In this lab we will

1. Download and install Maven.
2. Build a dropwizard microservice for hello-world
3. Use Docker to build a container to hold the microservice.

Tutorial I followed to build the dropwizare microservice is at:

<https://www.dropwizard.io/en/latest/getting-started.html>

Code at: <https://github.com/dropwizard/dropwizard/tree/master/dropwizard-example>

Use my code on the X drive (dropwizard-core.7z) as it follows exactly the tutorial , the git code is more extensive. Download and unzip it to your c drive – it should create a folder called dropwizard-core

Dropwizard with Maeven (there are possible tutorials in using it with intelliJ, Eclipse etc <https://blog.indrek.io/articles/running-a-dropwizard-application-in-intellij-eclipse-and-netbeans/>)

Download Maeven and install (<https://maven.apache.org/install.html>)

Add the unpacked distribution’s bin directory to your user PATH environment variable by opening up the system properties (WinKey + Pause), selecting the “Advanced” tab, and the “Environment Variables” button, then adding or selecting the *PATH* variable in the user variables with the value C:\Program Files\apache-maven-3.6.0\bin

Also ensure JAVA\_HOME environment variable is set and points to your JDK installation

See more on Maven at : <https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>

Create a new folder on C: called drop

Open a command widow and cd to drop

Run the command

mvn archetype:generate -DarchetypeGroupId=io.dropwizard.archetypes -DarchetypeArtifactId=java-simple -DarchetypeVersion=1.3.8

I worked it with version 1.3.8

Add the following when requested by Maven

groupId:io.dropwizard

artifactId:dropwizard-core

version:1

package:com.example.helloworld

name:HelloWorld

This should create a project under folder dropwizard-core where you will see a POM file which contains the dependencies

Download dropwizard-core.7z from the X drive and extract on your machine. Copy HelloWorldApplication.java from dropwizard-core\src\main\java\com\example\helloworld to drop\dropwizard-core\src\main\java\com\example\helloworld

Also copy HelloWorldConfiguration.java from X drive and put it in helloworld folder

Create an example.yml file with the following and put it in the same folder as the POM file

template: Hello, %s!

defaultName: Stranger

Copy Saying.java from dropwizard-core\src\main\java\com\example\helloworld\api and place it in the api folder of your project

Copy HelloWorldResource.java from dropwizard-core\src\main\java\com\example\helloworld\resources and place it in the resources folder of your project

This resource file is the most interesting as it implements the REST verbs

@Path("/hello-world")

@Produces(MediaType.APPLICATION\_JSON)

public class HelloWorldResource {

private final String template;

private final String defaultName;

private final AtomicLong counter;

public HelloWorldResource(String template, String defaultName) {

this.template = template;

this.defaultName = defaultName;

this.counter = new AtomicLong();

}

@GET

@Timed

public Saying sayHello(@QueryParam("name") Optional<String> name) {

final String value = String.format(template, name.orElse(defaultName));

return new Saying(counter.incrementAndGet(), value);

}

Copy TemplateHealthCheck.java from dropwizard-core\src\main\java\com\example\helloworld\health and put it in the health folder.

Check that the following code is in the run method of HelloWorldApplication.java above the final line.

**final TemplateHealthCheck healthCheck =**

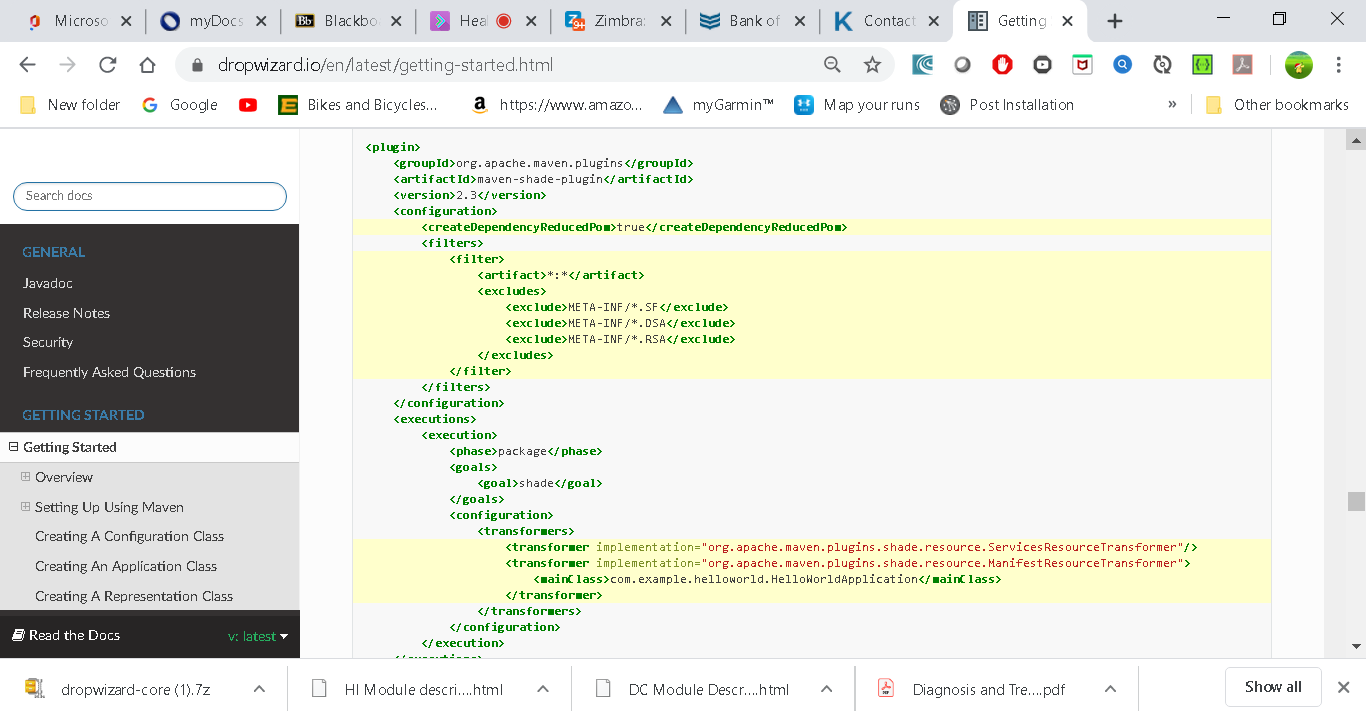
**new TemplateHealthCheck(configuration.getTemplate());**

**environment.healthChecks().register("template", healthCheck);**

**environment.jersey().register(resource);**

**}**

Now build the fat JAR as shown in the tutorial (<https://www.dropwizard.io/1.3.8/docs/getting-started.html> section Building Fat JAR) by checking the plugin lines (in yellow) are in the Maven POM file (they should be!)



Leave the maven-jar-plugins as it is.

Now cd to dropwizard-core and try and build and run your application with

mvn package

To run try this from your project directory

java -jar target\dropwizard-core-1.jar

You should see something like the following:

usage: java -jar hello-world-0.0.1-SNAPSHOT.jar

[-h] [-v] {server} ...

positional arguments:

{server} available commands

optional arguments:

-h, --help show this help message and exit

-v, --version show the service version and exit

The server command requires a configuration file (which is in example.yml which you can copy from the dropwizard-core project to your project)so let’s go ahead and give it [the YAML file we previously saved](https://www.dropwizard.io/1.3.8/docs/getting-started.html#gs-yaml-file):

java -jar target\dropwizard-core-1.jar server example.yml

Your Dropwizard application is now listening on ports 8080 for application requests and 8081 for administration requests. If you press ^C, the application will shut down gracefully, first closing the server socket, then waiting for in-flight requests to be processed, then shutting down the process itself.

However, while it’s up, let’s give it a whirl!

(If it fails to start there may be something already using port 8080 – you can see and kill it using netstat –ano)

In a browser navigate to <http://localhost:8080/hello-world>

You should get back a JSON object with the data

Try http://localhost:8080/hello-world?name=Firstname+Surname

One of the main reasons for using Dropwizard is the out-of-the-box operational tools it provides, all of which can be found on <http://localhost:8081/>

Look at Metrics and Threads.

**Hosting on Docker**

Now we will create a Docker container to host our microservice

First download **Docket Toolbox (for OS < Windows 10 Pro**) (or Docker for Windows – for OS 10 Pro needed!

We will use Docket Toolbox today – its available on the X drive.

<https://docs.docker.com/docker-for-windows/install/>

Manual at: <https://docs.docker.com/docker-for-windows/>

I used Docker Toolbox and followed instructions at : <https://docs.docker.com/toolbox/toolbox_install_windows/>

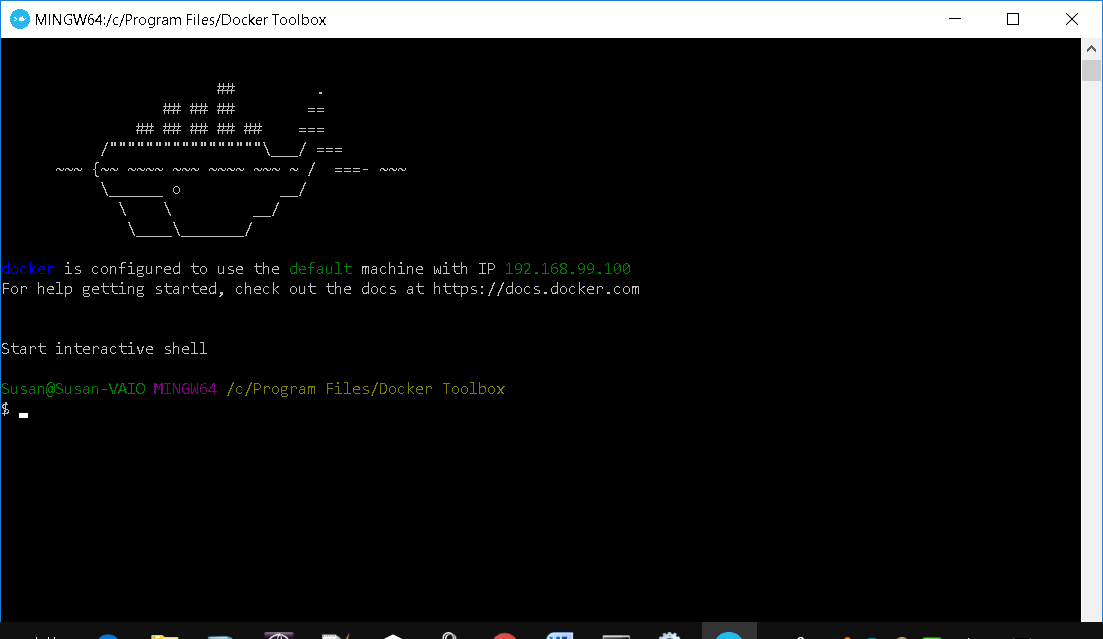
Next understand docker and test toolbox by reading:

<https://docs.docker.com/get-started/>

Start Docker (if using the Toolbox then start the Docker Quickstart Terminal)

Ig Hyper\_V termianal clicked in Windows, do the following

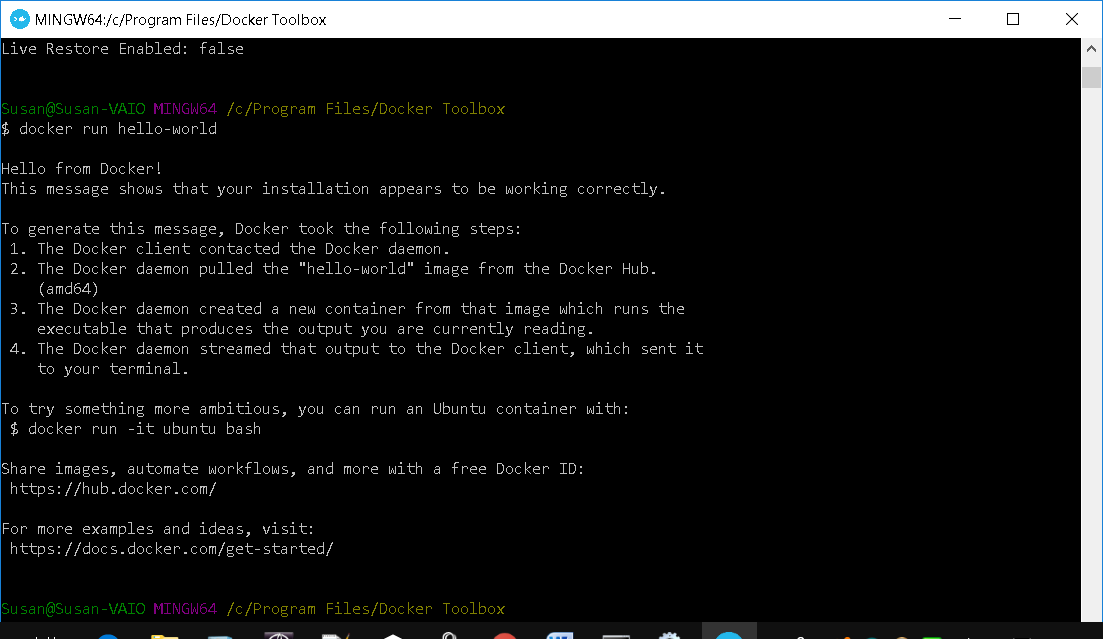
1. Hit the Windows Key and type “windows features”
2. Click on **Turn** Windows features on or **off**.
3. When the **Turn** Windows features on or **off** dialog appears, look for **Hyper-V** and deselect it.
4. Click OK.
5. Restart your computer when prompted.



As a test

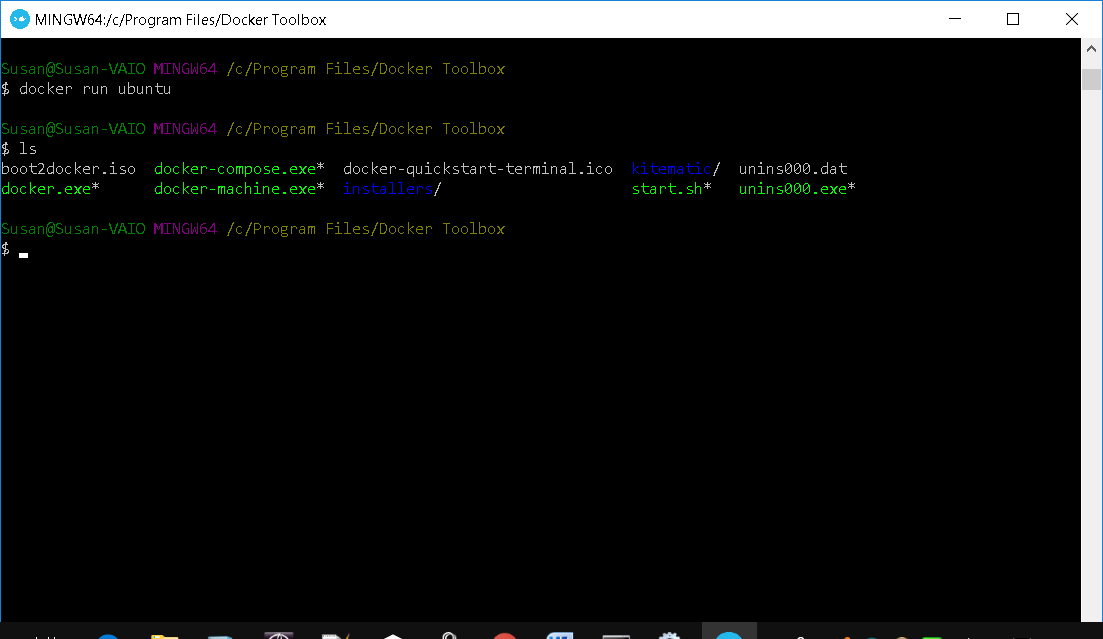
* Try to pull the the hello-world container application

docker run hello-world



* Try to pull and run a Ubuntu container from docker hub (<https://hub.docker.com/>) and run some linux commands

docker run Ubuntu



Next follow instructions in http://honstain.com/dropwizard-in-a-docker-container/

to dockerize the dropwizard application you built earlier. Steps I took are shown below…

I created a file called Dockerfile in C:\drop\dropwizard-core\target (i.e. where my jar is)

Also add example.yml to this folder

Dockerfile contains the following (could change the jdk to Oracle 8?)

FROM openjdk:8-jdk

ADD dropwizard-core-1.jar /data/dropwizard-core-1.jar

ADD example.yml /data/example.yml

RUN java -version

CMD ["java","-jar","/data/dropwizard-core-1.jar","server","/data/example.yml"]

EXPOSE 8080-8081

The Dropwizard Application needs a Java Runtime, so you can start from a base image already available at [Docker Hub](https://hub.docker.com/), for example: openjdk:8-jdk.

You must add the Dropwizard Application files to the image, using the ADD instruction in your Dockerfile.

Next, simply specify the commands of your Dropwizard Application, which you want to execute during image build and container runtime

RUN Just a basic sanity check of the Java version. You can omit.

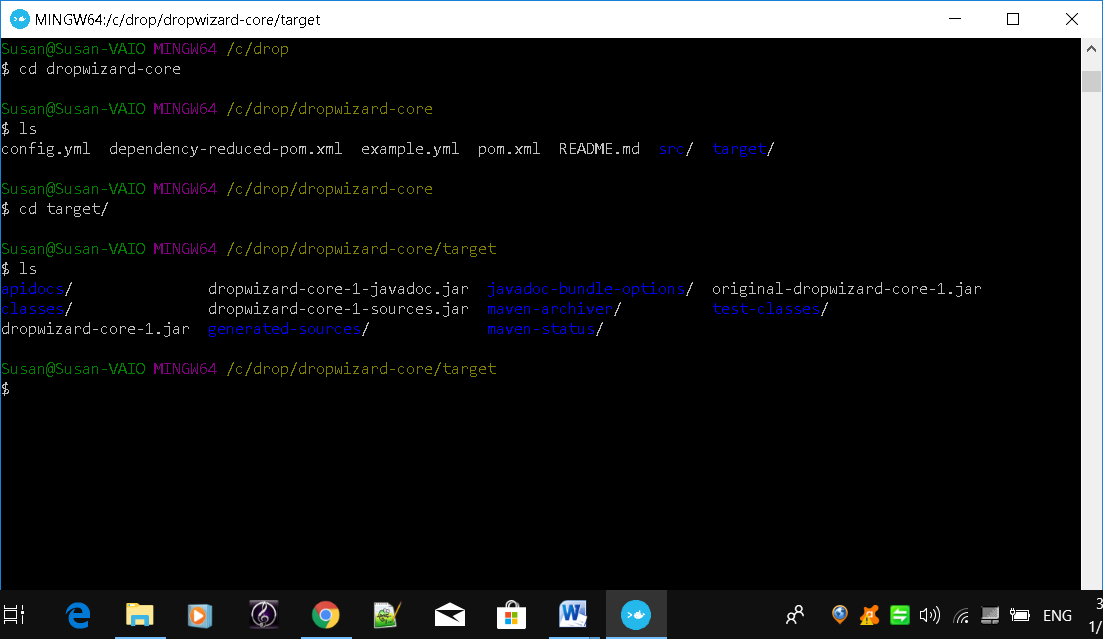
CMD ["java" etc] This is where things get interesting. Based on the best practices guide, I pass in the executable and all the parameters.

Finally, the EXPOSE instruction tells Docker that your container will listen on the specified port(s) at runtime (but this is just for documentation, it doesn’t do this – the run command below does)

You can cd to the target folder in docker command window

cd ..

pwd



### Step 2 - Build

From the root of your project, your going to want to run

# Replace with your desired tag.

docker build –-tag=<INSERT-YOUR-TAG> .

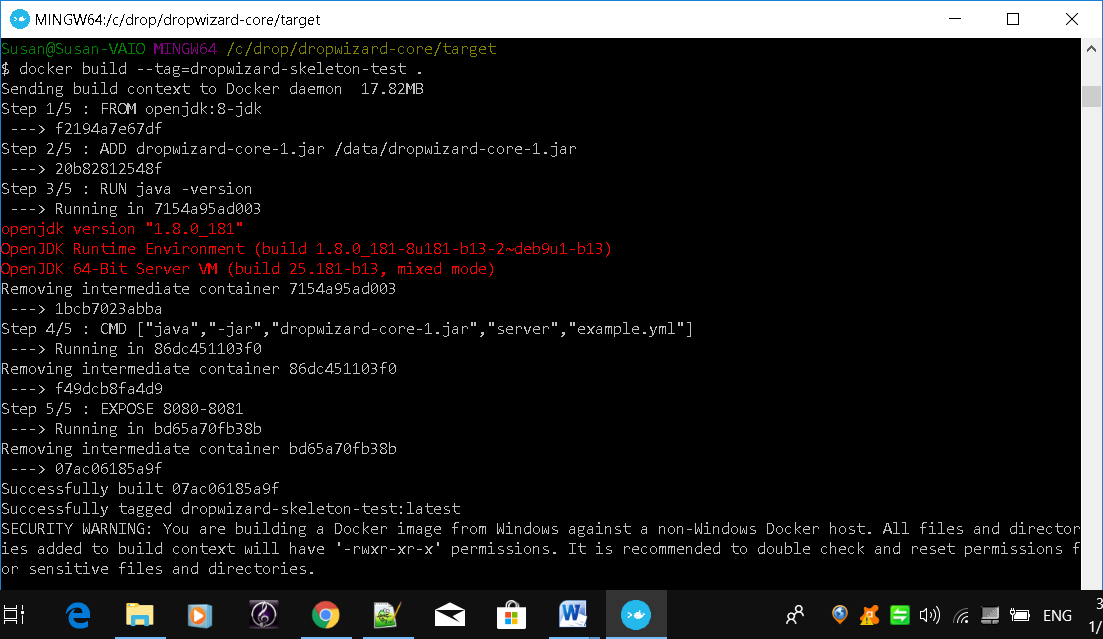
Nothing really crazy is going on here, you should pick your own tag to use, I used

docker build –-tag=dropwizard-skeleton-test .

Don’t forget . at end of command

If you want more info on the build command <https://docs.docker.com/engine/reference/commandline/build/>

Example results:



### Step 3 - Run

Now it is time to run the container.

# Run locally in interactive mode

docker run -p 8080:8080 -it dropwizard-skeleton-test

# Run in detached mode (you will need to look to docker see the log/stdout stream.

docker run -d -p 8080:8080 dropwizard-skeleton-test

Notice that I included the –p option to include a network port binding, which maps 8080 inside the container to port 8080 on the Docker host.  You can verify whether your container is running using the docker ps command.

**Test the application**

Now the application is ready for use. You can access the application using your **Docker host ip address** and the forward port 8080.

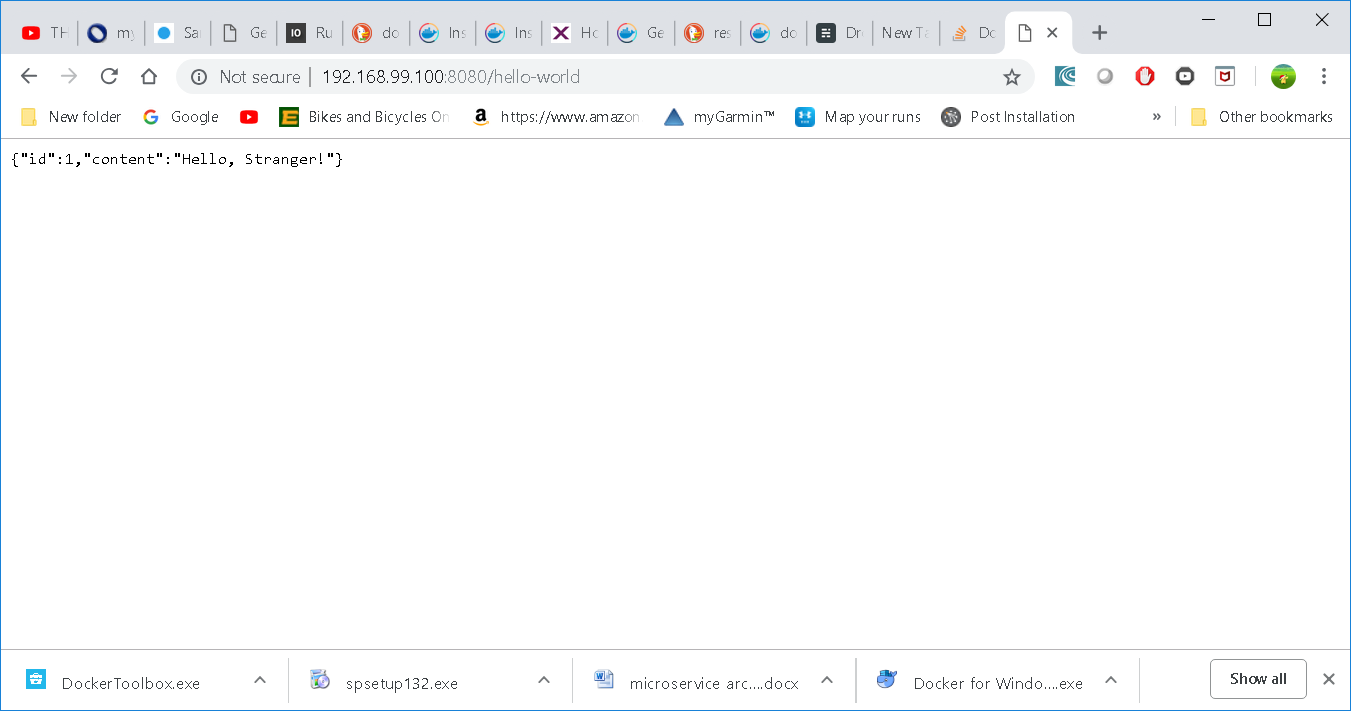
To get the Docker host IP do the following

From your host machine in program files/docker toolbox run

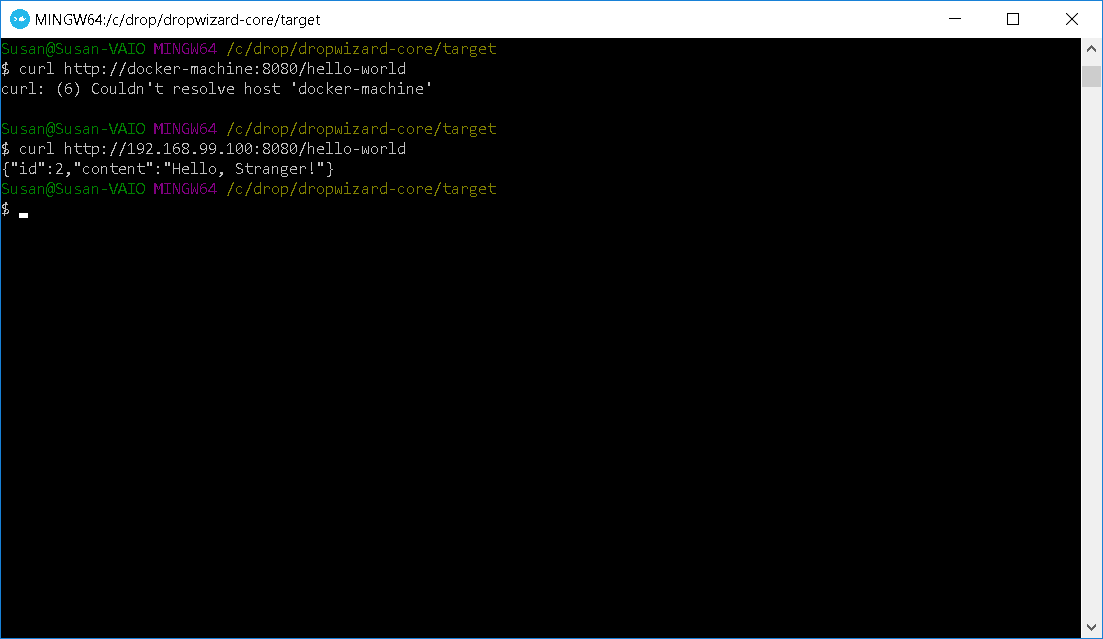
docker-machine.exe ip

(when I ran it I got 192.168.99.100 – or look at the top of the docker terminal above – it should show the IP used)

from your host machine browser try URL <http://ip:8080/hello-world>

or from within the container if you use detached mode

curl <http://ip:8080/hello-world>



You could also try Postman or the Google Advanced Rest Client App (<https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo> ) if you want to run POST commands.

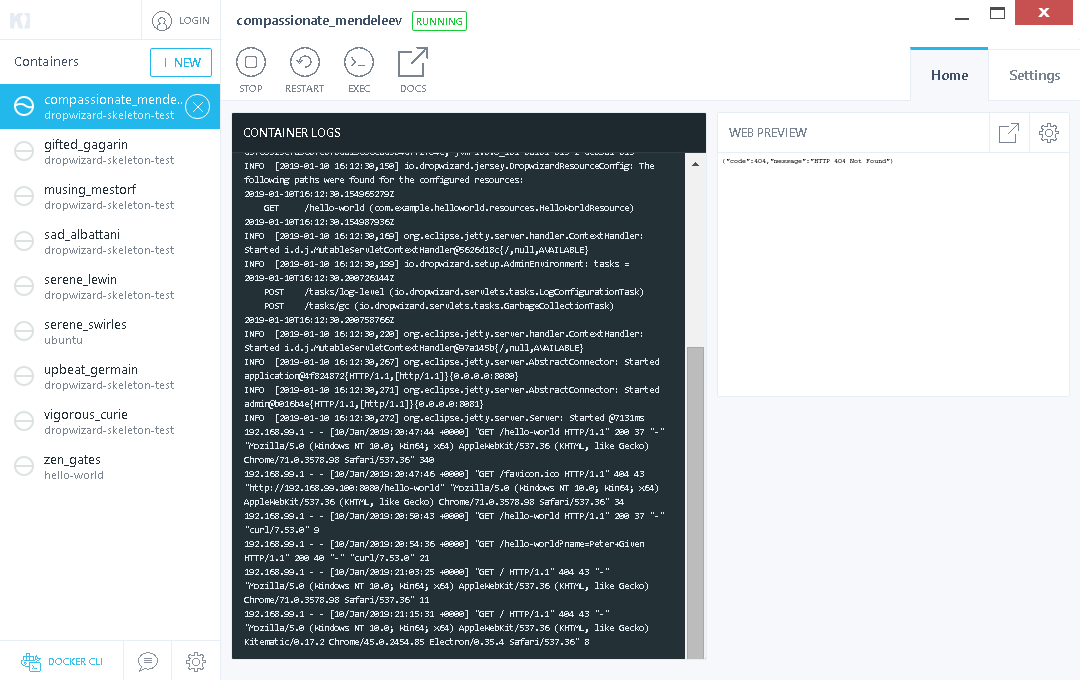
To stop the detached container get the container id

docker ps

docker stop <container id>

**Kitematic**

You can run kitematic (a GUI for Docker) from its folder in docker toolbox in the start menu to see statistics



**Other tutorials (might be of some help)**

https://xebia.com/blog/how-to-dockerize-your-dropwizard-application/

<https://www.blog.philipphauer.de/building-dropwizard-microservice-docker-maven/>

Reference:

Docker command line

<https://docs.docker.com/engine/reference/commandline/cli/>