July Wellman

CS 470 Full Stack Development II

Southern New Hampshire University

October 15, 2025

Project Two: Presentation Cloud Development

YouTube Link:

https://youtu.be/qmo78eCySek

Power Point Script:

Slide 1: Introduction

Hello, my name is July Wellman. In this presentation, I will walk through my experience migrating a full-stack Angular application into a cloud-native architecture using AWS microservices. My goal is to explain how modern web applications evolve from local development to scalable, serverless cloud environments.

Slide 2: Purpose & Overview

The purpose of this presentation is to describe the process and challenges of migrating a full-stack web app to AWS. I'll cover containerization, serverless architecture, scalability principles, and cloud security practices.

Slide 3: From Local Stack to Containers

The first step in my migration was containerization. I used PowerShell to initialize the Angular site and wrapped it with Docker containers. Each service: frontend, backend, and database, was isolated to ensure consistent environments for development and deployment.

Slide 4: Tools for Containerization

I used Docker Desktop, Docker Compose, and PowerShell to manage automation. Docker Compose simplified orchestration by defining services and networks in a single YAML file. The benefit to Docker is with one command, I could deploy or stop the entire stack, improving workflow efficiency.

Slide 5: Understanding Serverless Architecture

Next, I transitioned parts of the backend to AWS Lambda. Serverless computing removes the need for manual server management. Lambda functions automatically scale, run on demand, and you only pay for the actual execution time.

Slide 6: S3 Storage vs local Storage

I replaced local hosting with Amazon S3. It offers high durability, redundancy, and scalability. Hosting the Angular frontend in S3 eliminated the need for a dedicated web server and made it globally accessible.

Slide 7: Integrating Lambda with API Gateway

I used AWS Lambda for backend logic and API Gateway to handle requests. Each Lambda function represented a specific route, such as retrieving or updating trip data. The integration involved setting permissions, linking endpoints, and deploying the API.

Slide 8: Data Model Differences: MongoDB vs DynamoDB

Migrating data models required shifting from MongoDB's flexible schema to DynamoDB's key-value and document-based design. I used partition and sort keys to define efficient queries through the AWS SDK.

Slide 9: Scalability and Elasticity

AWS provides elasticity, allowing resources to scale automatically with demand. The pay-for-use model ensures you only pay for what you consume—ideal for dynamic workloads and cost management.

Slide 10: IAM Roles and Policies

Security was handled using IAM roles and policies. I followed the Principle of Least Privilege, creating roles that allowed Lambda to access only the necessary S3 buckets and DynamoDB tables.

Slide 11: Securing Cloud Connections

Connections between services were secured using HTTPS, IAM trust policies, and environment variables. This ensured that sensitive credentials were never hard-coded or exposed in the codebase.

Slide 12: Key Takeaways

To summarize: Docker provided environment consistency, AWS Lambda enabled serverless scalability, and IAM policies ensured secure access control. This migration strengthened my understanding of cloud-native development and deployment. My biggest obstacle was ensuring I was on correct versions when initially putting the project together.

References

- **Amazon Web Services. (2023).** ***AWS Lambda: Developer guide.*** **Amazon Web Services, Inc.** **https://docs.aws.amazon.com/lambda/latest/dg/welcome.html**
- **Docker, Inc. (2023).** ***Docker overview.*** **https://docs.docker.com/get-started/overview/**
- **Google Cloud. (2023).** ***Serverless computing overview.*** **Google Cloud Platform. https://cloud.google.com/serverless**
- **MongoDB, Inc. (2023).** ***MongoDB vs DynamoDB: A detailed comparison.*** **MongoDB, Inc.** **https://www.mongodb.com/compare/mongodb-vs-dynamodb**
- **Microsoft. (2023).** ***What is container orchestration?*** **Microsoft Learn. https://learn.microsoft.com/en-us/azure/architecture/guide/containers/container-orchestration**