Title: Docker-compose - build with maven that re-uses the maven repository

Post Body:

When building my Spring-boot image using Maven I now do this inside the Dockerfile. Maven will download all dependencies and then compile my files. This takes quite some time.

How can I specify that the build process via docker-compose (Dockerfile) re-uses my 'Windows10 Home' Maven repository? So, the number of (new) downloads is minimal. My dev context: I use the Docker quickstart terminal, so using a docker-machine.

The is is a part of my docker-compose.yml file:

```
version: '3' services:   spring-boot-app:     image: spring-boot-image     build:       context: ./       dockerfile: Dockerfi
```

My Dockerfile is:

```
FROM java:8 FROM maven:alpine WORKDIR /app COPY . /app RUN mvn -v RUN mvn clean install -DskipTests EXPOSE 8080 LABEL maintain
```

Accepted Answer:

**@Jack Gore** - thank you very much for pointing me the way. I had seen the post, but there was not a definitive answer yet. After diving in all seperate answers, the info provided me both an answer to the symptom as well with insights how to solve similar questions.

**ANSWER**: Via the Dockerfile you can build the final images via 1 or more subsequent image layers. To prevent the step from re-downloading the dependencies each time, you can make an image layer for the 'downloading the world' of dependencies. Re-downloading will only be done when a dependency changes in the pom.xml file.

To do so, you split the Dockerfile in 2 build steps: first copy the pom.xml file and build it. That will create an image layer with all dependencies. As a second step you build the application.

```
FROM java:8 FROM maven:alpine  # image layer WORKDIR /app ADD pom.xml /app RUN mvn verify clean --fail-never  # Image layer: w
```

Then you get the following build scenario's:

- The **first time** you build this (docker build .) the dependencies are downloaded and as step 2 the application jar is build.
- When you rebuild immediately the dependencies (pom.xml) and the application sources were not changed. So, the image layers don't need to be changed. The build is ready in no time.
- If you change 1 of your application source files, only a few downloads are downloaded and the application is build. So you are NOT downloading the world.
- If you change the pom.xml file, thus changing the dependencies, then all dependency downloads are done.

The impact of seperating image layers is shown via a number of very short videos.

The disadvantage of this way of building is of course that the final (production) image contains more than the appication. Not only the JAR is in it, but also a number of dependencies.

How to solve this iamge being far too big:

- Reduce the image size via a trick. Use the option: --squash. This is explained in this post.
- Seperate the build process (here: maven and/or ng build --prod) as a step before the docker build step. Then execute the docker build and put ONLY the jar file in final image.
- Use an CI/CD environment with e.g. a Jenkinsfile (pipeline). In the Jenkins pipeline you first build the image. All dependencies are already there. You only rebuild the application. The you perform a docker build, etc.

For my case this is the best option possible. You automate the process AND keep the image size low.

Highest Rated Answer: None