Title: Docker Compose with Zookeeper, Kafka, Redis, and Java Spring Boot

Post Body:

I'm having difficulty linking these containers together using Docker Compose, let me preface by saying that I am currently running on a Mac as well.

The application is currently working without the use of Docker Compose, in that if I run these all individually (not with Docker) the application works as intended.

As intended means that the application is reading from Redis as well as pulling data that comes across certain Kafka topics and displaying them on the front-end.

On to what I believe are the necessary files.

`docker-compose.yml`

```
version: '2' services:   zookeeper:     image: wurstmeister/zookeeper     ports:       - '2181:2181'   kafka:     image: wurst
```

The `Dockerfile` that is being referenced under `kafka-websocket-connector` is as follows:

`Dockerfile`

```
FROM frolvlad/alpine-oraclejdk8:slim VOLUME /tmp ADD target/kafka-websocket-connector-0.0.1-SNAPSHOT.jar app.jar ENV JAVA_OPTS
```

If I attempt to run the command `docker-compose up --build` I receive the following error:

```
ERROR: Service 'kafka-websocket-connector' failed to build: ADD failed:      stat /var/lib/docker/tmp/docker-builder838069739/
```

I don't know if this necessarily relates to the connecting these components, but my `build.gradle` for the Spring application is as follows:

`build.gradle`

```
buildscript {     ext {         springBootVersion = '1.5.4.RELEASE'     }     repositories {         mavenCentral()     }
```

So to restate the problem, I am having trouble connecting the following components using Docker Compose: Zookeeper, Kafka, Redis, and a Spring Boot application.

Accepted Answer: None

Highest Rated Answer:

The `KAFKA_ADVERTISED_HOST_NAME: localhost` environment variable in your `kafka` service is likely the cause of the problem. It needs to be reachable by other containers. Try change it to `KAFKA_ADVERTISED_HOST_NAME: kafka`.

All the containers of the services defined in your docker-compose.yml are added to a user-defined network, named after your compose project. The containers in this network are reachable and discoverable by other each, using hostnames that are identical to the container name. You can find out more information on Docker Compose networking in their documentation.