Assignment 1

*Container and Orchestration*

*INFO8995 - Spring 2025 - Section 1*

# Dockerizing a Python Flask App

Nidhun Murali - 8981611

**Steps**

- I built a simple Python Flask web app (app.py) and specified its dependency (Flask 3.0.3) in a requirements.txt file.
- To containerize it, I created a Dockerfile using the lightweight and secure python:3.11.8-alpine image, deliberately avoiding the :latest tag for consistency.
- I set up environment variables for Python, created a non-root user (nidhun) for better security, and designated /app as the working directory.
- To optimize Docker's caching, I copied the application files and dependencies in separate layers.
- After setting the non-root user (nidhun) as the owner of the files and switching to it, I exposed port 8080 and defined a volume at /app/data for persistent storage.
- The container runs the app using the command **[python app.py]**

**Volumes/bind mounts**

For persistent data, I used the VOLUME keyword in the Dockerfile to create a Docker volume. When running the container, I mounted the local **./data** directory to **/app/data** with the **-v $(pwd)/data:/app/data** flag. This ensures that data persists even if the container is deleted or recreated.

**Challenges and resolutions:**

One small challenge I faced was trying to remove the Dockerfile without stopping the container, which caused an issue. I learned that I either need to use the **-f** (force) flag or stop the container before deleting the file.