Anthony W Bradley

CS470 Project Presentation Script

A Journey into Cloud Development :

Hello everyone, my name is Anthony Bradley, and today I will be sharing my experiences migrating a full stack application to a cloud-native web application using AWS microservices. This presentation aims to articulate the intricacies of cloud development to both technical and nontechnical audiences.

Purpose of Presentation:

The purpose of this presentation is to provide a comprehensive overview of the migration process from a traditional full stack application to a cloud-native architecture using AWS microservices. We will explore various aspects of this journey, including containerization, serverless computing, cloud-based development principles, and security.

Containerization and Orchestration:

To migrate a full stack application to the cloud, we first need to containerize our application. This involves packaging the application and its dependencies into containers. Tools like Docker and Docker Compose are essential for this process. Docker Compose allows us to define and run multi-container Docker applications, making it easier to manage and orchestrate our services.

The Serverless Cloud:

Serverless computing is a cloud-computing execution model where the cloud provider dynamically manages the allocation of machine resources. One of the key advantages of serverless is that it allows developers to focus on writing code without worrying about infrastructure management. AWS Lambda is a popular serverless compute service that lets you run code without provisioning or managing servers. S3 storage, another AWS service, provides scalable object storage and is often compared to local storage for its durability and availability.

Serverless API and Lambda Logic:

Using a serverless API has several advantages, including reduced operational overhead and automatic scaling. AWS Lambda functions can be triggered by events from various AWS services, and API Gateway can be used to create, publish, maintain, monitor, and secure APIs. The integration of the frontend with the backend involves creating Lambda functions that handle

API requests and responses. Scripts are written to define the logic within these Lambda functions.

MongoDB vs. DynamoDB:

MongoDB and DynamoDB are both NoSQL databases, but they have different data models. MongoDB uses a document-oriented data model, while DynamoDB uses a key-value and document data model. During the migration, we performed various queries to interact with the data stored in DynamoDB. Scripts were written to handle data operations such as reading, writing, and updating records.

Cloud-Based Development Principles:

Cloud-based development principles include elasticity and the pay-for-use model. Elasticity allows the application to scale resources up or down based on demand, ensuring optimal performance and cost efficiency. The pay-for-use model means you only pay for the resources you consume, which can lead to significant cost savings.

Securing Your Cloud Application:

Security is a critical aspect of cloud development. Preventing unauthorized access involves defining roles and policies that control permissions. Custom policies were created to meet specific security requirements. Securing the connection between Lambda and API Gateway, as well as between Lambda and the database, is essential to protect data in transit. Additionally, S3 buckets must be configured with appropriate access controls to ensure data security.

Reflection:

Migrating to a cloud-native architecture involves several key steps: containerization and orchestration, leveraging serverless computing, understanding cloud-based development principles, and ensuring robust security measures. These elements are crucial for successfully transitioning to a cloud environment and reaping the benefits of scalability, flexibility, and cost efficiency.