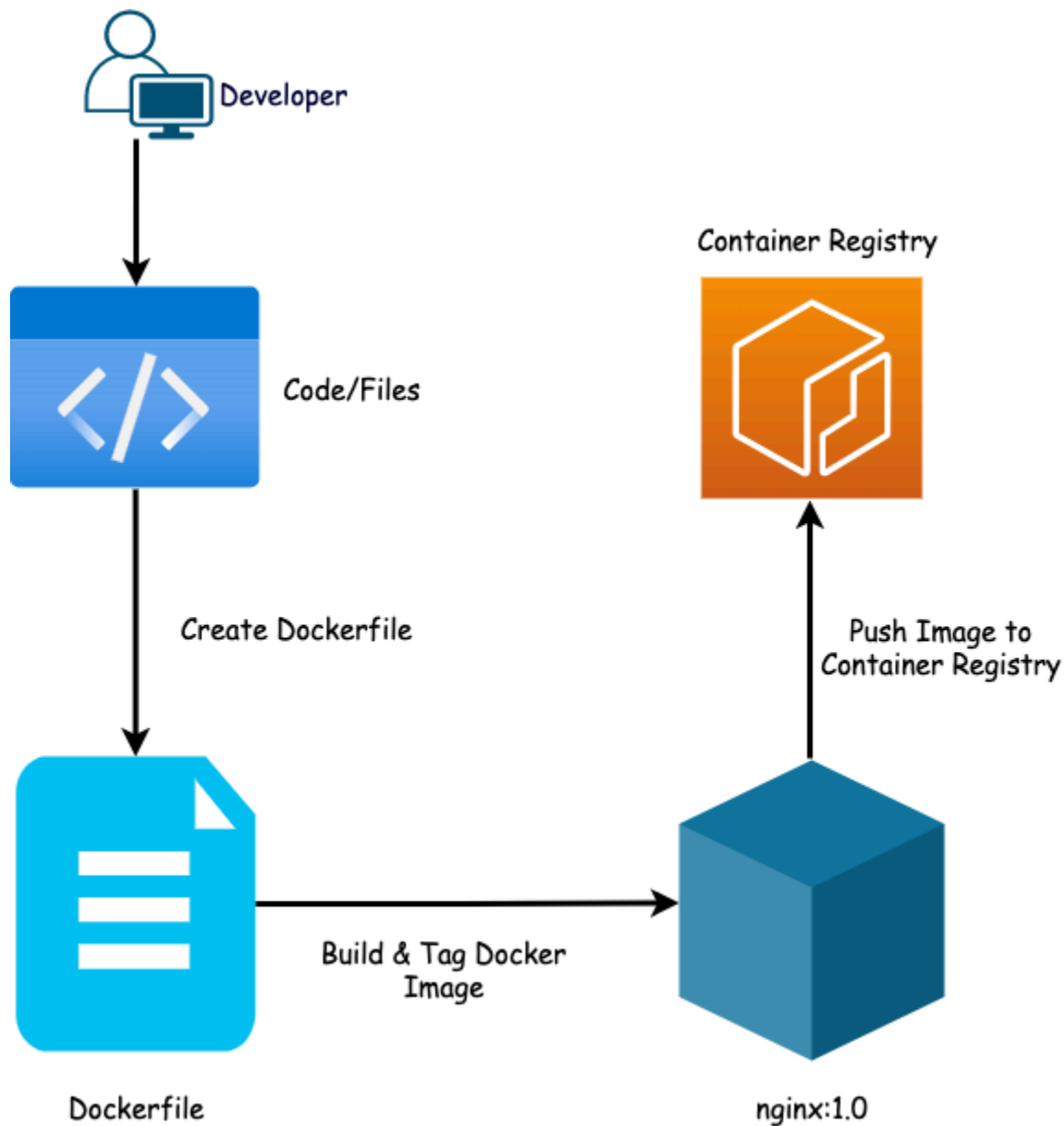


## AIM:

To create a Dockerfile for building a Docker image of a sample web application.

## Theory:

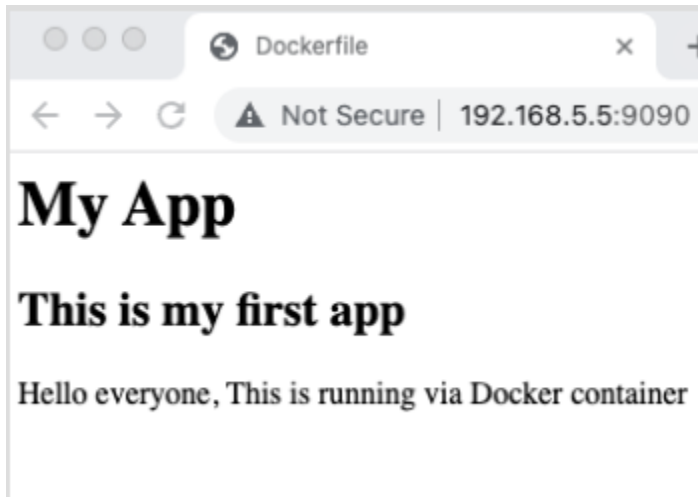
Dockerfile is a text file that contains instructions for building a Docker image. Each instruction in a Dockerfile represents a layer in the image. By creating a Dockerfile, we can define the environment and dependencies required to run an application inside a Docker container.



To build a Docker image for a sample web application, we typically follow these steps:

1. Specify a base image: Choose an existing Docker image that serves as the starting point for your application. This could be an official image from Docker Hub or a custom image.
2. Install dependencies: Use instructions like RUN to install any necessary software or packages required by your web application.
3. Copy application files: Use the COPY instruction to add your web application files into the Docker image.
4. Expose ports: Use the EXPOSE instruction to specify which ports your application listens on.
5. Define startup command: Use the CMD instruction to specify the command that should be executed when the container starts. This usually starts the web server or application.

```
vagrant@ubuntu-focal:~/docker-image-examples/nginx$ docker build -t nginx .  
Sending build context to Docker daemon 5.12kB  
Step 1/6 : FROM ubuntu:18.04  
--> 71cb16d32be4  
Step 2/6 : RUN apt-get -y update && apt-get -y install nginx  
--> Using cache  
--> 121ef9021ba8  
Step 3/6 : COPY files/default /etc/nginx/sites-available/default  
--> Using cache  
--> ca751c4cff9a  
Step 4/6 : COPY files/index.html /usr/share/nginx/html/index.html  
--> Using cache  
--> d653cd1df9da  
Step 5/6 : EXPOSE 80  
--> Using cache  
--> 6f11c34ffb5f  
Step 6/6 : CMD ["/usr/sbin/nginx", "-g", "daemon off;"]  
--> Using cache  
--> 45e69895763f  
Successfully built 45e69895763f  
Successfully tagged nginx:latest  
vagrant@ubuntu-focal:~/docker-image-examples/nginx$
```



```
Administrator: Windows Powe x + v
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API surface has changed to use Promises in 1.x.)
changed 10 packages in 351ms
C:\Users\15L\Desktop\welcome-to-docker>express docker-app

warning: the default view engine will not be jade in future releases
warning: use '--view=jade' or '--help' for additional options

destination is not empty, continue? [y/N] y

create : docker-app\
create : docker-app\public\
create : docker-app\public\javascripts\
create : docker-app\public\images\
create : docker-app\public\stylesheets\
create : docker-app\public\stylesheets\style.css
create : docker-app\routes\
create : docker-app\routes\index.js
create : docker-app\routes\users.js
create : docker-app\views\
create : docker-app\views\error.jade
create : docker-app\views\index.jade
create : docker-app\views\layout.jade
create : docker-app\app.js
create : docker-app\package.json
create : docker-app\bin\
create : docker-app\bin\www
```

```
Administrator: Windows Powe x + v - □ x
To address all issues (including breaking changes), run:
  npm audit fix --force

C:\Users\15L\Desktop\welcome-to-docker>npm start

> welcome-to-docker@0.1.0 start
> react-scripts start

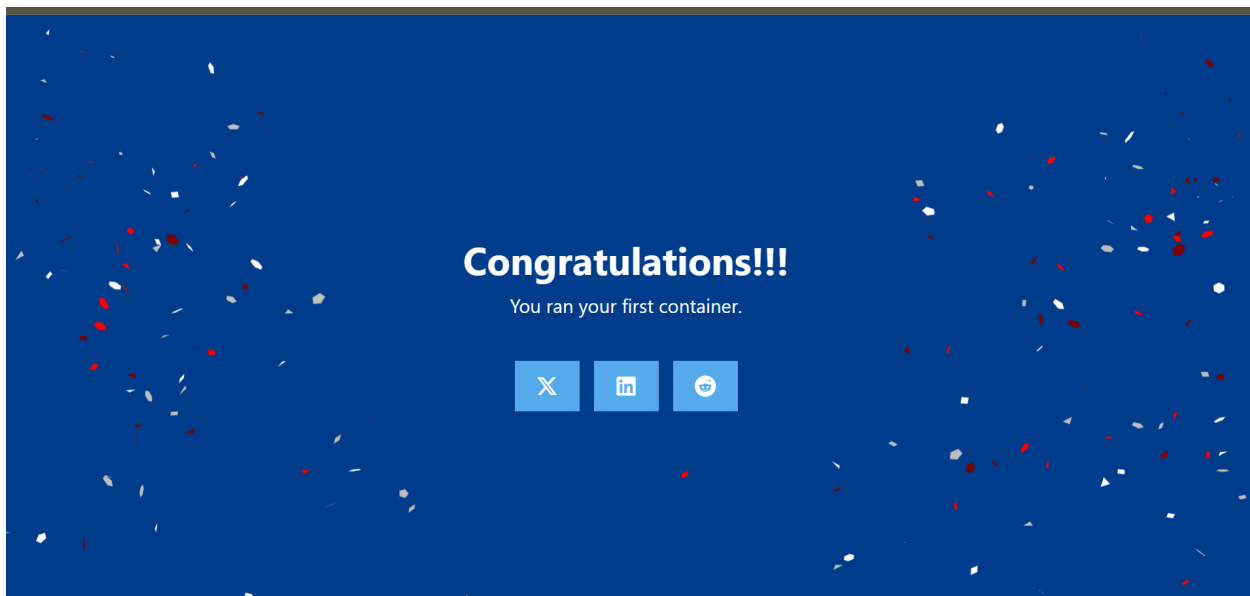
(node:10628) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is d
eprecated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:10628) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is
deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view welcome-to-docker in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.166:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



### Conclusion:

Creating a Dockerfile allows us to define the environment and dependencies required to run a web application in a Docker container. By following best practices and optimizing our Dockerfile, we can efficiently build Docker images that are portable, reproducible, and scalable. This enables us to deploy our applications consistently across different environments, making development and deployment processes more streamlined and reliable.