**Docker File for Simple Spring Boot App:**

FROM openjdk:8-jdk-alpine

VOLUME /tmp

ARG JAR\_FILE

COPY ${JAR\_FILE} app.jar

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]

* We added a VOLUME pointing to "/tmp" because that is where a Spring Boot application creates working directories for Tomcat by default. The effect is to create a temporary file on your host under **"/var/lib/Docker**" and link it to the container under "/tmp". This step is optional for the simple app that we wrote here, but can be necessary for other Spring Boot applications if they need to actually write in the file system.
* To reduce Tomcat startup time, we added a system property pointing to "/dev/urandom" as a source of entropy.
* if you are using boot2docker you need to run it first before you do anything with the Docker command line or with the build tools (it runs a daemon process that handles the work for you in a virtual machine).

**Build a Docker Image with Maven:**

**In the Maven pom.xml you should add a new plugin like this:**

**Pom.xml:**

<properties>

<docker.image.prefix>springio</docker.image.prefix>

</properties>

<build>

<plugins>

<plugin>

<groupId>com.spotify</groupId>

<artifactId>dockerfile-maven-plugin</artifactId>

<version>1.3.6</version>

<configuration>

<repository>${docker.image.prefix}/${project.artifactId}</repository>

<buildArgs>

<JAR\_FILE>target/${project.build.finalName}.jar</JAR\_FILE>

</buildArgs>

</configuration>

</plugin>

</plugins>

</build>

<executions>

<execution>

<id>default</id>

<phase>install</phase>

<goals>

<goal>build</goal>

<goal>push</goal>

</goals>

</execution>

</executions>

**The configuration specifies 3 things:**

* The repository with the image name, which will end up here as springio/gs-spring-boot-docker
* The name of the jar file, exposing the Maven configuration as a build argument for docker.
* Optionally, the image tag, which ends up as latest if not specified. It can be set to the artifact id if desired.

Before proceeding with the following steps (which use Docker’s CLI tools), make sure Docker is properly running by typing docker ps. If you get an error message, something may not be set up correctly. Using a Mac? Add $(boot2docker shellinit 2> /dev/null) to the bottom of your .bash\_profile (or similar env-setting configuration file) and refresh your shell to ensure proper environment variables are configured.

You can build a tagged docker image using the command line like this:

$ ./mvnw install dockerfile:build

And you can push the image to dockerhub with “ ./mvnw dockerfile:push” .

You do NOT have to register with Docker or publish anything to run a Docker image. You still have a locally tagged image, and you can run it like this:

$ docker run -p 8080:8080 -t springio/gs-spring-boot-docker

<http://www.springboottutorial.com/hibernate-jpa-tutorial-with-spring-boot-starter-jpa>