Babatunde Ali-Brown
August 23, 2024
CS 470 Final Reflection
https://youtu.be/-5AR1uEos7I

Experiences and Strengths

In this course, I learned how to apply cloud-based development principles and best practices in application development, migrating a full-stack application by refactoring to the cloud using Amazon S3, AWS Lambdas, DynamoDB, and Amazon API Gateway. I also learned about containerizing a MEAN full-stack application (Angular frontend, Node JS REST API backend, MongoDB database) using Docker and orchestrating the three containers using Docker Compose.

My strengths as a developer include coding to requirement specifications, practicing secure coding and code commenting, and conducting satisfactory testing. I also like to conduct rigorous research on coding bottlenecks to learn different ways to avoid similar problems in the future.

The level of skills I have acquired throughout my undergraduate degree program culminated by the MEAN full stack development in CS465 and cloud development in this course adequately prepares me for a productive role in full stack development, frontend engineering, system design and analysis, cloud development, and AWS cloud solutions architecting.

Planning for Growth

To produce efficiencies of management and scale in a full stack web application it can either be broken down into microservices or refactored to use serverless. Microservices and serverless can also be combined in a hybrid approach to enjoy the benefit of both methods.

Microservices architecture allows each microservice to be:

- independently developed and deployed,
- independently scaled up and down horizontally according to its own resource needs thereby facilitating resource optimization,
- isolated without affecting other microservices, thereby allowing the application, as a whole, to be more fault-tolerant,
- built with an appropriate technology stack and containerized to operate on diverse infrastructures.

Serverless allows developers to focus on code only without worrying about the management of servers needed to run backend services. Serverless allows the application to scale automatically, reduce its operational overhead and prevent over-provisioning, encourage cost efficiency by operating on a pay-as-you-go model, facilitate rapid development and iteration, and allow for event-driven architecture to trigger serverless functions only when they are needed.

Predicting the cost associated with using either microservices or serverless involves analyzing the costs of development, operation, and infrastructure (including storage, networking, scaling, and computing). Cost estimation tools provided by cloud providers may also be used to predict costs based on expected usage patterns, memory, and compute requirements. Other cost prediction strategy includes prototyping to a benchmarked usage and cost and using historical data on resource consumption before migration.

Microservices architecture has a steadier infrastructure cost than serverless which fluctuates based on usage, but has higher development and operational costs. Unexpected traffic spikes can cause unexpected costs in both serverless and microservices, and poorly architectured microservices can lead to resource inefficiencies, increasing costs.

The pros of using microservices include fine-grained scalability, polyglot programming and decoupled deployment, team specialization and clear ownership, and tailored resource optimization to individual container needs. The cons of using microservices include operational complexity from services orchestration, high upfront development cost and networking cost, managing data consistency for processes spanning multiple services, latency from distributed data access and networking, and complexity of managing security for many services.

The pros of serverless include cost-efficiency, focus on code and not infrastructure, autoscaling, faster time to market from rapid development and simplified DevOps, and easy global availability. The cons of serverless include limited portability and vendor lock-in, performance latency from cold-starts, imposed execution and memory limits, debugging event-driven challenges, and unpredictable cost spikes from unexpected high-volume traffic.

Deciding factors in the plan for expansion using either microservices or serverless include scalability needs (consistent or fluctuating), development and operational scope and complexity (larger or smaller organization), cost constraints (upfront costs or pay-as-you-go), needed time to market, application complexity (simple or modular).