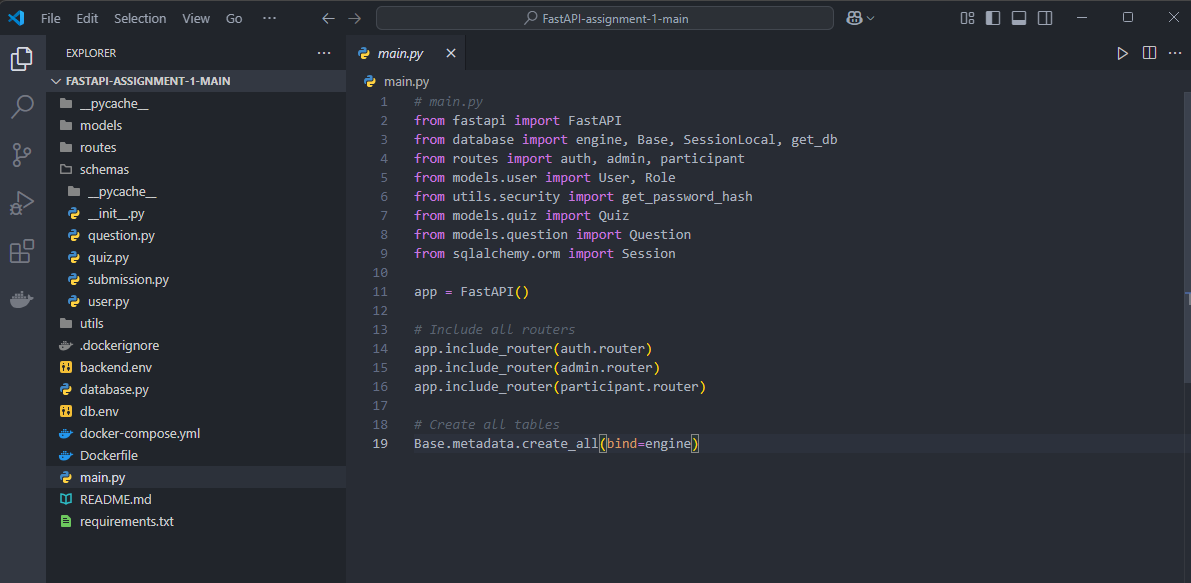
**Assignment 2**

**1. Prepare the Application:**

**Create a new application or use an existing one (apps demonstrated in labs module will not be considered).**

**If you are a backend developer, your app must use a Postgres database.**

-> FastAPI application using postgreql as database



**2. Define a single compose file:**

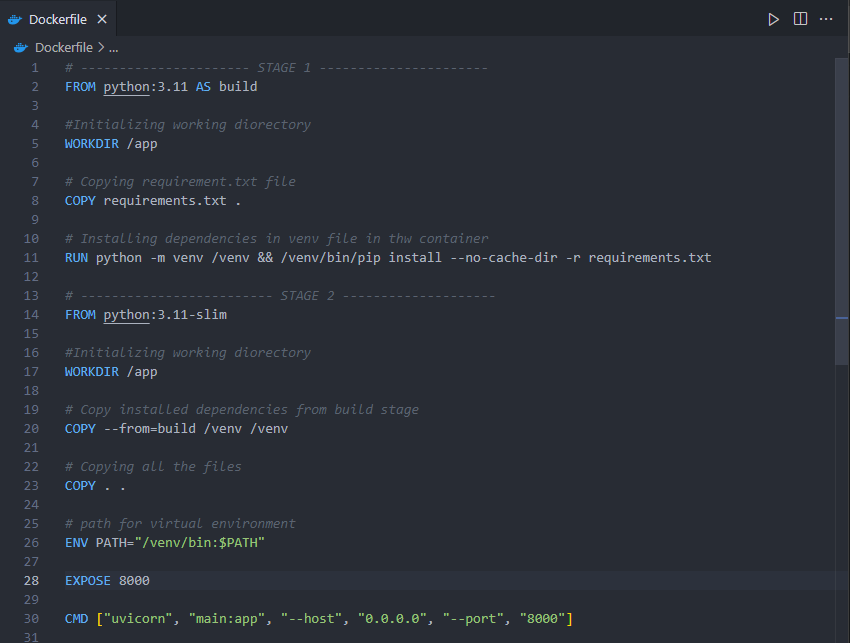
**Create a compose file to define services required for your application.**

**For backend developers:**

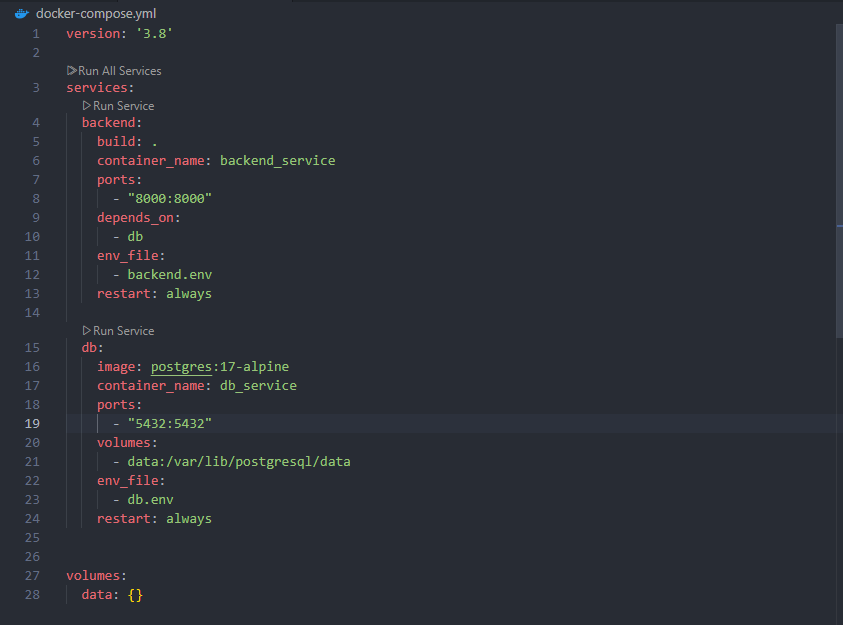
**Include at least two services: app (your application), database, and cache (optional).**

Ensure that cache service/database service data is preserved in case of a container crash.

-> Creating Dockerfile for the backend



-> Creating docker-compose.yml file to define services



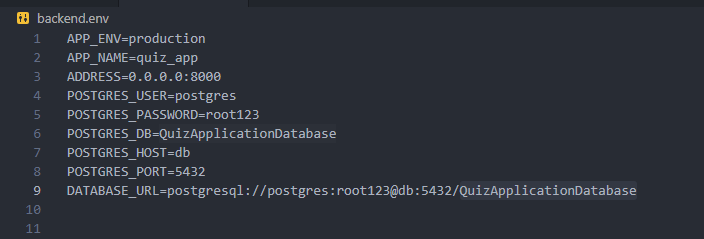
**3. Create .env Files for Each Service:**

**For each service, create a separate .env file to manage environment variables.**

**Avoid hardcoding environment variables directly in the code or the compose file.**

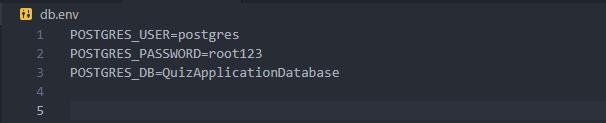
-> Creating backend.env file for backend service

-> Contains variables for FastAPI, like DATABASE\_URL



-> Creating db.env file for database service

-> Contains variables for PostgreSQL, like POSTGRES\_USER, POSTGRES\_PASSWORD, POSTGRES\_DB.

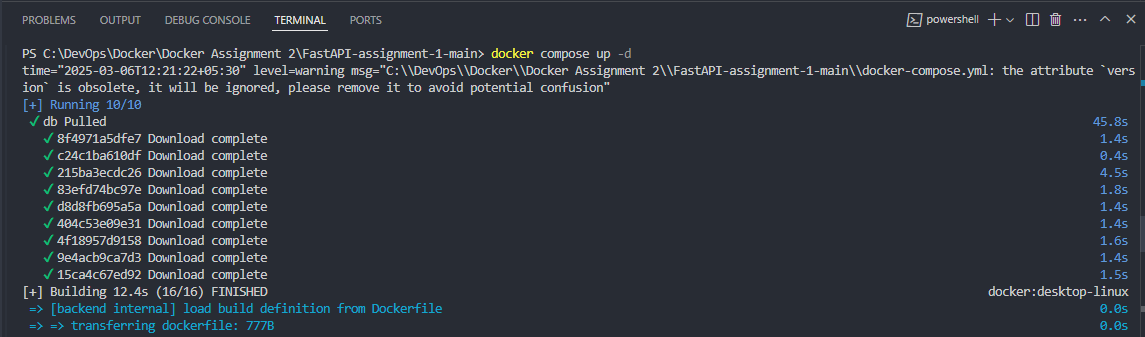


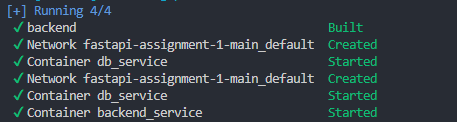
**4. Build and run the Docker Compose:**

**Use Docker Compose commands to build and run all the services defined in the compose file.**

-> Build the image and run services using the following command

-> docker compose up-d

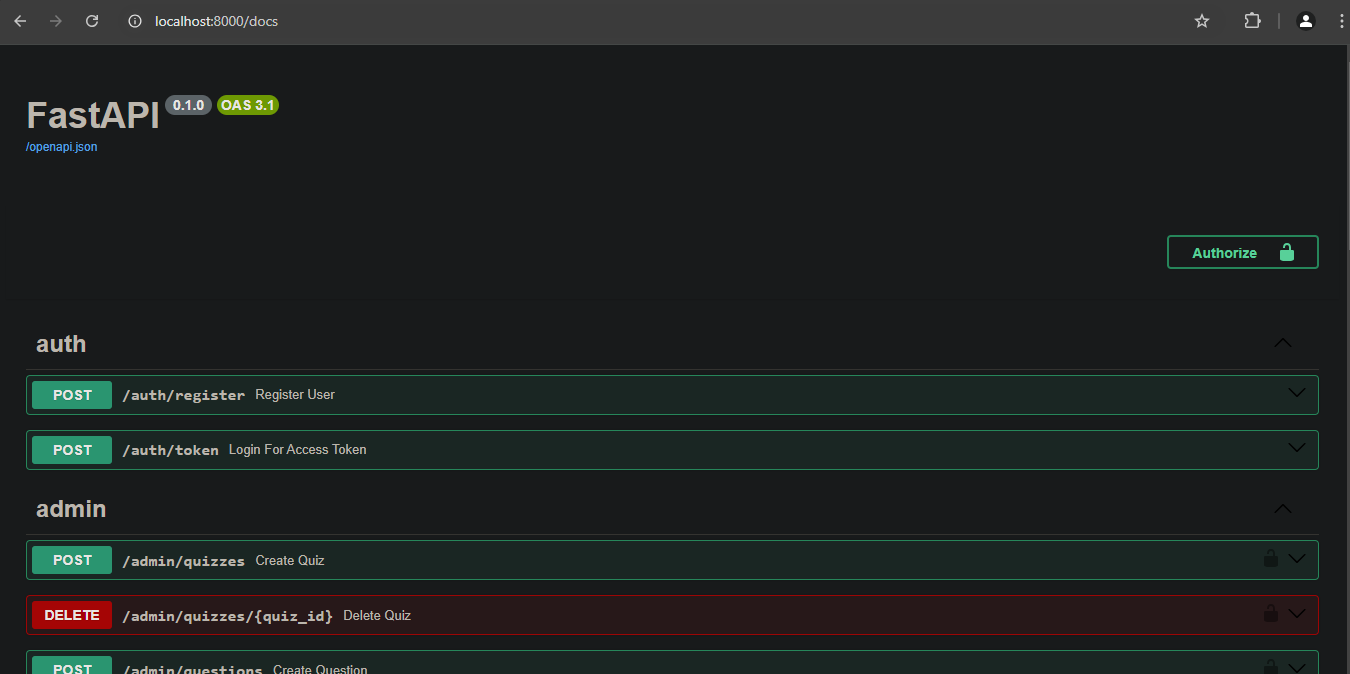




**5. Test Connectivity and Application Functionality:**

**Access the application in a browser to ensure it is functioning correctly.**

-> Open your FastAPI quiz app in a browser : <http://localhost:8000/>

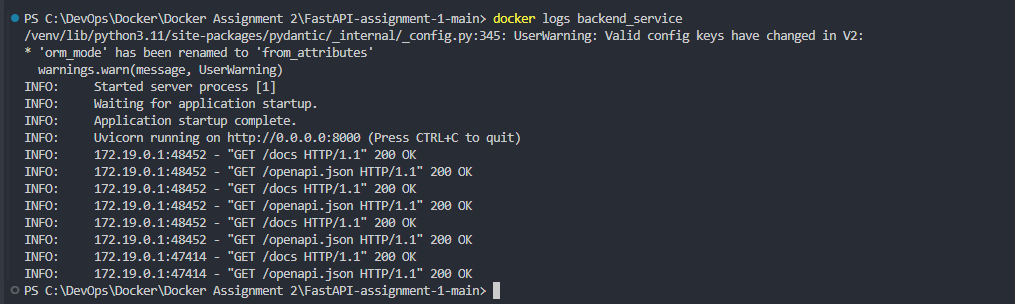


**Verify the connectivity between services (e.g., check if the app communicates with the database or cache correctly, or both).**

-> To check database connectivity we can check logs

-> docker logs backend\_service



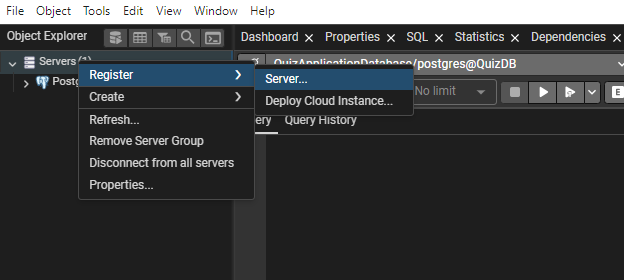


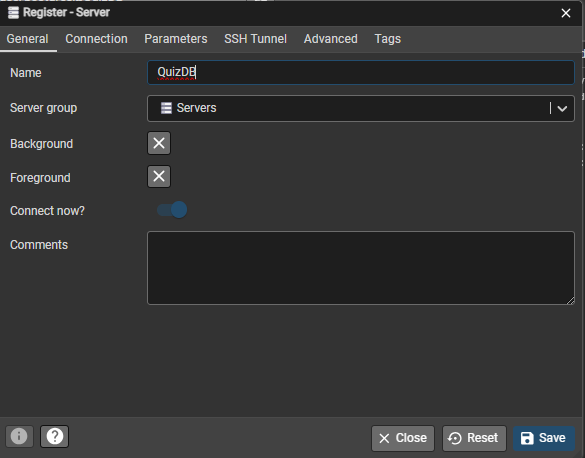
-> The message shows database is connected to backend service

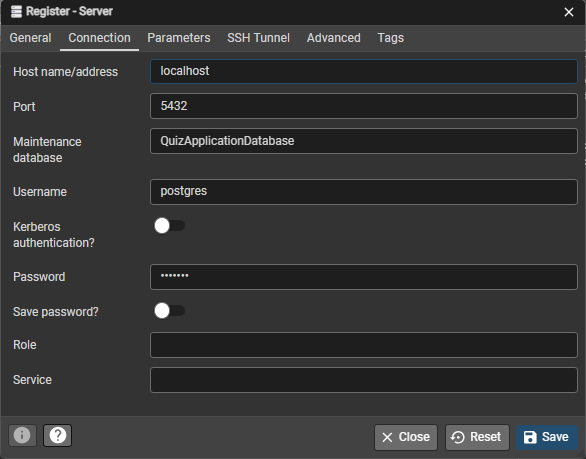
**6. Connect to Services for Verification:**

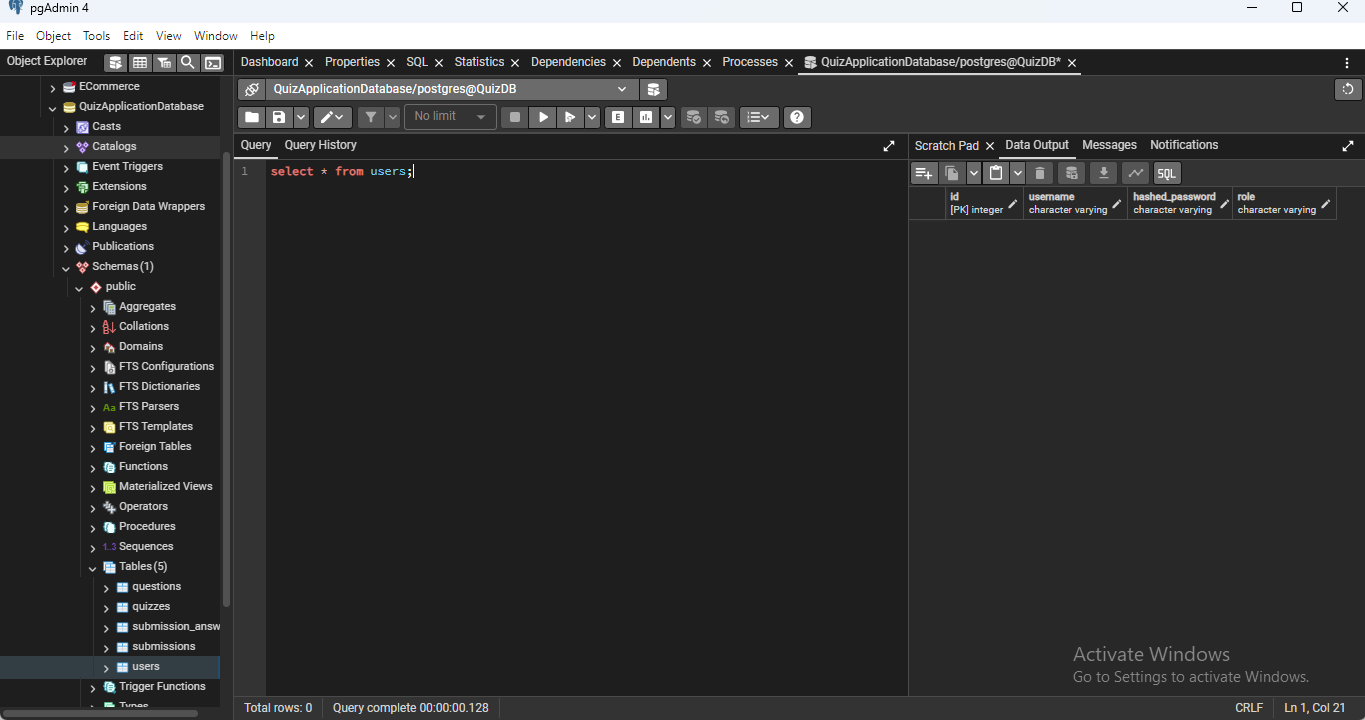
**Connect to the database using tools like Pg Admin for PostgreSQL**

-> Connect using pg admin





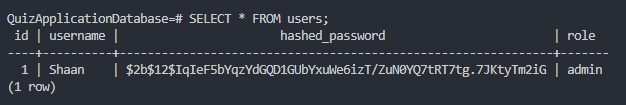




-> Using the docker exec command

-> docker exec -it db\_service psql -U quiz\_user -d quiz\_db

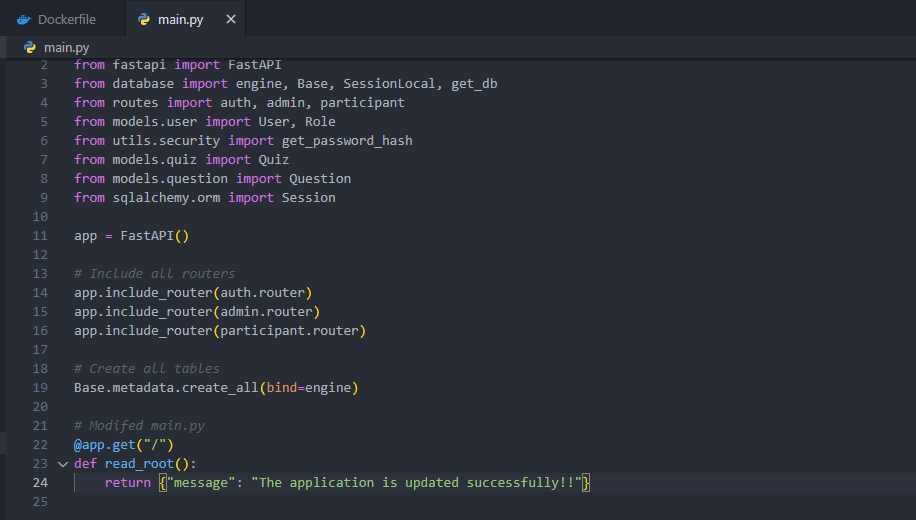




**7. Redeploy Changes**

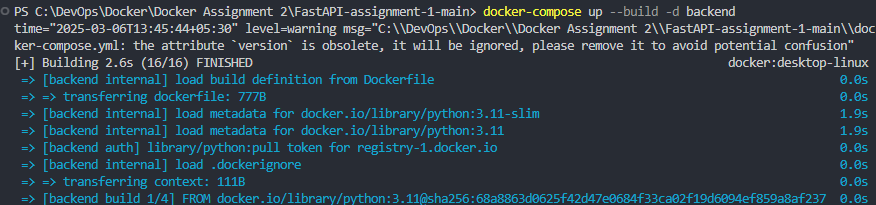
**Modify the application code, then save it and then redeploy only the application container without touching Redis/database container**

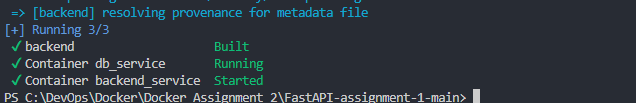
-> Modify main.py file



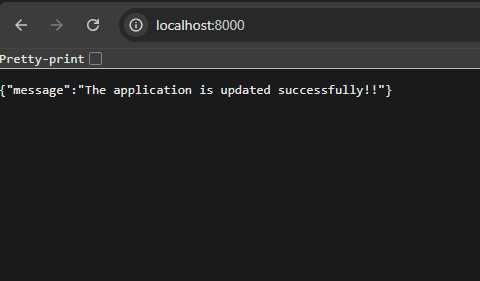
-> Rebuild and redeploy only the backend without affecting database

-> docker-compose up --build -d backend





-> open http://localhost:8000/

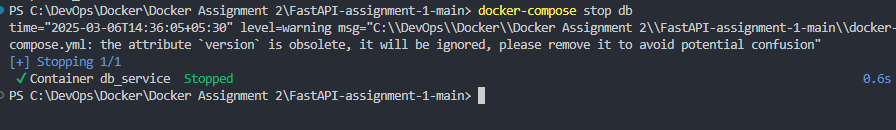


**8. Stop and delete the Redis/database container:**

**Stop the database/Redis service and delete these containers without touching the other containers. Ensure that the data is preserved.**

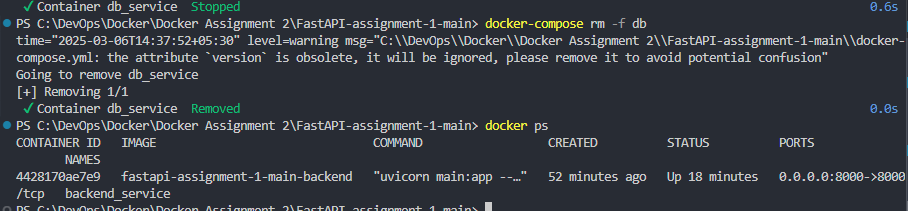
-> Stop database container

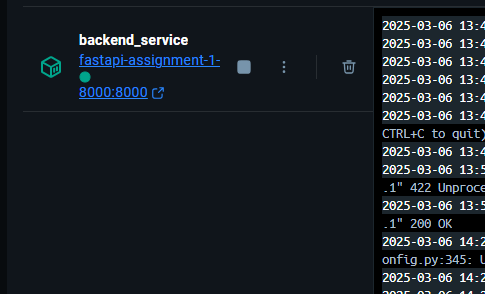
-> docker-compose stop db



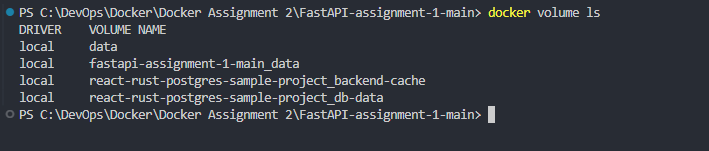
-> remove database container

-> docker-compose rm -f db





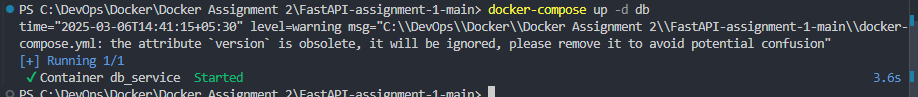
-> Verify That Data Is Still Present

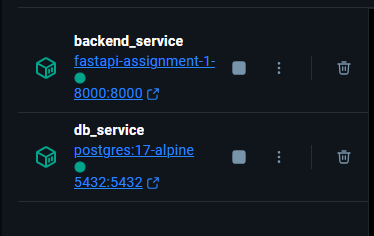


**Redeploy the Redis/database container ensuring the old data loads up automatically.**

-> Redeploy the Database Container

-> docker-compose up -d db





-> Verify That Old Data Is Still Present

-> docker exec -it db\_service psql -U postgres -d QuizApplicationDatabase

