Creating own Hello World container using Dockerfile

The first thing you need is to create a basic.java file, HelloWorld.java, and add these lines into it:

```
public class HelloWorld {
  public static void main(String[] args){
    System.out.println("Hello World :) ");
  }
}
```

Save and compile it in the command line. From the directory in which you have created your HelloWorld.java, run the command `javac HelloWorld.java`.

Once you do this, you will get the HelloWorld.class file, which later we will build in .jar. But before that, we need to create a simple manifest.txt to make it packed right.

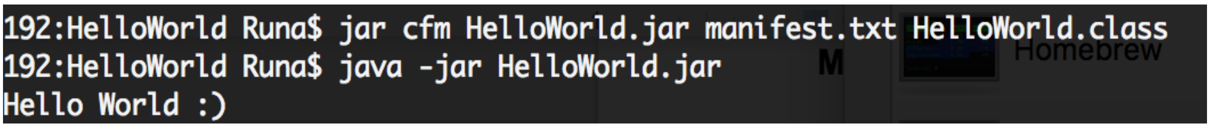
So now, in the same directory, create manifest.txt and place the following lines:

```
Manifest-Version: 1.0
Created-By: Me
Main-Class: HelloWorld
```

Then, in the command line, run the following: `jar cfm HelloWorld.jar manifest.txt HelloWorld.class`.

And to check if everything works correctly, type `java -jar HelloWorld.jar`.

If everything is okay, you should see the following:



```
192:HelloWorld Runa$ jar cfm HelloWorld.jar manifest.txt HelloWorld.class
192:HelloWorld Runa$ java -jar HelloWorld.jar
Hello World :)
```

The next step is to start Docker and create a Dockerfile, a text file that contains the instructions (or commands) used to build a Docker image.

To do that, create the file with the name "Dockerfile" and place the following text in it:

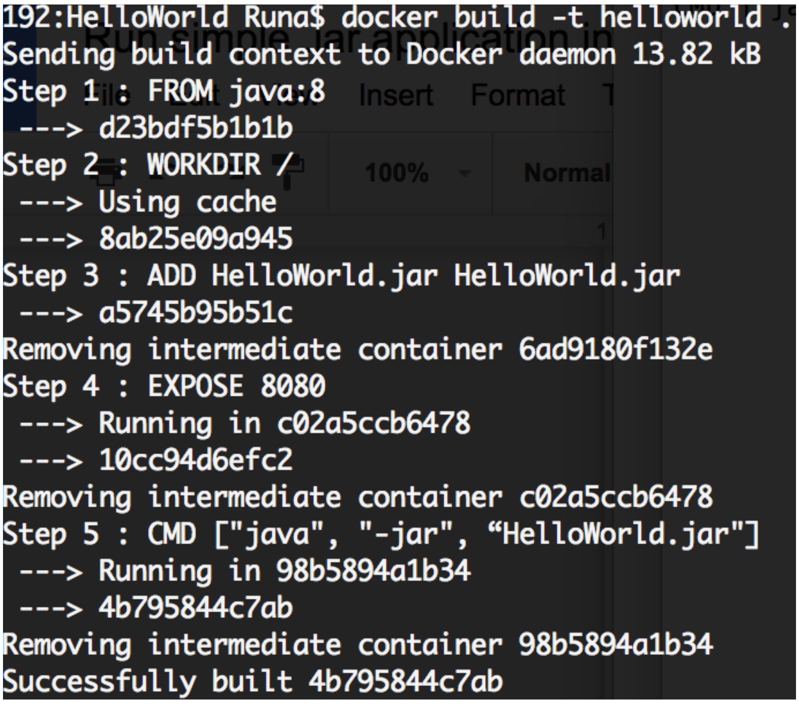
```
FROM java:8
WORKDIR /
ADD HelloWorld.jar HelloWorld.jar
EXPOSE 8080
CMD java - jar HelloWorld.jar
```

Don’t forget to leave the empty line at the end of the file.

Now you are ready to create a Docker image, the result of building a Dockerfile and executing the Dockerfile's commands. It is constructed from a root operating system, installed applications, and commands executed in such a way that it can run your application. A Docker image serves as the basis for Docker containers and is the static template from which they are created.

You need to run in command line the following: `docker build -t helloworld`

As a result, you should see this:

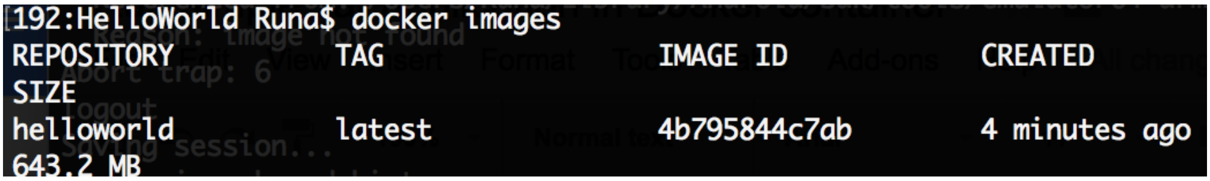


```
192:HelloWorld Runa$ docker build -t helloworld .
Sending build context to Docker daemon 13.82 kB
Step 1 : FROM java:8
 ---> d23bdf5b1b1b
Step 2 : WORKDIR /
 ---> Using cache
 ---> 8ab25e09a945
Step 3 : ADD HelloWorld.jar HelloWorld.jar
 ---> a5745b95b51c
Removing intermediate container 6ad9180f132e
Step 4 : EXPOSE 8080
 ---> Running in c02a5ccb6478
 ---> 10cc94d6efc2
Removing intermediate container c02a5ccb6478
Step 5 : CMD ["java", "-jar", “HelloWorld.jar"]
 ---> Running in 98b5894a1b34
 ---> 4b795844c7ab
Removing intermediate container 98b5894a1b34
Successfully built 4b795844c7ab
```

Then you have to create an account on dockerhub and create the repository "hello-world" to push your image to your repository. Once you register and create a repository, go to command line and log in there with `docker login`.

Then pull that repository: `docker pull /hello-world`

To push your Docker image to DockerHub you need to figure out your Docker_Image_ID. Run the following: `docker images`



```
192:HelloWorld Runa$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
helloworld          latest              4b795844c7ab        4 minutes ago
643.2 MB
```

So you may find your image and see you Image_Id. Now you need to tag and push your image: `docker tag 4b795844c7ab /hello-world`

To read more about working with it, you can go here.

Now you are ready to upload your Docker Image to DockerHub.
Just type: `docker push /hello-world:latest`

To check if everything works fine enter: `docker run /hello-world`

You must see the output: Hello World :)