# Project Report: Docker File Creation and Project Deployment

Prepared by: Abdul Rahiman

Project Title: DLF

Deployment Platform: Render and Docker

Duration: 13/10/2024 to 16/10/2025

## 1. Introduction

This report outlines the work carried out in the creation of a Dockerfile and the deployment of a software project using Docker. The purpose of this work was to containerize the application for easier distribution, consistent environments, and simplified deployment across different systems.

## 2. Objective

The primary objective was to:  
- Build a Dockerfile to containerize the project.  
- Create a lightweight and portable environment for running the application.  
- Deploy the containerized application in a reliable and scalable manner.

## 3. Work Description

### Step 1: Dockerfile Creation

A Dockerfile was created to define all the necessary configurations and dependencies required to run the project.  
Key tasks performed:  
- Selected an appropriate base image (e.g., python:3.10-slim or node:18-alpine).  
- Added project source code to the Docker image.  
- Installed all required dependencies using requirements.txt or package.json.  
- Set up environment variables and working directories.  
- Defined the entry point or command to start the application.

Example Dockerfile:

FROM python:3.10-slim  
WORKDIR /app  
COPY . /app  
RUN pip install --no-cache-dir -r requirements.txt  
EXPOSE 8000  
CMD ["python", "app.py"]

### Step 2: Building the Docker Image

After creating the Dockerfile, the image was built using:  
docker build -t dlf:latest .  
This step packaged the application and its dependencies into a single container image.

### Step 3: Testing the Container Locally

The built image was tested to ensure the application ran correctly using:  
docker run -d -p 8000:8000 dlf:latest  
This verified that the application was successfully containerized and accessible via the assigned port.

### Step 4: Deployment

The containerized project was deployed on Render using Docker. The image was pushed to a container registry and then deployed on the Render platform. Docker ensured consistency, and Render provided scalable hosting for the application.

## 4. Results

- The project was successfully containerized and deployed.  
- Deployment time was reduced significantly since dependencies were prepackaged.  
- The containerized setup ensured consistent performance across environments.  
- The project could be easily scaled or redeployed without reconfiguration.

## 5. Conclusion

The Docker-based deployment streamlined the overall development and delivery process. The containerization approach improved consistency, simplified maintenance, and enhanced portability. Future improvements may include implementing CI/CD pipelines and orchestration using Kubernetes for large-scale deployments.

## 6. Tools and Technologies Used

- Docker – Containerization platform  
- Render – Deployment platform  
- Python / Node.js – Application runtime environment  
- Git – Version control system