DZone

Maven Build Local Project With Docker: Why?

by Ivo Woltring · Jan. 22, 18 · Cloud Zone · Tutorial

The What and Why

Build your local project on a Mac with docker so that you can do the build with Linux and all it provides.

Now you might ask me why for heaven's sake?! Java is cross-platform, so why?

I needed a dependency in my maven build with native bindings (Rados) on the target system. These dependencies are not available for my local machine without doing some nasty stuff and installing it all on my machine. As I don't want to pollute my machine with software I only use professionally for a certain customer, I didn't want to do the installation.

What To Do

A couple of things:

- Docker image(s) needed
  - CentOs 7 base image (ivonet/centos-jdk:7-1.8.0) with java 1.8.0
  - Extended image with maven and the native bindings
- Entrypoint script to run default mvn clean install if no parameters provided
- Expose VOLUMES for easy mounting
- Bash script to run the docker maven on the local project and use the local repository

Docker Images

Base CentOs image with JDK 8

First create a base image with the target system and the needed Java version. In my case, this was CentOs, as we use Redhat servers on production.

- Create a directory called *centos-jdk*
- Create a *Dockerfile* in that directory

```
FROM centos:7

RUN yum -y update \
&& yum -y install \
   --setopt=tsflags=nodocs \
   --disableplugin=fastestmirror \
   epel-release \
&& yum -y install \
inotify-tools \
java-1.8.0-openjdk.i686 \
&& yum clean all \
&& rm -rf /etc/ld.so.cache

ENTRYPOINT ["/bin/true"]
```

Build and push it:

```
docker build -t ivonet/centos-jdk:7-1.8.0 .
docker push ivonet/centos-jdk:7-1.8.0
```

Extended Docker Images with Maven

Now that we have the base image we can create an extended version for our specific purposes.
So lets extend the just created image and add maven and the rados native bindings. This is of course something you might want to change
for your environment as this is just the reason I needed this.

- Create a directory called *centos-maven*
- Create a *Dockerfile* in that directory

```
FROM ivonet/centos-jdk:7-1.8.0

R            date \
&            tall \
   --setopt=tsflags=nodocs \
    --disableplugin=fastestmirror \
    librados2.x86_64 \
    maven \
 && yum clean all \
 && mv /usr/share/maven/conf/settings.xml /usr/share/maven/conf/settings.xml.orig \
 && rm -rf /etc/ld.so.cache

COPY setup/settings.xml /settings.xml
ADD setup/entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

WORKDIR /project
VOLUME ["/project", "/repository"]

ENTRYPOINT ["/entrypoint.sh"]
CMD ["mvn", "verify"]
```

This *Dockerfile* also tells us that we have two volumes we can mount as we like and one of the is the *repository* volume.

This is the only thing we overrode in the *settings.xml* file from the original *settings.xml*.

For those actually following the blog for their own purposes I will paste it in.

Note the *<localRepository>/repository</localRepository>* line in the file, as it maps the the volume declared in the *Dockerfile*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements.  See the NOTICE file
distributed with this work for additional information
regarding copyright ownership.  The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License.  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"           , WITHOUT WARRANTIES OR CONDITIONS OF ANY
K           express or implied.  See the License for the
s           uage governing permissions and limitations
under the License.
```

```
-->
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/


  <localRepository>/repository</localRepository>

  <!-- I removed all the commends to save space in this blog-->

  <pluginGroups>
  </pluginGroups>
  <proxies>
  </proxies>
  <servers>
  </servers>
  <mirrors>
  </mirrors>
  <profiles>
  </profiles>
</settings>
```

I added an *entrypoint.sh* file which looks like this in the folder *setup*:

```
#!/usr/bin/env bash
## Do your own config stuff here



## this copies our settings.xml to the default place where maven will look for it.
rm -f /usr/share/maven/conf/settings.xml >/dev/null
cp /settings.xml /usr/share/maven/conf/settings.xml

# Always end with this line so that the CMD of the Dockerfile will be executed...
exec "$@"
```

This is more or less just a placeholder as you can see, but in the version I use for my actual project I have stuff in there to set the proxy in the settings.xml file correctly based on ENV settings on the commandline, for the environment I am working on and the nexus repository for the client I'm working for.

I left the placeholder script as a reminder one can do lots more than this straightforward example.

Now we can build and deploy it:

```
d[illegible]-t ivonet/centos-maven:7-3.0.5 .
d[illegible]vonet/centos-maven:7-3.0.5
```

Maven Script

Maven Script

We are almost ready. The ingredients are there, now we can put it all together on the command line. Note that this script will not go into the docker image. This one is used to run the docker image with the correct parameters:

```
docker run -it \
           --rm  \
           -v $(pwd):/project \
           -v $(echo "$HOME/.m2/repository"):/repository \
           ivonet/centos-maven:7-3.0.5
```

This command will run the just-created image with the current directory mapped the the */project* volume in the docker image and the local maven repository mapped to the */repository* in the docker image.

As we throw away the state of this image after every command `--rm`, it would take a long time to build as no dependencies would be remembered and all would have to be downloaded every time. This is actually a good thing to do now and then, but not every time. That would be wasteful. So by mapping your own local maven repo to the docker image the dependencies will be downloaded there and state is preserved and the image becomes stateless.
As the default CMD is *mvn verify* it will be performed.

But there is more...

```
#!/bin/sh
###############################################################################
# Maven command with centos 7 image behind it.
###############################################################################
# This command makes it possible to do a maven build on your local maven
# project but with centos architecture. The reason for this command is that
# some projects need the Rados native bindings in order to be able to build and
# test. Rados is the protocol to talk to Ceph. On centos based images this
# native binding is easy to install but on windows and mac very difficult.
# This command makes it possible to do a "normal" maven build with the bindings
# available. As java makes bytecode it will be cross platform and completely
# ok.
###############################################################################
PARAMS="$@"
if [ -z "$PARAMS" ]; then
    echo "[ERROR] Syntaxis $0 [clean|install|package|verify|compile|test]"
    echo "Standard maven commands are allowed"
    echo "Do not do a release with this command but let the build server to it"
    echo "This command has no access to the git / nexus credentials"

f

docker run -it \
            --rm  \
            --memory=1024g \
            -v $(pwd):/project \
            -v $(echo "$HOME/.m2/repository"):/repository \
            ivonet/centos-maven:7-3.0.5 mvn $PARAMS
```

If you create a shell script based on the above bash commands with a name like ***mvncentos*** and make it executable (*chmod +x mvncentos*) and place it on your PATH it will be available for the projects you need it for.

Done

With these ingredients you will be able to build your project without polluting your system with stuff you do not want to install.
When you are done with the project you can just throw away your docker image and shell script and you are set again for the next project.

Feedback always welcome.

Learn how to auto-discover your containers and monitor their performance, capture Docker host and container metrics to allocate host resources, and provision containers.

Like This Article? Read More From DZone

Topics: DOCKER , JAVA , MAVEN

Published at DZone with permission of Ivo Woltring . See the original article here.
Opinions expressed by DZone contributors are their own.