Title: Development workflow for Spring Boot + Maven + Docker + IntelliJ

Post Body:

I would like to ask for some recommendations for development workflow for application with stack mentioned in the title. Before I switched to use Docker all I had to do was:

1. Go to start.spring.io and download project starter
2. Import it into intelliJ
3. Develop features, hit green arrow to start app or red square to stop and repeat it with every change in code

Now when I switched to docker, after step 2, I do this:

1. Create Dockerfile and docker-compose.yml (where I start my app and also mysql service).
2. Right click on docker-compose and hit run. Then it builds my app image (i use --build flag in my run configuration so it builds images every time it I hit run on docker-compose) and starts two services: app and mysql, and everything works.

The problem is when I change sth in my code then I have to:

1. Execute mvn clean and install steps manually, to produce new jar under /target folder
2. Then stop previous docker compose and run it again. Then it builds new images from what is in /target

I would rather like to have something like one-click solution, like it was before I started to use docker. So when I change code then I press only one button and new image is generated and run with all changes applied. Is it possible? Do I miss something? Could you tell me if your workflow is similar to mine? Maybe you could recommend some tools or different config?

Accepted Answer:

You can set up Spring Boot dev tools to live reload inside a Docker container.

Ensure spring-boot-devtools dependency is in your pom.xml:

```
<dependency>   <groupId>org.springframework.boot</groupId>   <artifactId>spring-boot-devtools</artifactId>   <scope>runtime</s
```

Then create a docker-compose.yml file with

```
version: '3.1' services:   backend:     image: maven:3.6.3-jdk-8     command: mvn spring-boot:run     ports:       - 8000:8000
```

This uses the Maven Docker image so when you run `docker-compose up`, it will run the image and map your source code folder as a volume. Then maven will run the application using `mvn spring-boot:run`

Whenever you make a change to the source code, it will reload with the same functionality as `mvn spring-boot:run`.

Highest Rated Answer:

1. Create a separate release module - *project-name*-release - which brings down the old image, build the new image and run/publish it
2. Use docker maven plugin in the config

Sample project - https://github.com/spring-guides/gs-spring-boot-docker