Title: How do I have my jar re-deployed and put into docker image every time I run compose?
Post Body:

So I know that there are a lot of tutorials on the topics, both docker and maven, but I'm having some confusion in combining them alltogether.

I created a multi-module Maven project with 2 modules, 2 spring applications, let's call them application 1 and application 2.

Starting each other via IntelliJ IDEA green "run" button works fine, now I'd like to automate things and run via docker.

I have Dockerfiles that looks the same in both cases: (in both modules it's the same, only JAR name's different)

```
FROM adoptopenjdk:11-jre-hotspot MAINTAINER *my name here lol* ADD https://github.com/ufoscout/docker-compose-wait/releases/do
```

I also have docker-compose:

```
version: '2.1' services:   application1:     container_name: app1     build:       context: ../app1     image: docker.io/mynam
```

What I do currently is:

- I run maven package for each module and receive files like "application1(-2)-0.0.1-SNAPSHOT-jar-with-dependencies.jar" in both target folders.
- "docker build -t springio/app1 ."
- "docker-compose up --build"

And it works, but I feel I do some extra steps. How can I do the project so that I ONLY have to run docker compose? (after each time I change things in the code)

Again, I know it's a quite simple thing but I kinda lost the logic.

Thanks!

P.S Ah, and about the "...docker-compose-wait/releases/download/2.9.0/wait /wait" It's important that app start one after another, tried different solutions, unfortunately, doesn't really work as good as I would like to. But I guess I'll leave it as is.

Accepted Answer: None
Highest Rated Answer:

So, again, if anyone ever wonders how to do the things I asked, here's the answer: you need multi-stage build Dockerfile. It'll look like this:

```
# # Build stage # FROM maven:3.6.0-jdk-11-slim AS build COPY src /home/app/src COPY pom.xml /home/app RUN mvn -f /home/app/pom
```

What it does is it basically first creates a jar file, copies it into package stage and eventually runs. That's allow you to run your app in docker by running only docker compose.