A Report on Comprehensive Docker, Maven, and Java Version Management

By,

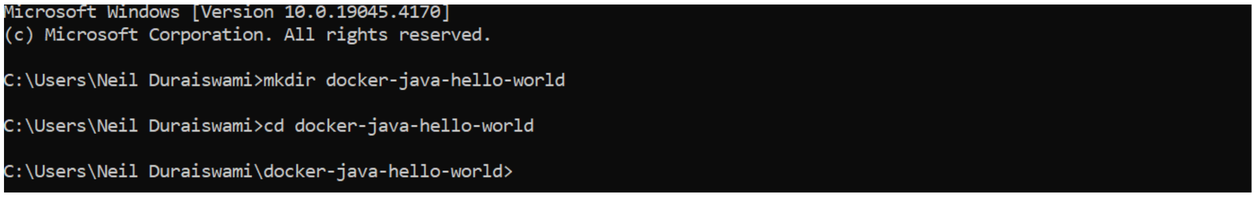
Neil Duraiswami.

Contents

Objective

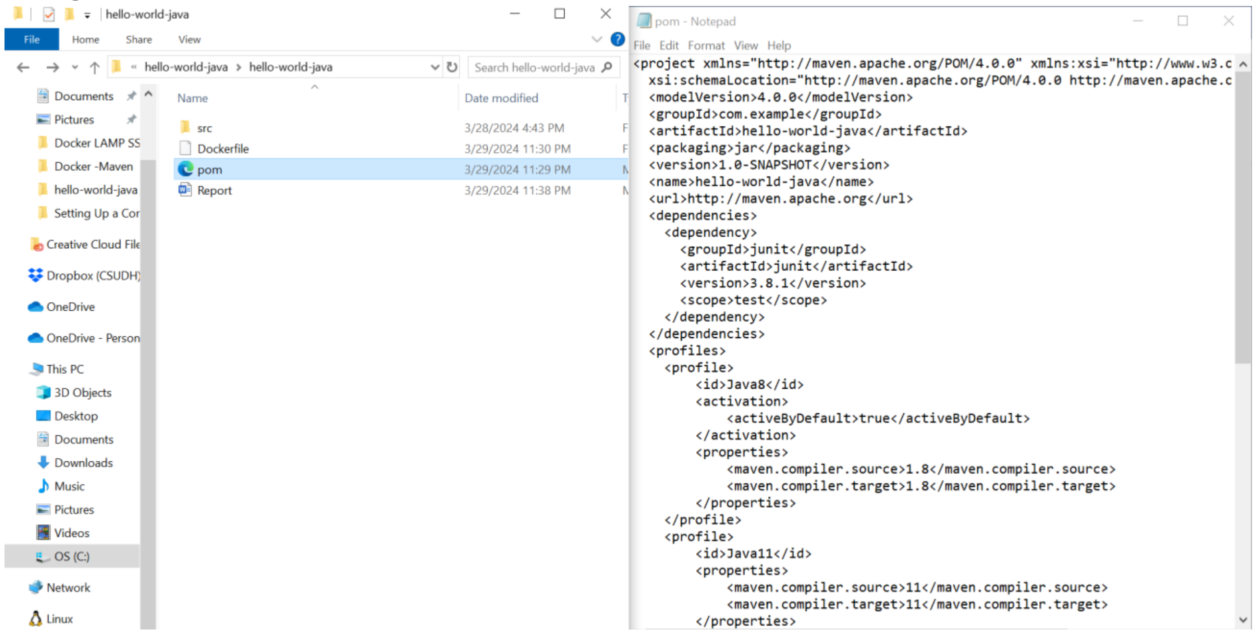
The objective of this project was to demonstrate the process of building and running a simple "Hello World" Java application using Docker and Maven across different Java environments: Java 8, Java 11, and Java 17.

Task 1: Java Project Setup and Maven Configuration



```
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Neil Duraiswami>mkdir docker-java-hello-world

C:\Users\Neil Duraiswami>cd docker-java-hello-world

C:\Users\Neil Duraiswami\docker-java-hello-world>
```

- Initialized a Maven project with a HelloWorld class that prints a message to the console using **mvn archetype:generate**.
- Configured Maven for multiple Java versions (Java 8, Java 11, and Java 17) by editing the pom.xml file. Profiles were created for each Java version with specific compiler configurations.

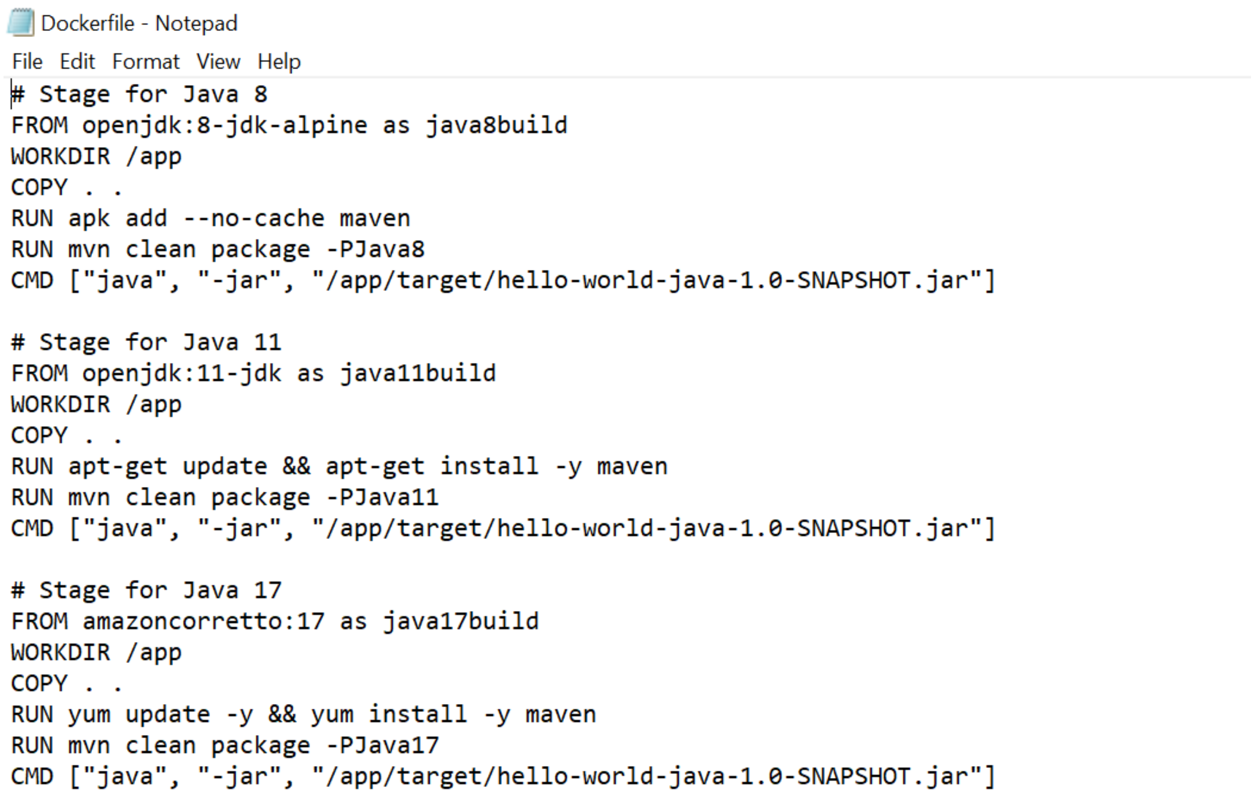


Name	Date modified
src	3/28/2024 4:43 PM
Dockerfile	3/29/2024 11:30 PM
pom	3/29/2024 11:29 PM
Report	3/29/2024 11:38 PM

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.c
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.c
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>hello-world-java</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>hello-world-java</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <profiles>
    <profile>
        <id>Java8</id>
        <activation>
            <activeByDefault>true</activeByDefault>
        </activation>
        <properties>
            <maven.compiler.source>1.8</maven.compiler.source>
            <maven.compiler.target>1.8</maven.compiler.target>
        </properties>
    </profile>
    <profile>
        <id>Java11</id>
        <properties>
            <maven.compiler.source>11</maven.compiler.source>
            <maven.compiler.target>11</maven.compiler.target>
        </properties>
```

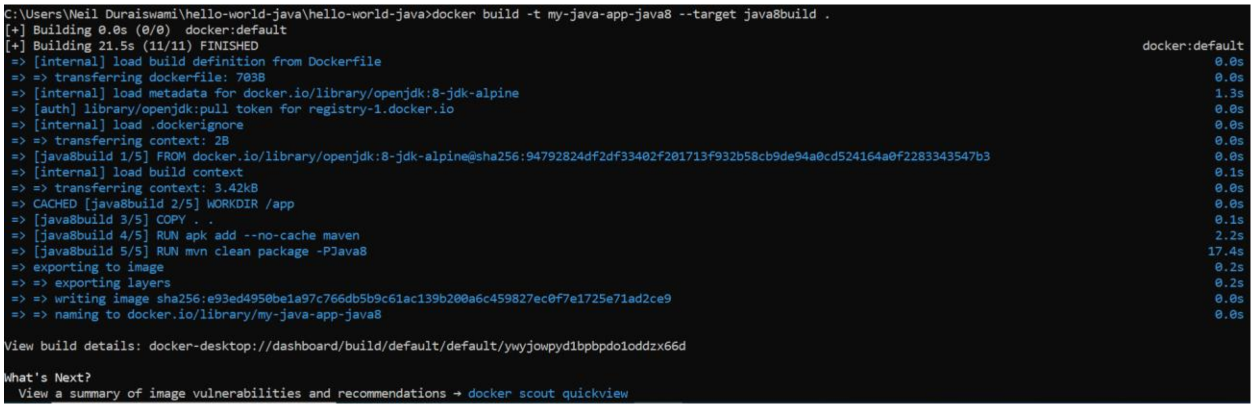
Task 2: Dockerfile Creation for Multi-Version Builds

- Created a multi-stage Dockerfile to build the Java application for each Java version.

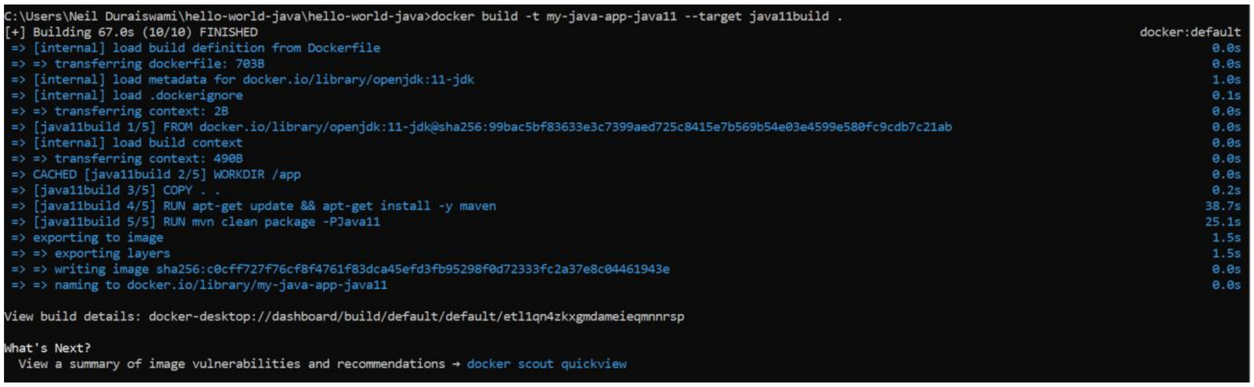


```
# Stage for Java 8
FROM openjdk:8-jdk-alpine as java8build
WORKDIR /app
COPY . .
RUN apk add --no-cache maven
RUN mvn clean package -PJava8
CMD ["java", "-jar", "/app/target/hello-world-java-1.0-SNAPSHOT.jar"]

# Stage for Java 11
FROM openjdk:11-jdk as java11build
WORKDIR /app
COPY . .
RUN apt-get update && apt-get install -y maven
RUN mvn clean package -PJava11
CMD ["java", "-jar", "/app/target/hello-world-java-1.0-SNAPSHOT.jar"]

# Stage for Java 17
FROM amazoncorretto:17 as java17build
WORKDIR /app
COPY . .
RUN yum update -y && yum install -y maven
RUN mvn clean package -PJava17
CMD ["java", "-jar", "/app/target/hello-world-java-1.0-SNAPSHOT.jar"]
```

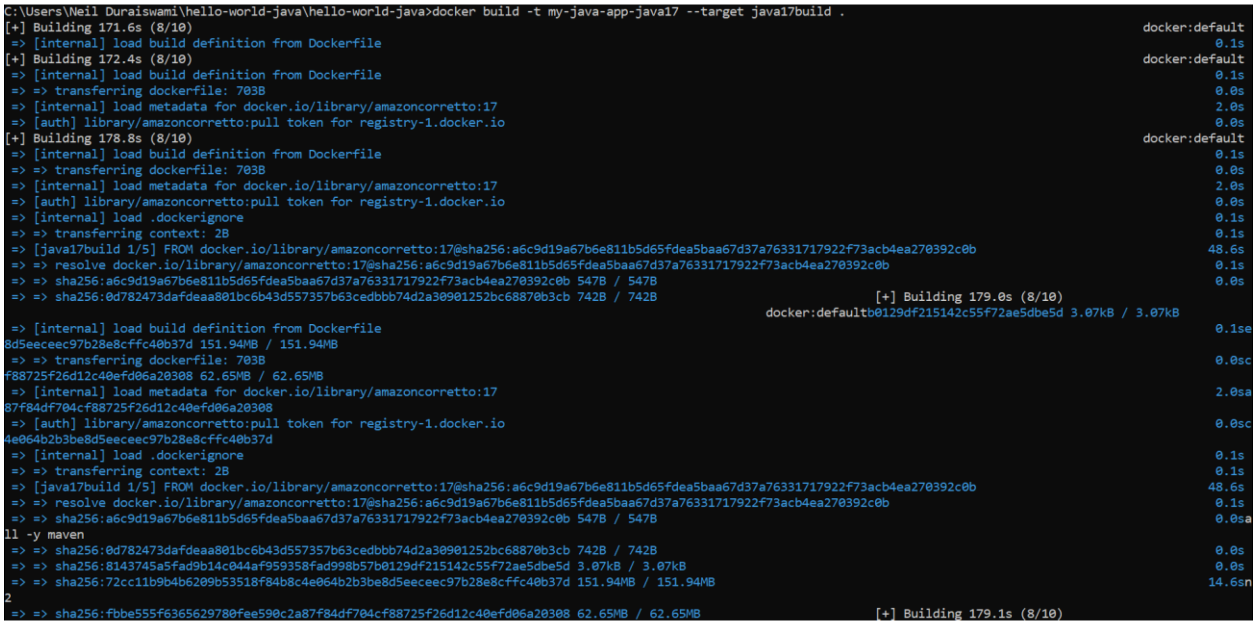
- Utilized official Maven Docker images for each Java version to build the application.
- Utilized multi-stage builds to optimize Docker images by including only necessary dependencies.
- The final image stage copied the compiled JAR file from each Java version stage to the final image.



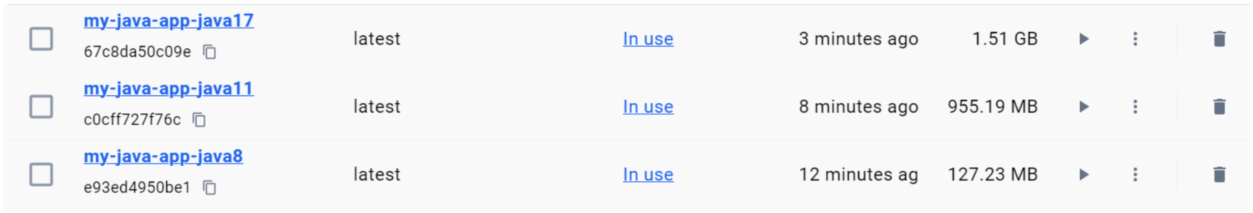
```
C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker build -t my-java-app-java8 --target java8build .
[+] Building 0.0s (0/0)  docker:default
[+] Building 21.5s (11/11) FINISHED                                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                                            0.0s
 => => transferring dockerfile: 703B                                                                                                            0.0s
 => [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine                                                                         1.3s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                                                                  0.0s
 => [internal] load .dockerignore                                                                                                               0.0s
 => => transferring context: 2B                                                                                                                 0.0s
 => [java8build 1/5] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3         0.0s
 => [internal] load build context                                                                                                               0.1s
 => => transferring context: 3.42kB                                                                                                             0.0s
 => CACHED [java8build 2/5] WORKDIR /app                                                                                                        0.0s
 => [java8build 3/5] COPY . .                                                                                                                   0.1s
 => [java8build 4/5] RUN apk add --no-cache maven                                                                                               2.2s
 => [java8build 5/5] RUN mvn clean package -PJava8                                                                                             17.4s
 => exporting to image                                                                                                                          0.2s
 => => exporting layers                                                                                                                         0.2s
 => => writing image sha256:e93ed4950be1a97c766db5b9c61ac139b200a6c459827ec0f7e1725e71ad2ce9                                                    0.0s
 => => naming to docker.io/library/my-java-app-java8                                                                                            0.0s

View build details: docker-desktop://dashboard/build/default/default/ywyjowpyd1bpbpdo1oddzx66d

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
```



```
C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker build -t my-java-app-java11 --target java11build .
[+] Building 67.0s (10/10) FINISHED                                                                       docker:default
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 703B                                                                                0.0s
 => [internal] load metadata for docker.io/library/openjdk:11-jdk                                                   1.0s
 => [internal] load .dockerignore                                                                                   0.1s
 => => transferring context: 2B                                                                                     0.0s
 => [java11build 1/5] FROM docker.io/library/openjdk:11-jdk@sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c21ab   0.0s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 490B                                                                                   0.0s
 => CACHED [java11build 2/5] WORKDIR /app                                                                           0.0s
 => [java11build 3/5] COPY . .                                                                                      0.2s
 => [java11build 4/5] RUN apt-get update && apt-get install -y maven                                               38.7s
 => [java11build 5/5] RUN mvn clean package -PJava11                                                               25.1s
 => exporting to image                                                                                              1.5s
 => => exporting layers                                                                                             1.5s
 => => writing image sha256:c0cff727f76cf8f4761f83dca45efd3fb95298f0d72333fc2a37e8c04461943e                        0.0s
 => => naming to docker.io/library/my-java-app-java11                                                               0.0s

View build details: docker-desktop://dashboard/build/default/default/etl1qn4zkxgmdameieqmnnrsp

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
```



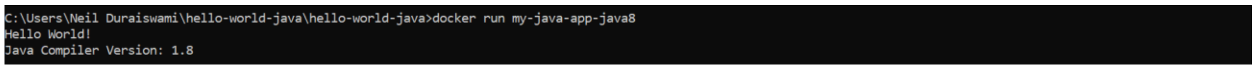
```
C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker build -t my-java-app-java17 --target java17build .
[+] Building 171.6s (8/10)                                                                                docker:default
 => [internal] load build definition from Dockerfile                                                                0.1s
[+] Building 172.4s (8/10)                                                                                docker:default
 => [internal] load build definition from Dockerfile                                                                0.1s
 => => transferring dockerfile: 703B                                                                                0.0s
 => [internal] load metadata for docker.io/library/amazoncorretto:17                                                2.0s
 => [auth] library/amazoncorretto:pull token for registry-1.docker.io                                               0.0s
[+] Building 178.8s (8/10)                                                                                docker:default
 => [internal] load build definition from Dockerfile                                                                0.1s
 => => transferring dockerfile: 703B                                                                                0.0s
 => [internal] load metadata for docker.io/library/amazoncorretto:17                                                2.0s
 => [auth] library/amazoncorretto:pull token for registry-1.docker.io                                               0.0s
 => [internal] load .dockerignore                                                                                   0.1s
 => => transferring context: 2B                                                                                     0.1s
 => [java17build 1/5] FROM docker.io/library/amazoncorretto:17@sha256:a6c9d19a67b6e811b5d65fdea5baa67d37a76331717922f73acb4ea270392c0b  48.6s
 => => resolve docker.io/library/amazoncorretto:17@sha256:a6c9d19a67b6e811b5d65fdea5baa67d37a76331717922f73acb4ea270392c0b   0.1s
 => => sha256:a6c9d19a67b6e811b5d65fdea5baa67d37a76331717922f73acb4ea270392c0b 547B / 547B                           0.0s
 => => sha256:0d782473dafdeaa801bc6b43d557357b63cedbbb74d2a30901252bc68870b3cb 742B / 742B          [+] Building 179.0s (8/10)
                                                                     docker:defaultb0129df215142c55f72ae5dbe5d 3.07kB / 3.07kB
 => [internal] load build definition from Dockerfile                                                                0.1se
8d5eeceec97b28e8cffc40b37d 151.94MB / 151.94MB
 => => transferring dockerfile: 703B                                                                                0.0sc
f88725f26d12c40efd06a20308 62.65MB / 62.65MB
 => [internal] load metadata for docker.io/library/amazoncorretto:17                                                2.0sa
87f84df704cf88725f26d12c40efd06a20308
 => [auth] library/amazoncorretto:pull token for registry-1.docker.io                                               0.0sc
4e064b2b3be8d5eeceec97b28e8cffc40b37d
 => [internal] load .dockerignore                                                                                   0.1s
 => => transferring context: 2B                                                                                     0.1s
 => [java17build 1/5] FROM docker.io/library/amazoncorretto:17@sha256:a6c9d19a67b6e811b5d65fdea5baa67d37a76331717922f73acb4ea270392c0b  48.6s
 => => resolve docker.io/library/amazoncorretto:17@sha256:a6c9d19a67b6e811b5d65fdea5baa67d37a76331717922f73acb4ea270392c0b   0.1s
 => => sha256:a6c9d19a67b6e811b5d65fdea5baa67d37a76331717922f73acb4ea270392c0b 547B / 547B                           0.0sa
ll -y maven
 => => sha256:0d782473dafdeaa801bc6b43d557357b63cedbbb74d2a30901252bc68870b3cb 742B / 742B                           0.0s
 => => sha256:8143745a5fad9b14c044af959358fad998b57b0129df215142c55f72ae5dbe5d 3.07kB / 3.07kB                       0.0s
 => => sha256:72cc11b9b4b6209b53518f84b8c4e064b2b3be8d5eeceec97b28e8cffc40b37d 151.94MB / 151.94MB                  14.6sn
2
 => => sha256:fbbe555f6365629780fee590c2a87f84df704cf88725f26d12c40efd06a20308 62.65MB / 62.65MB          [+] Building 179.1s (8/10)
```



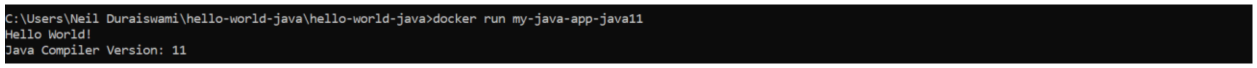
my-java-app-java17 67c8da50c09e	latest	In use	3 minutes ago	1.51 GB	▶	⋮
my-java-app-java11 c0cff727f76c	latest	In use	8 minutes ago	955.19 MB	▶	⋮
my-java-app-java8 e93ed4950be1	latest	In use	12 minutes ag	127.23 MB	▶	⋮

Task 3: Testing, Verification, and Documentation

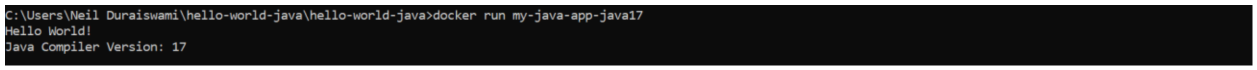
- Tested the application by running Docker containers for each Java version and verifying that the "Hello, World!" message was printed on the console.



```
C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker run my-java-app-java8
Hello World!
Java Compiler Version: 1.8
```

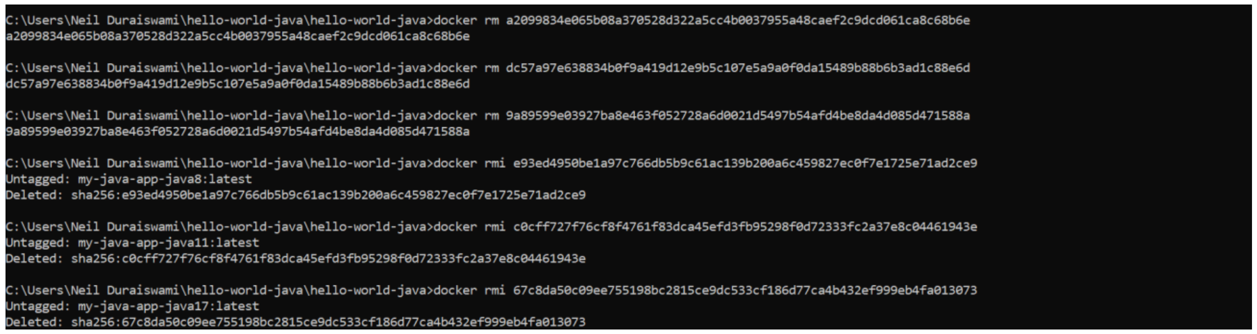


```
C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker run my-java-app-java11
Hello World!
Java Compiler Version: 11
```



```
C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker run my-java-app-java17
Hello World!
Java Compiler Version: 17
```

- Provided cleanup commands to stop and remove Docker containers and images after testing.



```
C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker rm a2099834e065b08a370528d322a5cc4b0037955a48caef2c9dcd061ca8c68b6e
a2099834e065b08a370528d322a5cc4b0037955a48caef2c9dcd061ca8c68b6e

C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker rm dc57a97e638834b0f9a419d12e9b5c107e5a9a0f0da15489b88b6b3ad1c88e6d
dc57a97e638834b0f9a419d12e9b5c107e5a9a0f0da15489b88b6b3ad1c88e6d

C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker rm 9a89599e03927ba8e463f052728a6d0021d5497b54afd4be8da4d085d471588a
9a89599e03927ba8e463f052728a6d0021d5497b54afd4be8da4d085d471588a

C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker rmi e93ed4950be1a97c766db5b9c61ac139b200a6c459827ec0f7e1725e71ad2ce9
Untagged: my-java-app-java8:latest
Deleted: sha256:e93ed4950be1a97c766db5b9c61ac139b200a6c459827ec0f7e1725e71ad2ce9

C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker rmi c0cff727f76cf8f4761f83dca45efd3fb95298f0d72333fc2a37e8c04461943e
Untagged: my-java-app-java11:latest
Deleted: sha256:c0cff727f76cf8f4761f83dca45efd3fb95298f0d72333fc2a37e8c04461943e

C:\Users\Neil Duraiswami\hello-world-java\hello-world-java>docker rmi 67c8da50c09ee755198bc2815ce9dc533cf186d77ca4b432ef999eb4fa013073
Untagged: my-java-app-java17:latest
Deleted: sha256:67c8da50c09ee755198bc2815ce9dc533cf186d77ca4b432ef999eb4fa013073
```

Conclusion

This project successfully demonstrated the process of building and running a simple Java application across different Java environments using Docker and Maven. The use of multi-stage Docker builds ensured efficient image sizes and dependencies. By leveraging Maven profiles, the project easily adapted to different Java versions, making it versatile and easily maintainable.