CSC 410/510 — Cloud Computing
Assignment: Containerized FastAPI Service with Reverse Proxy

Richard Kelley

February 13, 2026

Overview

In this assignment you will design, containerize, and deploy a small web service using Docker and Docker Compose. You will:

- Write a Dockerfile for a FastAPI application
- Configure persistent storage using Docker volumes
- Run Nginx as a reverse proxy in a separate container
- Orchestrate services using Docker Compose
- Deploy the system to a free-tier EC2 instance

Only the reverse proxy may be publicly exposed. The application container must remain internal.

Learning Objectives

By completing this assignment, you will be able to:

1. Author a production-oriented Dockerfile.
2. Use named Docker volumes for persistent data.
3. Configure inter-container networking via Docker Compose.
4. Deploy a multi-container system to EC2.

Starter Repository

Clone the provided repository:

```
git clone <REPO_URL>
```

Repository structure:

```
fastapi-docker-assignment/
|-- app/
|   |-- main.py
|   `-- requirements.txt
|
|-- nginx/
|   `-- nginx.conf
|
|-- docker-compose.yml
|-- .dockerignore
`-- README.md
```

You must complete:

- `app/Dockerfile`
- `docker-compose.yml`

Functional Requirements

1. FastAPI Service

Your container must:

- Listen on port 8000 internally
- Provide:
  - `GET /api/health`
  - `POST /api/items`
  - `GET /api/items`
- Persist data using SQLite or file storage
- Store persistent data in `/data`
- Run as a non-root user
- Use `uvicorn` without `--reload`

2. Persistent Storage

- Use a named Docker volume
- Mount the volume to `/data`
- Data must survive `docker compose down` followed by `up`

3. Reverse Proxy (Nginx)

- Run in a separate container
- Expose only port 80 to the host
- Proxy `/api/` to the FastAPI container using service-name DNS
- Do not expose port 8000 publicly

4. Docker Compose

- Define at least two services: `api` and `nginx`
- Define at least one named volume
- Ensure Nginx depends on the API
- Entire system must start with:

```
docker compose up -d --build
```

EC2 Deployment

Deploy your system to a free-tier EC2 instance.

Requirements:

- Install Docker and Docker Compose plugin
- Clone the provided repository on the instance
- Run:

```
docker compose up -d --build
```

- Configure Security Group to allow inbound port 80 only
- Demonstrate public access:

```
http://<EC2_PUBLIC_IP>/api/health
```

Testing Commands

Health Check

```
curl -i http://localhost/api/health
```

Add Item

```
curl -i -X POST http://localhost/api/items \
  -H "Content-Type: application/json" \
  -d '{"name":"alpha"}'
```

Retrieve Items

```
curl -i http://localhost/api/items
```

Persistence Test

```
docker compose down
docker compose up -d
curl http://localhost/api/items
```

Data must remain.

Live Demonstration (No Written Submission)

When you are ready, request a live checkoff.

You must demonstrate:

1. Both containers running
2. Successful GET and POST requests
3. Data persistence after restart
4. Port 8000 not publicly accessible

Note on Realism

This assignment is designed to model a realistic production architecture pattern:

- Application container
- Reverse proxy container
- Persistent storage
- Cloud deployment