Title: Install package in Docker image created by Spring Boot Maven plugin

Post Body:

My Spring Boot project contains the Spring Boot Maven Plugin which I use for building a Docker image by running `mvn spring-boot:build-image`.

```
<plugin>     <groupId>org.springframework.boot</groupId>     <artifactId>spring-boot-maven-plugin</artifactId>     <executions
```

When deploying this image to a Docker stack I need to run a healthcheck using the `curl` command but unfortunately `curl` is not installed by the default buildpack.

Is it possible to further tweak the image building process so that `curl` gets installed into the iamge? I couldn't find the necessary information

Accepted Answer: None

Highest Rated Answer:

TLDR;

Install `curl` into the build image with:

```
docker run --user="root" --entrypoint launcher my-app:0.0.1-SNAPSHOT "apt-get update && apt-get install curl -y"
```

Grab container id of the stopped container with `docker ps -a`:

```
$ docker ps -a CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS
```

Create a new container image based on the one we installed `curl` into with:

```
docker commit 2ff7db32825f my-app-with-curl
```

Fire up a new container defining the correct `ENTRYPOINT` to start Spring Boot app:

```
docker run --rm -p 8080:8080 --user="cnb" --entrypoint /cnb/process/web my-app-with-curl
```

Now `curl` should be ready inside your container.

Details of the solution:

The reasoning behind Cloud Native Buildpacks (CNBs) & Paketo.io, which are basically abstracted away by the `spring-boot-maven-plugins` `build-image` goal, is to free us from the need to write/maintain our own `Dockerfiles`. So the inversion of this is: It's easy to configure the build process, but it is not easy to change things like installed packages.

The reason is, that those packages are maintained in a so called stack, that manages the used `build`-time and `run`-time images. And if the stack doesn't define a Mixin for your OS-level dependency, then you can't simply add another package. It would also **not suffice** to create your own simple buildpack (I tried this approach). And creating your own stacks, buildpacks and/or builders would also negate the huge benefits that Cloud Native Buildpacks provide! Amongst other things we would be also forced to keep the images updated ourselves...

But there's another solution. As we don't want to create our own stacks/buildpacks, we can tweak the container image which has been created by CNBs/`spring-boot-maven-plugin`. Because the official docs show us how to hook into the startup process of the produced containers and run shell scripts for example. Let's assume our `mvn spring-boot:build-image` command produced a container image called `my-app:0.0.1-SNAPSHOT`.

Then first we install `curl` into the image with:

```
docker run --user="root" --entrypoint launcher my-app:0.0.1-SNAPSHOT "apt-get update && apt-get install curl -y"
```

We need to use `--user="root"` here in order that the command `apt-get update && apt-get install curl -y` will run successfully (otherwise we would run into errors like `List directory /var/lib/apt/lists/partial is missing. - Acquire (13: Permission denied)`). This will install curl, but we shouldn't use the resulting container in production. Because our Spring Boot app would run using the root user, which would introduce a variety of security problems. Also we've overwritten the `ENTRYPOINT` of our container, so it wouldn't be able to start our app.

Therefore we simply start this stopped container with a new command, entrypoint & user! Simply grab the container ID of the stopped container with `docker ps -a`:

```
$ docker ps -a CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS
```

And **create a new container image** based on the one we installed `curl` into with:

```
docker commit 2ff7db32825f my-app-with-curl
```

Finally **fire up a new container** based on this new image, defining the correct `ENTRYPOINT` to start our Spring Boot app and also using the `cnb` user again (as defined in the Cloud Native Buildpacks):

```
docker run --rm -p 8080:8080 --user="cnb" --entrypoint /cnb/process/web my-app-with-curl
```

Off topic but relevant:

There are ongoing discussions if it is desired to install curl in a production container. See this post for example.