Cloud Computing for Data Analysis
ITCS 6190/8190, Spring 2026
Marco Vieira

Hands-On L3: Installing Docker & Building a Multi-Container Microservice:
This Hands-On will help us install and set up Docker Desktop, deploy a Python Flask web application with a Redis cache, and run a PostgreSQL database. You'll practice container networking, multi-container orchestration with Docker Compose, and submit your results via GitHub.

What You'll Do

- Install and verify Docker Desktop.
- Run a PostgreSQL container.
- Set up Flask app and Redis cache files.
- Build and connect services with Docker Compose.
- Test in browser and capture screenshots of output as well as Docker Desktop showing the containers.
- Push project to GitHub.
- Create an Issue to errors faced during handson (**Note: There are some errors that are introduced intentionally in the code. You need to log these errors under Issues tab in GitHub**)

1. Install Docker Desktop

Windows

1. Go to Docker Desktop for Windows.
2. Download Docker Desktop Installer.exe.
3. Run it with default settings.
4. Start Docker Desktop from the Start Menu.
5. If prompted, enable **WSL 2** and restart.

macOS

1. Go to Docker Desktop for Mac.
2. Download the installer for your chip type (**Apple Silicon** or **Intel**).
3. Open the .dmg file and drag **Docker.app** into **Applications**.
4. Open **Docker.app** and follow security prompts.

Verify Installation

Open **Command Prompt** (Windows) or **Terminal** (Mac) and run:

```
docker -v
```

Steps:

1. **PostgreSQL Setup with Docker**
   a. Pull the PostgreSQL image:

   ```
   docker pull postgres
   ```

   b. Create and start a PostgreSQL instance:

   ```
   docker run -d -p 5432:5432 --name postgres1 -e POSTGRES_PASSWORD=pass12345 postgres
   ```

   c. Open a terminal to the container

   ```
   docker exec -it postgres1 bash
   ```

   d. Interact with PostgreSQL using `psql`:

   ```
   psql -d postgres -U postgres
   ```

2. **Build a Python Web App with Docker Compose in code environments (eg: Vs Code)**
   a. Define application dependencies in `requirements.txt`:

   ```
   flask

   redis
   ```

   b. Define application in app.py:

   ```
   import time
   import redis from flask
   import Flask app = Flask(__name__)
   cache = redis.Redis(host='redis', port=6379)
   def get_hit_count():
   retries = 5
   while True:
   try:
   return cache.incr('hits')
   except redis.exceptions.ConnectionError as  exc:
   if retries == 0:
   raise exc
   retries -= 1
   time.sleep(0.5)

   @app.route('/')
   def hello():
   count = get_hit_count()
   return 'Hello World! I have been seen {}
   times.\n'.format(count)
   ```

   c. Create a Dockerfile to containerize the application. Use the following template:

   ```
   FROM python:3.7-alpine
   WORKDIR /code
   ENV FLASK_APP=app.py
   ENV FLASK_RUN_HOST=0.0.0.0
   RUN apk add --no-cache gcc musl-dev linux-headers
   COPY requirements.txt requirements.txt
   RUN pip install -r requirements.txt
   EXPOSE 5000
   COPY . .
   CMD ["flask", "run"]
   ```

d. Define services in a `compose.yaml` file:

```
version: "3.9"
services:
  web:
    build: .
    ports:
      - "8000:5000"
    depends_on:
      - redis
  redis:
    image:
"redis:alpine"
```

e. Build and run the application:

```
docker compose up
```

f. Open the application in a browser at: `http://localhost:8000`

Expected submission format: One page report with screenshots from docker desktop app showing the containers log and the GitHub repo link in the document.

Submission:

1. Upload all files into Github.(app.py, Dockerfile, compose.yaml, requirement.txt, Readme.md, Document with screenshots.)
2. Readme.md : This file should contain the execution steps (**note: ensure to Quote the code. This can be found in further readings below.**) and details as to what you have learnt in this handson with appropriate formatting.
3. Log the errors faced in handson (**You will face errors**) under the Issues tab in GitHub.
4. Upload screenshots of the following in Readme file:
    a. Docker desktop app showing the containers.
    b. Output obtained after the handson.
5. Submit the link of the GitHub repository in canvas.

Further Readings:

Docker CLI Cheat Sheet: https://docs.docker.com/get-started/docker_cheatsheet.pdf

Basic writing and formatting syntax for Readme: https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax