Title: Error when running docker container 'NoClassDefFoundError'

Post Body:

I am trying to dockerize a simple Spring Boot Application, built with Maven.

Dockerfile:

```
FROM openjdk:latest COPY target/backend-1.0-SNAPSHOT.jar app.jar ENTRYPOINT ['java','-jar','app.jar']
```

When I run the .jar without the container (`java -jar target/backend-1.0-SNAPSHOT.jar`), everything works fine and the app is running.

Now I create the container with `docker build -t company/backend .`

But when I try to run the docker container with `docker run -p 8080:8080 company/backend` the following error occurs:

```
Exception in thread 'main' java.lang.NoClassDefFoundError: org/springframework/boot/SpringApplication         at de.company.ba
```

It seems like docker does not find the main class, even though it is defined in my pom.xml:

```
<properties>     <maven.compiler.source>1.8</maven.compiler.source>     <maven.compiler.target>1.8</maven.compiler.target>
```

Main Class:

```
package de.company.backend;  import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.
```

Accepted Answer:

In your pom.xml, the `copy-dependencies` goal is specified at the `install` phase : too late the package of the jar was already done.

> I am trying to dockerize a simple Spring Boot Application, built with Maven.

You don't need to declare any plugin to create a fat jar with spring boot that could be run by a docker container.
Declaring these plugins is error prone (and should be used only in corner cases) while the `repackage` goal of the spring boot maven plugin attached by default to the package phase of maven will create for you the fat jar :

> Repackages existing JAR and WAR archives so that they can be executed from the command line using java -jar

Juste remove these plugins declarations and execute `mvn clean package` and it should be good.

Side note :

> FROM openjdk:latest

Don't use `latest` as image version but favor a specific version of the image othewhise you could have bad surprises. As you use JDK 8, you could specify a JRE or a JDK 8 such as : `FROM openjdk:8-jre-alpine`.

Highest Rated Answer:

I had the same problem as you.

you need to add plugin in your pom.xml.

```
     <build>         <plugins>             <plugin>                 <groupId>org.springframework.boot</groupId>
```

If you input as above, it works normally.

and check MANIFEST.MF (in .jar file)

```
Main-Class: org.springframework.boot.loader.JarLauncher    Start-Class: {your main class}
```