Name: Shweta Singh
Sap Id: 500098159
Course & Batch: Btech CSE(DevOps)-B4

Submitted to: Dr Hitesh Kumar Sharma

EXPERIMENT- 3

AIM: Working with .dockerignore

Perform Docker Link and .dockerignore based scenario on Katacoda.

Working with .dockerignore

Step 1 - Docker Ignore

To prevent sensitive files or directories from being included by mistake in images, you can add a file named *.dockerignore*.

Example

A Dockerfile copies the working directory into the Docker Image. As a result, this would include potentially sensitive information such as a passwords file which we'd want to manage outside the image.

View the Dockerfile with `cat Dockerfile`

```
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$ cat Dockerfile
FROM httpd
WORKDIR /usr/local/apache2/htdocs/
COPY . .pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$
```

Build the image with `docker build -t password .`

```
COPY . .pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$ docker build -t password .
[+] Building 142.1s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 91B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/httpd:latest
 => [auth] library/httpd:pull token for registry-1.docker.io
 => [1/3] FROM docker.io/library/httpd@sha256:ed6db4a8c394d075c9c59a3dbd61a3818cd302d9948057f1e19046e5bffec027
 => => resolve docker.io/library/httpd@sha256:ed6db4a8c394d075c9c59a3dbd61a3818cd302d9948057f1e19046e5bffec027
 => => sha256:5f75f8a17f406c926d42ef7e5e0803a23b150c46f77cda9c83bff10db3d68f2f 31.40MB / 31.40MB
 => => sha256:ed6db4a8c394d075c9c59a3dbd61a3818cd302d9948057f1e19046e5bffec027 1.86kB / 1.86kB
 => => sha256:2f1ec45327a35711f293cf543c2e0efadfd44bc71e8081dcd76a558b99778005 1.37kB / 1.37kB
 => => sha256:75a48b16cd565cdaff0cfcbe3462c292e56108a22b9733c0a04a9dc1cbd7a774 9.09kB / 9.09kB
 => => sha256:c20157372e943d84bb5a0624e80395697de1f41ecd54b3bcead2b03bb6b13fe8 176B / 176B
 => => sha256:073cbcfef6634b5131786873a7a92a3b3bda43672e5830126dfa94352649358d 4.20MB / 4.20MB
 => => extracting sha256:c20157372e943d84bb5a0624e80395697de1f41ecd54b3bcead2b03bb6b13fe8
 => => sha256:39fc6f0c5be2c5b90e4d0881260800bc921cb4c1db44c7a55e0b638914f0842f 301B / 301B
 => => extracting sha256:073cbcfef6634b5131786873a7a92a3b3bda43672e5830126dfa94352649358d
 => => extracting sha256:5f75f8a17f406c926d42ef7e5e0803a23b150c46f77cda9c83bff10db3d68f2f
 => => extracting sha256:39fc6f0c5be2c5b90e4d0881260800bc921cb4c1db44c7a55e0b638914f0842f
 => [internal] load build context
 => => transferring context: 489B
 => [2/3] WORKDIR /usr/local/apache2/htdocs/
 => [3/3] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:aefcd6dbe2c6dd69887e9c73c9113e6f0d01730d1bb906e014c690a486748857
 => => naming to docker.io/library/password

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$
```

Look at the output using `docker run password ls /app`

```
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$ docker run password ls /usr/local/apache2/htdocs/
Dockerfile
aboutus.html
index.html
password.txt
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$
```

This will include the passwords file.

Ignore File

The following command would include passwords.txt in our *.dockerignore* file and ensure that it didn't accidentally end up in a container. The *.dockerignore* file would be stored in source control and share with the team to ensure that everyone is consistent.

```
echo passwords.txt >> .dockerignore
```

```
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$ echo password.txt >> .dockerignore
pavilion@shweta:~/Desktop/ACO/A
```

Docker Desktop

The ignore file supports directories and Regular expressions to define the restrictions, very similar to *.gitignore*. This file can also be used to improve build times which we'll investigate in the next step.

Build the image, because of the Docker Ignore file it shouldn't include the passwords file.

```
docker build -t nopassword .
```

```
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$ docker build -t nopassword .
[+] Building 4.8s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 91B
 => [internal] load .dockerignore
 => => transferring context: 53B
 => [internal] load metadata for docker.io/library/httpd:latest
 => [auth] library/httpd:pull token for registry-1.docker.io
 => [1/3] FROM docker.io/library/httpd@sha256:ed6db4a8c394d075c9c59a3dbd61a3818cd302d9948057f1e19046e5bffec027
 => [internal] load build context
 => => transferring context: 143B
 => CACHED [2/3] WORKDIR /usr/local/apache2/htdocs/
 => [3/3] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:fa58aca76800f7797a627e0154c478217db665b40cf34fb83607daee2fe1e387
 => => naming to docker.io/library/nopassword

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$
```

Look at the output using `docker run nopassword ls /app`

```
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$ docker run nopassword ls /usr/local/apache2/htdocs/
Dockerfile
aboutus.html
index.html
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$
```

Protip

If you need to use the passwords as part of a *RUN* command then you need to copy, execute and delete the files as part of a single RUN command. Only the final state of the Docker container is persisted inside the image.

Step 2 - Docker Build Context

The *.dockerignore* file can ensure that sensitive details are not included in a Docker Image. However they can also be used to improve the build time of images.

In the environment, a 100M temporary file has been created. This file is never used by the *Dockerfile*. When you execute a build command, Docker sends the entire path contents to the Engine for it to calculate which files to include. As a result sending the 100M file is unrequired and creates a slower build.

```
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$ dd if=/dev/zero of=tempfile bs=1M count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.0578615 s, 1.8 GB/s
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$
```

You can see the 100M impact by executing following the command.

```
docker build -t large-file-context .
```

```
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$ docker build -t large-file-context .
[+] Building 13.0s (9/9) FINISHED
 => [internal] load .dockerignore
 => => transferring context: 53B
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 91B
 => [internal] load metadata for docker.io/library/httpd:latest
 => [auth] library/httpd:pull token for registry-1.docker.io
 => [1/3] FROM docker.io/library/httpd@sha256:ed6db4a8c394d075c9c59a3dbd61a3818cd302d9948057f1e19046e5bffec027
 => [internal] load build context
 => => transferring context: 104.88MB
 => CACHED [2/3] WORKDIR /usr/local/apache2/htdocs/
 => [3/3] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:89b6588280d116ab3722f8c5701ad2e270629722d0add024f81926cf640a1f01
 => => naming to docker.io/library/large-file-context

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
pavilion@shweta:~/Desktop/ACO/ACO_LAB/my-web-app$
```

In the next step, we'll demonstrate how to improve the performance of the build.

Protip

It's wise to ignore *.git* directories along with dependencies that are downloaded/built within the image such as *node_modules*. These are never used by the application running within the Docker Container and just add overhead to the build process.

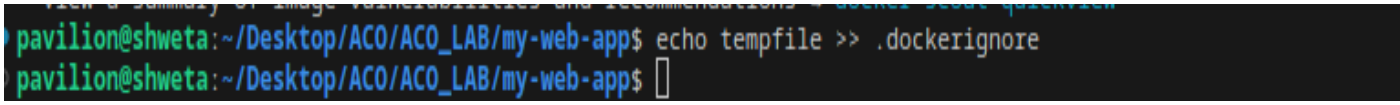
Step 3 - Optimised Build

In the same way, we used the *.dockerignore* file to exclude sensitive files, we can use it to exclude files which we don't want to be sent to the Docker Build Context during the build.

Optimizing

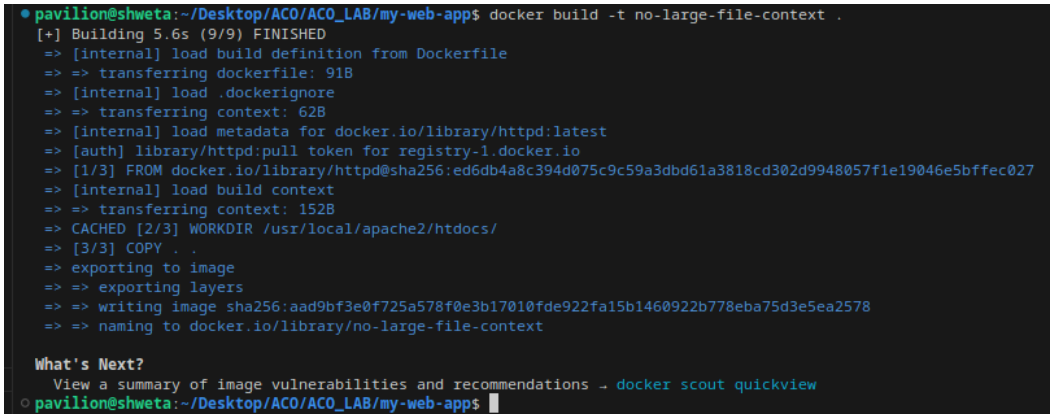
To speed up our build, simply include the filename of the large file in the ignore file.

```
echo big-temp-file.img >> .dockerignore
```



When we rebuild the image, it will be much faster as it doesn't have to copy the 100M file.

```
docker build -t no-large-file-context .
```



This optimisation has a greater impact when ignoring large directories such as *.git*