Title: Docker: Springboot container can not connect to PostgreSql Container Connection error

Post Body:

I am building my first Springboot 2.0 application. I am trying to put my Springboot application into one docker container and my PostgresDB into another container.

My Dockerfile

```
    FROM frolvlad/alpine-oraclejdk8:slim     VOLUME /tmp     ADD springboot-api-demo-0.1*.jar app.jar     RUN sh -c 'touch /ap
```

My docker-compose.yml file

```
version: '2.1'  services:   springboot-api-demo:     image: 'fw/springboot-api-demo'     mem_limit: 1024m     ports:       - '
```

I am using Springboot JPA Data 2.0 with below config data in my **application.properties**

```
spring.datasource.url= jdbc:postgresql://localhost:54321/java_learning spring.datasource.username=postgres spring.datasource.p
```

I can test that Both of the Images are up. Also from docker log and docker events, I see that postgres Container is running fine, even I can access it and also created a DB too. But springboot container started but i died because it could not connect to postgress and throwing error below.

> Unable to obtain connection from database: The connection attempt failed

Note that my host machine already has Postgres on port 5432 thats why I did a port mapping ofr 54321:5432 on my postgres container. Here is Proof :) -

```
➜  springboot-api-demo git:(master) ✗ lsof -i:54321              COMMAND     PID             USER   FD   TYPE             DE
```

I am not sure what is the problem. But my Springboot application is not able to connect my postgres container which is running fine with proper creadentials.

Accepted Answer: None

Highest Rated Answer:

Try with :

```
spring.datasource.url= jdbc:postgresql://pgdb:5432/java_learning
```

The postgres database is not running on localhost, it's running in the other container which has an other IP (yet unknown).

Thanksfully, docker-compose automatically create a network shared among all the containers in the docker-compose.yml (unless explicitly said to do not), as a result you can magically use the **service name as an hostname**.

Also, you have a typo in the port, Postgres use `5432` by default, not `54321`