

# Fullstack Development

# Preflight project - database

[Github Repo](#)

# Prerequisite

- Docker
  - Docker desktop
- Database management tools
  - Dbeaver

# Database choices

- Relational database (Comparison)
  - PostgreSQL
  - MariaDB
  - MySQL
  - SQLite
- NoSQL
  - Types
  - Vendors

# Containers

- Containers provide a way of creating an isolated environment in which applications and their dependencies can live.
- Why?
  - Portability
  - Consistency
  - Speed to deploy

# Docker




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

# Should you run database on docker container?

It depends.

# Spinning up database instance

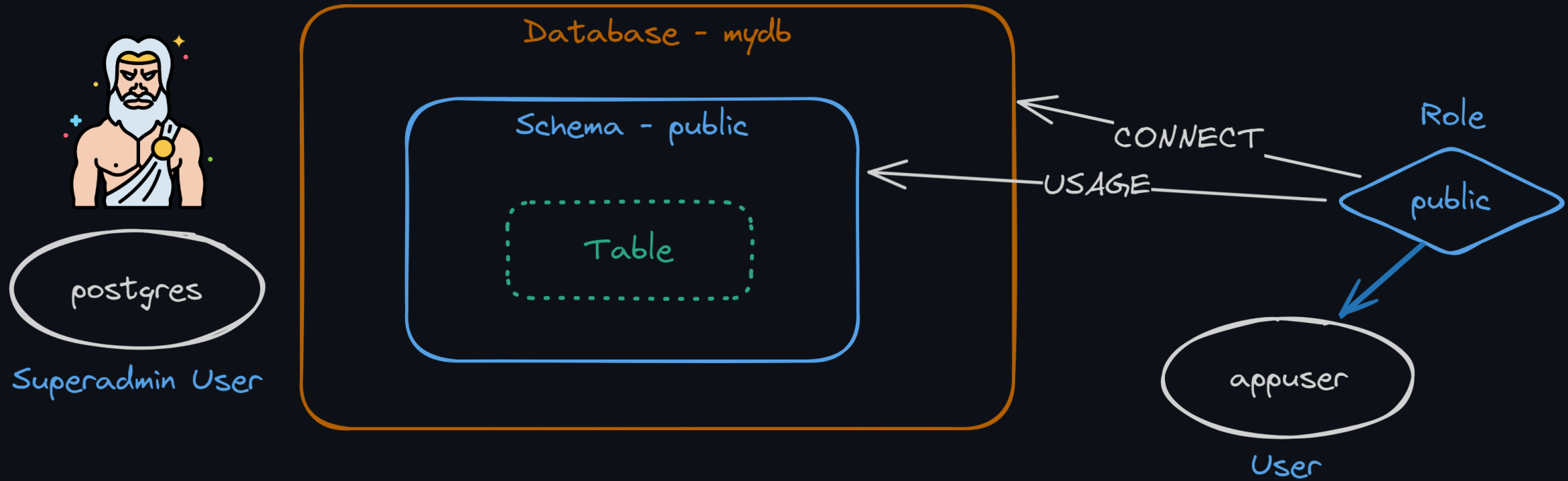
- Files

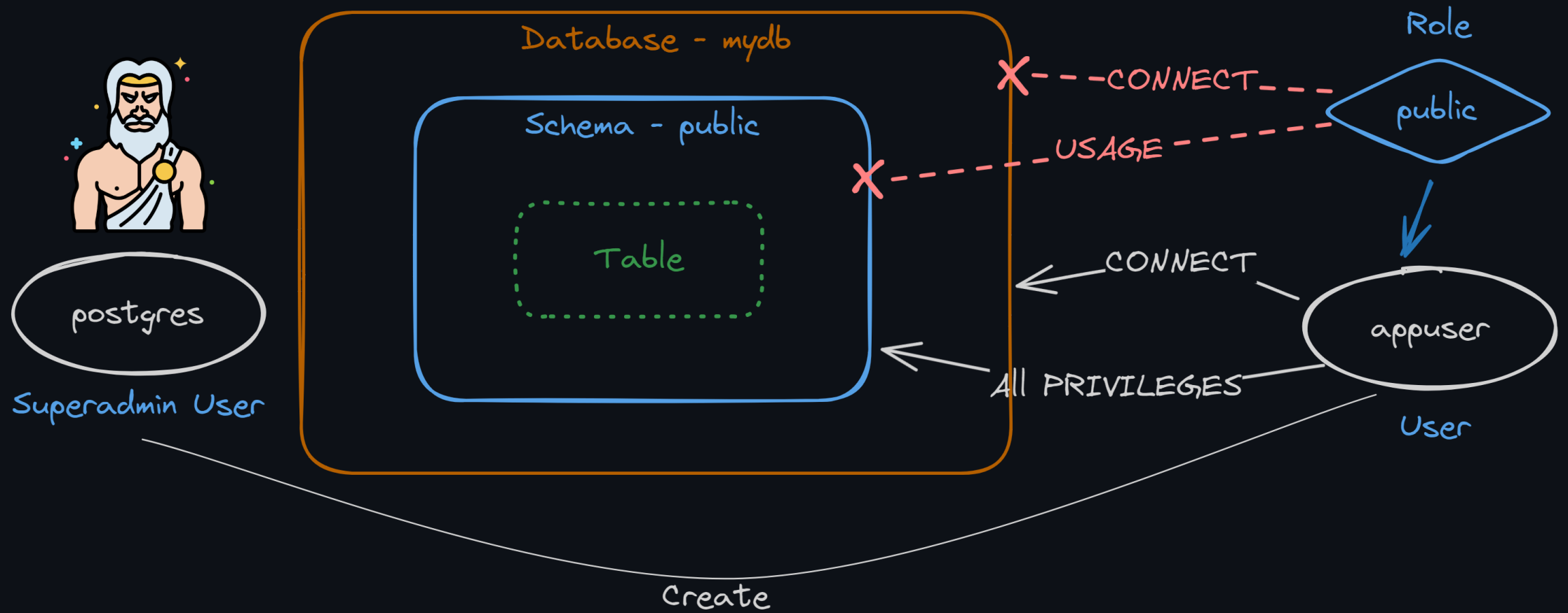
-  `./.env` Copy from [here](#).
-  `./.gitignore` ([link](#))
-  `./docker-compose.yml` ([link](#))

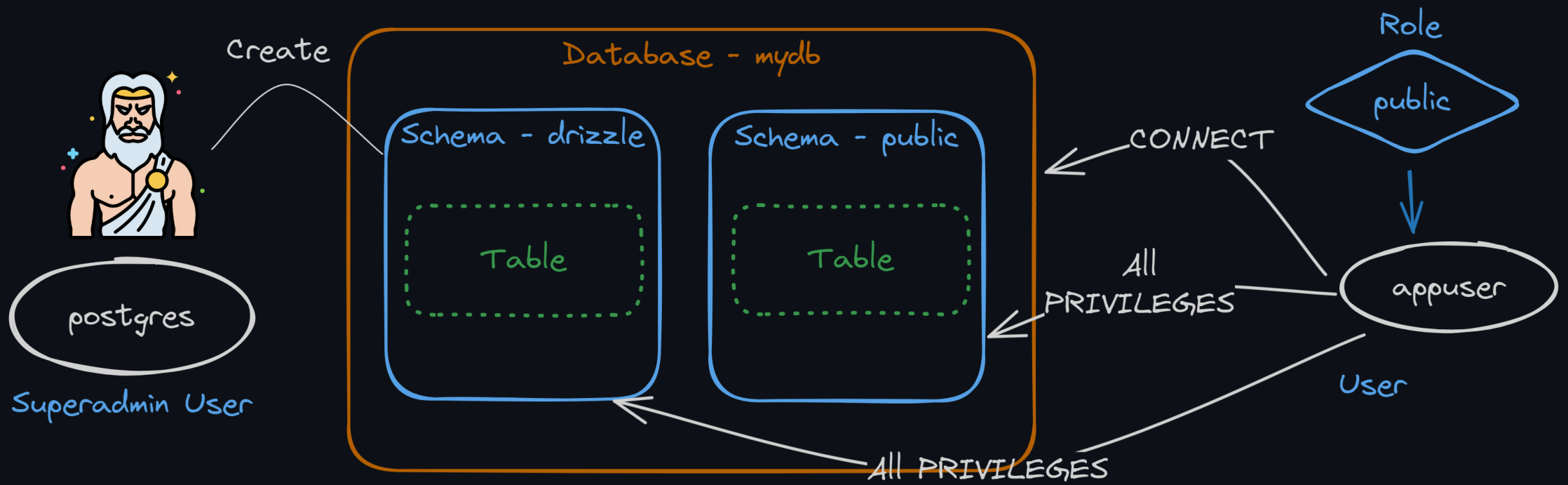
-  `docker compose up -d`



# Database user management







# DB user management

- `docker exec -it postgres_container bash`
- `psql -U postgres -d mydb`

```
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`

- `\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).

# Should you use ORM?

It depends.



# JavaScript / TypeScript ORM

- [List](#)

# Setting up Drizzle





- `npm init -y`
- `npm i drizzle-orm postgres dotenv`
- `npm i -D drizzle-kit`
- `npm i typescript ts-node tsconfig-paths`

# Typescript




- `npx tsc --init`
- Add this in `./tsconfig.json`

```
{
  "ts-node": {
    "require": ["tsconfig-paths/register"]
  },
  "compilerOptions": {
    // ...
    "baseUrl": "./",
    "paths": {
      "@db/*": [".db/*"]
    }
    // ...
  }
}
```



# Database initialization

- Files
  -  `./db/utils.ts` ([Link](#))
  -  `./db/schema.ts` ([Link](#))
  -  `./drizzle.config.ts` ([Link](#))
-  `npx drizzle-kit push`

# Migration

-  `./db/migrate.ts` ([Link](#))
-  `npx drizzle-kit generate`
-  `npx ts-node ./db/migrate.ts`

# CRUD

-  `./db/client.ts` ([Link](#))
-  `./db/prototype.ts` ([Link](#))
- `npx ts-node ./db/prototype.ts`

# Script

To save yourself some typing, add this in `package.json`

```
{  
  // ...  
  "scripts": {  
    "db:generate": "drizzle-kit generate",  
    "db:push": "drizzle-kit push",  
    "db:migrate": "ts-node ./db/migrate.ts",  
    "db:prototype": "ts-node ./db/prototype.ts"  
  }  
}
```