How to push a docker image to a private repository

I have a docker images tagged as me/my-image, and I have a private repo on the dockerhub named me-private. When I push my me/my-image, I end up always hitting the public repo.

What is the exact syntax to specifically push my image to my private repo?

docker docker-registry

edited May 27 '16 at 21:14

luchaninov
2,822 4 28 62

asked Feb 5 '15 at 16:42

Eugene Goldberg
2,585 5 29 65

docs.docker.com/engine/getstarted/step_six – Sudip Bhandari Nov 22 '16 at 9:41

3 Answers

You need to tag your image correctly first with your `registryhost` :

```
docker tag [OPTIONS] IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]NAME[:TAG]
```

Then docker push using that same tag.

```
docker push NAME[:TAG]
```

Example:

```
docker tag myImage myRegistry.com/myImage
docker push myRegistry.com/myImage
```

edited Feb 5 '15 at 17:04

answered Feb 5 '15 at 16:49

Abdullah Jibaly
23.4k 24 93 172

1 so, with this image: 518a41981a6a, what is the actual tag syntax to make it go to the me-private repo, please? – Eugene Goldberg Feb 5 '15 at 16:54

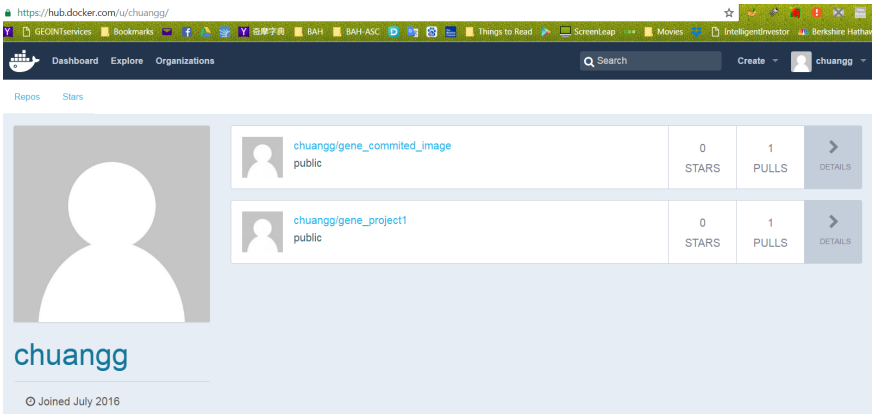
1 `docker tag 518a41981a6a me-private.com/myPrivateImage && docker push me-private.com/myPrivateImage` – Abdullah Jibaly Feb 5 '15 at 17:04

I fixed some syntax issues in my answer above as well. Also, fyi when you push to a registry you have to use an actual image name and not the image id. – Abdullah Jibaly Feb 5 '15 at 17:05

for some reason the image still goes to the public dockerhub registry instead of the private one. Just for the clarification - my private registry is also on the dockerhub. I tagged the image, and did the push, but the feedback was indicating that all layers have already been pushed, which would not be the case if the image was going to the private registry for the first time – Eugene Goldberg Feb 5 '15 at 17:14

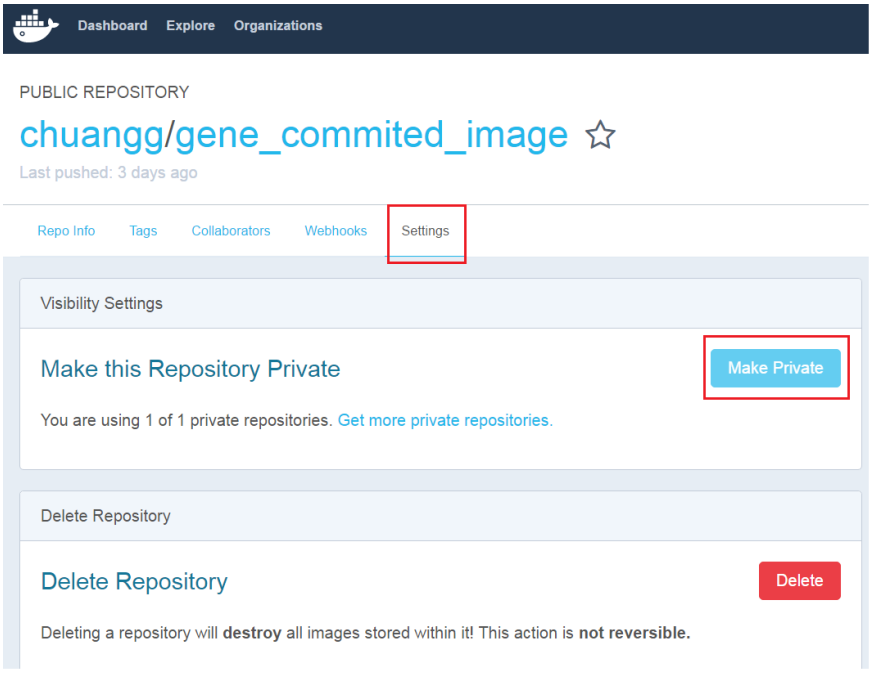
3 Oh, if you're using a private dockerhub registry it should be pretty simple. Make sure you do a `docker login` first, then tag your image: `docker tag 518a41981a6a me-private/myPrivateImage` and push: `docker push me-private/myPrivateImage` – Abdullah Jibaly Feb 5 '15 at 17:20

First go to your Docker Hub account and make the repo. Here is a screenshot of my Docker Hub account:



From the pic, you can see my repo is "chuangg"

Now go into the repo and make it private by clicking on your image's name. So for me, I clicked on "chuangg/gene_commited_image", then I went to Settings -> Make Private. Then I followed the on screen instructions



HOW TO UPLOAD YOUR DOCKER IMAGE ONTO DOCKER HUB

Method #1= Pushing your image through the command line (cli)

1) `docker commit <container ID> <repo name>/<Name you want to give the image>`

Yes, I think it has to be the container ID. It probably cannot be the image ID.

For example= `docker commit 99e078826312 chuangg/gene_commited_image`

2) `docker run -it chaung/gene_commited_image`

3) `docker login --username=<user username> --email=<user email address>`

For example= `docker login --username=chuangg --email=gc.genechaung@gmail.com`

Yes, you have to login first. Logout using “docker logout”

4) `docker push chuangg/gene_commited_image`

Method #2= Pushing your image using pom.xml and command line.

Note, I used a Maven Profile called “build-docker”. If you don’t want to use a profile, just remove the `<profiles>, <profile>, and <id>build-docker</id>` elements.

Inside the parent pom.xml:

```
<profiles>
        <profile>
            <id>build-docker</id>
            <build>
                <plugins>
                    <plugin>
                        <groupId>io.fabric8</groupId>
                        <artifactId>docker-maven-plugin</artifactId>
                        <version>0.18.1</version>
                        <configuration>
                            <images>
                                <image>
                                    <name>chuangg/gene_project</name>
                                    <alias>${docker.container.name}</alias>
                                    <!-- Configure build settings -->
                                    <build>

<dockerFileDir>${project.basedir}\src\docker\vending_machine_emulator</dockerFileDir

                                        <assembly>
                                            <inline>
                                                <fileSets>
                                                    <fileSet>

<directory>${project.basedir}\target</directory>
                                                        <outputDirectory>.
</outputDirectory>
                                                        <includes>

<include>*.jar</include>
                                                        </includes>
                                                    </fileSet>
                                                </fileSets>
                                            </inline>
                                        </assembly>
                                    </build>
                                </image>
                            </images>
                        </configuration>
                        <executions>
                            <execution>
                                <id>docker:build</id>
                                <phase>package</phase>
                                <goals>
                                    <goal>build</goal>
                                </goals>
                            </execution>
                        </executions>
                    </plugin>
                </plugins>
            </build>
        </profile>
    </profiles>
```

Docker Terminal Command to deploy the Docker Image (from the directory where your pom.xml is located)= `mvn clean deploy -Pbuild-docker docker:push`

Note, the difference between Method #2 and #3 is that Method #3 has an extra `<execution>` for the deployment.

Method #3= Using Maven to automatically deploy to Docker Hub

Add this stuff to your parent pom.xml:

```
    <distributionManagement>
        <repository>
            <id>gene</id>
            <name>chuangg</name>
            <uniqueVersion>false</uniqueVersion>
            <layout>legacy</layout>
            <url>https://index.docker.io/v1/</url>
        </repository>
    </distributionManagement>


    <profiles>
```

```
        <profile>
            <id>build-docker</id>
            <build>
                <plugins>

                    <plugin>
                        <groupId>io.fabric8</groupId>
                        <artifactId>docker-maven-plugin</artifactId>
                        <version>0.18.1</version>
                        <configuration>
                            <images>
                                <image>
                                    <name>chuangg/gene_project1</name>
                                    <alias>${docker.container.name}</alias>
                                    <!-- Configure build settings -->
                                    <build>

<dockerFileDir>${project.basedir}\src\docker\vending_machine_emulator</dockerFileDir>

                                        <assembly>
                                            <inline>
                                                <fileSets>
                                                    <fileSet>

<directory>${project.basedir}\target</directory>
                                                        <outputDirectory>.
</outputDirectory>
                                                        <includes>

<include>*.jar</include>
                                                        </includes>
                                                    </fileSet>
                                                </fileSets>
                                            </inline>
                                        </assembly>
                                    </build>
                                </image>
                            </images>
                        </configuration>
                        <executions>
                            <execution>
                                <id>docker:build</id>
                                <phase>package</phase>
                                <goals>
                                    <goal>build</goal>
                                </goals>
                            </execution>
                            <execution>
                                <id>docker:push</id>
                                <phase>install</phase>
                                <goals>
                                    <goal>push</goal>
                                </goals>
                            </execution>

                        </executions>
                    </plugin>
                </plugins>
            </build>
        </profile>
    </profiles>
</project>
```

Go to C:\Users\Gene.docker\ directory and add this to your config.json file:

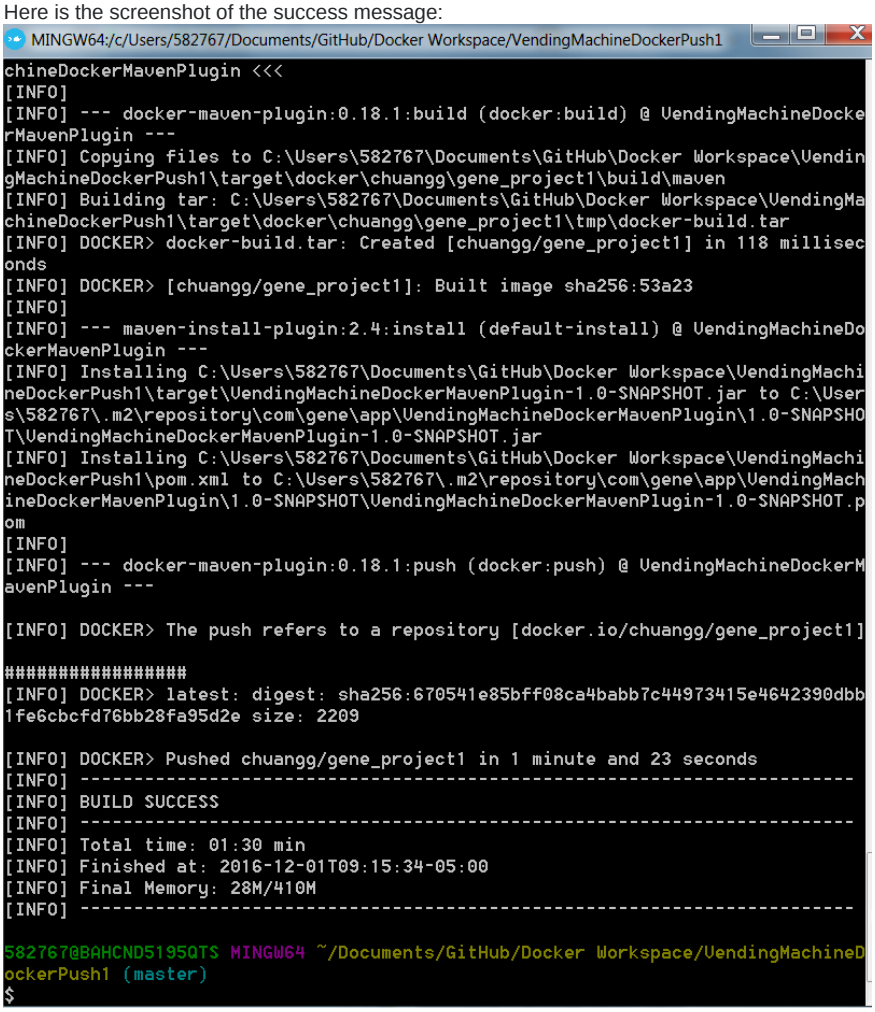
config.json

```
{
    "auths": {
        "https://hub.docker.com": {
            "auth": "Y2h1YW5nZzpEb2NrZXI="
        },
        "https://index.docker.io/v1/": {
            "auth": "Y2h1YW5nZzpEb2NrZXI=",
            "email": "gc.genechuang@gmail.com"
        }
    }
}
```

Now in your Docker Quickstart Terminal type= `mvn clean install -Pbuild-docker`

For those of you not using Maven Profiles, just type `mvn clean install`

Here is the screenshot of the success message:

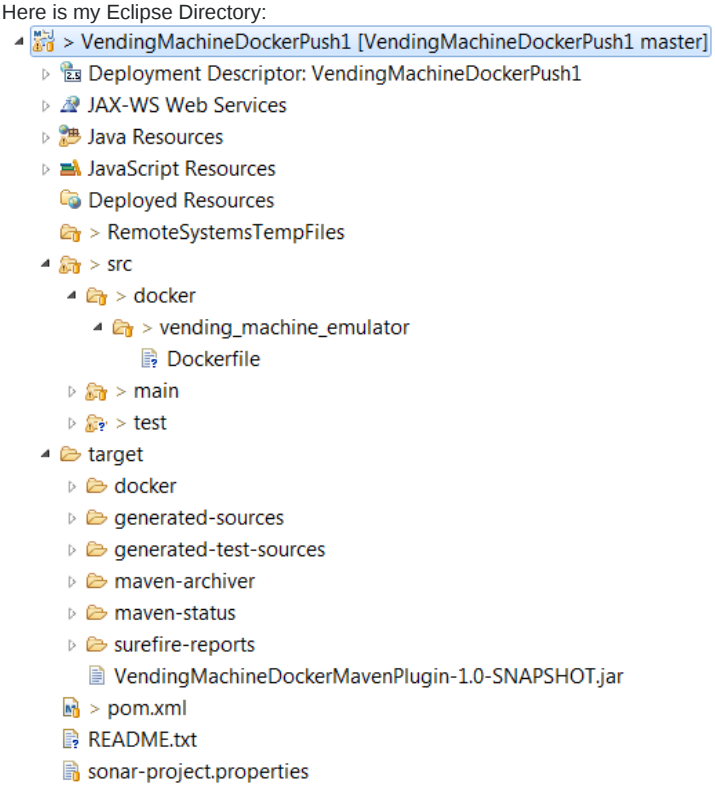


Here is my full pom.xml and a screenshot of my directory structure:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>com.gene.app</groupId>
<artifactId>VendingMachineDockerMavenPlugin</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>

<name>Maven Quick Start Archetype</name>
<url>www.gene.com</url>

<build>
    <pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <configuration>
                    <archive>
                        <manifest>
                            <mainClass>com.gene.sample.Customer_View</mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>

                    <source>1.7</source>
                    <target>1.7</target>

                </configuration>
            </plugin>
        </plugins>
    </pluginManagement>
```

```
</build>


<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.8.2</version>
        <scope>test</scope>
    </dependency>

</dependencies>

<distributionManagement>
    <repository>
        <id>gene</id>
        <name>chuangg</name>
        <uniqueVersion>false</uniqueVersion>
        <layout>legacy</layout>
        <url>https://index.docker.io/v1/</url>
    </repository>
</distributionManagement>


<profiles>
    <profile>
        <id>build-docker</id>
        <properties>
            <java.docker.version>1.8.0</java.docker.version>
        </properties>
        <build>
            <plugins>

                <plugin>
                    <groupId>io.fabric8</groupId>
                    <artifactId>docker-maven-plugin</artifactId>
                    <version>0.18.1</version>
                    <configuration>
                        <images>
                            <image>
                                <name>chuangg/gene_project1</name>
                                <alias>${docker.container.name}</alias>
                                <!-- Configure build settings -->
                                <build>

<dockerFileDir>${project.basedir}\src\docker\vending_machine_emulator</dockerFileDir>

                                    <assembly>
                                        <inline>
                                            <fileSets>
                                                <fileSet>

<directory>${project.basedir}\target</directory>
                                                    <outputDirectory>.
</outputDirectory>
                                                    <includes>
                                                        <include>*.jar</include>
                                                    </includes>
                                                </fileSet>
                                            </fileSets>
                                        </inline>
                                    </assembly>
                                </build>
                            </image>
                        </images>
                    </configuration>
                    <executions>
                        <execution>
                            <id>docker:build</id>
                            <phase>package</phase>
                            <goals>
                                <goal>build</goal>
                            </goals>
                        </execution>
                        <execution>
                            <id>docker:push</id>
                            <phase>install</phase>
                            <goals>
                                <goal>push</goal>
                            </goals>
                        </execution>

                    </executions>
                </plugin>
            </plugins>
        </build>
    </profile>
</profiles>
```

Here is my Eclipse Directory:



Here is my Dockerfile:

```
FROM java:8

MAINTAINER Gene Chuang
RUN echo Running Dockerfile in src/docker/vending_machine_emulator/Dockerfile
directory

ADD maven/VendingMachineDockerMavenPlugin-1.0-SNAPSHOT.jar /bullshitDirectory/gene-
app-1.0-SNAPSHOT.jar

CMD ["java", "-classpath", "/bullshitDirectory/gene-app-1.0-SNAPSHOT.jar",
"com/gene/sample/Customer_View" ]
```

Common Error #1:

```
[INFO]
[INFO] --- maven-deploy-plugin:2.7:deploy (default-deploy) @ VendingMachineDocke
rMavenPlugin ---
Downloading: https://index.docker.io/v1/com/gene/app/VendingMachineDockerMavenPl
ugin/1.0-SNAPSHOT/maven-metadata.xml
Uploading: https://index.docker.io/v1/com/gene/app/VendingMachineDockerMavenPlug
in/1.0-SNAPSHOT/VendingMachineDockerMavenPlugin-1.0-20161201.151617-1.jar
Uploading: https://index.docker.io/v1/com/gene/app/VendingMachineDockerMavenPlug
in/1.0-SNAPSHOT/VendingMachineDockerMavenPlugin-1.0-20161201.151617-1.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.515 s
[INFO] Finished at: 2016-12-01T10:16:17-05:00
[INFO] Final Memory: 28M/723M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.7:
deploy (default-deploy) on project VendingMachineDockerMavenPlugin: Failed to de
ploy artifacts: Could not find artifact com.gene.app:VendingMachineDockerMavenPl
ugin:jar:1.0-20161201.151617-1 in gene (https://index.docker.io/v1/) -> [Help 1]

[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e swit
ch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please rea
d the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionE
xception
```

Solution for Error #1= Do not sync the `<execution>` with maven deploy phase because then maven tries to deploy the image 2x and puts a timestamp on the jar. That's why I used `<phase>install</phase>` .

edited Dec 1 '16 at 19:43 answered Dec 1 '16 at 19:07


Gene
3,650 22 32

1 Awesome answer ! Helped me in first shot. – Mayur Patil Jan 15 at 19:13

There are two options:

1. Go into the hub, and create the repository first, and mark it as private. Then when you push to that repo, it will be private. This is the most common approach.
2. log into your docker hub account, and go to your global settings. There is a setting that allows you to set what your default visability is for the repositories that you push. By default it is set to public, but if you change it to private, all of your repositories that you push will be marked as private by default. It is important to note that you will need to have enough private repos available on your account, or else the repo will be locked until you upgrade your plan.

answered May 29 '15 at 15:06


Ken Cochrane
35.1k 7 34 52