Programming Assignment 2 – Report

Group Members: Sayojya Patil, Neeraj Saini

GitHub Repository: https://github.com/SayojyaPatil/AdvOS-Programming-Assignment-2

Docker Hub: https://hub.docker.com/r/sayojya2000/vending-machine-microservices

Major Steps Performed

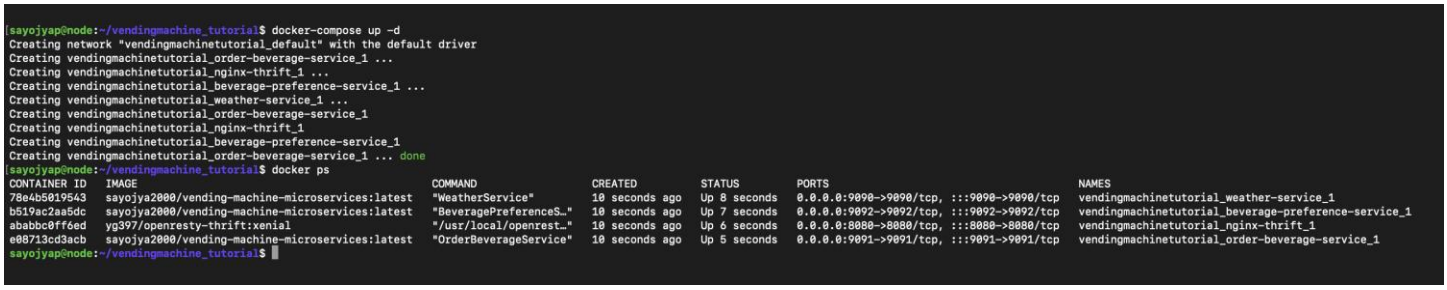
- Project Setup and Baseline Testing:
  Cloned the base vending machine microservices project. Used docker-compose to build and run all services. Verified the original functionality using the generate_request.sh script.

- Weather Service Enhancement:
  Replaced random weather selection with logic based on city ID:
  - Odd ID → WARM
  - Even ID → COLD
  Updated the WeatherServiceHandler accordingly.

- Development of Beverage Preference Service:
  Created a new microservice: beverage-preference-service.
  Implemented getBeverage (BeverageType btype) to randomly return:
  - For Hot: cappuccino, latte, or espresso
  - For Cold: lemonade, iced tea, or soda

- Order Beverage Service Integration:
  Updated placeOrder(city) to:
  - Call getWeather () to determine beverage type
  - Use getBeverage () to select the appropriate drink
  - Return the drink name to the NGINX reverse proxy.

- Thrift and Build System Modifications:
  Extended vending_machine. thrift with new service definitions. Regenerated code using Thrift compiler. Updated CMakeLists.txt and rebuilt containers.
- Final Testing and Results:
  Ran the generate_request.sh script and saved output to output.txt. Captured a screenshot showing all four services running with docker ps -a. Uploaded the working code to GitHub and pushed the Docker image to Docker Hub.



Reflection

Working on this assignment gave us a comprehensive understanding of how microservices operate in a containerized environment. By modifying and extending a pre-existing microservices-based vending machine system, we experienced the full development lifecycle—from setting up services, writing Thrift definitions, integrating components and deploying them using Docker.

One of the key learnings was how to leverage Thrift for defining service interfaces and generating language-specific bindings, allowing seamless communication between services. This exposed us to real-world practices of service definition and serialization protocols. Additionally, we gained hands-on experience with Docker and Docker Compose for managing multiple microservices, understanding the importance of isolated containers and declarative service orchestration.

Implementing the beverage-preference service and integrating it into the existing workflow highlighted the challenges and best practices of inter-service communication, especially handling RPC chaining and ensuring data flow integrity. We also learned how to troubleshoot issues in multi-container setups using logs and container statuses.

Overall, this assignment significantly improved our collaboration, system design thinking, and debugging skills in complex environments. It simulated the kinds of problems engineers face when building scalable, modular applications in production systems.

---------- END OF DOCUMENT----------