**Ex. 2 - Docker and Java with a Docker file**

1. The sandbox you will use for Docker is the same Sandbox from Exercise 1. It is located at <https://learning.oreilly.com/scenarios/docker-sandbox/9781492086161/>.
2. We will use some common commands to build a Docker file and build, run a Java 8 image. A good reference for some of these commands and for Docker in general is at <https://docs.docker.com/engine/reference/builder/>. Feel free to use any reference that helps your understanding.
3. Check for java on Linux by typing which java. We see a directory /usr/bin/java. So we have Java on the system, another nice thing about Linux, but we will pull a specific image below.
4. From the terminal run command docker images to see images. We do not see a Java image. Why don’t we pull a Java 8 image?
5. To get a Java image we can type docker pull openjdk:8. But we don’t have to do this because we will be building an application with a Docker file that accomplishes the same thing in a later step.
6. Type docker images again. We now see openjdk 8 and we will use this.
7. Type pwd to see the current directory. Another command you can type to have the prompt show you the present working directory is PS1='$(pwd)>'.
8. Change the PS1 primary command prompt to that in Step 7 and see what you get for the prompt now. Remember the command env in Linux. What does it output? Type the command and see if you can see PS1.
9. Normally we pull an image as in step 5. From Dockerfile1Java or the code below, click on the New File icon in the sandbox workspace directory (top frame editor - Explorer) with a name of Dockerfile. Copy the code below or from Dockerfile1Java if you have access to this file. Better alternative is to type the commands into the file.

FROM openjdk:8

COPY . /usr/src/myapp

WORKDIR /usr/src/myapp

RUN javac Main.java

CMD ["java", "Main"]

1. Can you state what all the commands do in this Dockerfile? Check the reference documentation in step 2 or search another location.
2. Now create a Java file like below. Click on the New File icon in the Explorer file frame and type Main.java for the title. Copy the code from below or in Main.java if you have access to the file.
3. This can be any Java application actually. The sample below is basic as you see.

public class Main {

public static void main(String[] args) {

System.out.println("This is a test");

}

}

1. Now click again on the New File icon in Explorer frame and create a shell script file with a name docker1.sh. From this file or the code below copy and paste the code. Please understand what the script does!

#docker1.sh

docker build -t java-app . # tag build of Dockerfile - this will be name of new image

docker run -it java-app # run app

1. We need to add execute permission to the script so we can execute it and run the contents of the file. Do you remember how to do this? Run the command chmod +x docker1.sh Do you know what the command states? This says to add execute permission to all three groups of users – the owner, group, and all others. Run ls -al or ll (alias) to see the execute permission added.
2. Now from the terminal run ./docker1.sh After you run this file you should see the output of the application.
3. Now run docker images | grep java and you should see the java-app image you just created.
4. Notice our Java application is in the default package and this is not usually best practice. In other words there is no package statement. Why don’t we add one?
5. Copy the new code or just add the package statement to Main.java.

**package com.hcl;**

public class Main {

public static void main(String[] args) {

System.out.println("This is a test");

}

}

1. Now we must change the Docker file. Why is this? Do you remember when you compiled a Java program from the command line with the command javac? This compiled the source file to a class file with a class file extension. Then you actually ran the Java application with the command java. But when we have the default package we don’t need to be concerned with where or how we run the application, in terms of the **classpath** (defaults to current directory but we can use a switch to specify other directories also) or folder structure. Remember, a package is a folder structure below the **classpath**. So in the Docker file below (or Docker2Java) we need another command WORKDIR to change the directory (like cd to Docker) to the directory where the application was compiled (/usr/src/myapp). (Remember one of your tasks earlier was to find what the commands in **Dockerfile** mean so you should know about WORKDIR anyway.) This directory is the parent directory to the package directories (com/hcl). Note also that in the CMD command we need to specify the fully qualified class name (package and class) as the second parameter, just like when you executed the application from the command line. If this is strange to you or you don’t know or remember these concepts, please review!

FROM openjdk:8

COPY . /usr/src/myapp/com/hcl

WORKDIR /usr/src/ myapp/com/hcl

RUN javac Main.java

WORKDIR /usr/src/myapp

CMD ["java", "com.hcl.Main"]

1. Run the script docker1.sh as before, follow the screen output along the way, and then you should get the application output again.
2. Some more questions for you here and below. Do we have to specify the Docker file with the default name **Dockerfile** or can we tell Docker to use another name with a switch or argument?
3. Can you understand some other Docker file commands that you see in the documentation?
4. What about the directory /usr/src/myapp that we used in the Docker file? Can you find this directory on your system? If not, why not and where is it?
5. Do you feel comfortable in a Linux environment. DevOps and other tools, applications favor Linux or some form of Unix. Some of you have Macs and that is Linux based. But Docker, as stated before, can be used on Windows. When we go through some tutorials, or search answers to your questions, many are explained assuming you are running Linux.
6. Do not worry if this and other exercises seem long, tedious, have a lot of information. These are meant just as an introduction and to familiarize you with Docker and DevOps in general. So as you practice and get more experience you will gain confidence.
7. (Optional) Some you may know or have an interest in Python. You can find a Docker file with name Dockerfilepython. There is a Python code file – args.py, script file – docker2.sh. Can you use the information in this file to create a Python image, build and run the application?