

EXPERIMENT : 9

Date of Performance	
Date of Submission	

AIM

To learn Dockerfile instructions, build an image for a sample web application using Dockerfile.

PROBLEM DEFINITION

Master Dockerfile instructions to build an image for a sample web application.

THEORY

Jenkins is an open-source automation tool created with Java. It is extensively used as a Continuous Integration (CI) and Continuous Deployment (CD) tool.

Maven:

- Maven primarily intends to provide developers with:
 1. A comprehensive, reusable, easily maintainable model for projects.
 2. Plugins or tools to interact with and operate within this model.
- Maven is a POM (Project Object Model)-based build automation and project management tool written in Java. However, it is compatible with projects written in C#, Python, Ruby, etc.

A few Maven features worth mentioning are:

1. Maven can be used to build any number of projects into predefined output types like .jar, .war, metadata, etc.
2. Maven can automatically download necessary files from the repository when building a project.

Selenium using Maven in Jenkins: Selenium is a widely used test automation framework for validating web applications across different combinations of browsers, platforms, and devices (or emulators). It is widely used for testing areas such as functional testing, end-to-end testing, and more.

Why Jenkins and Selenium?

- Running Selenium tests in Jenkins allows you to run your tests every time your software changes and deploy the software to new environments when the tests pass.

Advantages of using Maven and Jenkins with Selenium:

- Whenever a change is made in the implementation, the changes are deployed on the test environment. Automation testing is performed continuously, and developers are kept informed about the build and test stage results.
- Test suites that comprise many test scenarios might take longer duration testing. A nightly build run can be scheduled for build and execution on the Jenkins server in such cases.

Traditionally, the Dockerfile is called Dockerfile and located in the root of the context. You use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

```
$ docker build -f /path/to/a/Dockerfile .
```

You can specify a repository and tag at which to save the new image if the build succeeds:

```
$ docker build -t shykes/myapp .
```

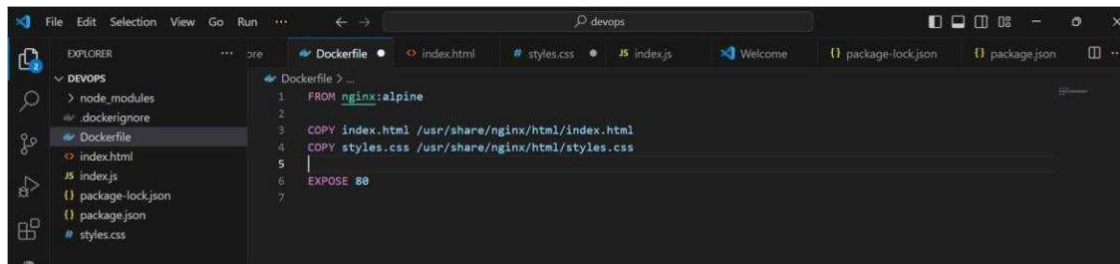
To tag the image into multiple repositories after the build, add multiple -t parameters when you run the build command:

```
$ docker build -t shykes/myapp:1.0.2 -t shykes/myapp:latest .
```

Before the Docker daemon runs the instructions in the Dockerfile, it performs a preliminary validation of the Dockerfile and returns an error if the syntax is incorrect:

```
$ docker build -t test/myapp . Syntax: syntax=[remote image reference]
```

OUTPUT

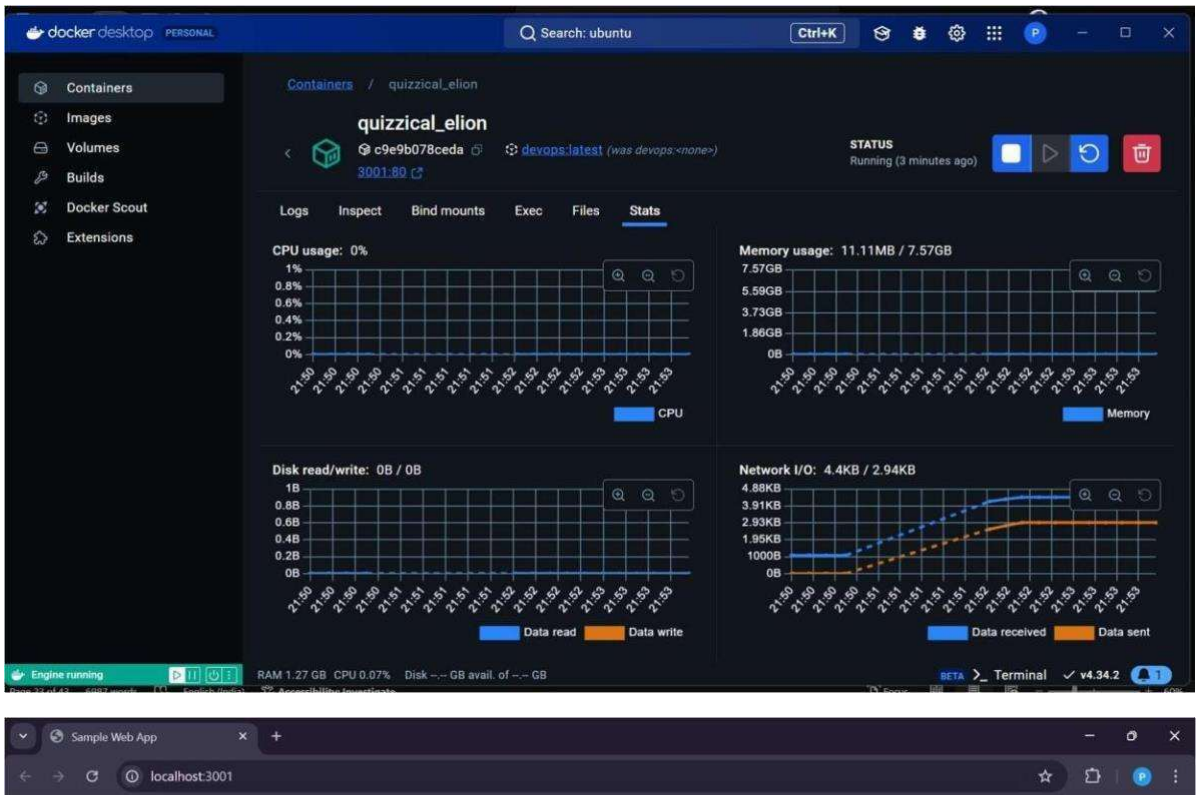


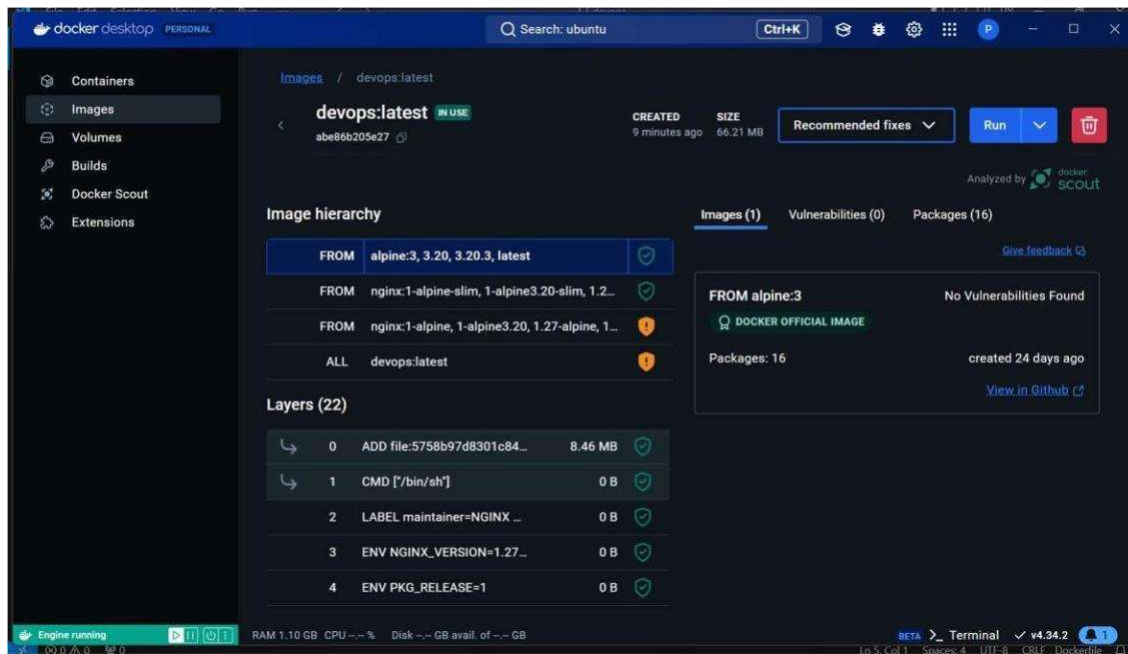
```
1 FROM nginx:alpine
2
3 COPY index.html /usr/share/nginx/html/index.html
4 COPY styles.css /usr/share/nginx/html/styles.css
5
6 EXPOSE 80
7
```

```
Windows PowerShell
PS C:\Users\Piyush\OneDrive\Desktop\devops> docker build -t devops .
[+] Building 6.9s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 334B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 69B
=> [1/3] FROM docker.io/library/nginx:alpine@sha256:a5127daff3d6f4606be3100a252419bfa84fd6ee5cd74d0feacala50668f97dcf
=> resolve docker.io/library/nginx:alpine@sha256:a5127daff3d6f4606be3100a252419bfa84fd6ee5cd74d0feacala50668f97dcf
=> sha256:0c02f601d0eed2923ae2087212c9c0753846732b22db5f2088ec0daf62387e12 13.19MB / 13.19MB
=> sha256:bb16f69e8876d046e20b50c0873ac84b46e7b60926bbcc72a32765ad981cc732 393B / 393B
=> sha256:c298c5a0cd21956f1dec93f16c6968b7b009b43f22add9e78d18273bb91661f5 1.40kB / 1.40kB
=> sha256:6f0b7faa0055e50ddac110c0d0b6286235e9bd9c0d4de0f0dc5860dd5833a6 1.21kB / 1.21kB
=> sha256:51076974aef5e1f3c046f2d40fa8e10d03a4c37e962e00f46bcfb5af242e81ad 956B / 956B
=> sha256:652d09a25e853a561388e4eaf55072df1747066277ef8310aff10d601150385 629B / 629B
=> sha256:5b19511a843df5d68c62b357426dd4e99e48fbeb9c085260de375065b969561f 1.75MB / 1.75MB
=> extracting sha256:5b19511a843df5d68c62b357426dd4e99e48fbeb9c085260de375065b969561f
=> extracting sha256:652d09a25e853a561388e4eaf55072df1747066277ef8310aff10d601150385
=> extracting sha256:51076974aef5e1f3c046f2d40fa8e10d03a4c37e962e00f46bcfb5af242e81ad
=> extracting sha256:bb16f69e8876d046e20b50c0873ac84b46e7b60926bbcc72a32765ad981cc732
=> extracting sha256:6f0b7faa0055e50ddac110c0d0b6286235e9bd9c0d4de0f0dc5860dd5833a6
=> extracting sha256:c298c5a0cd21956f1dec93f16c6968b7b009b43f22add9e78d18273bb91661f5
=> extracting sha256:0c02f601d0eed2923ae2087212c9c0753846732b22db5f2088ec0daf62387e12
=> [internal] load build context
=> => transferring context: 453B
=> [2/3] COPY index.html /usr/share/nginx/html/index.html
=> [3/3] COPY styles.css /usr/share/nginx/html/styles.css
=> exporting to image
=> exporting layers
=> exporting manifest sha256:da353bbb24208c2ba66554b54f648ad31c5e488ba8b5ec73d7b1f92ac22f9209
=> exporting config sha256:13cfb0fd87876195e92d4f0bae16177aecf838735a04c3248f0870810775885c0
=> exporting attestation manifest sha256:e7a93eb35c84aa3bb2e9db9bca696b0121eb442c3217b6af016b7091168080c
=> exporting manifest list sha256:abe86b205e2704e087412435b637ab0d51369e19df36458f518a900ea893805
=> naming to docker.io/library/devops:latest
=> unpacking to docker.io/library/devops:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/n7t5b7126rn2jxcyqukggty

What's next:
View a summary of image vulnerabilities and recommendations + docker scout quickview
PS C:\Users\Piyush\OneDrive\Desktop\devops> docker run -d -p 3000:80 devops
2c64a2547efb7720fab7d88aba80aebf48a3c1ed6d8a7ead0405c3ec0c1b77
docker: Error response from daemon: Ports are not available: exposing port TCP 0.0.0.0:3000 -> 0.0.0.0:0: listen tcp 0.0.0.0:3000: bind: Only one usage of each socket address (protocol/network address/port) is normally permitted.
PS C:\Users\Piyush\OneDrive\Desktop\devops> docker run -d -p 3001:80 devops
c9e9b078cedabf31dd1db4bec10ee8fd0602cd26eb71c361a0c98ce11dc42eb1
PS C:\Users\Piyush\OneDrive\Desktop\devops> |
```





CONCLUSION:

Consequently, learning Dockerfile instructions and building an image for a sample web application provided practical experience in containerization and application deployment.