Title: Proper setup of docker compose with spring boot and postgres

Post Body:

I have a docker compose file where i have two services the `flamup` service is used for building my spring boot application and links to a postgres image defined by the `db` service.

Here's the docker-compose.yml file

```
version: '3.8'   volumes:   postgres_data:  services:    flamup:     build: .     container_name: flamup     environment:
```

Here's the Dockerfile for the spring boot application

```
FROM maven:3.6.1-jdk-8-slim AS build RUN mkdir -p workspace WORKDIR workspace COPY pom.xml /workspace COPY src /workspace/src
```

I am using the frontend-maven-plugin for bootstrapping the frontend with the spring boot application. Everything works fine locally.

However when i try to run it via docker-compose i made some changes in the application-properties.yml

```
server:   error:     include-message: always     include-binding-errors: always   spring:   datasource:     url : jdbc:postgre
```

However the build get's failed everytime, because of this

```
Caused by: org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'entityManagerFactory' defin
```

Things i have tried

- Hard coding the `DB_SERVER` variable and replacing with `localhost:5432` , `db` , `db:5432`. None of the seems to work.
- Adding depends on field in the `flamup` service.
- Adding networks field in both the services and connecting them via a bridge.
- Tried different versions of postgres and maven

Nothing seems to be the solution, any help will be highly appreciated.

Accepted Answer: None

Highest Rated Answer:

So a working solution to the problem (though may not be that efficient), is running the maven build by skipping the tests. Apparently hibernate results in running the JPA statements which results in failure of JDBC postgres exception as there is yet no container of postgres running.

Thus changing the dockerfile to

```
RUN mvn -f pom.xml clean install -DskipTests=true
```

instead of

```
RUN mvn -f pom.xml clean package
```

caused in successful creation of the jar file which could be run as a docker container.