CS 470 Project Two Conference Presentation: Cloud Development https://youtu.be/H6JDzl5dWkY

Nicholle Caudy

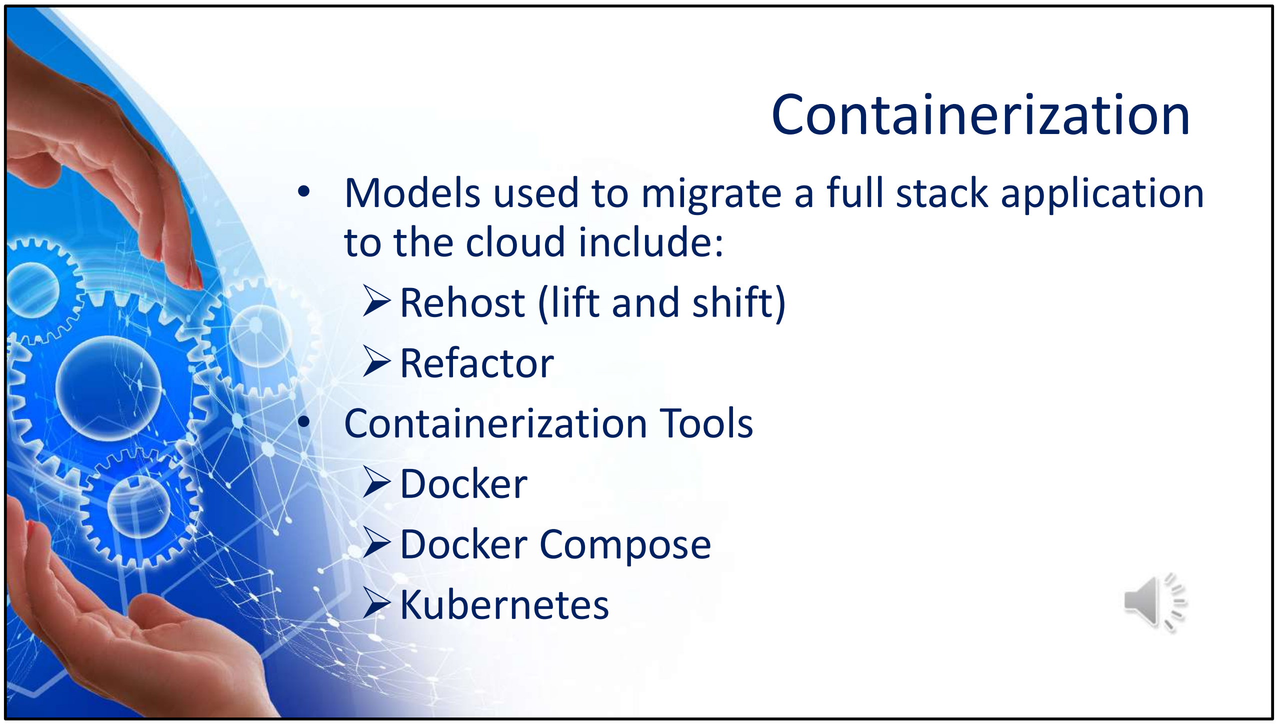
February/2026

CS 470 Project Two Conference Presentation: Cloud Development

Overview

- My name is Nikki and I am in my final stretch of earning my bachelor’s in Computer Science degree at SNHU.
- In this presentation I will discussing and illustrating the intricacies of migrating a full stack to a serverless cloud system that includes containerization, Serverless Cloud storage, APIs, database, and security of our cloud application.

Hello everyone, my name is Nikki and I am in my final stretch of earning my bachelor’s in computer science degree at SNHU. In this presentation I will be discussing and illustrating the intricacies of migrating a full stack to a serverless cloud system that includes containerization, particularly using Docker Compose, serverless cloud storage, APIs, database, and security of our cloud application.



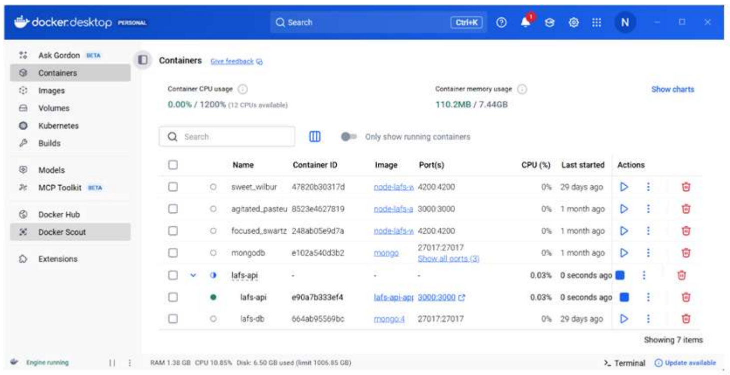
Containerization

- Models used to migrate a full stack application to the cloud include:
  - Rehost (lift and shift)
  - Refactor
- Containerization Tools
  - Docker
  - Docker Compose
  - Kubernetes

There are few models that can be used to migrate a full stack application to the cloud today I will be discussing rehosting or lift and shift and refactoring. Rehosting or lift and shift involves moving the application to the cloud with minimal changes, this allows for quick migration with the application's existing functionality. Refactoring involves breaking the application down into smaller pieces and restructuring the application code to make it more suitable for the cloud environment, allowing it to take full advantage of the cloud capabilities like scalability, performance and cost efficiency. Docker, Docker Compose, and Kubernetes are tools that developers use to perform containerization. In this instance we used Docker to perform the containerization, creating the frontend, API, and database containers and Docker Compose to orchestrate the containers allowing them to communicate with each other.

Orchestration

- Docker Compose
  - ➢ Simplified Management
  - ➢ Consistent Environment
  - ➢ Efficient Collaboration
  - ➢ Streamlined Networking
  - ➢ Testing and CI/CD Integration



Containers

Container CPU usage
0.00% / 1200% (12 CPUs available)

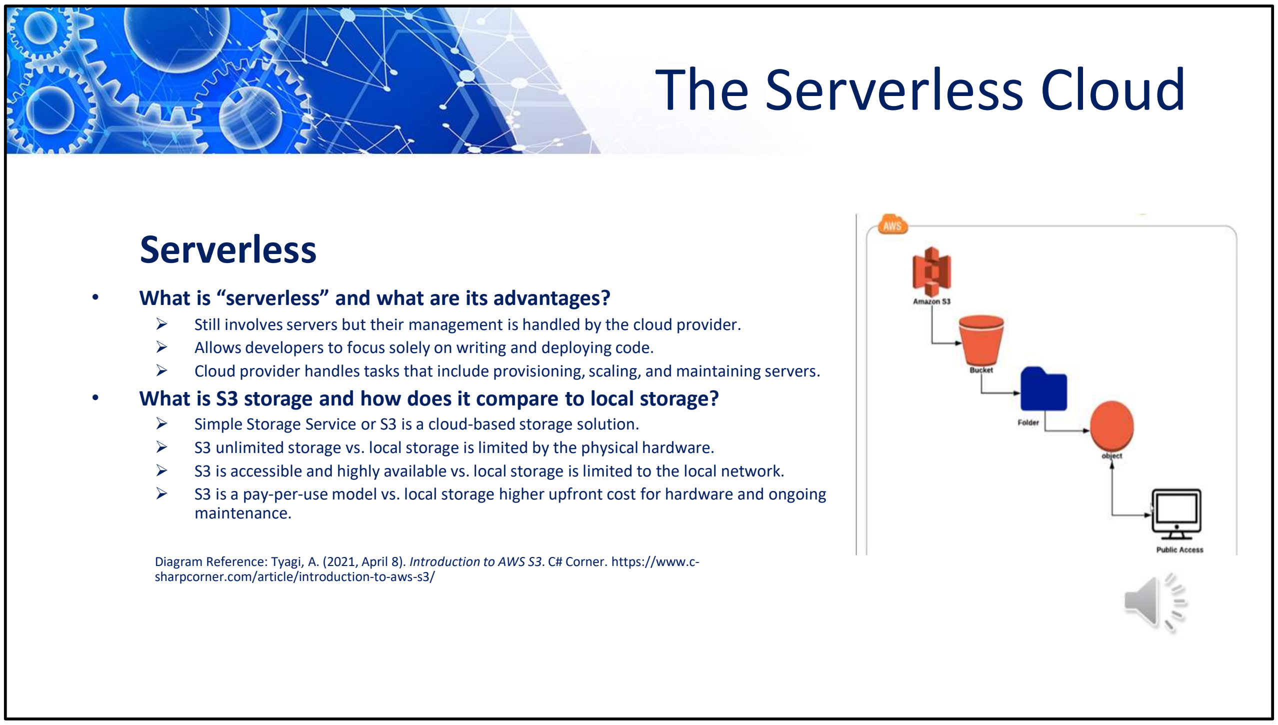
Container memory usage
110.2MB / 7.44GB

Name	Container ID	Image	Port(s)	CPU (%)	Last started
sweet_wilbur	47820b30317d	node-lafs-w	4200:4200	0%	29 days ago
agitated_pasteu	8523e4627819	node-lafs-a	3000:3000	0%	1 month ago
focused_swartz	248ab05e9d7a	node-lafs-w	4200:4200	0%	1 month ago
mongodb	e102a540d3b2	mongo	27017:27017 Show all ports (3)	0%	1 month ago
lafs-api	-	-	-	0.03%	0 seconds ago
lafs-api	e90a7b333ef4	lafs-api-api	3000:3000	0.03%	0 seconds ago
lafs-db	664ab95569bc	mongo:4	27017:27017	0%	29 days ago

Showing 7 items

RAM 1.38 GB CPU 10.85% Disk: 6.50 GB used (limit 1006.85 GB)

Docker Compose allows developers to manage and define multi-container applications using a single YAML file encapsulating all services, networks, and volumes making it easier to orchestrate applications with multiple components. Using a single docker-compose.yml file ensures that all developers are working in the same environment which reduces compatibility issues and streamlines collaboration. The YAML configuration file is sharable in nature and promotes better collaboration among developing team members. Docker Compose also sets up a default network for all services automatically, which simplifies communication between containers. Docker Compose is very useful for creating isolated testing environments that can be easily spun up and torn down which is beneficial for automated testing and continuous integration/continuous deployment workflows. The picture provided is of my Docker Dashboard with all the containers we created and used.



The Serverless Cloud

Serverless

- **What is “serverless” and what are its advantages?**
  - ➢ Still involves servers but their management is handled by the cloud provider.
  - ➢ Allows developers to focus solely on writing and deploying code.
  - ➢ Cloud provider handles tasks that include provisioning, scaling, and maintaining servers.
- **What is S3 storage and how does it compare to local storage?**
  - ➢ Simple Storage Service or S3 is a cloud-based storage solution.
  - ➢ S3 unlimited storage vs. local storage is limited by the physical hardware.
  - ➢ S3 is accessible and highly available vs. local storage is limited to the local network.
  - ➢ S3 is a pay-per-use model vs. local storage higher upfront cost for hardware and ongoing maintenance.

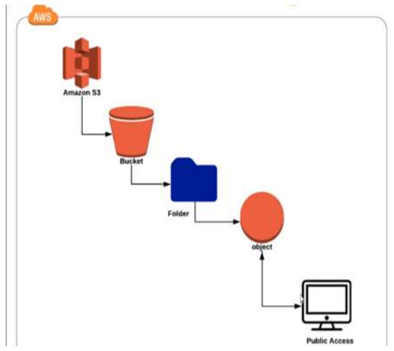
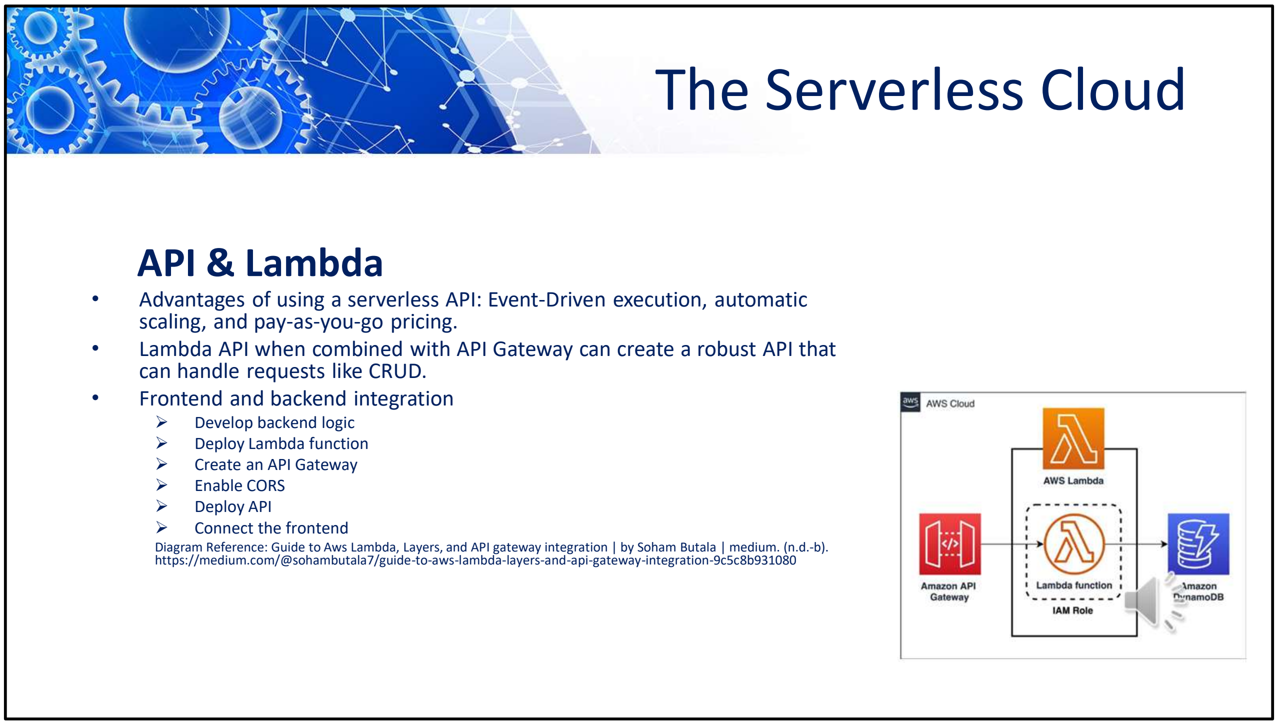


Diagram Reference: Tyagi, A. (2021, April 8). *Introduction to AWS S3*. C# Corner. https://www.c-sharpcorner.com/article/introduction-to-aws-s3/

Serverless, while despite its name still involves servers, is a cloud-native development model that allows developers to build and run applications without managing the underlying server infrastructure. This allows developers to focus solely on writing and deploying code for their applications. The cloud provider handles the scaling, maintaining, and provisioning of the servers. S3 or Amazon Simple Storage Service is a cloud-based storage solution that allows users to store and retrieve any amount of data from anywhere on the web and offers scalability, availability, and security. Some differences to note when comparing S3 Storage to local storage is S3 offers nearly unlimited storage while local storage is limited by the physical machine. Amazon S3 can be accessed from anywhere in the world providing high availability and accessibility with local storage accessibility is limited to the local network. S3 offers robust security but some organizations may be concerned about storing sensitive data off-site while local storage provides complete control over data and privacy.

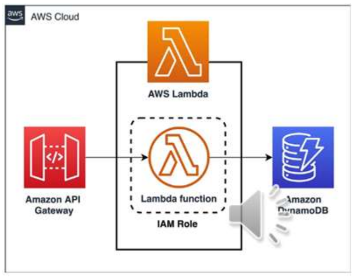


The Serverless Cloud

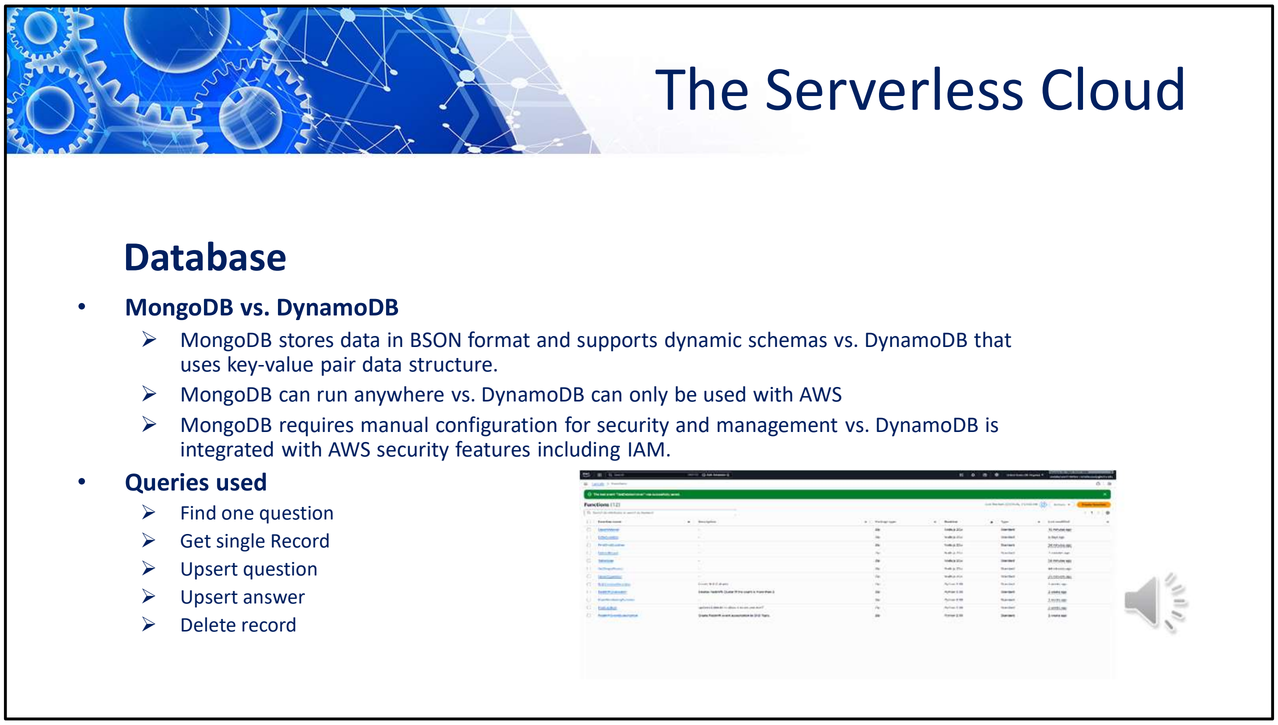
API & Lambda

- Advantages of using a serverless API: Event-Driven execution, automatic scaling, and pay-as-you-go pricing.
- Lambda API when combined with API Gateway can create a robust API that can handle requests like CRUD.
- Frontend and backend integration
  - ➢ Develop backend logic
  - ➢ Deploy Lambda function
  - ➢ Create an API Gateway
  - ➢ Enable CORS
  - ➢ Deploy API
  - ➢ Connect the frontend

Diagram Reference: Guide to Aws Lambda, Layers, and API gateway integration | by Soham Butala | medium. (n.d.-b). https://medium.com/@sohambutala7/guide-to-aws-lambda-layers-and-api-gateway-integration-9c5c8b931080



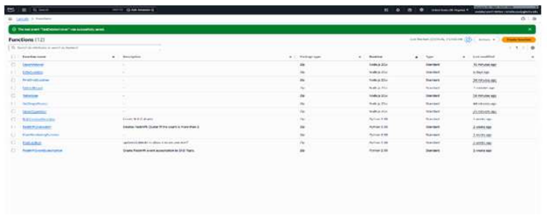
Serverless APIs are an event-driven model each endpoint corresponds to a function such as AWS Lambda that is triggered by an event like HTTP requests or data changes. Serverless APIs scale automatically based on demand and are cost effective with a pay-as-you-go pricing model. Serverless APIs also allow for faster deployment with rapid iterations and deployments of individual endpoints. Using AWS Lambda with API Gateway you can create a serverless REST API that invokes Lambda functions for handling requests and perform operations like create, read, update, and delete using the HTTP requests such as Get, Post, Put, and Delete. Setting out endpoints for each operation called the Lambda functions to perform the requested operation on the database. Endpoints were created to search for and retrieve different types of data from the database. To allow communication between the backend and the frontend, using the API Gateway, CORS or Cross-Origin Resource Sharing was used using the Options method of the API requests. Each of the various Lambdas had policies created to ensure only authorized users could gain access.



The Serverless Cloud

Database

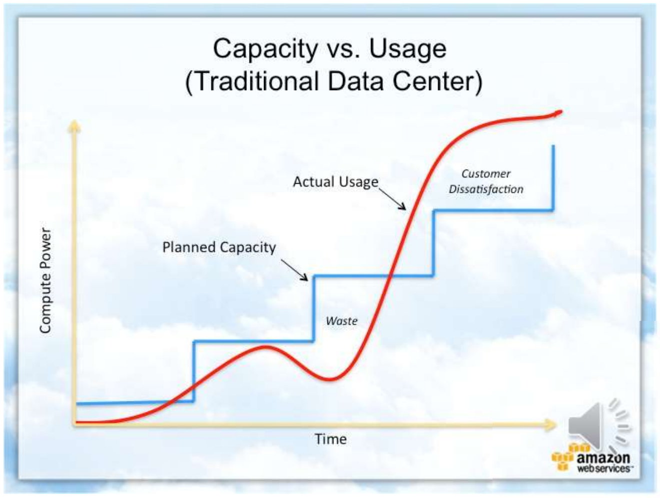
- **MongoDB vs. DynamoDB**
  - MongoDB stores data in BSON format and supports dynamic schemas vs. DynamoDB that uses key-value pair data structure.
  - MongoDB can run anywhere vs. DynamoDB can only be used with AWS
  - MongoDB requires manual configuration for security and management vs. DynamoDB is integrated with AWS security features including IAM.
- **Queries used**
  - Find one question
  - Get single Record
  - Upsert question
  - Upsert answer
  - Delete record



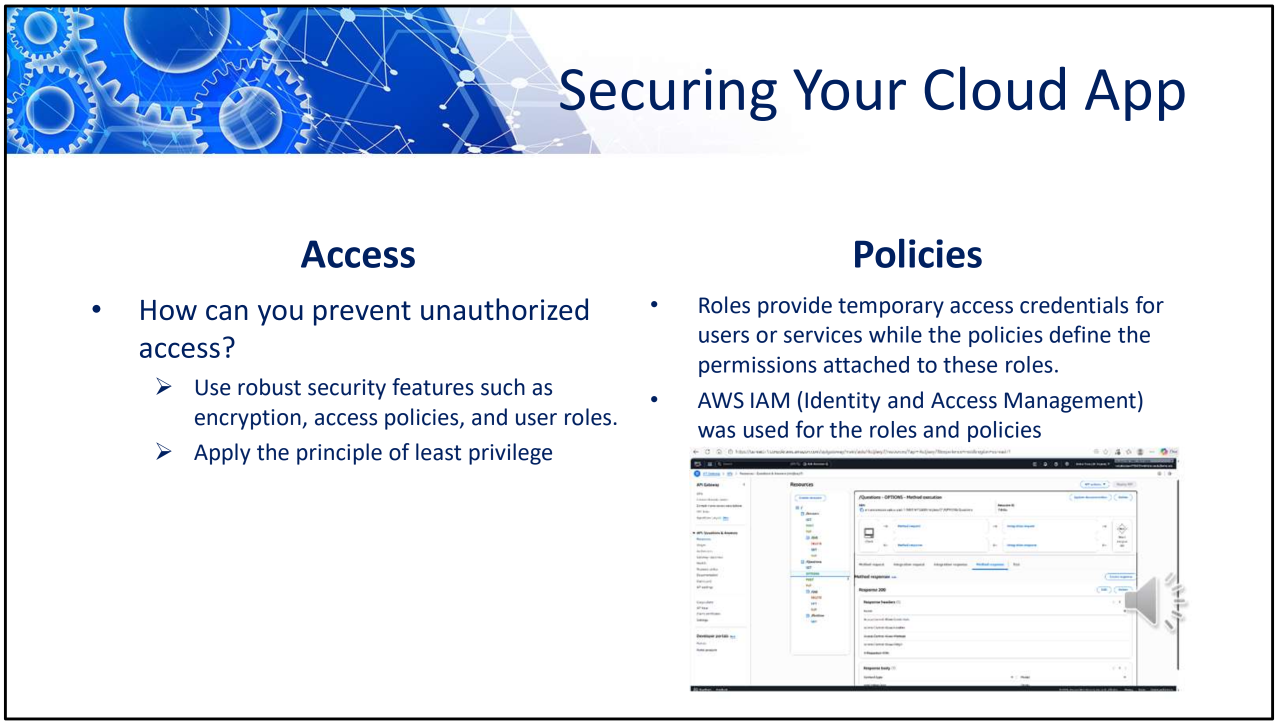
MongoDB and DynamoDB are both No-SQL databases. MongoDB is an open-source document-oriented database that stores data in BSON format while DynamoDB is a fully managed key-value pair data structure. DynamoDB is a database service offered by AWS and can only be used in this platform while MongoDB can be utilized anywhere. Security for MongoDB requires manual configuration and management but DynamoDB is integrated with AWS security features that includes IAM for access control and encryption at rest. We performed all the CRUD query operations by creating key-value pairs reading the pairs back using the unique IDs. We added questions and answers to the database and deleted records from the database as well. To do this we created several Lambdas that are listed on the slide and performed tests on each Lambda. To perform these tasks we created Get, Post, Put, and Delete scripts. The picture provided demonstrates the queries that were used.

Cloud-Based Development Principles

- Elasticity
- Pay-for-use model



Elasticity enables the cloud system to automatically adjust their resources to match changing workloads, scaling up or down based on current needs for your applications. The pay-for-use model allows users to only pay for the resources and time they actually use making it a cost-effective option.



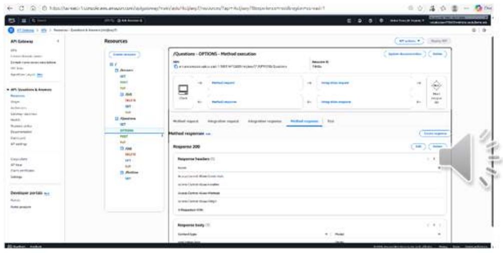
Securing Your Cloud App

Access

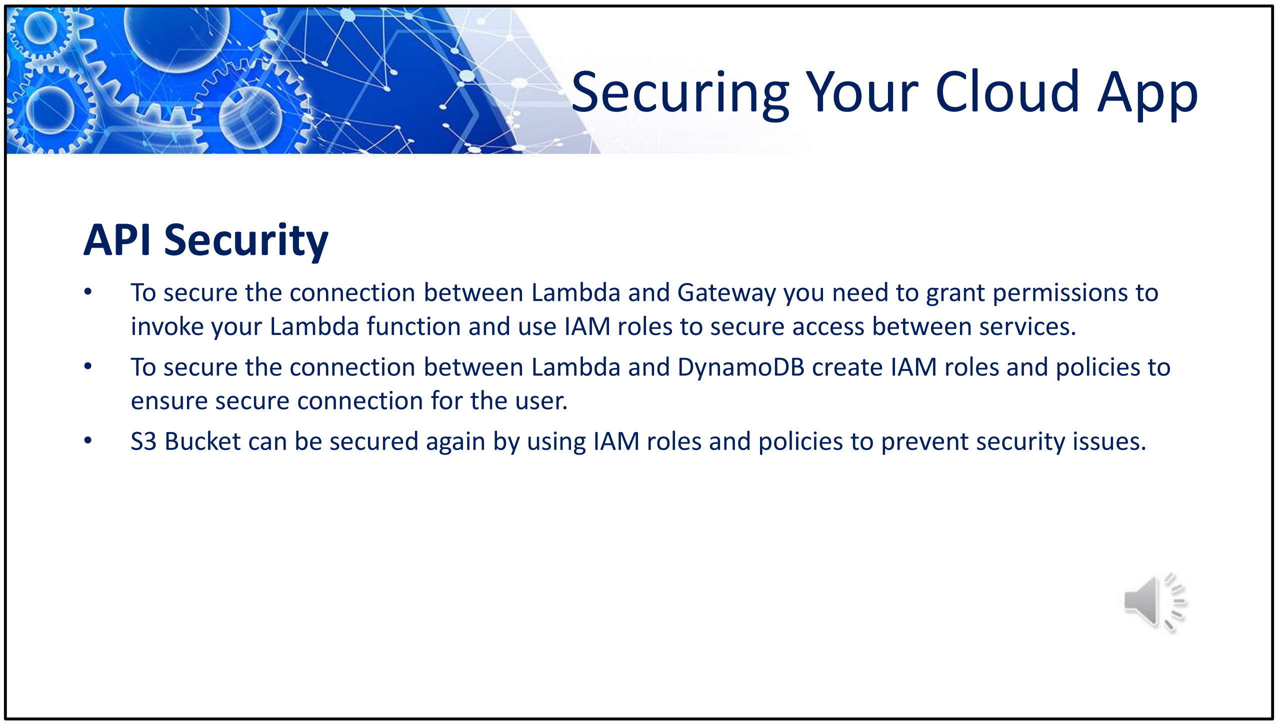
- How can you prevent unauthorized access?
  - ➢ Use robust security features such as encryption, access policies, and user roles.
  - ➢ Apply the principle of least privilege

Policies

- Roles provide temporary access credentials for users or services while the policies define the permissions attached to these roles.
- AWS IAM (Identity and Access Management) was used for the roles and policies



To prevent unauthorized access, you need to implement robust security features that include encryption, access policies, and role-based access control. To do this you can create AWS IAM or Identity and Access Management roles and policies. With these policies you can apply the principle of least privilege to ensure user cannot access things they should not. The relationship between roles and policies are that roles provide temporary access credentials for users or services while the policies define the permissions attached to these roles again ensuring that the principle of least privilege is followed. We used AWS IAM to initiate a lab role and created custom policies that allowed that role to access the database. This was accomplished by setting up CORS or Cross-Origin Resource sharing allowing us to specify the type of sharing that is allowed. We set up policies that allowed headers, credentials, origin, and methods through the options method in the REST configuration. The picture provided is demonstrating the CORS methods used.

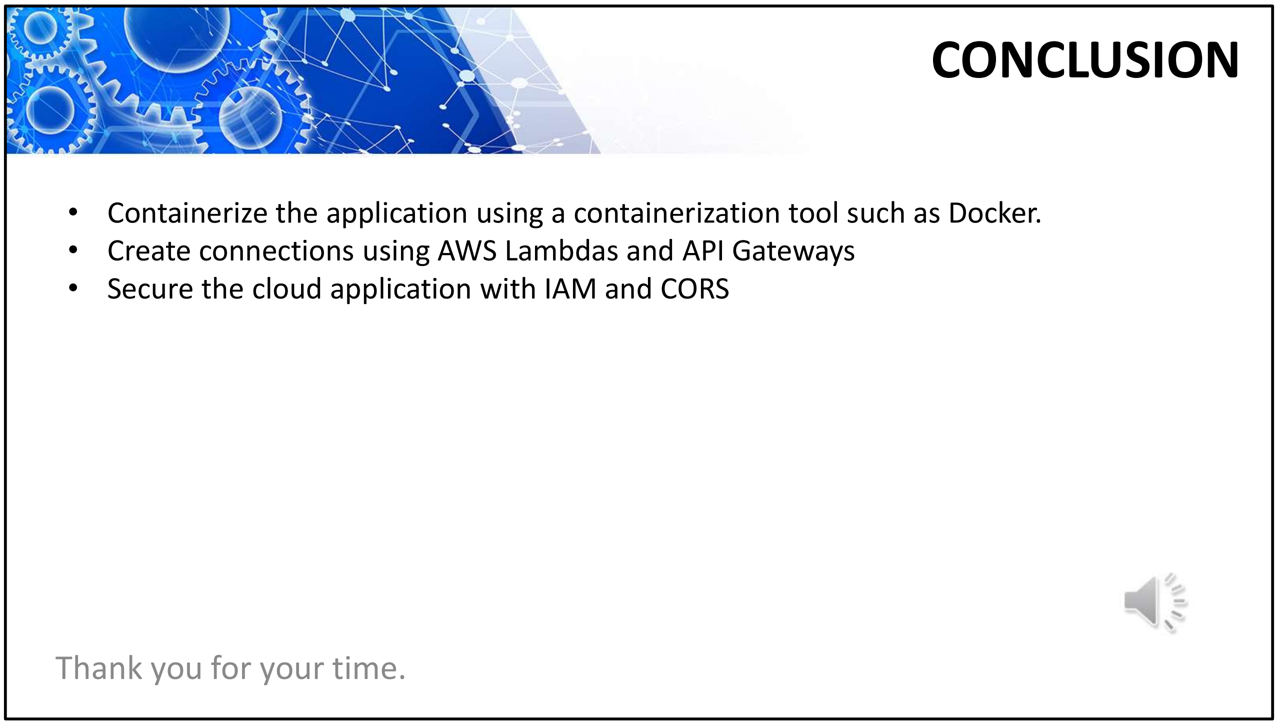


Securing Your Cloud App

API Security

- To secure the connection between Lambda and Gateway you need to grant permissions to invoke your Lambda function and use IAM roles to secure access between services.
- To secure the connection between Lambda and DynamoDB create IAM roles and policies to ensure secure connection for the user.
- S3 Bucket can be secured again by using IAM roles and policies to prevent security issues.

To secure the connection between Lambda and Gateway you need to grant permissions to invoke your Lambda function and use IAM roles to secure access between services. The connection between Lambda and DynamoDB is secured by creating IAM roles and policies to ensure a secure connection for the user. IAM roles and policies are also used to secure the connections for the S3 buckets.



CONCLUSION

- Containerize the application using a containerization tool such as Docker.
- Create connections using AWS Lambdas and API Gateways
- Secure the cloud application with IAM and CORS

Thank you for your time.

In conclusion there are a few focus points when refactoring your application to the Cloud. The first step is containerizing your application, in this case we used Docker, to ensure it is modular and all parts are functioning properly. Following this step, you need to create Lambdas and APIs to create the connections between the various modules of your application. Lastly, ensuring your application is secure is paramount, to secure your cloud application use the IAM roles and policies to define who can have access and under what conditions, using the principle of least privilege. Thank you for your time.