Title: Docker : java.net.ConnectException: Connection refused - Application running at port 8083 is not able to access other application on port 3000

Post Body:

I have to consume an external rest API(using restTemplate.exchange) with Spring Boot. My rest API is running on port 8083 with URL http://localhost:8083/myrest (Docker command : `docker run -p 8083:8083 myrest-app`)

External API is available in form of public docker image and after running below command , I am able to pull and run it locally.

```
 docker pull dockerExternalId/external-rest-api docker     run -d -p 3000:3000 dockerExternalId/external-rest-api
```

a) If I enter external rest API URL, for example http://localhost:3000/externalrestapi/testresource directly in chrome, then I get valid JSON data.

b) If I invoke it with myrest application from eclipse(Spring Boot Application), still I am getting valid JSON Response. (I am using Windows Platform to test this)

c) But if I run it on Docker and execute myrest service (say http://localhost:8083/myrest), then i am facing `java.net.ConnectException: Connection refused`

**More details :**

```
org.springframework.web.client.ResourceAccessException: I/O error on GET request for 'http://localhost:3000/externalrestapi/te
```

**P.S - I am using Docker on Windows.**

Accepted Answer:

~~# The problem~~

~~You run with:~~

~~`docker run -p 8083:8083 myrest-app`~~

~~But you need to run like:~~

~~`docker run --network 'host' --name 'app' myrest-app`~~

~~So passing the flag `--network` with value `host` will allow you container to access your computer network.~~

Please ignore my first approach, instead use a better alternative that does not expose the container to the entire host network... is possible to make it work, but is not a best practice.

## A Better Alternative

Create a network to be used by both containers:

```
docker network create external-api
```

Then run both containers with the flag `--network external-api`.

```
docker run --network 'external-api' --name 'app' -p 8083:8083 myrest-app
```

and

```
docker run -d --network 'external-api' --name 'api' -p 3000:3000 dockerExternalId/external-rest-api
```

The use of flag `-p` to publish the ports for the `api` container are only necessary if you want to access it from your computers browser, otherwise just leave them out, because they aren't needed for 2 containers to communicate in the `external-api` network.

> **TIP**: docker pull is not necessary, once docker run will try to pull the image if does not found it in your computer. Let me know how it went...

## Call the External API

So in both solutions I have added the `--name` flag so that we can reach the other container in the network.

So to reach the external api from my rest app you need to use the url `http://api:3000/externalrestapi/testresource`.

Notice how I have replaced `localhost` by `api` that matches the value for `--name` flag in the docker run command for your external api.

Highest Rated Answer:

From your `myrest-app` container if you try to access `http://localhost:3000/externalrestapi/testresource`, it will try to access `3000` port of the same `myrest-app container`.

Because each container is a separate running Operating System and it has its own network interface, file system, etc.

`Docker is all about Isolation`.

There are 3 ways by which you can access an API from another container.

1. Instead of `localhost`, provide the IP address of the external host machine (i.e the IP address of your machine on which docker is running)
2. Create a docker network and attach these two containers. Then you can provide the `container_name` instead of `localhost`.
3. Use --link while starting the container (deprecated)