***Spring Boot application***

The java file can be found at following location;

src/main/java/hello/Application.java

package hello;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.bind.RelaxedPropertyResolver;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication

@RestController

public class Application {

@RequestMapping("/")

public String home() {

return "Hello Docker World";

}

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

The class is flagged as a @SpringBootApplication and as a @RestController, meaning it’s ready for use by Spring MVC to handle web requests. @RequestMapping maps / to the home() method which just sends a response back. The main() method uses Spring Boot’s SpringApplication.run() method to launch an application.

**Containerize the Spring Boot application**

We will containerize the above application using the DockerFile. The project JAR file is added to the container as “app.jar” and then executed in the ENTRYPOINT.We have added a VOLUME pointing to /tmp because it is where a Spring Boot application creates working directories for Tomcat by default.

Dockerfile

This file is needed to create a docker image;

FROM java:8

VOLUME /tmp

ADD gs-spring-boot-docker-0.1.0.jar app.jar

RUN bash -c 'touch /app.jar'

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom",

"-jar","/app.jar"]

**Docker build**

We can build a tagged docker image using the below command :

$ docker build .

**Docker run**

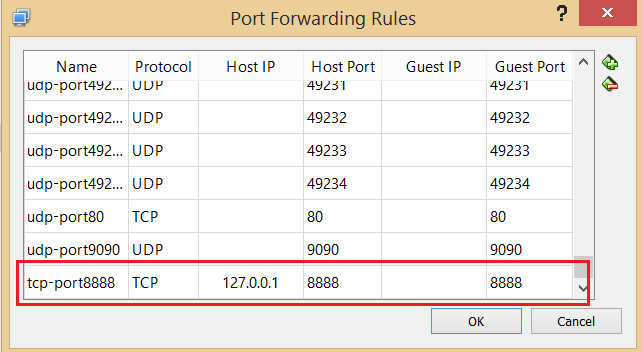
We do NOT have to register with docker or publish anything to run a docker image. We can still have a locally tagged image, and run it like this:

$ docker run -p 8888:8080 -t <Image-ID>

**VirtualBox settings**

We need to add a port forwarding rule in the VirtualBox settings.

boot2docker vm > settings > Network > NAT adapter > Port forwarding



**VOLUME Comand**

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While bind mounts are dependent on the directory structure of the host machine, volumes are completely managed by Docker. Volumes have several advantages over bind mounts:

* Volumes are easier to back up or migrate than bind mounts.
* You can manage volumes using Docker CLI commands or the Docker API.
* Volumes work on both Linux and Windows containers.
* Volumes can be more safely shared among multiple containers.
* Volume drivers allow you to store volumes on remote hosts or cloud providers, to encrypt the contents of volumes, or to add other functionality.
* A new volume’s contents can be pre-populated by a container.

