Southern New Hampshire University

CS-470: Full Stack Development II

Project Two Conference Presentation

Chris Wong

June 23, 2024

https://youtu.be/2KpZYiDF5Aw

Slide 1:
Introduction

Hello, my name is Chris Wong, I am a senior at SNHU majoring in Computer Science.

Slide 2:

Overview

Today, I'll share insights into migrating from a traditional full-stack application to a cloud-native solution using AWS. This presentation aims to articulate the intricacies of cloud development to both technical and non-technical audiences. Let's start by discussing the foundational technology that facilitates this migration: containerization.

Slide 3:
Containerization

Containerization models are crucial for ensuring consistency and efficiency when migrating applications to the cloud. Using Docker, we package our applications and their dependencies into containers. This method guarantees that our application performs uniformly across different environments, facilitating the 'lift and shift' approach, where we move applications to the cloud without modifying them. However, for some components, we opt for 're-platforming,' which involves making minor tweaks to optimize performance in the cloud environment without redesigning the whole application. Managing these containers, especially when they scale, requires effective orchestration tools. Let's explore how Docker Compose aids in this process.

Slide 4: Orchestration

Docker Compose simplifies the management of multi-container applications by allowing us to define our entire stack configuration in a single YAML file. This not only makes deployment easier but also ensures our services interact seamlessly. With our containers managed effectively, we could focus on enhancing our application's scalability and efficiency. This orchestration tool was instrumental in our project, allowing us to manage multiple services such as our web app and database efficiently.

Slide 5:

Serverless

Serverless architectures offer a significant shift in how we deploy and manage applications. By eliminating the need to manage servers, we can focus solely on code while AWS handles the scaling. This approach is not only efficient but also cost-effective as it follows a pay-for-use model. Additionally, when comparing AWS S3 storage to traditional local storage, S3 provides scalable, durable, and cost-efficient cloud storage solutions, which is a stark contrast to local storage systems that can be limited in scalability and often involve upfront costs for provisioning and maintenance.

Slide 6:
API & Lambda

Building on the serverless model, AWS Lambda and API Gateway form the backbone of our serverless architecture. These services enable our application to operate without the need for provisioning or managing servers. AWS Lambda executes code in response to events, such as HTTP requests from API Gateway, and only charges for the compute time consumed. This setup allows for both high availability and scalability. The seamless integration of Lambda with API Gateway facilitates smooth interactions within our application. We employ scripts that define how each API request is processed—whether fetching data, posting updates, or managing user interactions. These scripts, primarily using Node.js, handle the business logic of the application, ensuring efficient operation. Integrating this with the frontend involves defining API endpoints in the API Gateway that connect to Lambda functions, setting up CORS to handle cross-origin requests, and ensuring the frontend components can interact with these endpoints smoothly.

Slide 7:
Database

As we delve deeper into our serverless architecture, managing data efficiently becomes crucial. Transitioning from MongoDB to DynamoDB brought to light their differing data models. MongoDB's flexible, document-oriented approach allows easy storage of diverse data structures, ideal for complex queries. On the other hand, DynamoDB’s key-value and document database structure, optimized for high performance and scalability within AWS's managed environment, offers operational ease without the overhead of manual database management. To make the most of DynamoDB’s features, we crafted queries and Lambda functions that interact seamlessly through Amazon API Gateway. This shift not only streamlined our operations but also significantly enhanced the performance and scalability of our application, especially in a serverless setting where efficiency is paramount.

Slide 8:
Cloud-Based Development Principles

As we've optimized our database interactions for performance and scalability, it's crucial to recognize the underlying cloud development principles that support these enhancements. Two fundamental principles are elasticity and the pay-for-use model. Elasticity allows our cloud resources to dynamically scale in response to application demands, ensuring optimal performance without manual intervention. Meanwhile, the pay-for-use model enhances financial efficiency by ensuring we only pay for the resources we actively use, eliminating waste.

Slide 9:
Securing Your Cloud Application

Building on our foundation of robust cloud principles, securing our application becomes the next critical step. We implement comprehensive security measures to prevent unauthorized access, employing AWS Identity and Access Management (IAM) to define precise roles and policies. This framework ensures that only authorized personnel and services have access to specific resources under strict conditions. We've developed custom policies tailored specifically for our needs. For instance, certain policies enable our Lambda functions to safely interact with DynamoDB and API Gateway, while preventing any unintended operations. A specific example is a policy that allows Lambda functions to retrieve data from DynamoDB, demonstrating our proactive approach to fine-tuned access control. Another policy meticulously defines how the API Gateway can invoke Lambda functions, ensuring that every access point is secure. To further fortify our application, measures were taken to secure all connections between our services. For example, all communication between Lambda and API Gateway is conducted over HTTPS, encrypting data in transit with TLS to prevent interception and tampering. Similarly, interactions between Lambda functions and DynamoDB are safeguarded with IAM roles that control access, coupled with encryption at rest within DynamoDB, thus protecting our data from unauthorized access.

Slide 10:

Conclusion

In conclusion, cloud development offers unparalleled flexibility and scalability. With the pay-as-you-go model, costs are directly tied to usage, reducing the overhead by eliminating upfront investments in infrastructure. Moreover, serverless architectures further decrease the need for ongoing hardware management, streamlining operations significantly. Security is another cornerstone of cloud development; managed services ensure that infrastructure is continually updated and patched with the latest security fixes. By establishing robust security policies, we ensure that our application remains secure in the cloud, giving us peace of mind and safeguarding our data and processes.

Thank you for joining me today to explore the transformative benefits and efficiencies of cloud development, illustrating how modern cloud practices can optimize and secure applications.