EXPERIMENT 10

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

```
anuvagarg@Anuvas-MacBook-Air ~ % cd docker
anuvagarg@Anuvas-MacBook-Air docker % mkdir lab10
anuvagarg@Anuvas-MacBook-Air docker % cd lab10
anuvagarg@Anuvas-MacBook-Air lab10 % vi Dockerfile
anuvagarg@Anuvas-MacBook-Air lab10 % vi password.txt
```

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

```
anuvagarg@Anuvas-MacBook-Air lab10 % cat dockerfile
FROM alpine
RUN mkdir /app
COPY password.txt /app
WORKDIR /app
```

Build the image with docker build -t password .

```
anuvagarg@Anuvas-MacBook-Air lab10 % docker build -t password:1.0 .
[+] Building 0.6s (9/9) FINISHED

 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 105B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/alpine:latest
 => CACHED [1/4] FROM docker.io/library/alpine
 => [internal] load build context
 => => transferring context: 51B
 => [2/4] RUN mkdir /app
 => [3/4] COPY password.txt /app
 => [4/4] WORKDIR /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:9a607eb9e4a853868d5c750e007be5263bb12c4c9065b0a62c6e355cd8e8a6f1
 => => naming to docker.io/library/password:1.0
```

Look at the output using docker run password ls /app

```
anuvagarg@Anuvas-MacBook-Air lab10 % docker run password:1.0 ls /app
password.txt
```

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 shared with the team to ensure that everyone is consistent.

echo passwords.txt >> .dockerignore

```
anuvagarg@Anuvas-MacBook-Air lab10 % docker run password:1.0 ls -a /app
.
..
password.txt
anuvagarg@Anuvas-MacBook-Air lab10 % echo password.txt >> .dockerignore
anuvagarg@Anuvas-MacBook-Air lab10 % vi .dockerignore
anuvagarg@Anuvas-MacBook-Air lab10 % docker run password:1.0 ls -a /app
.
..
password.txt
```

docker build -t nopassword .

```
anuvagarg@Anuvas-MacBook-Air lab10 %
docker build -t nopassword .
[+] Building 0.3s (9/9) FINISHED

 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 36B
 => [internal] load .dockerignore
 => => transferring context: 59B
 => [internal] load metadata for docker.io/library/alpine:latest
 => [1/4] FROM docker.io/library/alpine
 => [internal] load build context
 => => transferring context: 114B
 => CACHED [2/4] RUN mkdir /app
 => [3/4] COPY . /app
 => [4/4] WORKDIR /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:84b622b423b12c962e460de07e3ac799432ec9ef7ce5f91414af30cf1be98359
 => => naming to docker.io/library/nopassword

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
anuvagarg@Anuvas-MacBook-Air lab10 % docker run nopassword ls -a /app
.
..
.dockerignore
.pgpass
Dockerfile
```

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.

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

```
anuvagarg@Anuvas-MacBook-Air lab10 % vi .dockerignore
anuvagarg@Anuvas-MacBook-Air lab10 % docker run nopassword ls -a /app
.
..
.dockerignore
.pgpass
Dockerfile
anuvagarg@Anuvas-MacBook-Air lab10 % docker run password:1.1 ls -a /app
.
..
.pgpass
Dockerfile
password.txt
anuvagarg@Anuvas-MacBook-Air lab10 % docker run nopassword ls -a /app
.
..
.dockerignore
.pgpass
Dockerfile
anuvagarg@Anuvas-MacBook-Air lab10 % docker build -t nopassword:1.1 .
[+] Building 0.3s (9/9) FINISHED

 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 36B
 => [internal] load .dockerignore
 => => transferring context: 67B
 => [internal] load metadata for docker.io/library/alpine:latest
 => [1/4] FROM docker.io/library/alpine
 => [internal] load build context
 => => transferring context: 96B
 => CACHED [2/4] RUN mkdir /app
 => [3/4] COPY . /app
 => [4/4] WORKDIR /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:f04bf28002d4e43997ad9e60ec56cb36a1306e26ffee74df3dcb0928e86a3486
 => => naming to docker.io/library/nopassword:1.1

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
anuvagarg@Anuvas-MacBook-Air lab10 % docker run nopassword:1.1 ls -a /app
.
..
.dockerignore
Dockerfile
```

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.