# Assignment 1: Containerizing an Application with Docker

**Name: Jithi Jyothi**

**Github link:** [**https://github.com/JithinJyothi95/Assignment1-CNO**](https://github.com/JithinJyothi95/Assignment1-CNO)

For this assignment1 , I decided to containerize a basic Python Flask application that I created myself. The goal was to make sure the app runs consistently inside a Docker container while meeting all the requirements like using a lightweight image, setting up a non-root user, and making the container portable using volumes.

## Step 1: Selecting and Setting Up the Application I created a simple Flask app called 'app.py' that returns a welcome message. I additionally wrote a'requirements.txt' file to identify dependencies ('flask' and 'gunicorn') and a 'boot.sh' script to launch the app using Gunicorn, which is preferable for production.

## Step 2: Writing the Dockerfile

I used the official `python:3.11-slim` image because it's secure and lightweight, as required. I followed best practices by:

* Installing only what's needed
* Copying the code using `COPY`
* Setting a non-root user (`flaskuser`)
* Adding environment variables
* Exposing port 5000
* Using a clear `CMD` to run the app via `boot.sh`

## Step 3: Building the Image

I built the image using:

**docker build -t flask-app:v1 .**

The version tag `**v1**` was used instead of `**latest**` to follow the guidelines.

## Step 4: Running the Container with Volume

To make it portable and avoid bind mounts, I used a Docker volume:

**docker run -d -p 5000:5000 --name my-flask-app -v flask-data:/data flask-app:v1**

This volume helps keep runtime data persistent and ensures the app works in Codespaces or on another machine.

**Step 5: Resolving Issues**I got an error about 'gunicorn' not being found, so I added it to'requirements.txt' and rebuilt the image.

Everything now works correctly, and I’ve included all the required screenshots in the submission.