip -c --best > ".mariadb_bck/$(date +%Y%m%d%H%M%S).$db.sql.gz"; done
```

Para ejecutarlo hay que recordar hacerlo ejecutable (`chmod u+x backup.sh`) y muy recomendable añadirlo a `cron` para que se haga periódicamente.

# PostgreSQL

Fuente: [PostgreSQL](#) y [pgAdmin](#)

Al igual que en [MariaDB](#), uno de los elementos importantes de un entorno son las bases de datos (BBDD). Es donde se almacenan los datos necesarios para que las aplicaciones puedan guardar y gestionar persistentemente en el disco.

Habitualmente, cuando creo un Docker de motor de BBDD, añado también un gestor gráfico o tipo web para agilizar la gestión de la misma. Así que en este caso he añadido [pgAdmin](#).

También prefiero centralizarlo todo en un contenedor y que el resto se conecte a este, para así no duplicar las instalaciones de motores de BBDD y ser más conservador en los recursos de memoria, centralizar copias de seguridad...

```
---
services:
  postgres:
    container_name: postgresql
    image: postgres:15
    restart: always
    environment:
      POSTGRES_PASSWORD: ${POSTGRESQL_PWD}
    volumes:
      - ./postgresql:/var/lib/postgresql/data
    networks:
      - postgresqlnet

  pgadmin:
    container_name: pgadmin
    image: dpage/pgadmin4
    restart: always
    environment:
      PGADMIN_DEFAULT_EMAIL: ${PGADMIN_USR}
      PGADMIN_DEFAULT_PASSWORD: ${PGADMIN_PWD}
    ports:
      - 8181:80
```

```
volumes:
  - ./pgadmin:/var/lib/pgadmin

networks:
  - postgresqlnet

depends_on:
  - postgres

networks:
  postgresqlnet:
    external: true
```

## Variables

- `POSTGRESQL_PWD`: Contraseña del usuario administrador de PostgreSQL.
- `PGADMIN_USR`: Usuario de pgAdmin.
- `PGADMIN_PWD`: Contraseña del usuario pgAdmin.

## Notas

- **Importante**: el directorio `./pgadmin` se le han de otorgar permisos para usuario y grupo `5050` (`sudo chown -R 5050:5050 ./pgadmin`), ya que de lo contrario el servicio `pgadmin` no arrancará el servicio web de inicio.
- `image`: Escogeremos una versión concreta como la `15`, `16`, `17`... ya que si le ponemos `latest` (o nada), la imagen se actualizará en algún momento a la última versión y tendremos problemas para levantar las BBDD. Actualmente no hay retro/post/compatibilidad entre versiones. Si quieres pasar de una versión a otra la única opción es hacer una copia de seguridad en una versión y restaurarla en la otra.
- En mi caso particular, tengo varias versiones en marcha, según las necesidades de cada aplicación. Pero procuro tener las menos posibles.
- Otra opción es tener sólo una versión y centralizar todas las aplicaciones ahí y en algún momento, migrar todas las BBDD a otra versión de PostgreSQL, apuntar las aplicaciones al nuevo y eliminar el más antiguo.
- `networks`: Todos los contenedores que creemos a partir de ahora que necesiten conectarse a MariaDB, se han de añadir a la red `postgresqlnet`.

## Copias de seguridad

Hacer copias de seguridad es una obligación si no quieres perder tus datos. En este caso se hace una copia de seguridad de todas las BBDD y objetos. Tengo en mente hacer como en la copia de seguridad de [MariaDB](#) que se haga por separado, pero de momento es esto lo que tengo.

```
#!/bin/bash

# Eliminamos los archivos más antiguos de 20 días
```

```
find -L ./postgresql_bck -mtime +20 -type f -delete
```

```
# Hacemos copia de todas las BBDD y objetos
```

```
docker exec postgresql /usr/bin/pg_dumpall -U postgres >
```

```
./postgresql_bck/postgresql_bck_$(date +%Y%m%d%H%M%S).sql
```

```
##
```

```
# Para restaurar todas las BBDD: cat ./postgresql_bck/postgresql_bck_XXXXXXXXX.sql | docker  
exec -i postgresql psql -U postgres
```

```
# Ojo con restaurarlo todo, ya que también afectaría a los esquemas de sistema y no es válido  
si es de una versión a otra.
```

```
# La restauración habría que hacerla BD por BD si queremos pasar de una a otra versión.
```

```
##
```

Para ejecutarlo hay que recordar hacerlo ejecutable (`chmod u+x backup.sh`) y muy recomendable añadirlo a [cron](#) para que se haga periódicamente.