

Docker Java Application Example

As, we have mentioned earlier that docker can execute any application.

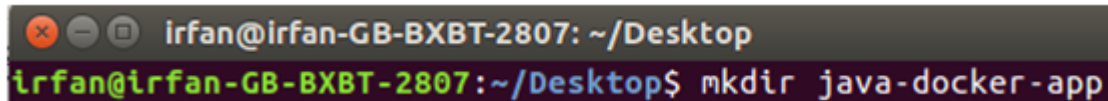
Here, we are creating a Java application and running by using the docker. This example includes the following steps.

1. Create a directory

Directory is required to organize files. Create a director by using the following command.

```
$ mkdir java-docker-app
```

See, screen shot for the above command.

A terminal window with a dark background. The prompt is 'irfan@irfan-GB-BXBT-2807: ~/Desktop'. The command 'mkdir java-docker-app' is entered and executed, with the output 'irfan@irfan-GB-BXBT-2807:~/Desktop\$ mkdir java-docker-app' shown in green text.

2. Create a Java File

Now create a Java file. Save this file as **Hello.java** file.

// **Hello.java**

```
class Hello{  
    public static void main(String[] args){  
        System.out.println("This is java app \n by using Docker");  
    }  
}
```

Save it inside the directory **java-docker-app** as Hello.java.

3. Create a Dockerfile

After creating a Java file, we need to create a Dockerfile which contains instructions for the Docker. Dockerfile does not contain any file extension. So, save it simple with **Dockerfile** name.

// **Dockerfile**

```
FROM java:8  
COPY . /var/www/java  
WORKDIR /var/www/java  
RUN javac Hello.java  
CMD ["java", "Hello"]
```

Write all instructions in uppercase because it is convention. Put this file inside **java-docker-app** directory. Now we have Dockerfile parallel to Hello.java inside the **java-docker-app** directory.

See, your folder inside must look like the below.



Dockerfile



Hello.java

4. Build Docker Image

After creating Dockerfile, we are changing working directory.

```
$ cd java-docker-app
```

See, the screen shot.

```
irfan@irfan-GB-BXBT-2807: ~/Desktop/java-docker-app
irfan@irfan-GB-BXBT-2807:~/Desktop$ cd java-docker-app/
irfan@irfan-GB-BXBT-2807:~/Desktop/java-docker-app$
```

Now, create an image by following the below command. we must login as root in order to create an image. In this example, we have switched to as a root user. In the following command, **java-app** is name of the image. We can have any name for our docker image.

```
$ docker build -t java-app .
```

See, the screen shot of the above command.

```
root@irfan-GB-BXBT-2807: /home/irfan/Desktop/java-docker-app
root@irfan-GB-BXBT-2807:/home/irfan/Desktop# cd java-docker-app/
root@irfan-GB-BXBT-2807:/home/irfan/Desktop/java-docker-app# docker build -t java-app .
Sending build context to Docker daemon 3.072 kB
Step 1/5 : FROM java:8
--> a001fc27db5a
Step 2/5 : COPY . /var/www/html
--> Using cache
--> 8052cbc8ca31
Step 3/5 : WORKDIR /var/www/html
--> Using cache
--> 07977e913cb3
Step 4/5 : RUN javac Hello.java
--> Using cache
--> 0d6e2c3fbd77
Step 5/5 : CMD java Hello
--> Using cache
--> 3f6fb4241c06
Successfully built 3f6fb4241c06
root@irfan-GB-BXBT-2807:/home/irfan/Desktop/java-docker-app#
```

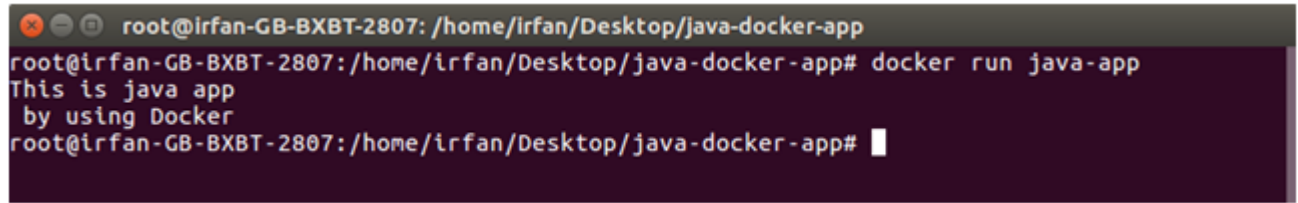
After successfully building the image. Now, we can run our docker image.

5. Run Docker Image

After creating image successfully. Now we can run docker by using run command. The following command is used to run java-app.

```
$ docker run java-app
```

See, the screen shot of the above command.

A terminal window with a dark background and light text. The title bar shows 'root@irfan-GB-BXBT-2807: /home/irfan/Desktop/java-docker-app'. The terminal content shows the command 'docker run java-app' being executed, followed by the output 'This is java app by using Docker'. The prompt 'root@irfan-GB-BXBT-2807: /home/irfan/Desktop/java-docker-app#' is visible at the end of the line.

```
root@irfan-GB-BXBT-2807: /home/irfan/Desktop/java-docker-app
root@irfan-GB-BXBT-2807: /home/irfan/Desktop/java-docker-app# docker run java-app
This is java app
by using Docker
root@irfan-GB-BXBT-2807: /home/irfan/Desktop/java-docker-app#
```

Here, we can see that after running the java-app it produced an output.

Now, we have run docker image successfully on your system. Apart from all these you can also use other commands as well.