

Fullstack Development

Preflight project - database

[Github Repo](#)

Prerequisite

- Docker
 - Docker desktop
- Database management tools
 - Dbeaver

Database choices

- Relational database (Comparison) (SO Survey 2024)
 - PostgreSQL
 - MariaDB / MySQL
 - SQLite
- NoSQL
 - Types
 - Vendors

Docker 101

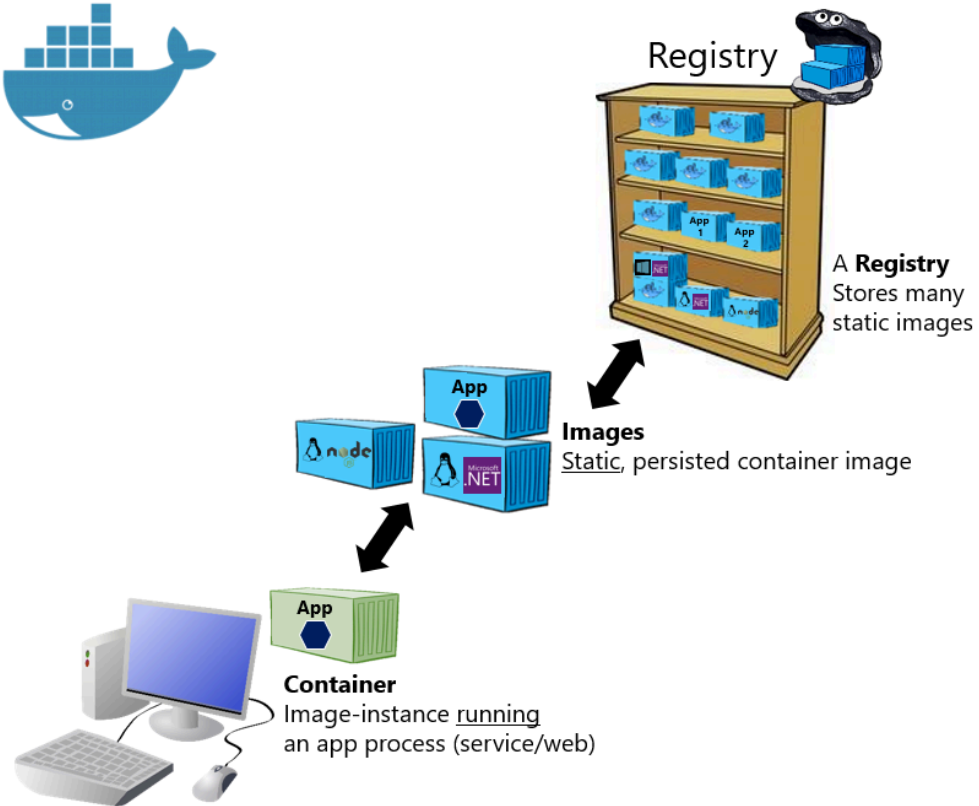
Containers

- *Virtualization technology*
- Provide a way of creating an isolated environment in which applications and their dependencies can live.
- Why?
 - Portability (save container to registry or even USB)
 - Consistency (works everywhere)
 - Easy deployment (can test on local machine)
 - More efficient (than virtual machines).

Docker

- A containerization platform
 - Leading player
- Alternative `Podman`

Basic taxonomy in Docker



Hosted Docker Registry

Docker Trusted Registry on-prem.

On-premises
(‘n’ private organizations)

Docker Hub Registry

Docker Trusted Registry on-cloud

Azure Container Registry

AWS Container Registry

Public Cloud
(specific vendors)

Google Container Registry

Quay Registry

Other Cloud

Should you run database on docker container?

| It depends.


Spinning up database instance

- Files

-  `./.env` Copy from [here](#).

-  `./.gitignore` [\(link\)](#)

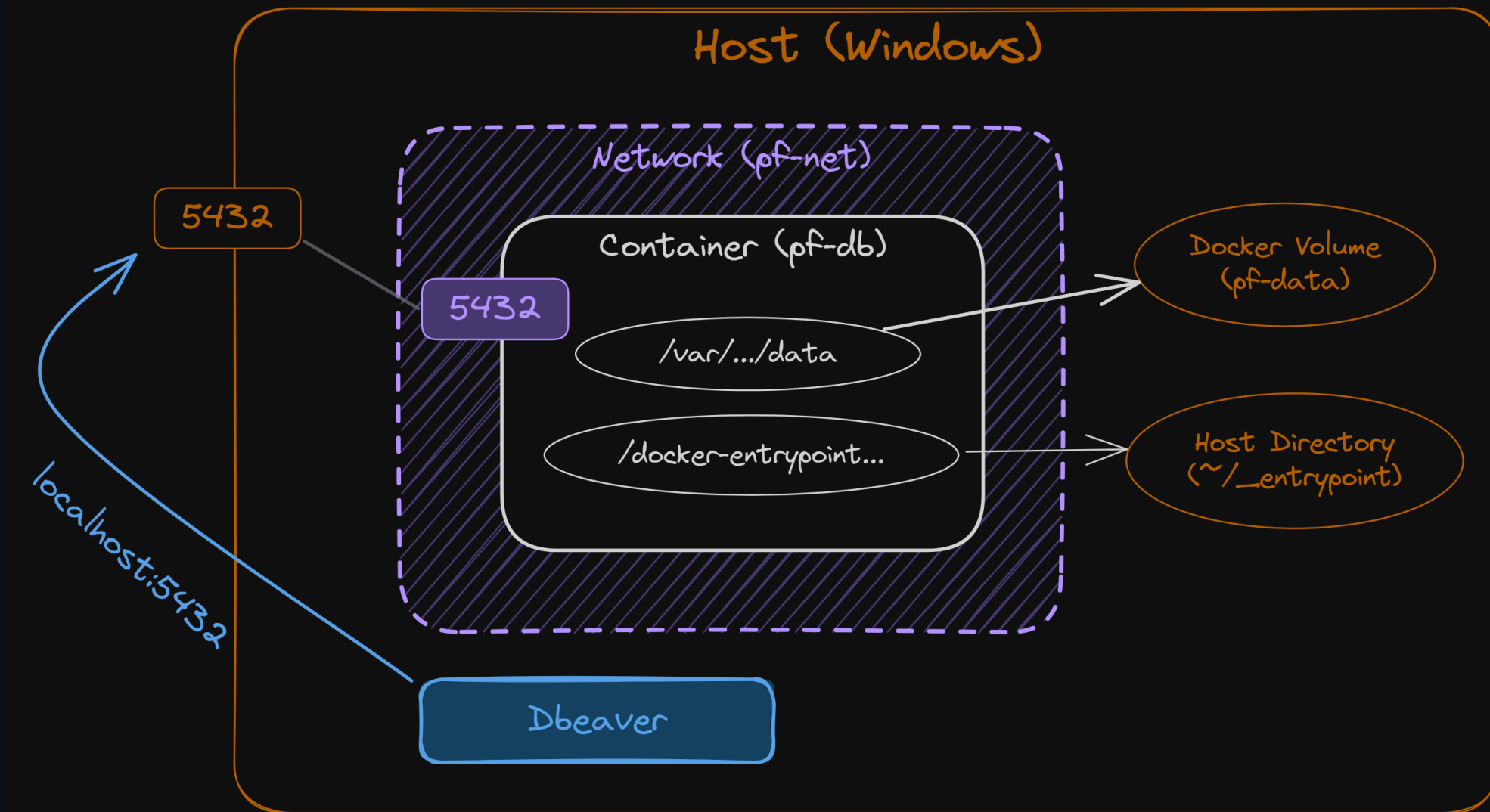
-  `./docker-compose.yml` [\(link\)](#)

-  `./_entrypoint/init.sh` [\(link\)](#)

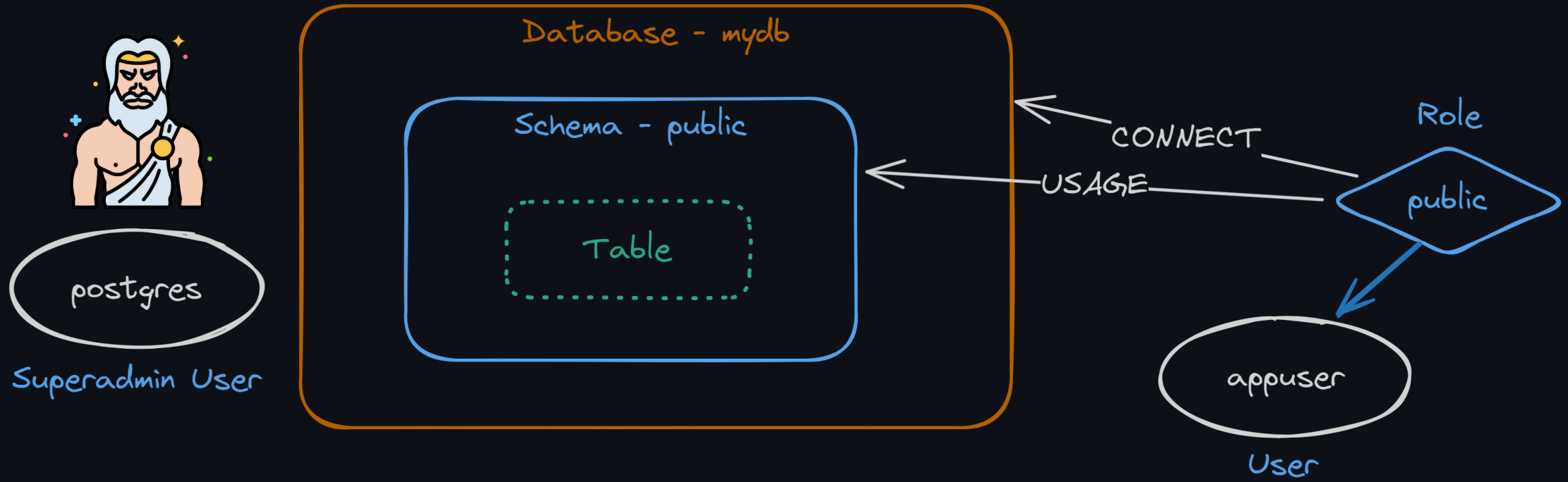
- *Windows: Make sure that you save with LF option. [\(What?\)](#)*

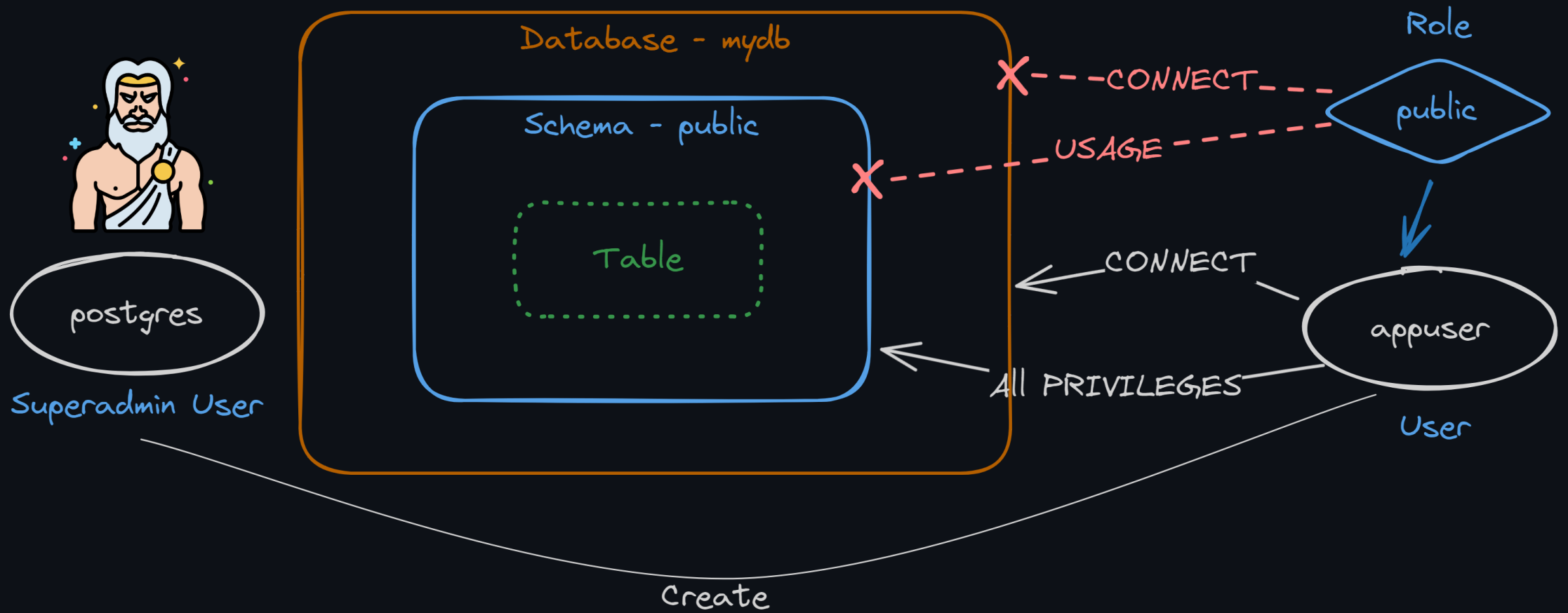
- *Mac:* `chmod +x ./_entrypoint/init.sh`

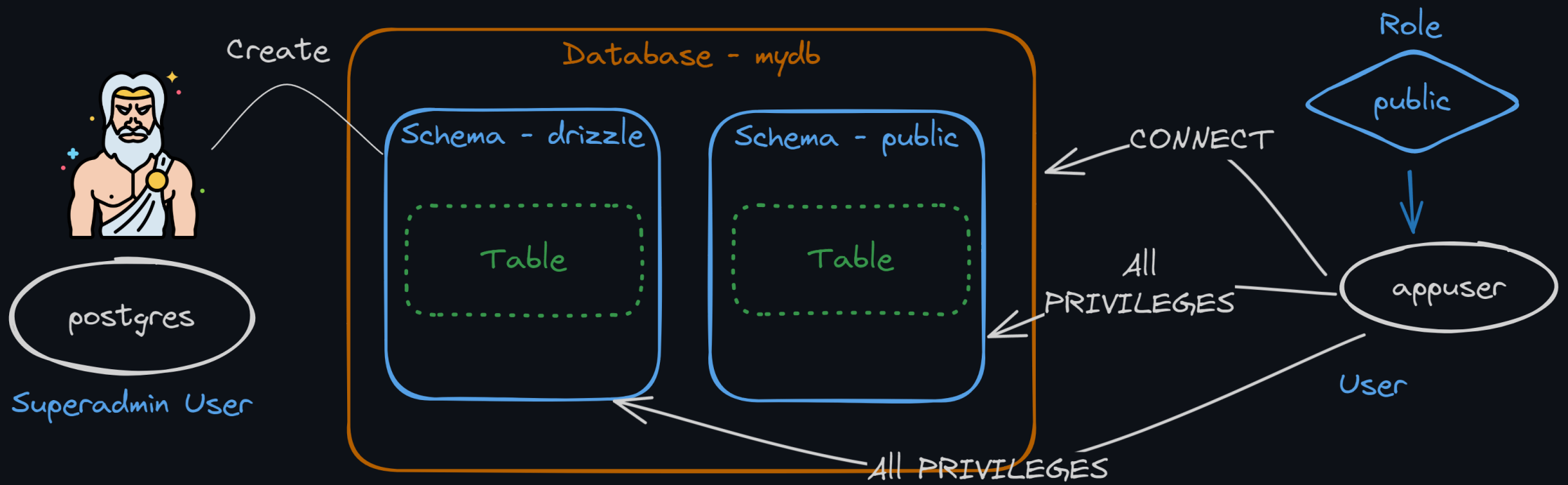
-  `docker compose up -d`



Database user management



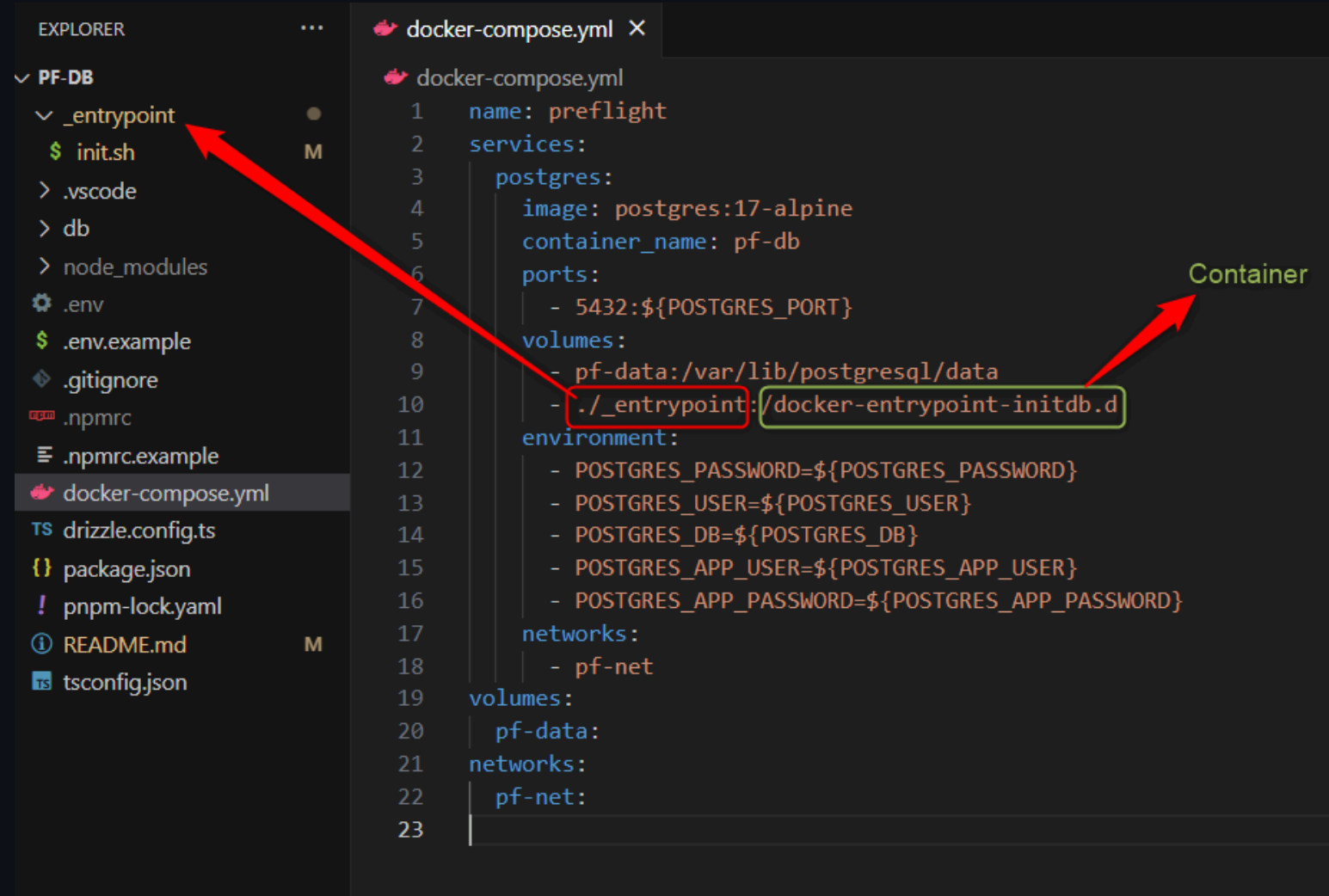




- We want to execute this when a postgres container is freshly created.
(Not restarted)

```
REVOKE CONNECT ON DATABASE mydb FROM public;  
REVOKE ALL ON SCHEMA public FROM PUBLIC;  
CREATE USER appuser WITH PASSWORD '1234';  
CREATE SCHEMA drizzle;  
GRANT ALL ON DATABASE mydb TO appuser;  
GRANT ALL ON SCHEMA public TO appuser;  
GRANT ALL ON SCHEMA drizzle TO appuser;
```


Any script files in `_entrypoint` will be executed automatically, when a docker container is freshly created. (Not restarted)



The image shows a VS Code interface with two panels. The left panel, titled 'EXPLORER', displays the file structure of a project named 'PF-DB'. It includes a directory named `_entrypoint` which contains a script file `init.sh`. The right panel, titled 'docker-compose.yml', shows the configuration for a service named 'preflight'. Within the 'services' section, the 'postgres' service is defined with the following configuration:

```
1  name: preflight
2  services:
3    postgres:
4      image: postgres:17-alpine
5      container_name: pf-db
6      ports:
7        - 5432:${POSTGRES_PORT}
8      volumes:
9        - pf-data:/var/lib/postgresql/data
10       - ./_entrypoint:/docker-entrypoint-initdb.d
11      environment:
12        - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
13        - POSTGRES_USER=${POSTGRES_USER}
14        - POSTGRES_DB=${POSTGRES_DB}
15        - POSTGRES_APP_USER=${POSTGRES_APP_USER}
16        - POSTGRES_APP_PASSWORD=${POSTGRES_APP_PASSWORD}
17      networks:
18        - pf-net
19  volumes:
20    pf-data:
21  networks:
22    pf-net:
```

A red arrow points from the `_entrypoint` directory in the Explorer to the `./_entrypoint` entry in the `volumes` section of the `postgres` service. A green box highlights the path `./_entrypoint:/docker-entrypoint-initdb.d`, and a green arrow points from this box to the word 'Container'.

Manual DB user management (not needed now)

- `docker exec -it pf-db bash`
- `psql -U postgres -d mydb`
 - Note that you do not need to input password here due to how the image is [setup](#). (See section in `POSTGRES_PASSWORD`)
- Don't forget to change the password for `appuser` .

```
REVOKE CONNECT ON DATABASE mydb FROM public;
REVOKE ALL ON SCHEMA public FROM PUBLIC;
CREATE USER appuser WITH PASSWORD '1234';
CREATE SCHEMA drizzle;
GRANT ALL ON DATABASE mydb TO appuser;
GRANT ALL ON SCHEMA public TO appuser;
GRANT ALL ON SCHEMA drizzle TO appuser;
```

Note on `psql` (not needed now)

- `\l` to list all databases
- `\du` to list users
- `\dn` to list schema
- `\dt` to list tables
- `\c` to view connected database or change to another db.
- `\q` to quit

ORM

- Object Relational Mapper
- A piece of software designed to translate between the data representations used by databases and those used in programming (in our case, Typescript).

Why ORM?

- Get type information when interacting with database.
- Write schema file
 - Good for documentation
- Nice Tooling
 - Database synchronization
 - Schema generation from existing database
 - Database viewer
 - Migration tool

Should you use ORM?

| It depends.

JavaScript / TypeScript ORM

- Ranking

Setting up Drizzle

- `npm init es6`
- `pnpm install dotenv drizzle-orm postgres`
- `pnpm install -D drizzle-kit typescript tsx @types/node @tsconfig/node-lts @tsconfig/node-ts cross-env`





TypeScript

./tsconfig.json

```
{
  "extends": [
    "@tsconfig/node-lts/tsconfig.json",
    "@tsconfig/node-ts/tsconfig.json"
  ],
  "compilerOptions": {
    "outDir": "./dist",
    "baseUrl": "./",
    "paths": {
      "@db/*": [".db/*"]
    }
  }
}
```

Database initialization

- Files


-  `./db/utils.ts` [\(Link\)](#)
-  `./db/schema.ts` [\(Link\)](#)
-  `./drizzle.config.ts` [\(Link\)](#)
-  `./npmrc` from `./npmrc.example` [\(Link\)](#) [\(What?\)](#)
 - Windows: you must choose a terminal.
 - Mac: optional

Database initialization

package.json

```
{
  "scripts": {
    "db:generate": "cross-env NODE_OPTIONS='--import tsx' drizzle-kit generate",
    "db:push": "cross-env NODE_OPTIONS='--import tsx' drizzle-kit push",
    "db:migrate": "cross-env NODE_OPTIONS='--import tsx' drizzle-kit migrate",
    "db:prototype": "tsx ./db/prototype.ts",
    "eol": "eolConverter _entrypoint/*.sh"
  }
}
```



Database synchronization

-  `npm run db:push`

Database migration

- Migration is like a *version control*.
 - Why do we need it since we already have database schema?
- Migrations ensures that database schema changes are tracked and reversible.


Migration

-  `npm run db:generate`
-  `npm run db:migrate`

Side note

- Migration rollback is not yet [available officially](#). (July 2025)
- You can try [community version](#).

CRUD

-  `./db/client.ts` [\(Link\)](#)
-  `./db/prototype.ts` [\(Link\)](#)
- `npm run db:prototype`