CS 470 Final Reflection

Nathalie Morales

April 2025

Assignment: CS 470 Final Reflection

YouTube Presentation Link: https://www.youtube.com/watch?v=chNmXD6Rt3I

Experiences and Strengths

This course has been instrumental in preparing me for my professional goals in software development, especially in the growing field of cloud-based architecture. Through hands-on development of a full stack serverless application on AWS, I gained valuable skills in containerization, microservices, cloud storage, serverless API creation, and security configuration using IAM roles and policies. These skills are directly transferable to real-world cloud projects and are highly relevant to many modern development roles.

I developed and mastered skills including Docker and Docker Compose, configuring Lambda functions, setting up API Gateway routes, designing DynamoDB data models, and deploying frontends using Amazon S3. I also practiced infrastructure management concepts such as scalability, elasticity, and secure deployment in the cloud. By presenting my work in a conference-style format, I further enhanced my ability to clearly communicate technical concepts to a diverse audience.

As a software developer, I consider my strengths to be adaptability, thoroughness, and clarity. I take pride in building secure and maintainable systems that scale well. My ability to explain my design decisions also adds value in collaborative environments. Thanks to CS 470, I am now confident stepping into roles such as:

- Cloud Application Developer
- Full Stack Developer with a focus on AWS
- Backend Developer using serverless infrastructure
- QA Engineer for testing and optimizing distributed services

Planning for Growth

Through CS 470, I gained a deep understanding of how cloud services allow for flexible, efficient, and scalable application development. To plan for future growth, I would build on this knowledge by expanding the web application's architecture using serverless and microservices techniques.

Microservices and Serverless Efficiency: By decomposing the application into smaller, independent services (e.g., separate Lambda functions for question handling, answer management, and user interactions), I can manage them independently and deploy updates without affecting other services. Serverless functions can automatically scale to meet growing demand, eliminating the need to manually provision infrastructure.

Handling Scale and Errors: I would implement AWS Step Functions or Amazon EventBridge to orchestrate workflows and handle retries or fallbacks for failed executions. Monitoring and alerts would be set up via Amazon CloudWatch to track performance and catch errors early.

Cost Prediction: To forecast cost, I would use AWS's pricing calculators and CloudWatch logs to track function invocations and storage usage over time. Serverless is typically more cost-effective and predictable for spiky or low-to-medium traffic apps, whereas containers may offer better cost control for high-throughput systems with consistent traffic.

Containers vs. Serverless:

- **Serverless Pros:** Pay only when in use, automatic scaling, low maintenance

- **Serverless Cons:** Cold start latency, execution timeout limits
- **Containers Pros:** More control, persistent environments, better for long-running processes
- **Containers Cons:** Requires orchestration and infrastructure oversight

Elasticity and Pay-for-Service: Elasticity is critical in designing cloud systems that can handle traffic bursts and user growth. AWS services like Lambda, S3, and DynamoDB scale automatically and help reduce downtime risks. Pay-for-service ensures we only pay for what we use, making it easy to start small and scale as needed. These principles guide key decisions in cloud planning and expansion.

In summary, this course has prepared me not only to build a full stack cloud-native application, but also to plan intelligently for its continued growth, stability, and cost-efficiency in the real world.