Fullstack Development

Preflight project - database

Github Repo

Prerequisite

- Docker
  - Docker desktop
- Database management tools
  - Dbeaver

Database choices

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

Docker 101

Containers

- Containers 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)

Docker

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

Basic taxonomy in Docker

A **Registry**
Stores many
static images

Images
Static, persisted container image

Container
Image-instance running
an app process (service/web)

- **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
- Google Container Registry
- Quay Registry
- Other Cloud

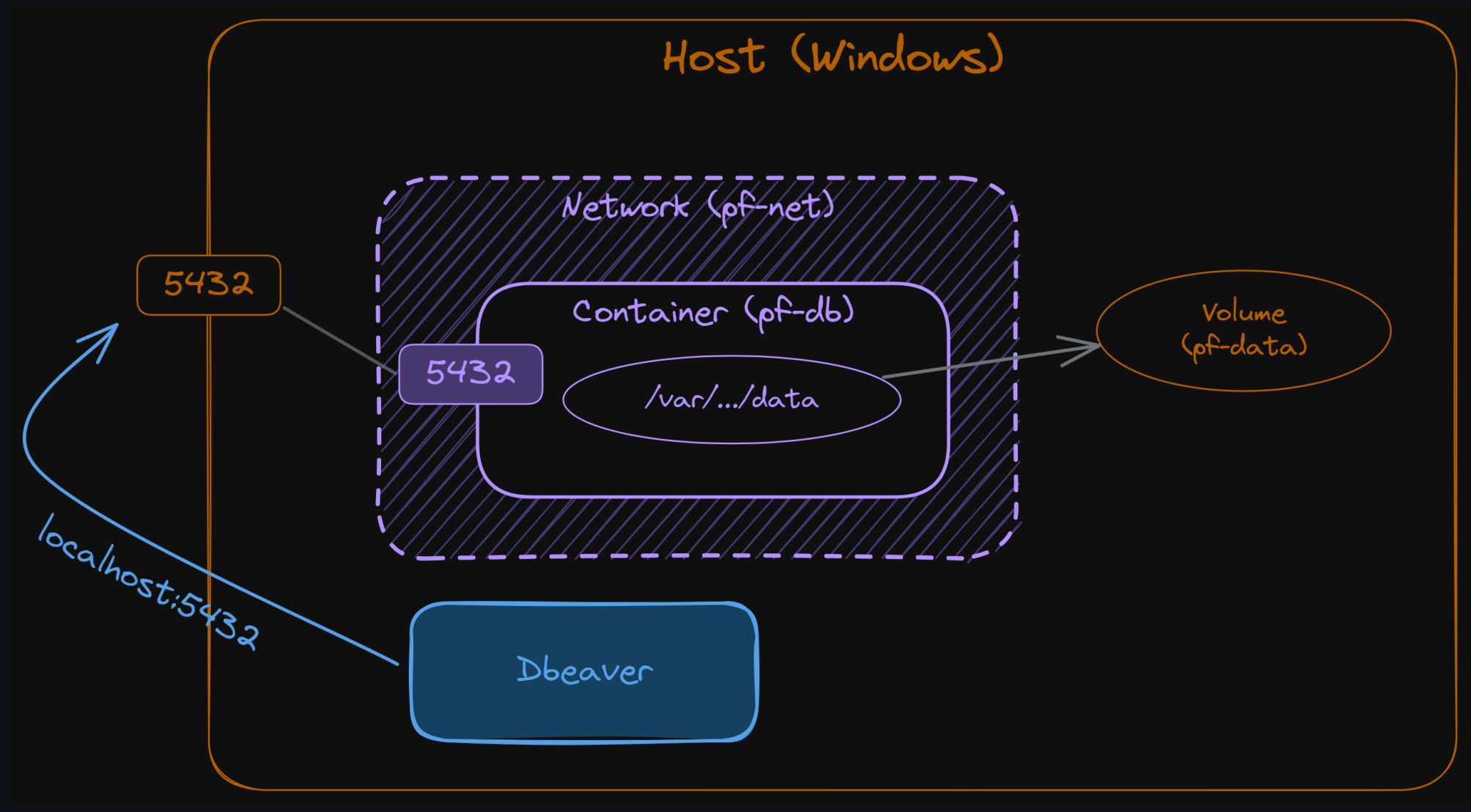
→ **Public Cloud**
(specific vendors)

Should you run database on docker container?

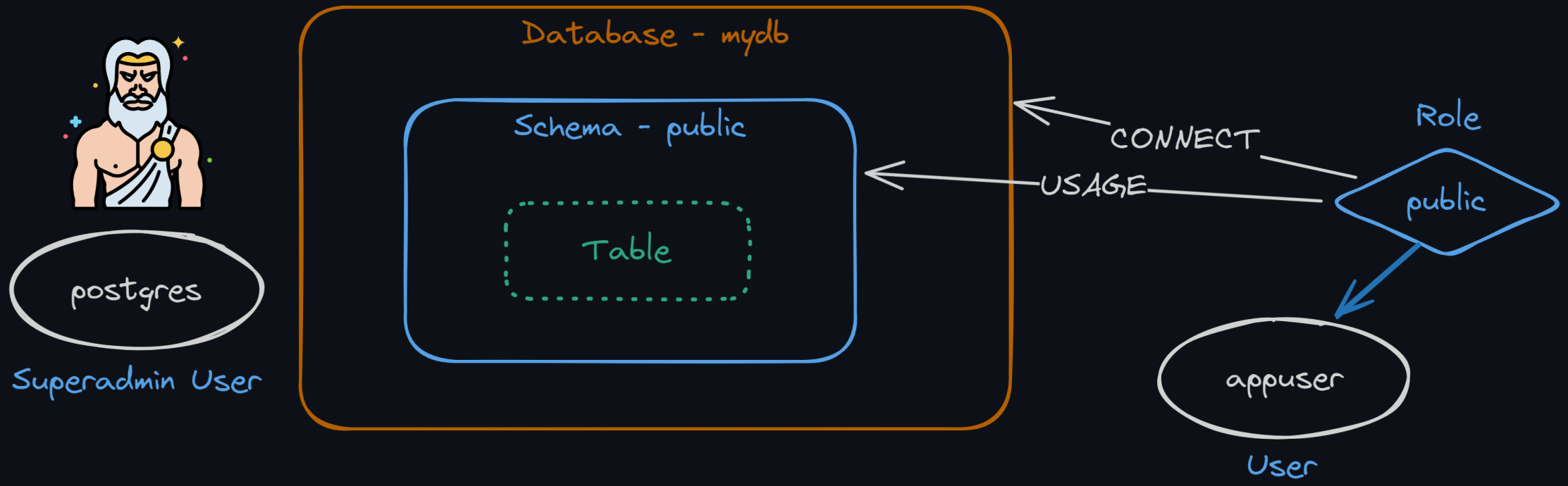
> It depends.

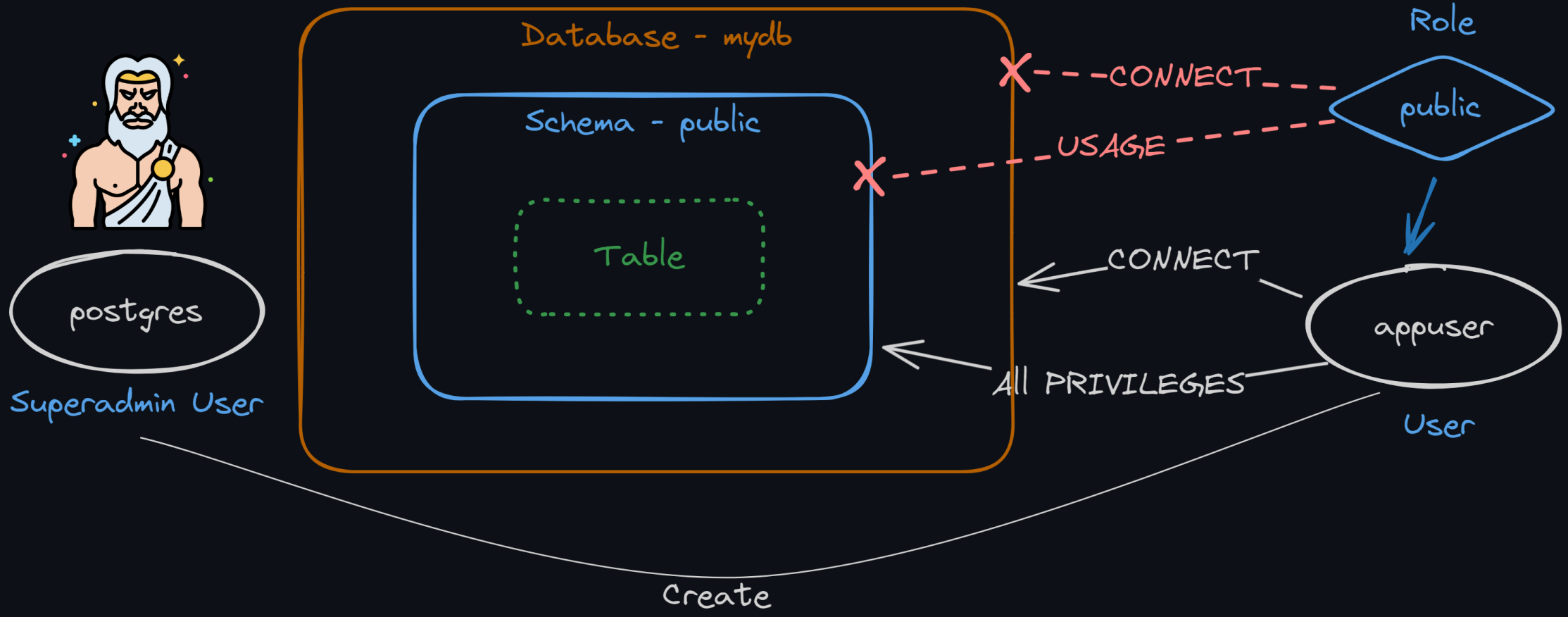
Spinning up database instance

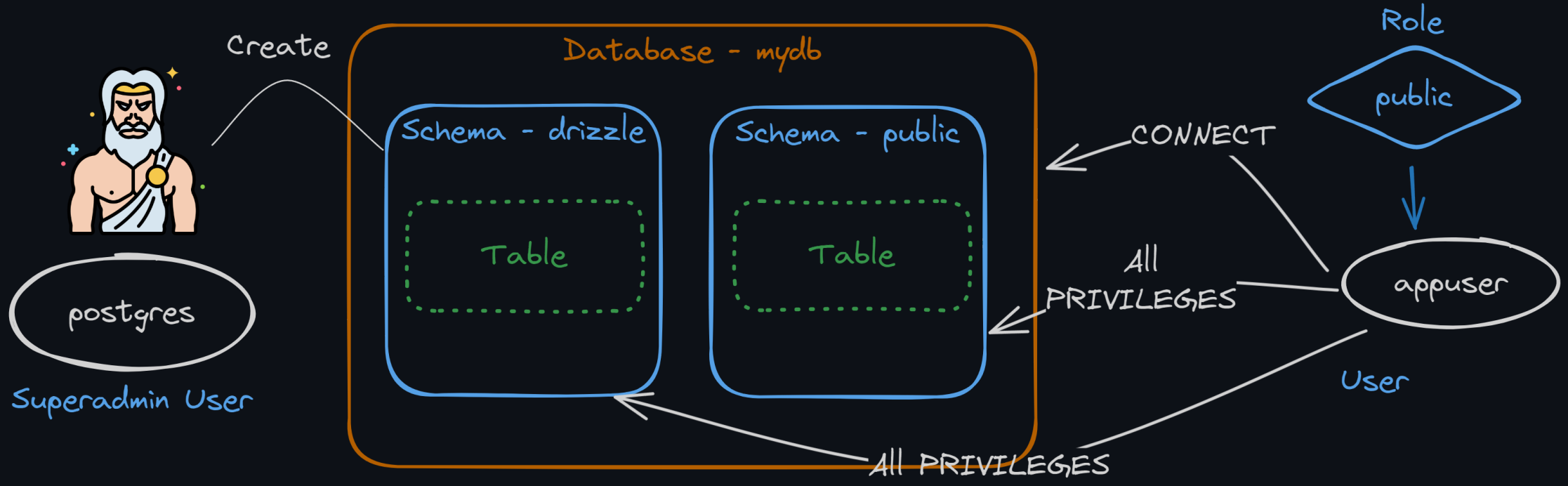
- Files
  - 💾 `./.env` Copy from here.
  - 💾 `./.gitignore` (link)
  - 💾 `./docker-compose.yml` (link)
  - 💾 `./_entrypoint/init.sql` (link)
- ⌨️ `docker compose up -d`


Host (Windows)
Network (pf-net)
5432
Container (pf-db)
5432
/var/.../data
Volume
(pf-data)
localhost:5432
Dbeaver

Database user management


Database - mydb
Schema - public
Table
CONNECT
USAGE
Role
public
appuser
User
postgres
Superadmin User

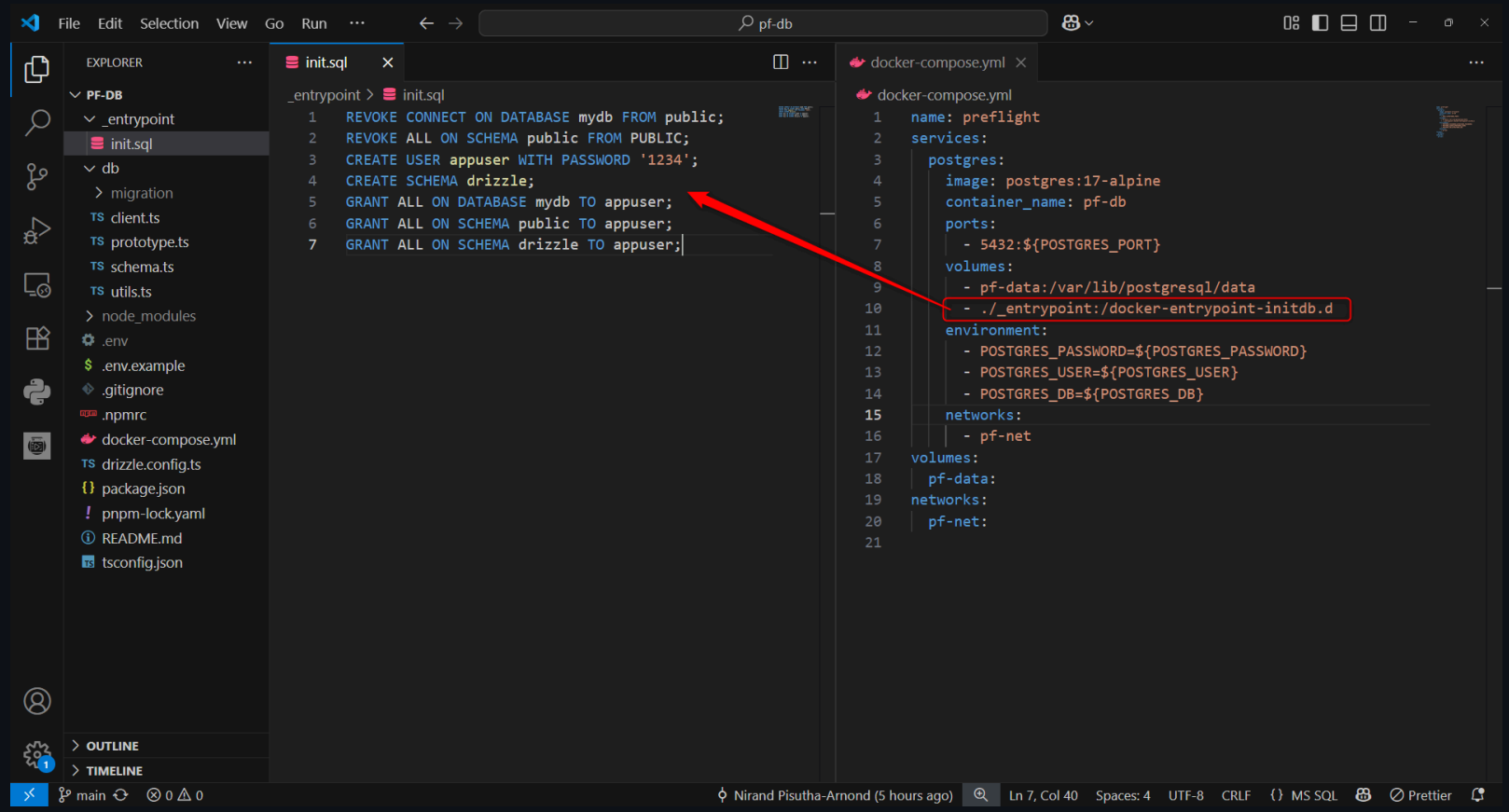

Database - mydb
Schema - public
Table
Role
public
CONNECT
USAGE
CONNECT
All PRIVILEGES
appuser
User
postgres
Superadmin User
Create


Create
Database - mydb
Schema - drizzle
Table
Schema - public
Table
Role
public
CONNECT
All
PRIVILEGES
appuser
User
postgres
Superadmin User
All PRIVILEGES

- We want to execute this when a postgres container is created.

```
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 SQL files in `_entrypoint` will be executed automatically, when a docker container is created.



DB user management (manual step)

- `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`

- `\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

- List

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` (Link)

Database initialization

package.json

```
{
  "scripts": {
    "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"
    }
  }
}
```

Database initialization

- ⌨ `npm run db:push`

Migration

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

CRUD

- 💾 `./db/client.ts` (Link)
- 💾 `./db/prototype.ts` (Link)
- `npm run db:prototype`