

# Maven 插件之 docker-maven-plugin 的使用

 [blog.csdn.net/aixiaoyang168/article/details/77453974](https://blog.csdn.net/aixiaoyang168/article/details/77453974)

版权声明：本文为博主原创文章，未经博主允许不得转载。

<https://blog.csdn.net/aixiaoyang168/article/details/77453974>

## 目录

- docker-maven-plugin 介绍
- 环境、软件准备
- Demo 示例
  - 配置 DOCKER\_HOST
  - 示例构建镜像
    - 指定构建信息到 POM 中构建
    - 使用 Dockerfile 构建
  - 使用命令
  - 绑定 Docker 命令到 Maven 各个阶段
  - 使用私有 Docker 仓库地址
  - 安全认证配置
- FAQ

## 1、docker-maven-plugin 介绍

在我们持续集成过程中，项目工程一般使用 Maven 编译打包，然后生成镜像，通过镜像上线，能够大大提供上线效率，同时能够快速动态扩容，快速回滚，着实很方便。docker-maven-plugin 插件就是为了帮助我们在 Maven 工程中，通过简单的配置，自动生成镜像并推送到仓库中。

## 2、环境、软件准备

本次演示环境，我是在本机 Mac OX 上操作，以下是安装的软件及版本：

1. Docker : version 17.03.1-ce
2. Maven : version 3.3.9
3. Java : version 1.8.0\_91
4. docker-maven-plugin : 1.0.0

注意：这里我们要测试 Java Maven 项目用 docker-maven 插件打镜像，上传镜像等操作，所以需要先安装一下 Docker、Maven、Java，这里忽略安装过程。

## 3、Demo 示例

### 3.1 配置 DOCKER\_HOST

docker-maven-plugin 插件默认连接本地 Docker 地址为：localhost:2375，所以我们需要先设置下环境变量。

```
DOCKER_HOST=tcp://<host>:2375  
1
```

注意：如果没有设置 DOCKER\_HOST 环境变量，可以命令行显示指定 DOCKER\_HOST 来执行，如我本机指定 DOCKER\_HOST : `DOCKER_HOST=unix:///var/run/docker.sock mvn clean install docker:build` 。

### 3.2 示例构建镜像

构建镜像可以使用一下两种方式，第一种是将构建信息指定到 POM 中，第二种是使用已存在的 Dockerfile 构建。第一种方式，支持将 `FROM` , `ENTRYPOINT` , `CMD` , `MAINTAINER` 以及 `ADD` 信息配置在 POM 中，不需要使用 Dockerfile 配置。但是如果使用 `VOLUME` 或其他 Dockerfile 中的命令的时候，需要使用第二种方式，创建一个 Dockerfile，并在 POM 中配置 `dockerDirectory` 来指定路径即可。

这里我们以一个 Java Maven 项目 mavendemo 作为示例演示一下。

### 3.2.1 指定构建信息到 POM 中构建

```
<build>
  <plugins>
    <plugin>
      <groupId>com.spotify</groupId>
      <artifactId>docker-maven-plugin</artifactId>
      <version>1.0.0</version>
      <configuration>
        <imageName>mavendemo</imageName>
        <baseImage>java</baseImage>
        <maintainer>docker_maven docker_maven@email.com</maintainer>
        <workdir>/ROOT</workdir>
        <cmd>["java", "-version"]</cmd>
        <entryPoint>["java", "-jar", "${project.build.finalName}.jar"]</entryPoint>

        <resources>
          <resource>
            <targetPath>/ROOT</targetPath>
            <directory>${project.build.directory}</directory>
            <include>${project.build.finalName}.jar</include>
          </resource>
        </resources>
      </configuration>
    </plugin>
  </plugins>
</build>


- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

```

### 3.2.2 使用 Dockerfile 构建

```
pom.xml配置
<build>
  <plugins>
    <plugin>
      <groupId>com.spotify</groupId>
      <artifactId>docker-maven-plugin</artifactId>
      <version>1.0.0</version>
      <configuration>
        <imageName>mavendemo</imageName>
        <dockerDirectory>${basedir}/docker</dockerDirectory>

        <resources>
          <resource>
            <targetPath>/ROOT</targetPath>
            <directory>${project.build.directory}</directory>
            <include>${project.build.finalName}.jar</include>
          </resource>
        </resources>
      </configuration>
    </plugin>
  </plugins>
</build>
```

`\${basedir}/docker/Dockerfile 配置

```
FROM java
MAINTAINER docker_maven docker_maven@email.com
WORKDIR /ROOT
CMD ["java", "-version"]
ENTRYPOINT ["java", "-jar", "${project.build.finalName}.jar"]
• 1
• 2
• 3
• 4
• 5
• 6
• 7
• 8
• 9
• 10
• 11
• 12
• 13
• 14
• 15
• 16
• 17
• 18
• 19
• 20
• 21
• 22
• 23
• 24
• 25
• 26
• 27
• 28
• 29
• 30
• 31
```

以上两种方式执行 `docker:build` 效果是一样的，执行输出过程大致如下：

```
[INFO] --- docker-maven-plugin:1.0.0:build (default-cli) @ mavenDemo ---
[INFO] Building image mavendemo
Step 1/5 : FROM java
--> d23bdf5b1b1b
Step 2/5 : MAINTAINER docker_maven docker_maven@email.com
--> Using cache
--> 2faf180d4a50
Step 3/5 : WORKDIR /ROOT
--> Using cache
--> 862210f7956a
Step 4/5 : ENTRYPOINT java -jar mavenDemo.jar
--> Running in 96bbe83de6ec
--> c29009c88993
Removing intermediate container 96bbe83de6ec
Step 5/5 : CMD java -version
--> Running in f69b8d2a75b1
--> bc8d54014325
Removing intermediate container f69b8d2a75b1
Successfully built bc8d54014325
• 1
• 2
• 3
• 4
• 5
• 6
• 7
• 8
• 9
• 10
• 11
• 12
• 13
• 14
• 15
• 16
• 17
• 18
• 19
```

执行完成后，使用 `docker images` 查看生成的镜像：

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mavendemo	latest	333b429536b2	38 minutes ago	643 MB

- 1
- 2

### 3.3 执行命令

`mvn clean package docker:build` 只执行 build 操作

`mvn clean package docker:build -DpushImage` 执行 build 完成后 push 镜像

`mvn clean package docker:build -DpushImageTag` 执行 build 并 push 指定 tag 的镜像

注意：这里必须指定至少一个 imageTag，它可以配置到 POM 中，也可以在命令行指定。命令行指定如下：  
`mvn clean package docker:build -DpushImageTags -DdockerImageTags=imageTag_1 -DdockerImageTags=imageTag_2`，POM 文件中指定配置如下：

```

<build>
  <plugins>
    ...
    <plugin>
      <configuration>
        ...
        <imageTags>
          <imageTag>imageTag_1</imageTag>
          <imageTag>imageTag_2</imageTag>
        </imageTags>
      </configuration>
    </plugin>
    ...
  </plugins>
</build>


- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15

```

### 3.4 绑定Docker 命令到 Maven 各个阶段

我们可以绑定 Docker 命令到 Maven 各个阶段，我们可以把 Docker 分为 build、tag、push，然后分别绑定 Maven 的 package、deploy 阶段，此时，我们只需要执行 `mvn deploy` 就可以完成整个 build、tag、push 操作了，当我们执行 `mvn build` 就只完成 build、tag 操作。除此此外，当我们想跳过某些步骤或者只执行某个步骤时，不需要修改 POM 文件，只需要指定跳过 docker 某个步骤即可。比如当我们工程已经配置好了自动化模板了，但是这次我们只需要打镜像到本地自测，不想执行 push 阶段，那么此时执行要指定参数 `-DskipDockerPush` 就可跳过 push 操作了。

```

<build>
  <plugins>
    <plugin>
      <groupId>com.spotify</groupId>
      <artifactId>docker-maven-plugin</artifactId>
      <version>1.0.0</version>
      <configuration>
        <imageName>mavendemo</imageName>
        <baseImage>java</baseImage>
        <maintainer>docker_maven docker_maven@email.com</maintainer>
        <workdir>/ROOT</workdir>
        <cmd>["java", "-version"]</cmd>
        <entryPoint>["java", "-jar", "${project.build.finalName}.jar"]</entryPoint>
        <resources>
          <resource>
            <targetPath>/ROOT</targetPath>
            <directory>${project.build.directory}</directory>
            <include>${project.build.finalName}.jar</include>
          </resource>
        </resources>
      </configuration>
    </plugin>
  </plugins>
</build>

```

```

<executions>
  <execution>
    <id>build-image</id>
    <phase>package</phase>
    <goals>
      <goal>build</goal>
    </goals>
  </execution>
  <execution>
    <id>tag-image</id>
    <phase>package</phase>
    <goals>
      <goal>tag</goal>
    </goals>
    <configuration>
      <image>mavendemo:latest</image>
      <newName>docker.io/wanyang3/mavendemo:${project.version}</newName>
    </configuration>
  </execution>
  <execution>
    <id>push-image</id>
    <phase>deploy</phase>
    <goals>
      <goal>push</goal>
    </goals>
    <configuration>
      <imageName>docker.io/wanyang3/mavendemo:${project.version}</imageName>
    </configuration>
  </execution>
</executions>
</plugin>
</plugins>
</build>


- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28

```

- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49
- 50
- 51
- 52
- 53
- 54

以上示例，当我们执行 `mvn package` 时，执行 build、tag 操作，当执行 `mvn deploy` 时，执行build、tag、push 操作。如果我们想跳过 docker 某个过程时，只需要：

- `-DskipDockerBuild` 跳过 build 镜像
- `-DskipDockerTag` 跳过 tag 镜像
- `-DskipDockerPush` 跳过 push 镜像
- `-DskipDocker` 跳过整个阶段

例如：我们想执行 package 时，跳过 tag 过程，那么就需要 `mvn package -DskipDockerTag`。

### 3.5 使用私有 Docker 仓库地址

实际工作环境中，我们需要 push 镜像到我们私有 Docker 仓库中，使用docker-maven-plugin 插件我们也是很容易实现，有几种方式实现：

#### 一、修改 POM 文件 imageName 操作

```
...
<configuration>
  <imageName>registry.example.com/wanyang3/mavendemo:v1.0.0</imageName>
  ...
</configuration>
...


- 1
- 2
- 3
- 4
- 5
- 6

```

#### 二、修改 POM 文件中 newName 操作

```
...
<configuration>
    <imageName>mavendemo</imageName>
    ...
</configuration>
<execution>
    <id>tag-image</id>
    <phase>package</phase>
    <goals>
        <goal>tag</goal>
    </goals>
    <configuration>
        <image>mavendemo</image>
        <newName>registry.example.com/wanyang3/mavendemo:v1.0.0</newName>
    </configuration>
</execution>
...


- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17

```

### 3.6 安全认证配置

当我们 push 镜像到 Docker 仓库中时，不管是共有还是私有，经常会需要安全认证，登录完成之后才可以进行操作。当然，我们可以通过命令行 `docker login -u user_name -p password docker_registry_host` 登录，但是对于自动化流程来说，就不是很方便了。使用 docker-maven-plugin 插件我们可以很容易实现安全认证。

首先在 Maven 的配置文件 setting.xml 中增加相关 server 配置，主要配置 Docker registry 用户认证信息。

```

<servers>
  <server>
    <id>my-docker-registry</id>
    <username>wanyang3</username>
    <password>12345678</password>
    <configuration>
      <email>wanyang3@mail.com</email>
    </configuration>
  </server>
</servers>


- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

```

然后只需要在 pom.xml 中使用 server id 即可。

```

<plugin>
  <plugin>
    <groupId>com.spotify</groupId>
    <artifactId>docker-maven-plugin</artifactId>
    <version>1.0.0</version>
    <configuration>
      <imageName>registry.example.com/wanyang3/mavendemo:v1.0.0</imageName>
      ...
      <serverId>my-docker-registry</serverId>
    </configuration>
  </plugin>
</plugins>


- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12

```

### 3.7 其他参数

docker-maven-plugin 插件还提供了很多很实用的配置，稍微列举几个参数吧。

<forceTags>true</forceTags>	build 时强制覆盖 tag，配合 imageTags 使用	false
<noCache>true</noCache>	build 时，指定 -no-cache 不使用缓存	false
<pullOnBuild>true</pullOnBuild>	build 时，指定 -pull=true 每次都重新拉取基础镜像	false
<pushImage>true</pushImage>	build 完成后 push 镜像	false

<code>&lt;pushImageTag&gt;true&lt;/pushImageTag&gt;</code>	build 完成后，push 指定 tag 的镜像，配合 imageTags 使用	false
<code>&lt;retryPushCount&gt;5&lt;/retryPushCount&gt;</code>	push 镜像失败，重试次数	5
<code>&lt;retryPushTimeout&gt;10&lt;/retryPushTimeout&gt;</code>	push 镜像失败，重试时间	10s
<code>&lt;rm&gt;true&lt;/rm&gt;</code>	build 时，指定 -rm=true 即 build 完成后删除中间容器	false
<code>&lt;useGitCommitId&gt;true&lt;/useGitCommitId&gt;</code>	build 时，使用最近的 git commit id 前7位作为tag，例如：image:b50b604，前提是不配置 newName	false

## 4、FAQ

### 1、执行 build images 时，报错情况一：

```
[INFO] Building image mavendemo
org.apache.http.impl.execchain.RetryExec execute
I/O exception (java.io.IOException) caught when processing request to {}->unix://localhost:80: No such file or directory
[ERROR] Failed to execute goal com.spotify:docker-maven-plugin:1.0.0:build (default-cli) on project mavenDemo: Exception
caught: java.util.concurrent.ExecutionException: com.spotify.docker.client.shaded.javax.ws.rs.ProcessingException:
java.io.IOException: No such file or directory -> [Help 1]
• 1
• 2
• 3
• 4
```

这是因为 Docker 服务没有启动造成的，启动 Docker 即可。

### 2、执行 build images 时，报错情况二：

```
ERROR] Failed to execute goal com.spotify:docker-maven-plugin:1.0.0:build (default-cli) on project mavenDemo: Exception
caught: Request error: POST unix://localhost:80/build?t=mavenDemo: 500, body: {"message":"Error parsing reference:
\"mavenDemo\" is not a valid repository/tag: repository name must be lowercase"}: HTTP 500 Internal Server Error -> [Help 1]
1
```

这是因为镜像名字不正确，Docker 镜像名称需匹配[a-z0-9-\_.]。

## 参考资料

[docker-maven-plugin](#)

[maven打包项目到docker的时候：ADD failed: stat xxx : no such file or directory](#)

[11-17](#)

[- 问答](#)

[Failed to execute goal com.spotify:docker-maven-plugin:1.2.0:build \(build-image\)](#)

[07-02](#)

[求大佬，求解决方法。 论坛](#)

[使用Maven插件构建Docker镜像 - keketrtr的专栏 - CSDN博客](#)

[7-5](#)

[Maven 插件之 docker-maven-plugin 的使用](#) 08-21 阅读数 2万+ 在我们持续集成...博文 来自: 哎\_小羊的博客 使用 maven插件dockerfile-maven-plugin进行镜像的bu...

## [使用dockerfile-maven-plugin - qq\\_36059306的博客 - CSDN博客](#)

11-6

[Maven 插件之 docker-maven-plugin 的使用 - 哎\\_小羊的博客\(学会发现,学会记录,学会分享。\)](#) 08-21 1.3万 在我们持续集成过程中,项目工程一般使用 Maven 编...

## [使用docker-maven-plugin插件将项目编译为docker镜像到... CSDN博客](#)

7-12

[Maven 插件之 docker-maven-plugin 的使用](#) 08-21 阅读数 2万+ 在我们持续集成...博文 来自: 哎\_小羊的博客 二. Spring Boot使用DockerFile maven插件自动化...

## [4.docker-maven-plugin - 元元的博客 - CSDN博客](#)

11-1

[来自: 哎\\_小羊的博客 使用docker-maven-plugin插件将项目编译为docker镜像到远程...Docker maven插件方式构建微服务镜像有两种方式:a.maven集成构建docker镜像 b.外部...](#)

## [使用docker-maven-plugin插件构建和推送Docker映像 - d... CSDN博客](#)

10-30

[58 She lock 阅读数:4486 标签: docker-maven-plugin使用docker快捷...docker-maven-plugin 插件... 来自: 哎\\_小羊的博客 maven构建docker镜像三...](#)

## [利用docker-maven-plugin自动化部署 - wings的博客 - CSDN博客](#)

4-16

[在编译工程时,能自动生成docker image,还可以远程推送至仓库,只需要在maven的配置...博文 来自: 哎\\_小羊的博客 使用 docker-maven-plugin插件将项目编译为docker镜像...](#)

## [docker-maven-plugin插件设置Docker的buildArgs - 添经... CSDN博客](#)

11-22

## [使用Maven 插件构建docker 镜像和推送仓库 - 风.foxwho... CSDN博客](#)

10-23

[--使用pom 配置--> java <entryPoint>"\["java","-jar","/\\${project.build...docker-maven-plugin 插件... 来自: 哎\\_小羊的博客 maven构建docker镜像三...](#)

## [...dockerfile-maven-plugin - Buynow\\_Zhao的博客 - CSDN博客...](#)

7-9

## [docker使用host模式 主机却无法ping通容器](#)

01-01

- 问答

[Failed to com.spotify:docker-maven-plugin:HttpHostConnectException:Connect to localhost:2375](#)

---

[阅读数 6253](#)

[在用docker部署SpringCloud的时候，pom.xml:org.springframework.boot:spring-boot-maven-plugin...  
博文来自：Java & Basketball](#)

**原创**

[71](#)

**粉丝**

254

**喜欢**

204

**评论**

475

**等级：**

**访问：**

60万+

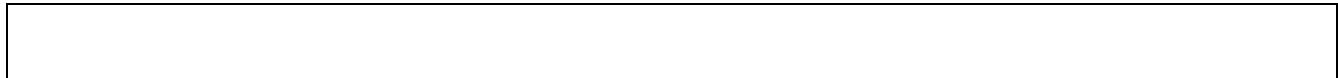
**积分：**

5268

**排名：**

9341

**勋章：**



**最新文章**

---

**个人分类**

[展开](#)

**归档**

[展开](#)