CS 470 Final Reflection

Name: Justin Dougherty

Date: 15 December 2024

Assignment: CS 470 Final Reflection

Presentation Link: https://youtu.be/s9eVM0yBnRA

Experiences and Strengths

Professional Goals

CS 470 has been instrumental in advancing my skills as a full stack developer and has prepared me to confidently transition into cloud-native development roles. By integrating cloud services with a modern full stack web application, I now have experience in building scalable, secure, and efficient serverless solutions—key skills sought after in today's tech industry.

Skills Learned and Mastered

Throughout this course, I have learned:

Serverless Architecture: Implementing AWS Lambda, API Gateway, and S3 to create a serverless backend for automatic scaling and cost optimization.

Containerization: Using Docker and Docker Compose to containerize applications for portability, scalability, and efficient orchestration.

API Development and Security: Designing, testing, and securing APIs with authentication, encryption, and rate limiting using AWS services.

Cloud Deployment: Successfully deploying a MEAN stack application to AWS and integrating cloud-native services.

Elasticity and Scalability: Understanding and applying elasticity and pay-for-use principles to optimize resource allocation and costs.

These skills position me as a versatile candidate capable of delivering high-quality software while adapting to modern cloud infrastructure practices.

Strengths as a Software Developer

My strengths as a software developer include:

Problem-Solving: Addressing complex challenges like scalability, integration, and security in serverless cloud environments.

Adaptability: Quickly adopting and implementing new tools, such as AWS cloud services, to stay ahead of industry trends.

Collaboration: Leveraging tools like GitHub and CI/CD pipelines to collaborate and ensure smooth deployments.

Attention to Detail: Writing clean, secure code and optimizing cloud configurations to minimize costs while maintaining performance.

Roles I Am Prepared For

With these skills, I am prepared to take on the following roles:

Cloud Application Developer: Building and maintaining scalable cloud-based applications.

Full Stack Developer: Designing frontend and backend solutions integrated with serverless APIs.

DevOps Engineer: Leveraging containerization, CI/CD, and automation to streamline development workflows.

API Developer: Securing and managing APIs with robust serverless architectures.

Planning for Growth

In planning for future growth of my web application, the use of microservices and serverless architecture will be central to scaling efficiently.

Handling Scale and Error Management

Scale: AWS Lambda provides automatic scaling, where functions trigger based on demand without manual intervention. Using API Gateway as the entry point ensures that the load is managed seamlessly, enabling elasticity to support higher traffic.

Error Management: Implementing retries, dead-letter queues (DLQs) for Lambda, and logging with AWS CloudWatch will ensure visibility into failures and quick recovery.

Cost Predictability

Serverless: AWS Lambda and S3 follow a pay-for-use model. Costs are incurred only when the functions are triggered or storage is used, making it highly predictable for sporadic workloads.

Containers: Containers using AWS ECS or EKS are better for consistent, long-running workloads but may incur higher costs due to baseline resource reservations.

Which is More Predictable?

Serverless provides better cost predictability for applications with varying traffic because resources scale automatically, and costs are usage-based.

Containers offer more predictable costs for steady workloads, but they require constant management and provisioned infrastructure.

Pros and Cons for Future Expansion

Factors	Serverless	Containers
Pros	- Automatic scaling	- Better for long-running workloads
	- Pay-as-you-go cost model	- Fine-grained control over resources
	- Minimal operational overhead	- Suitable for consistent traffic
Cons	- Cold start latency in some cases	- Requires resource provisioning
	- Not ideal for complex, long tasks	- Higher costs for idle resources

Elasticity and Pay-for-Service

Elasticity enables the application to scale up or down dynamically based on demand, ensuring resources are used efficiently. The pay-for-service model aligns costs with actual usage, making serverless architecture ideal for unpredictable growth. This flexibility will allow my application to expand without over-provisioning or incurring unnecessary costs.

Conclusion

In summary, CS 470 has equipped me with modern skills to build, scale, and secure cloud-native applications. Leveraging serverless solutions like AWS Lambda and S3 ensures efficient resource management and cost optimization. My strengths in problem-solving, adaptability, and cloud development position me for roles where innovation and scalability are key. Moving forward, microservices and serverless will remain central to my approach, enabling sustainable and cost-effective application growth.

YouTube Presentation Link: https://youtu.be/s9eVM0yBnRA