RUNNING A JAVA APPLICATION ON DOCKER

We are going to see an example to run a Java application in different ways. There is a base image for Java in the following link
https://hub.docker.com/_/openjdk?tab=description&page=2

Let's create a folder called **java_app** with a single file Main.java whose content is:

```
public class Main {

 public static void main(String[] args) {
 // Prints "Hello, World" to the terminal window.
 System.out.println("Hello, World");
 }

}
```

1. Run using a Dockerfile

It is possible to create an image which compiles and runs the file. We will use the last **openjdk** version. The image will be created in the folder **java_app**

```
FROM openjdk
COPY . /usr/src/myapp
WORKDIR /usr/src/myapp
RUN javac Main.java
CMD ["java", "Main"]
```

Basically, we copy the contents from the current directory into **/usr/src/myapp**. Then, the Java application will be complied and run.

Let's build the image from the current folder **java_app**:

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

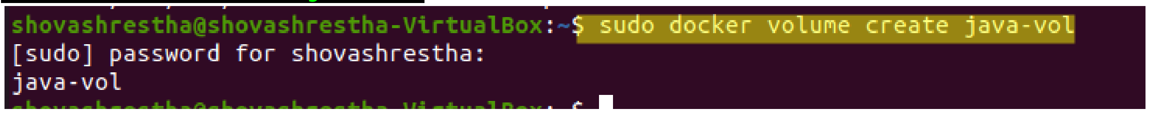
And then we will start a container to run the Java file.

```
docker run --rm --name my-running-app my-java-app
```

2. Run using volumes non-interactively

First, let's create a **volume** to save the Java application

```
docker volume create java-vol
```

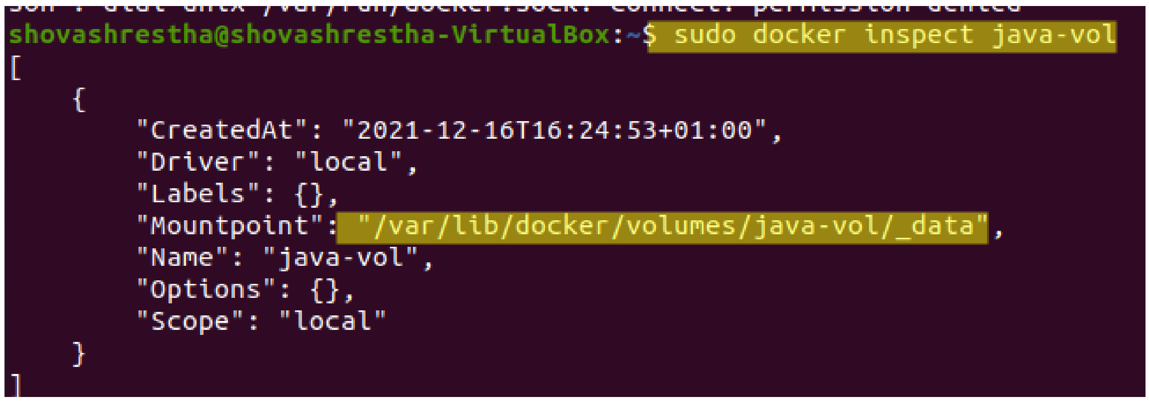


```
shovashrestha@shovashrestha-VirtualBox:~$ sudo docker volume create java-vol
[sudo] password for shovashrestha:
java-vol
```

Before running a container, we should check where it is really located with **docker inspect**.

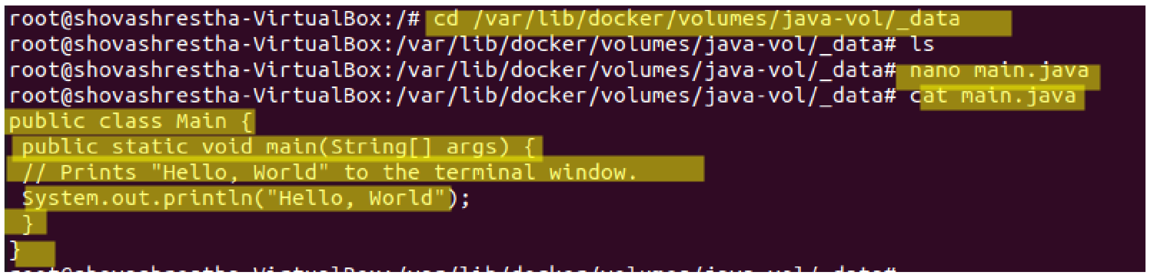
```
docker inspect java-vol
```

For example, in this case, the real content location will be **/var/lib/docker/volumes/java vol/_data**



```
shovashrestha@shovashrestha-VirtualBox:~$ sudo docker inspect java-vol
[
    {
        "CreatedAt": "2021-12-16T16:24:53+01:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/java-vol/_data",
        "Name": "java-vol",
        "Options": {},
        "Scope": "local"
    }
]
```

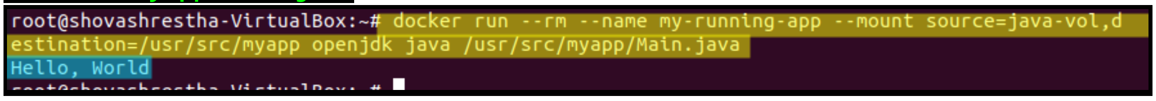
Then, you should copy the Java file into the volume folder.



```
root@shovashrestha-VirtualBox:/# cd /var/lib/docker/volumes/java-vol/_data
root@shovashrestha-VirtualBox:/var/lib/docker/volumes/java-vol/_data# ls
root@shovashrestha-VirtualBox:/var/lib/docker/volumes/java-vol/_data# nano main.java
root@shovashrestha-VirtualBox:/var/lib/docker/volumes/java-vol/_data# cat main.java
public class Main {
 public static void main(String[] args) {
 // Prints "Hello, World" to the terminal window.
 System.out.println("Hello, World");
 }
}
```

At this point we can compile the Java file associating the volume location.

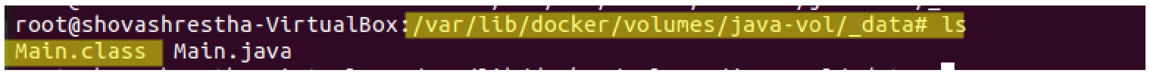
```
docker run --rm --name my-running-app --mount
source=java-vol,destination=/usr/src/myapp openjdk javac
/usr/src/myapp/main.java
```



```
root@shovashrestha-VirtualBox:~# docker run --rm --name my-running-app --mount source=java-vol,d
estination=/usr/src/myapp openjdk java /usr/src/myapp/Main.java
Hello, World
```

It is not necessary to create an image, but the main disadvantage is that we need to create two different containers to compile and run.

The first container should have created the **Main.class** in the volume folder.



```
root@shovashrestha-VirtualBox:/var/lib/docker/volumes/java-vol/_data# ls
Main.class  Main.java
```

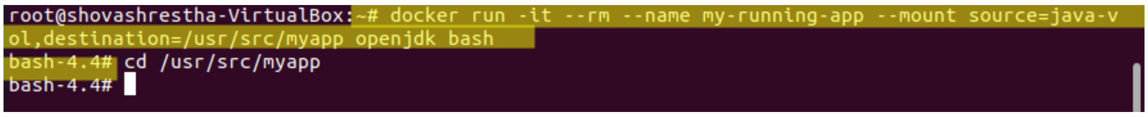
Finally, we can run the compiled file.

```
docker run --rm --name my-running-app --mount source=java
vol,destination=/usr/src/myapp openjdk java /usr/src/myapp/Main.java
```

3. Run using volumes interactively

Using the same volume as in part 2, it is possible to run a container with an interactive bash from an **openjdk** image (default option without bash opens a jshell, whose commands we have not studied).

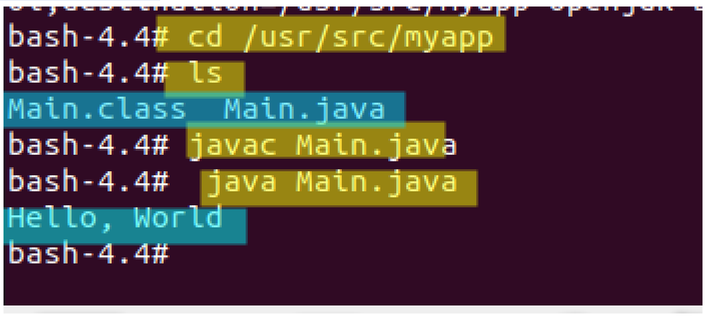
```
docker run -it --rm --name my-running-app --mount source=java
vol,destination=/usr/src/myapp openjdk bash
```



```
root@shovashrestha-VirtualBox:~# docker run -it --rm --name my-running-app --mount source=java-v
ol,destination=/usr/src/myapp openjdk bash
bash-4.4# cd /usr/src/myapp
bash-4.4#
```

Then, we can compile and run from bash using the **javac** and **java** command in the directory **/usr/src/myapp** in which we have the volume.

```
bash-4.4# cd /usr/src/myapp
bash-4.4# javac Main.java
bash-4.4# java Main.java
Hello, World
```



```
bash-4.4# cd /usr/src/myapp
bash-4.4# ls
Main.class  Main.java
bash-4.4# javac Main.java
bash-4.4#  java Main.java
Hello, World
bash-4.4#
```

Then, we finish the bash process with the corresponding **exit** command.

Regardless of the alternative, this example shows the utility of Docker. We do not need to install Java in our computer. And every employee has the same container with the same configuration.