

Title: Docker and Eureka with Spring Boot failing to register clients

Post Body:

I have a very simple demo of using Spring Boot + Docker Compose + Eureka.

My server runs on port 8671 with the following application properties:

```
server:    port: 8761 eureka:    instance:    prefer-ip-address: true    client:    registerWithEureka: false    fetchRegistry
```

My Eureka client runs on port 9000 with the following application properties:

```
server:    port: 9000 spring:    application:    name: user-registration eureka:    client:    registerWithEureka: true    fet
```

When I start up my docker.compose file in the parent maven project, this is the contents of my docker-compose file:

```
eureka-server:    image: rosenthal/eureka-server ports:    - '8761:8761' user-registration:    image: rosenthal/user-registratio
```

When I run my application by first starting the eureka server, following by the client via

```
mvn spring-boot:run
```

The server successfully registers my client (I call it user-registration).

When I run my application through docker-compose, the client fails to register with the following output:

```
DiscoveryClient_USER-REGISTRATION/0fd640cbc3ba:user-registration:9000:    registering service... user-registration_1 | 2017-
```

My first assumption was that running docker-compose ran into a race condition on waiting for the server to start, but my eureka client seems to have a heartbeat trying to call home to the server it's configured with. This means it's just not able to find the Eureka server I have registered (and is running, I can navigate to it on localhost:8671).

What am I missing here? Everything runs fine running locally with spring-boot starting up with it's own embedded tomcat containers. As soon as I start to do it with docker-compose, it doesn't want to work.

EDIT

I realized my problem, I believe. So docker doesn't run on localhost, it runs on the public IP it is assigned when I start up docker. Navigating to this ip + port shows my service running for Eureka Server. The client still doesn't register.

SO, I made changes to the application.yml file for my eureka client to:

```
serviceUrl:    defaultZone: http://192.168.59.103:8761/eureka/
```

That IP is the one my docker daemon is running under. Now, it misses the first registration when I do docker-compose, but the second heartbeat picks up my client.

How can I ensure the client waits until the server is FULLY up? I used the proper docker 'links' field in my docket compose file, but it didn't work as I hoped. Additionally, how can I see the defaultZone file to be my DOCKER_HOST IP?

Final result

The resulting docker-compose file that got everything working for me was:

```
eureka-server:    image: thorris/eureka-server    ports:    - '8761:8761' user-registration:    image: thorris/user-registratio
```

Accepted Answer:

If useful for you, that's in some way how I configure it for my production environment (docker-compose file version 2):

```
version: '2' services:    eureka-server:    image: rosenthal/eureka-server    expose:    - '8761'    user-registration:    i
```

From the [docs](#) that's what expose does:

Expose ports without publishing them to the host machine - they'll only be accessible to linked services. Only the internal port can be specified.

As you've got everything in the same network, containers can see each other with no links between them.

SIDE NOTE

Keep in mind that with this configuration port 9000 will be publicly accessible at the host machine, and mapped to the 8080 port of the user-registration container.

Highest Rated Answer:

Set an environment property to override the `eureka.client.serviceUrl.defaultZone` to match the service name in your docker compose file.

```
eureka-server:  image: rosenthal/eureka-server  ports:      - '8761:8761' user-registration:  image: rosenthal/user-registrat
```

This will override the property from the packaged `application.properties`.

NOTE: As mentioned in the comments you don't need the `links` section in the compose file. I removed it as such. See <https://docs.docker.com/compose/networking/> for info on that.