### **Slide 1: Title Slide**

Hello, my name is Owen McCooey Capak. Welcome to my presentation on migrating from 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.

### **Slide 2: Overview**

In this presentation, I will walk you through the process of migrating from a full stack application to a cloud-native web application using AWS microservices. We will cover containerization, orchestration, serverless architecture, cloud-based development principles, and security.

### **Slide 3: Containerization**

Containerization involves transitioning from a monolithic architecture to microservices. Docker is a key tool in this process, packaging each microservice with its dependencies into containers. This ensures consistent deployment across various environments. The tools necessary for containerization include Docker for packaging applications, Docker Hub for storing and sharing images, and Kubernetes for orchestrating containerized applications.

### **Slide 4: Orchestration**

Docker Compose simplifies the setup of multi-container applications by allowing us to define and run these applications using a YAML file. It eases development by managing all services with a single command, ensures consistent environments across development, testing, and production, and provides scalability and portability for applications.

### **Slide 5: The Serverless Cloud**

Serverless computing allows developers to build and run applications without managing infrastructure. Its advantages include no server management, automatic scalability, and cost-efficiency, as you only pay for execution time. AWS S3 provides scalable object storage in the cloud, offering durability, accessibility, and automatic scaling compared to local storage.

### **Slide 6: The Serverless Cloud - API & Lambda**

Using a serverless API offers cost-efficiency, automatic scalability, and reduced management overhead. AWS Lambda functions are event-driven, supporting various languages and execution models. To create this, we produce Lambda handlers that define function behavior and configure the API Gateway to set up endpoints. The integration steps include creating Lambda functions, setting up the API Gateway, and connecting the frontend to call these APIs.

### **Slide 7: The Serverless Cloud - Database**

MongoDB and DynamoDB are both NoSQL databases, but MongoDB is document-oriented with a flexible schema, while DynamoDB is a key-value and document store with fast response times. Queries performed include CRUD operations and complex queries like aggregations in MongoDB and key-based queries in DynamoDB. Scripts produced include schema definitions and query scripts for performing these operations.

### **Slide 8: Cloud-Based Development Principles**

Elasticity allows cloud applications to automatically scale resources based on demand, ensuring cost-effectiveness. The pay-for-use model means you only pay for actual usage, aiding in cost management and budgeting by predicting costs based on usage patterns.

### **Slide 9: Securing Your Cloud App**

Preventing unauthorized access involves setting up IAM roles and policies to define access permissions, and using Multi-Factor Authentication (MFA) for added security. Roles are sets of permissions for users or services, while policies are specific permissions attached to roles. Custom policies include tailored access permissions and resource-based policies for resources like S3.

### **Slide 10: API Security**

Securing the connection between Lambda and the API Gateway can be done through IAM authorizations to restrict API access and VPC integration for secure communication. For Lambda and the database, IAM roles grant permissions, and encryption secures data in transit and at rest. S3 buckets are secured using bucket policies to control access and encryption to ensure data is protected.

### **Slide 11: Conclusion**

In conclusion, cloud-native applications provide scalability, automatically adjusting resources based on demand. Serverless architecture reduces management overhead and costs, allowing developers to focus on code while the cloud provider handles infrastructure. Security is ensured through IAM roles, policies, and encryption, protecting access and data in the cloud. Thank you for your time, and if you have any questions, please feel free to ask.