Title: docker-compose inter container communication

Post Body:

I'm currently experimenting with Spring Boot-based microservices and getting to grips with docker, but I'm hitting a snag.

Basically what I'm trying to do is containerize 2 small services: a spring cloud config service and a spring cloud eureka service (discovery service). The eureka service fetches its configuration from the config service.

Both services are separate projects with their own Dockerfiles:

Dockerfile-cloud-config-service:

```
FROM openjdk:10.0.2-13-jre-sid ENV APP_FILE cloud-config-service.jar ENV APP_HOME /usr/apps EXPOSE 8888 COPY target/$APP_FILE
```

Dockerfile-discovery-service:

```
FROM openjdk:10.0.2-13-jre-sid ENV APP_FILE discovery-service.jar ENV APP_HOME /usr/apps EXPOSE 8761 COPY target/$APP_FILE $AP
```

Using docker-compose I'm trying to tie them together using the following **docker-compose.yml:**

```
version: '3.7' services:   cloud-config-service:     container_name: cloud-config-service     build:       context: cloud-conf
```

At first I configured the discovery-service to fetch its configuration from *http://localhost:8888*, but after some digging I found that localhost in a container refers to the container itself and found in the Docker documentation that services can refer to each other using their names. So I changed discovery-service's properties to fetch its config from *http://cloud-config-service:8888*. That doesn't work, hence this post.

Both Dockerfiles build and run just fine, except the fact that the discovery-service fails to GET the config-service on *http://cloud-config-service:8888*.

It does work if I use the `host` network driver and the *http://localhost:8888* endpoint, but this 'feels' hacky and not how it is supposed to be done.

I'm probably missing something trivial, but I'm afraid I can't find what.

EDIT: Small snippet of discovery-service's console log:

```
discovery-service       | 2018-10-02 13:14:26.798  INFO 1 --- [           main] c.c.c.ConfigServicePropertySourceLocator : Fet
```

Accepted Answer: None

Highest Rated Answer:

Firstly, communication between docker containers is a subset of a much bigger problem prevalent in distributed services - You don't know what service (and hence their dependencies) will go down at any moment, and so you should take into account such failures when you build your application.

The problem that you are facing is common, even more so with Docker containers, and I believe inter-container communication is a major piece in Docker that is under frequent development changes.

To address your problem, first, I would like to put forth some points -

1. `localhost` from within a container will refer to that container itself.
2. `localhost` on your machine does in fact refer to your local host and will be mapped with services you map through the `ports` configuration for each of your services in the `docker-compose` file.
3. `depends_on` only waits for the container to start and not for the actual process to start running - which might mean that the service you are waiting for isn't necessarily up and running yet, and so, cause timeouts for the dependent service(s).

What you need is to wait for the service to start running, not just the container to be up. There are two possible ways you can accomplish this -

> Specify a `restart` policy for your `discovery-service` based on failure. In your case, failure would be when it times out while connecting to the `cloud-config-service`. Something like `restart: on-failure:10` which means you are asking docker to restart the `discover-service` when it fails with a maximum retry of 10. That way, you would have given reasonable time for the other container (service) to be up and running and make sure that the container with the restart policy *eventually* connects to that one.
>
> Use another tool like dockerize that allows you to wait on other services before starting up the container.

Also, to make sure you are debugging the problem correctly, be sure to check the logs of your container to see what the issue really is - `docker logs -f --tail=100 <container_name/container_id>`.

Hope this helps.