Project Two Conference Presentation: Cloud Development - *Script*

Johnathan Phillips

10/27/2024

Video Link: https://youtu.be/rK_BgdWdI7I

Slide 1: Cloud Development with AWS and Docker Compose

"Hello, everyone! My name is Johnathan Phillips, and I'm Computer Science major at Southern New Hampshire University. Today, I'll be presenting my recent work on building a cloud-hosted website using AWS and Docker Compose. We'll walk through the main concepts, from containerization to serverless cloud and security, to give you a clear picture of what goes into cloud development."

Slide 2: Introduction

"To start, a bit about me: I'm studying computer science with a focus on cloud development, cybersecurity, and artificial intelligence. In this project, I launched a containerized, cloud-hosted application. We'll go over the some of the setup and benefits of using containers and serverless technologies for this purpose."

Slide 3: Containerization

"Containerization is a method of packaging an application with its dependencies, ensuring consistent behavior across environments. To migrate our full stack application to the cloud, I used a few key models.

First, I separated the frontend and backend into their own containers, creating isolated and portable environments. This setup aligns with a microservices model, which allows each service to be developed, deployed, and scaled independently.

Finally, Docker Compose was used to orchestrate these containers. By configuring everything in a single YAML file, Docker Compose enables seamless communication between services, making deployment to the cloud efficient and scalable – more on YAML files in the next two slides.”

Slide 4: Orchestration

"Once we have containers, the next step is orchestration. Each container operates independently, which prevents conflicts and keeps resources isolated. Docker Compose simplifies orchestration by using a YAML file to manage multiple containers and their network settings. This setup allowed me to link the frontend and backend with a closed, internal network, ensuring smooth data flow between services."

Slide 5: YAML Example Files

"Here’s a couple of the Docker Compose YAML files used in this project. A single configuration file specifies both the application containers and network requirements, making deployment much easier. By defining services and ports in a YAML file, I can launch and manage the entire application environment with one command, docker-compose up."

Slide 6: The Serverless Cloud – Serverless Architecture

"Moving on to the serverless cloud. Serverless is a cloud computing model where the provider, in this case, AWS, manages the infrastructure. This allows developers like us to focus on code rather than server setup or maintenance. Serverless solutions are scalable, adjusting automatically with demand, and cost-efficient, as you pay only for the time your code runs rather than for idle server capacity."

Slide 7: The Serverless Cloud – What is S3 Storage?

"An essential part of this project was S3 storage. S3, or Simple Storage Service, is AWS’s scalable cloud storage solution. Unlike local storage, S3 is accessible over the internet and can expand automatically to meet storage needs. It’s secure and provides an ideal place for storing application files, databases, and backups that can be accessed globally."

Slide 8: The Serverless Cloud – What is S3 Storage (Diagram)

"This slide shows a very simplified version of how S3 storage works within the AWS cloud. AWS S3, or Simple Storage Service, allows users to store and retrieve data over the internet. Here, you see how a user can interact with a service that uploads objects to an S3 bucket. Once stored in the S3 bucket, these objects can be searched and retrieved whenever needed.

This approach provides two main advantages. First, availability—data stored in S3 is accessible from anywhere with internet access, making it ideal for applications with a global audience. Second, scalability—S3 expands automatically to meet data storage demands, so it grows alongside your application without requiring manual intervention. This flexibility makes S3 an essential tool for cloud-hosted applications."

Slide 9: The Serverless Cloud – API & Lambda

"AWS also offers serverless APIs through API Gateway, which we combined with Lambda functions. Serverless APIs handle requests without dedicated servers, scaling automatically with traffic. This is cost-efficient because it charges based on actual usage.

Specifically, this project uses several API calls like GET, POST, DELETE, and PUT to interact with a questions and answers database. These calls are handled by endpoints like /Questions and /Answers, enabling efficient backend operations."

Slide 10: The Serverless Cloud – Lambda Functions

"Let's go deeper into Lambda functions. Lambda is AWS's event-driven compute service, which runs code in response to specific events—such as HTTP requests from the API Gateway. In this project, Lambda functions handle backend logic, responding to user inputs and querying databases. For instance, I created functions like EchoFunction and TableScan to manage interactions and data retrieval.

The integration connects Lambda functions to the API Gateway, allowing the frontend to seamlessly trigger these backend actions without needing dedicated servers."

Slide 11: The Serverless Cloud – Specific Lambda Functions

"Here are some of the key Lambda functions used in the project. Functions like GetSingleRecord, TableScan, and UpsertQuestion enable us to manage and interact with data in DynamoDB effectively. By implementing Lambda functions like DeleteRecord and FindOneQuestion, we gain full CRUD functionality to handle all user requests on the backend."

Slide 12: The Serverless Cloud – Database Differences

"Now, let's look at the database. In this project, I used DynamoDB, an AWS-managed NoSQL database that's optimized for high-performance queries. Unlike MongoDB, which stores data in JSON-like documents, DynamoDB uses a key-value structure that works well for quick lookups and scalability."

Slide 13: The Serverless Cloud - Database

"The primary queries we used were single record retrievals through GetSingleRecord and broader scans with TableScan, which allowed us to search for specific data points within tables. By using these queries in Lambda functions, we achieve high performance and efficient data access.

You can also see some example JSON queries on the right, specifically, one to delete a question and one to delete an answer in DynamoDB."

Slide 14: The Serverless Cloud – Example Function Code Snippet for TableScan

"Here's an example code snippet for the TableScan function. This Lambda function performs a scan across our DynamoDB table, retrieving records based on specified conditions. By leveraging DynamoDB's primary key structure, TableScan efficiently filters results, improving the application's response time and query efficiency."

Slide 15: Cloud-Based Development Principles

"Cloud development has two key principles: elasticity and the pay-for-use model. Elasticity means resources can scale up or down automatically based on demand, ensuring applications remain responsive while also being cost-effective during low-usage periods.

In the pay-for-use model, AWS only charges for the resources actually consumed. For example, Lambda charges based on the time a function runs, and S3 costs are based on the actual data stored. This aligns costs with usage, which is highly efficient."

Slide 16: Securing Your Cloud Application

"Security is critical in cloud applications, and AWS provides several tools to control access to resources. For this project, I implemented access controls using IAM roles and policies. IAM roles determine who can access which resources, while policies define what actions are permitted.

Custom policies were created to restrict access to specific resources like S3, Lambda functions, and DynamoDB tables. Additionally, I secured connections between API Gateway and Lambda by configuring IAM permissions to control which users and services have access. This layered approach protects sensitive data and ensures secure interactions between components."

Slide 17: Conclusion

"So, in conclusion, this project covered concepts from cloud migration, to containerization, serverless APIs and database management. We saw how serverless solutions enable scalability and cost savings, while AWS IAM roles and policies help secure our application.

Thank you for your time, and I hope this presentation has highlighted the benefits of serverless cloud solutions and the potential AWS offers for modern web applications."