Packaging a Java Application in a Docker Image with Maven – Take 2, Fabric8

By Ivan Krizsan | March 12, 2017 0 Comment

Contents [show]

This article is an addition to my previous article on how to package a Java application in a Docker image with Maven. This time I will use the Fabric8 Docker Maven plug-in instead of the Spotify Dockerfile Maven plug-in, since I found out that it does support creation of Docker images from a Dockerfile. After having tried the Fabric8 plug-in out I also found that it is better than the Spotify plug-in in that it deletes any old images without having been explicitly told to. As in the previous article, the example will be a Spring Boot web application which will be packaged into a Docker image.

Preparations

The preparations consists completing the example project I used in the previous article.

Maven Profile

To create the Docker image containing the Spring Boot application, I'll use the following Maven plug-ins:

- maven-resources-plug-in
  Copies files to a directory in which the Docker image will be built.
  The files needed to build the Docker image are copied to a directory in the target directory of the project.
- maven-dependency-plug-in
  Copies the JAR file containing the application and any additional Maven artifacts to the Docker image build directory.
- Fabric8 Docker Maven plug-in
  Builds the Docker image. The GitHub project is located here and the documentation can be found here.

As in the previous article, I use a Maven profile to contain what is needed to build the Docker image.
The updated Docker image building Maven profile looks like this:

```
    <profiles>
```

Privacy & Cookies Policy

```xml
        <!--
            This profile builds a Docker image with the Spring Boot application.
            The Docker image is built using the following command:
            mvn -Pdockerimage docker:build

            If a Docker image with the image name and tag (project version) already
            exists then it will be replaced.
        -->
        <profile>
            <id>dockerimage</id>
            <dependencies>
                <!--
                    Here you declare dependencies to additional artifacts that
                    are to be copied into the Docker image.
                    No need to add a dependency to the Spring Boot application JAR
                    file here.
                -->
            </dependencies>
            <properties>
                <!-- Name of Docker image that will be built. -->
                <docker.image.name>springboot-webapp</docker.image.name>
                <!--
                    Directory that holds Docker file and static content
                    necessary to build the Docker image.
                -->
                <docker.image.src.root>src/main/resources/docker</docker.image.src.root
                <!--
                    Directory to which the Docker image artifacts and the Docker
                    file will be copied to and which will serve as the root directory
                    when building the Docker image.
                -->
                <docker.build.directory>${project.build.directory}/dockerimgbuild</dock
                <!-- URL to the Docker host used to build the Docker image. -->
                <docker.host.url>http://localhost:2376</docker.host.url>
                <!-- Name of the Dockerfile the Docker image will be built from. -->
                <docker.file.name>Dockerfile</docker.file.name>
            </properties>
            <build>
                <plugins>
                    <!--
                        Copy the directory containing static content to build directory
                    -->
                    <plugin>
                        <artifactId>maven-resources-plugin</artifactId>
                        <executions>
                            <execution>
                                <id>copy-resources</id>
                                <phase>package</phase>
                                <goals>
                                    <goal>copy-resources</goal>
                                </goals>
                                <configuration>
                                    <outputDirectory>${docker.build.directory}</outputD
                                    <resources>
                                        <resource>
                                            <directory>${docker.image.src.root}</direct
                                            <filtering>false</filtering>
                                        </resource>
                                    </resources>
                                </configuration>
                            </execution>
                        </executions>
                    </plugin>
                    <!--
                        Copy the JAR file containing the Spring Boot application
                        to the application/lib directory.
                    -->
                    <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-dependency-plugin</artifactId>
                        <executions>
                            <execution>
                                <id>copy</id>
                                <phase>package</phase>
                                <goals>
                                    <goal>copy</goal>
                                </goals>
                                <configuration>
                                    <artifactItems>
                                        <artifactItem>
                                            <!--
                                                Specify groupId, artifactId, version an
                                                artifact you want to package in the Doc
                                                In the case of a Spring Boot applicatio
                                                the same as the project group id, artif
                                                and version.
                                            -->
                                            <groupId>${project.groupId}</groupId>
                                            <artifactId>${project.artifactId}</artifact
                                            <version>${project.version}</version>
                                            <type>jar</type>
                                            <overWrite>true</overWrite>
                                            <outputDirectory>${docker.build.directory}/
                                            <!--
                                                Specify the destination name as to have
                                                to refer to in the Dockerfile.
                                            -->
                                            <destFileName>springboot-webapp.jar</destFi
                                        </artifactItem>
                                        <!-- Add additional artifacts to be packaged in

                                    </artifactItems>
                                    <outputDirectory>${docker.build.directory}</outputD
                                    <overWriteReleases>true</overWriteReleases>
                                    <overWriteSnapshots>true</overWriteSnapshots>
                                </configuration>
                            </execution>
                        </executions>
                    </plugin>

                    <!--
                        Build the Docker image.
                    -->
                    <plugin>
                        <groupId>io.fabric8</groupId>
                        <artifactId>docker-maven-plugin</artifactId>
                        <version>0.19.0</version>
                        <configuration>
                            <dockerHost>${docker.host.url}</dockerHost>
                            <images>
                                <image>
                                    <name>${docker.image.name}</name>
                                    <build>
                                        <tags>
                                            <tag>${project.version}</tag>
                                            <tag>latest</tag>
                                        </tags>
                                        <dockerFile>${docker.build.directory}/${docker.
                                    </build>
                                </image>
                            </images>
                        </configuration>
                    </plugin>
                </plugins>
            </build>
        </profile>
    </profiles>
```

Privacy & Cookies Policy

Noteworthy about this new version of the profile:

- The value of the docker.build.directory property has changed.
  The reason for this is that the Fabric8 Docker plug-in uses the directory previously used, so I had to change it.
- A property docker.host.url has been introduced.
  With the Fabric8 Docker plug-in you do not have to rely on an environment variable, but can configure the URL to the Docker host API in the Maven plug-in.
- A property docker-file.name has been introduced.
  Although I do not expect this to change frequently, I created this property that specifies the name of the Dockerfile from which the Docker image is to be created.
- The configuration of the Maven resources plug-in has not changed.
- The configuration of the Maven dependencies plug-in has not changed.
- The Spotify Docker-Maven plug-in has been removed.
  Since the Fabric8 plug-in is capable of deleting existing Docker images with the same name, the Spotify Docker-Maven plug-in is no longer needed.
- Finally there is the Fabric8 Docker Maven plug-in.
  This is the plug-in responsible for building the Docker image. In fact, as configured in the example above, there will be two Docker images built; one with the tag being the same as the version of the project and one with the latest tag. If you do not want to create an image with the latest tag, remove the line containing <tag>latest</tag> in the configuration.
- The latest version of the Fabric8 Docker Maven plug-in was, at the time of writing, 0.19.0.
  You may want to update to a more recent version if one is available.

Create the Docker Images

Creating the Docker images with the Fabric8 plug-in is a little different, at least with the configuration above. Open a terminal window and go to the directory that contains the pom.xml file of the example Spring Boot web application and build the Docker image using this Maven command:

```
mvn -Pdockerimage docker:build
```

After some time, the message BUILD SUCCESS should appear in the terminal and if you list the Docker images, you should see two new images listed:

```
$ docker images
REPOSITORY             TAG                  IMAGE ID             CREATED              SIZE
springboot-webapp      0.0.1-SNAPSHOT       da04079ef50a         6 seconds ago        194 M
springboot-webapp      latest               da04079ef50a         6 seconds ago        194 M
```

Use the Docker Images

The Docker images are launched in the same manner as described in the previous article. Depending on which of the two images you want to use, you change the tag accordingly.
To start a Docker container using one of the newly produced Docker images, I use the command:

```
docker run -p 8080:8080 springboot-webapp:0.0.1-SNAPSHOT
```

I can also use the following command:

Privacy & Cookies Policy

```
docker run -p 8080:8080 springboot-webapp:latest
```

Happy coding!

Category: Java Tags: docker , java , maven