Documentation: Building a FastAPI Application with PostgreSQL and Docker

# **Introduction**

This documentation guides you through the process of creating a FastAPI application that uses a PostgreSQL database. The application allows you to manage books, authors, and clients, demonstrating CRUD operations. The entire setup is containerized using Docker for easy deployment.

## **1. Database and Table Creation**

### **1.1 PostgreSQL Database**

Database Name: booklib

Username: postgres

Password: 123456789

### **1.2 Models Definition**

The database models (Author, Book, Client, Borrow) are defined in the models.py file using SQLAlchemy, a powerful SQL toolkit and Object-Relational Mapping (ORM) library for Python.

These models represent entities such as authors, books, clients, and book borrowing relationships.

### **1.3 Session Management**

A SessionLocal class is created in the database.py file to manage the database sessions using SQLAlchemy's sessionmaker.

### **1.4 Table Creation**

The Base.metadata.create\_all(bind=engine) statement in models.py initializes the database tables when the application starts.

## **2. FastAPI Application**

### **2.1 Main Application (main.py)**

The FastAPI application is defined in main.py.

It initializes the FastAPI instance and includes the API routers defined in api.py.

### **2.2 API Router (api.py)**

The API router (api.py) contains endpoints for creating books, authors, clients, and retrieving information about books.

Endpoints are secured using OAuth2 token-based authentication.

### **2.3 OAuth2 Token Authentication**

The application uses OAuth2 token authentication to secure API endpoints.

A simple token verification mechanism is implemented for demonstration purposes.

## **3. Dockerization**

### **3.1 Dockerfile**

The Dockerfile specifies the steps to build the Docker image for the FastAPI application.

It uses the official Python 3.9 slim image, sets the working directory, copies the application code, installs dependencies, and exposes port 80.

### **3.2 Docker Compose**

The docker-compose.yml file defines services for the FastAPI application (fastapi-app) and PostgreSQL database (postgres).

The services are interconnected, and the FastAPI application depends on PostgreSQL.

### **3.3 Environment Variables**

Environment variables are utilized to configure the PostgreSQL service, providing database name, username, and password.

## **4. Deployment**

After defining the application, running docker-compose up deploys both the FastAPI application and PostgreSQL database.

The FastAPI application is accessible on http://localhost:80.

# **Conclusion**

This documentation outlines the process of creating a FastAPI application with PostgreSQL, incorporating database and table creation, OAuth2 token authentication, and Dockerization for seamless deployment. It serves as a comprehensive guide for developers looking to build and deploy similar applications.