interweb

Generated on 15/11/2026 01:42

KNOWLEDGE BASE · PUBLIC DOCUMENT

Docker Compose: an application with its database

Docker & Containers

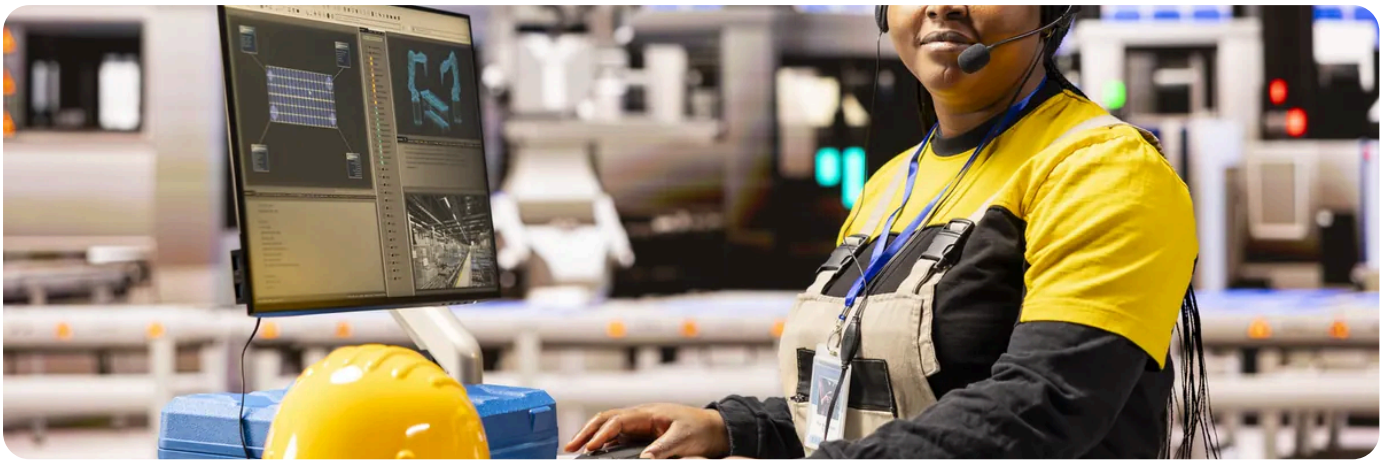
CODE	VERSION	COUNTRY · LANGUAGE
RLX-MZ-KB-882	v2026.10.10	Mozambique · English

Interweb Mozambique
RLX Holdings, S.A. · Group RLX Holdings
Tax ID (NUIT) 401 191 607 · Legal entity no. (NUEL) 105 045 974 · Economic activity code (CAE) 62022

https://www.meu.interweb.co.mz/knowledgebase/882

Docker Compose: an application with its database

Docker & Containers Created on **October 10, 2026** 3 min read



Docker Compose describes a whole application in a single file: the program, the database, the networks between them and the volumes where the data lives. One command, `docker compose up -d`, brings it all up. Here is an example with an application and PostgreSQL, on a **VPS** that already has Docker (**here is how**). The same layout works for MariaDB or MySQL: you change the image and the variables.

Putting the project together

1. **Create a folder just for the project** and go into it: `mkdir my-project && cd my-project`. The folder name becomes the project name in Compose.

2. **Create the `.env` file** with the passwords, and protect it:

```
DB_PASSWORD=put-a-long-random-password-here
```

`chmod 600 .env`. If you use Git, add `.env` to `.gitignore`. Never publish it.

3 **Create the `compose.yaml` file** (the name `docker-compose.yml` works too):

```
services:
  app:
    build: .
    restart: unless-stopped
    env_file: .env
    environment:
      DB_HOST: db
    ports:
      - "127.0.0.1:3000:3000"
    depends_on:
      db:
        condition: service_healthy
  db:
    image: postgres:16
    restart: unless-stopped
    environment:
      POSTGRES_USER: appuser
      POSTGRES_PASSWORD: ${DB_PASSWORD}
      POSTGRES_DB: appdb
    volumes:
      - db_data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U appuser -d appdb"]
      interval: 10s
      timeout: 5s
      retries: 5
volumes:
  db_data:
```

4 **Bring it up** and check the state:

```
docker compose up -d
docker compose ps
docker compose logs -f app
```

Compose creates a private network for the project.

What each part does

Part	What it does	Why it is like that
`DB_HOST: db`	The application finds the database by the **service name**.	Inside the Compose network you do not use `localhost` or an IP: each service is a separate computer.
`127.0.0.1:3000:3000`	Publishes the application port **on the server itself only**.	What shows it to the world is a proxy with HTTPS: see **nginx as a reverse proxy**.

Part	What it does	Why it is like that
No port on `db`	The database cannot be reached from outside.	That is what you want. Other services reach it over the internal network.
`db_data` (volume)	Where the database data is kept.	Without a volume, deleting the container deletes the data.
`healthcheck` and `service_healthy`	The application starts only when the database already accepts connections.	`depends_on` alone only waits for the container to *exist*, not to be ready.

The connection address, inside the application, looks like this: `postgres://appuser:PASSWORD@db:5432/appdb`. If the password has special characters (`@` , `:` , `/`), they must be encoded in the address, or pick one without them.

The `POSTGRES_*` variables only count the first time.
They create the database when the volume is empty. If you change the password in the file after the database exists, the database **does not change**, and the application can no longer log in. You change it with a command inside the database. And `docker compose down -v` **deletes the volumes**, meaning the data: without `-v` , a `down` only removes the containers.

Pick a version of the database image and pin it
(`postgres:16` rather than `postgres:latest`). Moving from one major version to the next is not done by changing only the number: it needs a dump and a restore. See **backups with mysqldump and pg_dump**, and how to update without losing anything in **updating a Docker application**. Choosing between engines is covered in **PostgreSQL or MySQL**.

RECOMMENDED
The VPS has to be yours, with root. If you do not have one yet, see the plans: Docker and the application are on your side, the network and the server on ours.
See the VPS servers →

SEE ALSO
- Nginx as a reverse proxy in front of a container →
- Updating a Docker application without losing data →

Backing up with mysqldump and pg_dump, and restoring the copy

interweb

Interweb Mozambique

RLX Holdings, S.A. · Group RLX Holdings

TAX ID (NUIT)	401 191 607
LEGAL ENTITY NO. (NUEL)	105 045 974
ECONOMIC ACTIVITY CODE (CAE)	62022
REGISTERED OFFICE	Bairro 7 de Setembro Vilankulo 1304 · CEP 0411-02, Inhambane, Mozambique
CONTACT	info@interweb.co.mz · +258 84 635 1111 · +258 86 635 1111 · www.interweb.co.mz
OPENING HOURS	Monday to Friday - 08 AM to 04 PM Saturday & Sunday - Closed

Online version of this document

RLX-MZ-KB-882 · v2026.10.10 · Generated on 15/11/2026 01:42