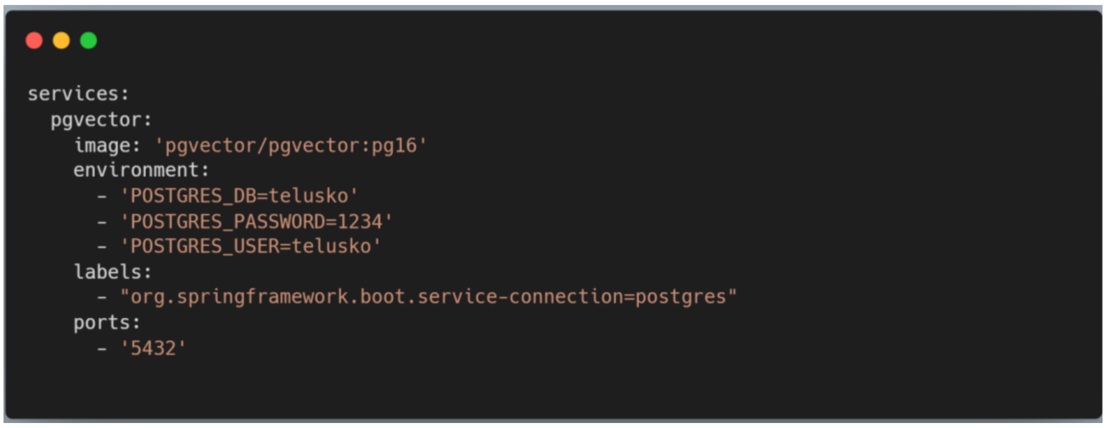
Lecture 25: PG Vector Store Configuration

Why Docker?

- We use **Docker Compose** to set up **PGVector** quickly.
- Running PGVector in a container saves us from manual installation.
- Command to start: **docker compose up -d**
- This launches a PostgreSQL instance with PGVector support.

Docker Compose File:

- Define the PGVector service in docker-compose.yml:

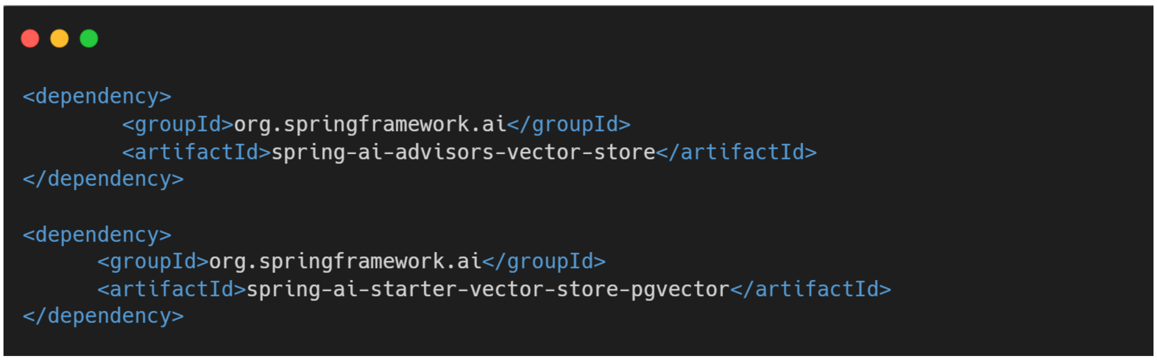


```yaml
services:
  pgvector:
    image: 'pgvector/pgvector:pg16'
    environment:
      - 'POSTGRES_DB=telusko'
      - 'POSTGRES_PASSWORD=1234'
      - 'POSTGRES_USER=telusko'
    labels:
      - "org.springframework.boot.service-connection=postgres"
    ports:
      - '5432'
```

- **Database Name:** telusko
- **User:** telusko
- **Password:** 1234
- **Port:** 5432
- Once started, verify that the container is running.

Dependencies:

- Apart from the base vector store, add the **PGVector starter** in pom.xml.
- This allows Spring AI to integrate with PGVector



```xml
<dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-advisors-vector-store</artifactId>
</dependency>

<dependency>
      <groupId>org.springframework.ai</groupId>
      <artifactId>spring-ai-starter-vector-store-pgvector</artifactId>
</dependency>
```

TELUSKO

Application Properties:

- Configure database connection in application.properties:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/telusko
spring.datasource.username=postgres
spring.datasource.password=1234
spring.jpa.show-sql=true
spring.datasource.driver-class-name=org.postgresql.Driver
```

Key Takeaways:

- PGVector setup requires:
  - **Docker Compose** for containerized PGVector.
  - **Maven dependency** for Spring AI PGVector starter.
  - **Database configuration** in application.propertie

TELUSKO