**Lab 4  
REST web service in a Docker container**

This lab will show you how you can run your Spring Boot REST service in a Docker container.

On top of that, we will verify how secure the container is, using an open source tool named Claire.

The starting point for this lab is to have the provided VirtualBox machine up-and-running. You are logged in under user/password developer/welcome1.

# Building the Docker container

We will use the service that we developed in lab 3. You can find that completed code in:

/home/developer/projects/SIGSpringBoot101/lab 4/dronebuzzers

The following steps will be done:

* Step 1: change the maven pom.xml
* Step 2: add the Dockerfile
* Step 3: build the container

**Step 1: change the maven pom.xml**

The maven pom will have 2 changes:

1. Add a property ‘docker.image.prefix’
2. Add the Spotify plugin for building the Docker container

@1:the property line:

<docker.image.prefix>docker</docker.image.prefix>

@2: the plugin:

<plugin>

<groupId>com.spotify</groupId>

<artifactId>dockerfile-maven-plugin</artifactId>

<version>1.3.4</version>

<configuration>

<repository>${docker.image.prefix}/${project.artifactId}</repository>

</configuration>

</plugin>

The above changes can be found in the file:

/home/developer/projects/SIGSpringBoot101/lab 4/input/mvn-changes.txt

Insert them into the appropriate places in the pom file:

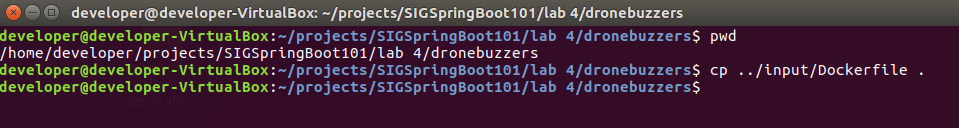
/home/developer/projects/SIGSpringBoot101/lab 4/dronebuzzers/pom.xml

/home/developer/projects/SIGSpringBoot101/lab 4/input/mvn-changes.txt

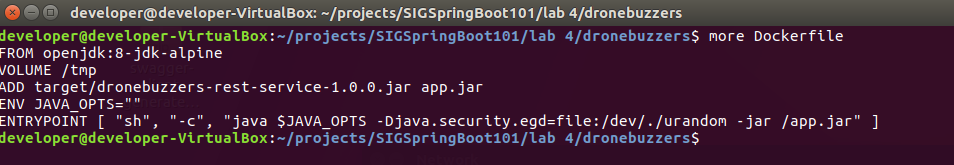
**Step 2: add the Dockerfile**

In order to build a Docker container, we need a file named Dockerfile that specifies the container.

Copy the Dockerfile from the input to the project directory:



Have a look at the Dockerfile:

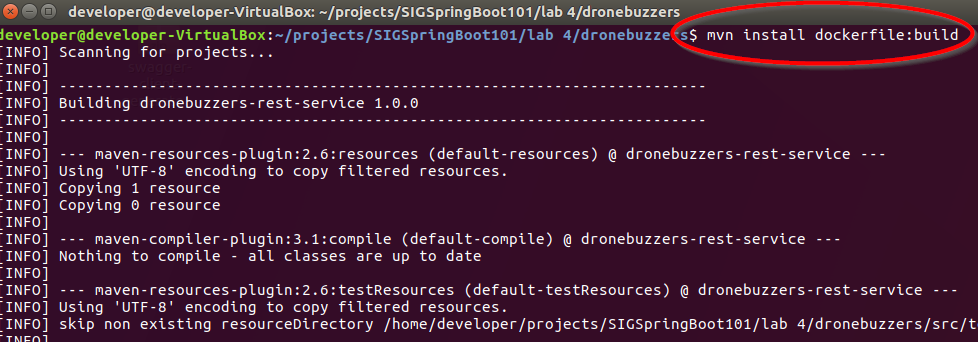


The keywords in the Dockerfile:

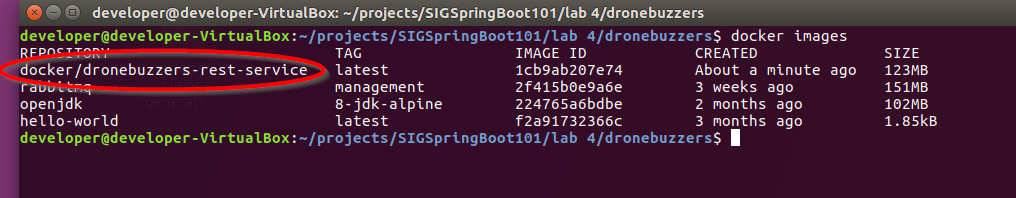
|  |  |
| --- | --- |
| Keyword | Meaning |
| FROM | sets the base image for the container |
| VOLUME | external mounted volume |
| ADD | adds files/directories from the source to the container image |
| ENV | environment variable |
| ENTRYPOINT | command that the container will run when started |

**Step 3: build the container**

The Docker container is built with the command mvn install dockerfile:build

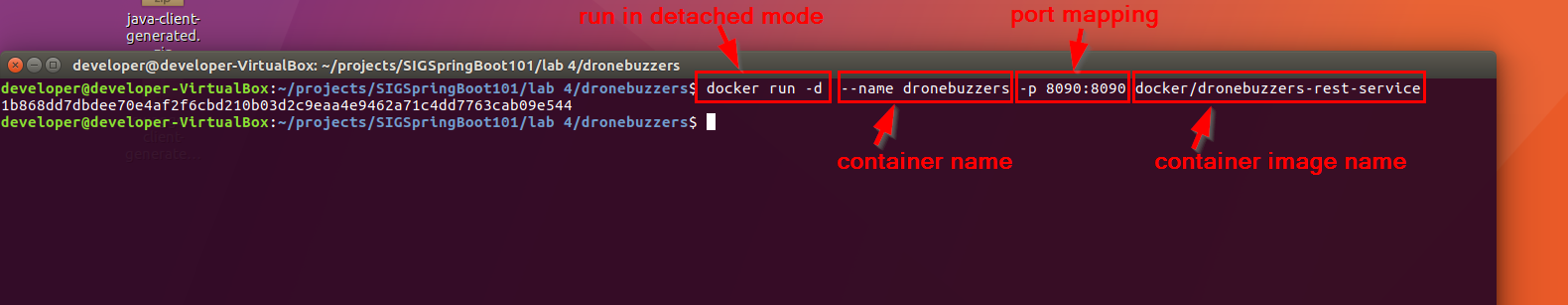


Check that the Docker container is present with the command docker images:



# Running the Docker container

Now, the container that we’ve built can be started with the docker run command:

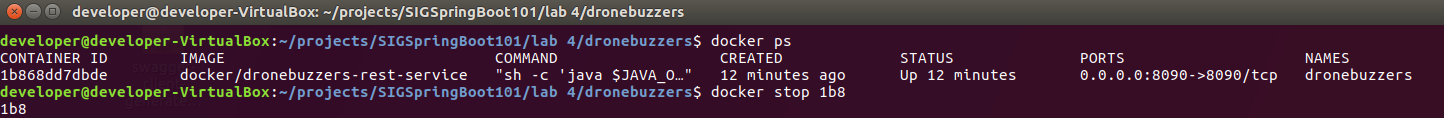


The command used was:

docker run -d --name dronebuzzers -p 8090:8090 docker/dronebuzzers-rest-service

* -d: the container runs in de-tached mode, i.e. in the background
* -p 8090:8090: specifies port exposure, i.e. how a local host port (first 8090)is mapped to the internal container port (second 8090)

Should you want to stop the container: look up the Container ID with the docker ps command. Then stop the container with the docker stop command. Notic that you only have to enter the first couple of characters of the Container ID.



There is a good Docker cheat sheet: <https://github.com/wsargent/docker-cheat-sheet>

# Testing the Docker container

Once the container is up and running, we can test it. We will do that with the same postman tests from lab 3.

For testing, start Postman  and import the Collection of Postman tests for lab 3 from location

/home/developer/projects/SIGSpringBoot101/lab 3/postman

Test the interface with the last 5 operations:

