OS Lab 6: Multi-Container Application Deployment with Docker

Course: Operating Systems

---

1. Objective

The objective of this lab is to understand the principles of OS-level virtualization by containerizing a 3-tier web application. You will gain hands-on experience with:

- **Process Isolation:** Managing application dependencies within containers.
- **Networking:** Enabling communication between isolated containers.
- **Volume Management:** Persisting data and mapping host files into containers.
- **Orchestration:** (Bonus) Managing multi-container environments using Docker Compose.

2. The Scenario

You are provided with a simple full-stack system consisting of three components:

1. **Database:** A MySQL initialization script (init.sql).
2. **Backend:** A Node.js API (server.js) that connects to the database and exports data to CSV.
3. **Frontend:** A React application (App.jsx) for managing users.

Your task is to "Dockerize" this application so it can run on any machine without installing Node.js or MySQL locally.

---

3. Required Tasks

Task 1: Database Containerization

- Pull the official mysql:8.0 image from Docker Hub.
- Run the container in a way that automatically executes the provided init.sql script upon startup to create the user's table.
  - *Hint: Look up the /docker-entrypoint-initdb.d/ directory in the official MySQL Docker documentation.*
- **Deliverable:** A running MySQL container with the schema loaded.

Task 2: Backend Containerization

- Create a Dockerfile for the Node.js application found in the backend/ folder.
- The application listens on port 3000. Ensure this port is exposed.
- **Challenge:** The backend code uses Environment Variables to find the database (DB_HOST, DB_USER, etc.). You must pass these variables when running the container.
- **Deliverable:** A running Node container that successfully connects to the MySQL container.

Task 3: Frontend Containerization

- Create a Dockerfile for the React application found in the frontend/ folder.
- For this lab, you may run the application using the development server (npm run dev).
- **Important:** Ensure the development server binds to 0.0.0.0 (host) and not just localhost, or it will not be accessible from your browser.
- **Deliverable:** A running Frontend container accessible via your browser (e.g., http://localhost:5173).

Task 4: Docker Registry

- Tag all three images following the convention: <your-dockerhub-username>/<image-name>.
- Login to Docker Hub using the CLI.
- Push these images to your public Docker Hub repository.

Bonus: Docker Compose & Volumes

Create a docker-compose.yml file that orchestrates the entire stack with a single command (docker compose up).

1. **Networking:** Define a dedicated network for the services.
2. **Persistence:** Mount Volume for the Database so data survives if the container is deleted.
3. **Host Access (Bind Mount):** Map a folder on your local host machine (e.g... /downloads) to the backend container's export directory.
    - *Goal:* When you click "Save to CSV" in the app, the file should appear on your actual laptop, not just inside the container.

---

4. Provided Source Code

Copy the code below into files on your local machine to begin the lab.

A. Database (db/init.sql)

SQL

```sql
CREATE DATABASE IF NOT EXISTS user_db;
USE user_db;

CREATE TABLE IF NOT EXISTS users (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(255) NOT NULL UNIQUE,
  firstname VARCHAR(100),
  lastname VARCHAR(100),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO users (email, firstname, lastname) VALUES ('admin@oslab.com', 'System', 'Admin');
```

B. Backend (backend/server.js)

Requires npm install express mysql2 cors body-parser csv-writer

JavaScript

```javascript
const express = require('express');
const mysql = require('mysql2');
const cors = require('cors');
const bodyParser = require('body-parser');
const createCsvWriter = require('csv-writer').createObjectCsvWriter;
const path = require('path');

const app = express();
app.use(cors());
app.use(bodyParser.json());

// Environment Variables for Database Connection
const dbConfig = {
  host: process.env.DB_HOST || 'localhost',
  user: process.env.DB_USER || 'root',
  password: process.env.DB_PASSWORD || 'root',
  database: process.env.DB_NAME || 'user_db',
  port: 3306
};

const db = mysql.createPool(dbConfig);

app.post('/users', (req, res) => {
  const { email, firstname, lastname } = req.body;
  const query = 'INSERT INTO users (email, firstname, lastname) VALUES (?, ?, ?)';
  db.query(query, [email, firstname, lastname], (err, result) => {
    if (err) return res.status(500).json({ error: err.message });
    res.status(201).json({ message: 'User created', id: result.insertId });
  });
});

app.get('/users', (req, res) => {
  db.query('SELECT * FROM users', (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results);
  });
});

app.get('/save-csv', (req, res) => {
  // Ensure this path exists in your container or is mounted to a volume
  const filePath = path.join(__dirname, 'data', 'users_dump.csv');

  db.query('SELECT * FROM users', (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    const csvWriter = createCsvWriter({
      path: filePath,
      header: [
        {id: 'user_id', title: 'ID'},
        {id: 'email', title: 'EMAIL'},
        {id: 'firstname', title: 'FIRSTNAME'},
        {id: 'lastname', title: 'LASTNAME'}
      ]
    });
    csvWriter.writeRecords(results)
      .then(() => res.json({ message: `File saved to ${filePath}` }))
      .catch(err => res.status(500).json({ error: err }));
  });
});

app.listen(3000, () => console.log(`Backend running on port 3000`));
```

C. Frontend (frontend/src/App.jsx)

Initialize using npm create vite@latest frontend -- --template react

JavaScript

```javascript
import { useState, useEffect } from 'react'
import './App.css'

const API_URL = "http://localhost:3000";

function App() {
  const [users, setUsers] = useState([]);
  const [form, setForm] = useState({ email: '', firstname: '', lastname: '' });
  const [status, setStatus] = useState('');

  const fetchUsers = async () => {
    try {
      const res = await fetch(`${API_URL}/users`);
      setUsers(await res.json());
    } catch (err) { console.error(err); }
  };

  useEffect(() => { fetchUsers(); }, []);

  const handleSubmit = async (e) => {
    e.preventDefault();
    await fetch(`${API_URL}/users`, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(form)
    });
    setForm({ email: '', firstname: '', lastname: '' });
    fetchUsers();
  };
```

```
 const handleExport = async () => {
   const res = await fetch(`${API_URL}/save-csv`);
   const data = await res.json();
   setStatus(data.message);
 };

 return (
   <div style={{ padding: '2rem' }}>
     <h1>OS Lab Docker Manager</h1>
     <form onSubmit={handleSubmit}>
       <input placeholder="Email" value={form.email} onChange={e => setForm({...form, email: e.target.value})}
/>
       <input placeholder="First Name" value={form.firstname} onChange={e => setForm({...form, firstname:
e.target.value})} />
       <input placeholder="Last Name" value={form.lastname} onChange={e => setForm({...form, lastname:
e.target.value})} />
       <button type="submit">Add User</button>
     </form>
     <button onClick={handleExport}>Save Users to CSV</button>
     {status && <p>{status}</p>}
     <ul>
       {users.map(u => <li key={u.user_id}>{u.email} - {u.firstname} {u.lastname}</li>)}
     </ul>
   </div>
 )
}
export default App
```

---

5. Deliverables & Submission Guidelines

Deliverables

You are required to submit your source code and configuration files.

1. **Part 1 (Mandatory):** Complete Dockerfiles for both Backend, Frontend and Database beside the copied code.
2. **Part 2 (Bonus):** The docker-compose.yml file connecting all services with volumes and networks.
3. **Report:** A file named REPORT.md containing the links to your pushed images on Docker Hub.

Required File Structure:
Your submission must be filed exactly according to the convention below:

- Backend dir (Contain the copied code and the Dockerfile)
- Frontend dir (Contain the copied code and Dockerfile)
- Database dir (Contain the SQL code and Dockerfile)
- docker-compose.yml (if applicable)
- REPORT.md

Submission

- **GitHub Link:** https://classroom.github.com/a/VK9aVgIY
- **Deadline:** Friday 12th December, 11:59 PM

Notes

- **Individual Work:** Students must work individually. You may talk together on the logic or commands being used but are **NOT** allowed to look at anybody's code.
- **Academic Integrity:** Revise the academic integrity note found on the class web page. Plagiarism in Dockerfiles or Compose configurations will be detected.