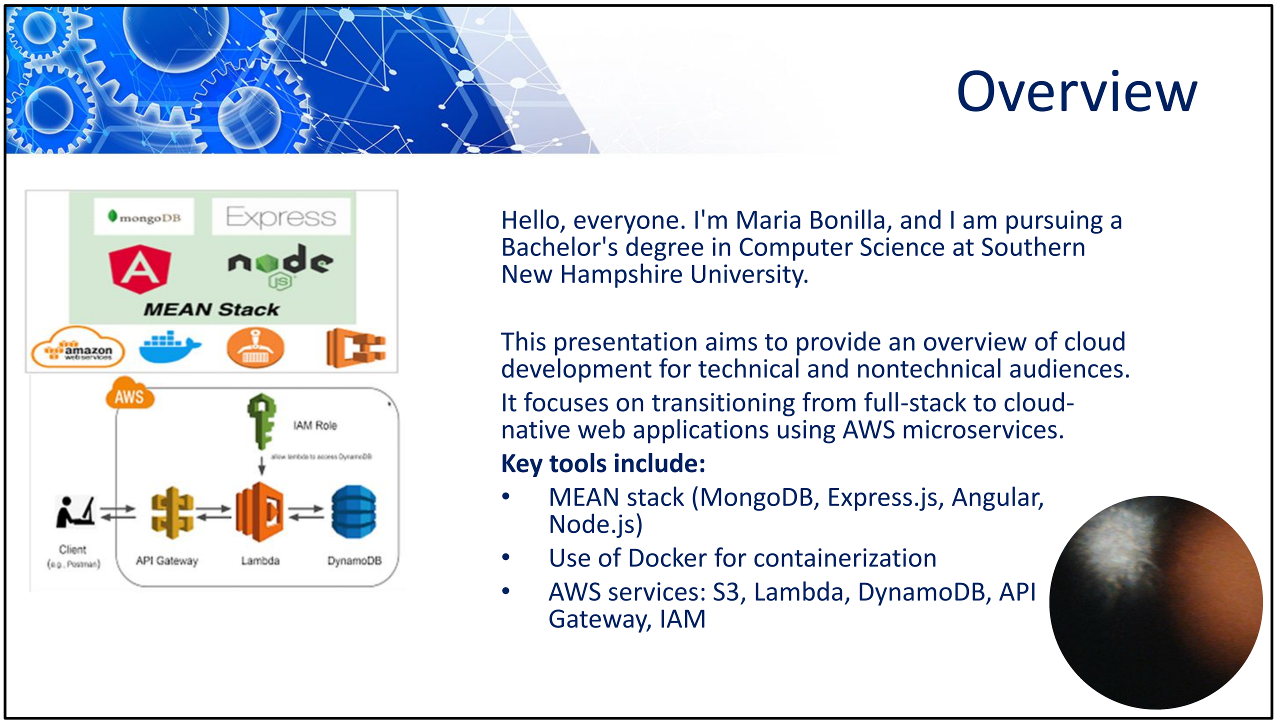
Name: Maria Bonilla
Date: April 20, 2025
Assignment: Project Two Conference Presentation: Cloud Development
Link of the video: https://www.youtube.com/watch?v=QFc4xEAA3SM

CS 470 Project Two Conference Presentation: Cloud Development

Maria Bonilla

April 20, 2025



Overview

Hello, everyone. I'm Maria Bonilla, and I am pursuing a Bachelor's degree in Computer Science at Southern New Hampshire University.

This presentation aims to provide an overview of cloud development for technical and nontechnical audiences.

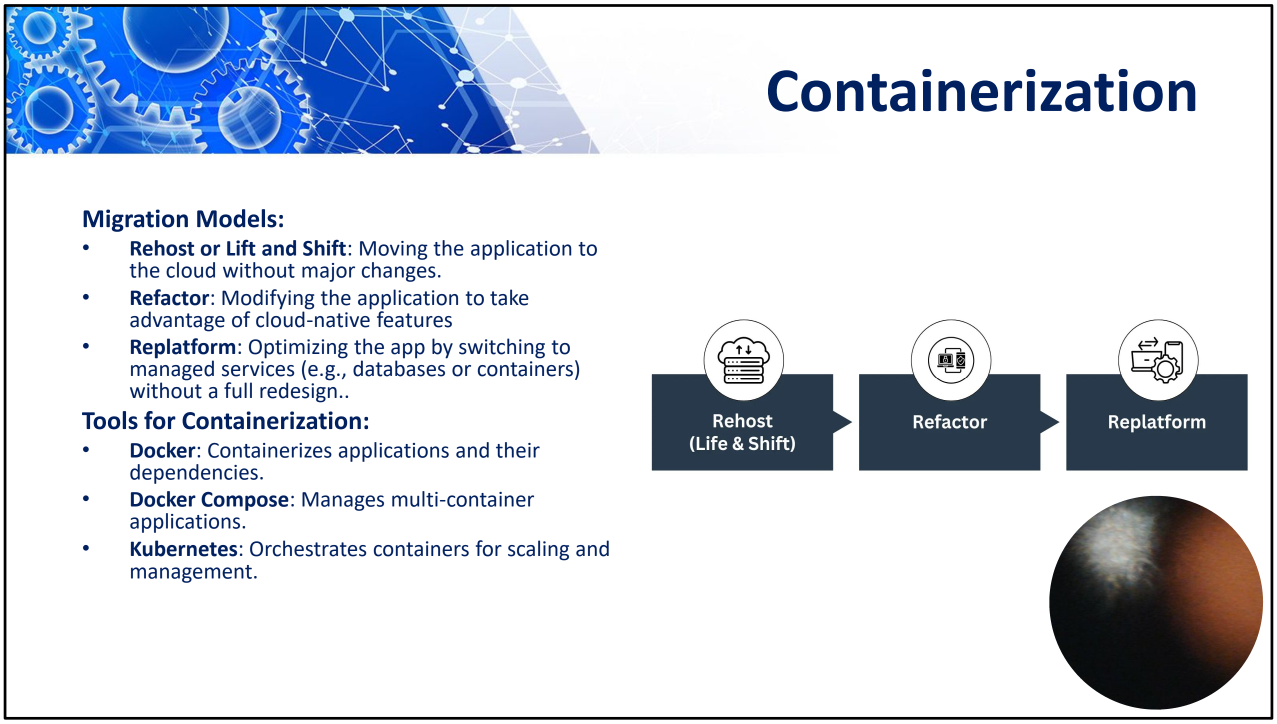
It focuses on transitioning from full-stack to cloud-native web applications using AWS microservices.

Key tools include:

- MEAN stack (MongoDB, Express.js, Angular, Node.js)
- Use of Docker for containerization
- AWS services: S3, Lambda, DynamoDB, API Gateway, IAM

Hello, everyone. I'm Maria Bonilla, and I am pursuing a Bachelor's degree in Computer Science at Southern New Hampshire University.

Today, I'll provide an overview of cloud development, focusing on shifting from full-stack to cloud-native web applications. We'll discuss the MEAN stack—MongoDB, Express.js, Angular, and Node.js, and the tools needed for this transition include Docker and AWS services like S3, Lambda, DynamoDB, API Gateway, and IAM.



Containerization

Migration Models:

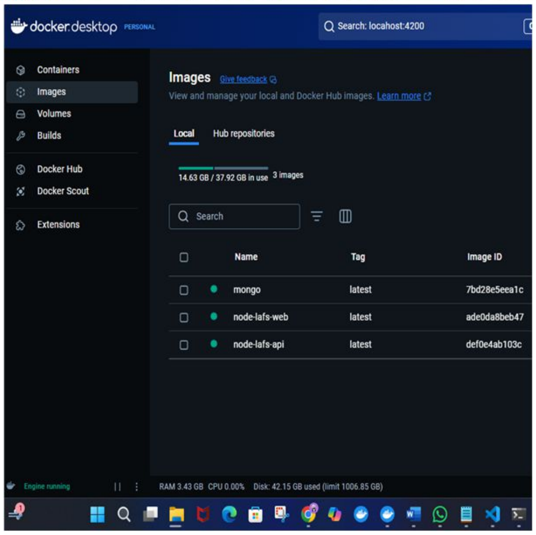
- **Rehost or Lift and Shift**: Moving the application to the cloud without major changes.
- **Refactor**: Modifying the application to take advantage of cloud-native features
- **Replatform**: Optimizing the app by switching to managed services (e.g., databases or containers) without a full redesign..

Tools for Containerization:

- **Docker**: Containerizes applications and their dependencies.
- **Docker Compose**: Manages multi-container applications.
- **Kubernetes**: Orchestrates containers for scaling and management.

Two primary methods for migrating the application to the cloud are Lift and Shift and Refactoring. Lift and Shift involves moving an existing app, like a MEAN stack application, directly to a cloud provider without significant changes. Refactoring focuses on optimizing the app for the cloud to enhance performance and scalability. Docker was used to create containers for the front end, API, and database, while Docker Compose managed container communication. The app was built using the MEAN stack—MongoDB, Express.js, Angular, and Node.js.

Orchestration



docker desktop PERSONAL

Search: localhost:4200

Images Give feedback

View and manage your local and Docker Hub images. Learn more

14.63 GB / 37.92 GB in use 3 images

	Name	Tag	Image ID
□ ●	mongo	latest	7bd28e5eea1c
□ ●	node-lafs-web	latest	ade0da8beb47
□ ●	node-lafs-api	latest	def0e4ab103c

Engine running RAM 3.43 GB CPU 0.00% Disk: 42.15 GB used (limit 1006.85 GB)

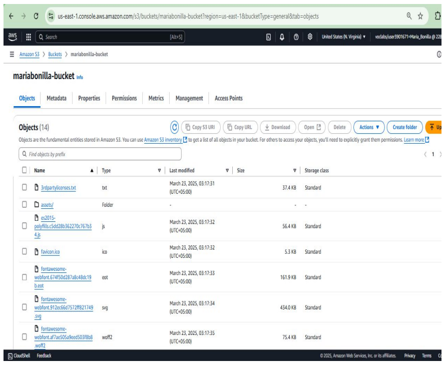
What is the value of using Docker Compose?

- Docker Compose is a tool for managing and linking multiple containers.
- It defines how the microservices that form the application should be deployed and interact.
- It enables easy scaling of services.

The view illustrated here is from Docker Desktop on Windows, specifically showing Docker Compose. Each end node in this tree repres iner using the MEAN stack (MongoDB, Angular Front End, and Node.js/Express.js).

Docker Compose is an orchestration tool that simplifies the management of application containers. By creating a configuration file, you can control deployment, define interactions between microservices, and scale your application based on demand, streamlining the development and deployment processes.

The Serverless Cloud



us-east-1.console.aws.amazon.com/s3/buckets/mariabonilla-bucket?region=us-east-1&bucketType=general&tab=objects

Search
[Alt+S]
United States (N. Virginia) ▾
voclabs/user3901671=Maria_Bonilla @ 228
Amazon S3 › Buckets › mariabonilla-bucket

mariabonilla-bucket Info

Objects (14)

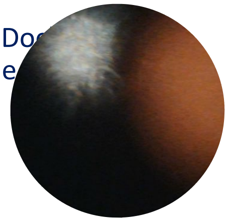
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type	Last modified	Size	Storage class
□	3rdpartylicenses.txt	txt	March 23, 2025, 03:17:31 (UTC+05:00)	37.4 KB	Standard
□	assets/	Folder	-	-	-
□	es2015-polyfills.c5dd28b362270c767b34.js	js	March 23, 2025, 03:17:32 (UTC+05:00)	56.4 KB	Standard
□	favicon.ico	ico	March 23, 2025, 03:17:32 (UTC+05:00)	5.3 KB	Standard
□	fontawesome-webfont.674f50d287a8c48dc19b.eot	eot	March 23, 2025, 03:17:33 (UTC+05:00)	161.9 KB	Standard
□	fontawesome-webfont.912ec66d7572ff821749.svg	svg	March 23, 2025, 03:17:34 (UTC+05:00)	434.0 KB	Standard
□	fontawesome-webfont.af7ae505a9eed503f8b8.woff2	woff2	March 23, 2025, 03:17:35 (UTC+05:00)	75.4 KB	Standard

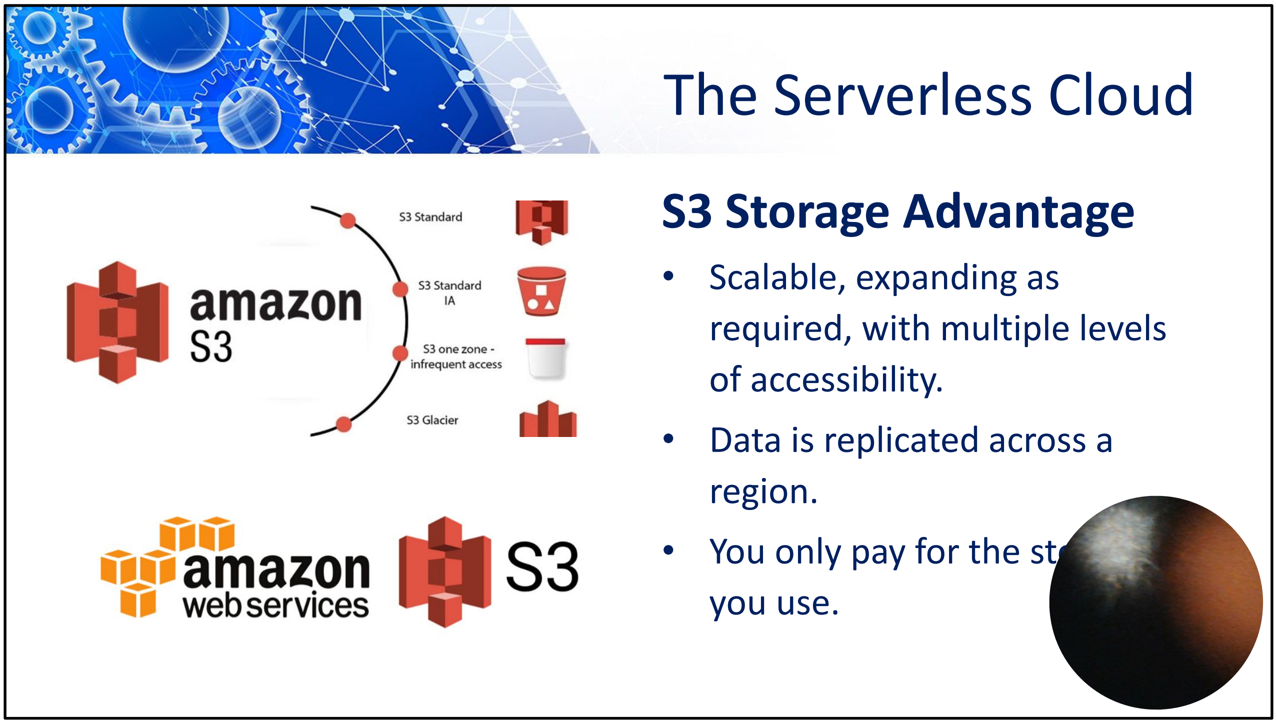
Serverless

- Applications and services operate on servers managed by the cloud provider.
- This eliminates the need for server maintenance, updates, and patching on your part.
- The system automatically scales to meet demand.

The view on the left shows the Do
Compose bucket, created on the
serverless.



Cloud and serverless architectures allow applications to run on servers managed by the cloud provider, which handles maintenance and updates. A significant advantage of serverless architecture is its automatic scaling according to the application's demand.



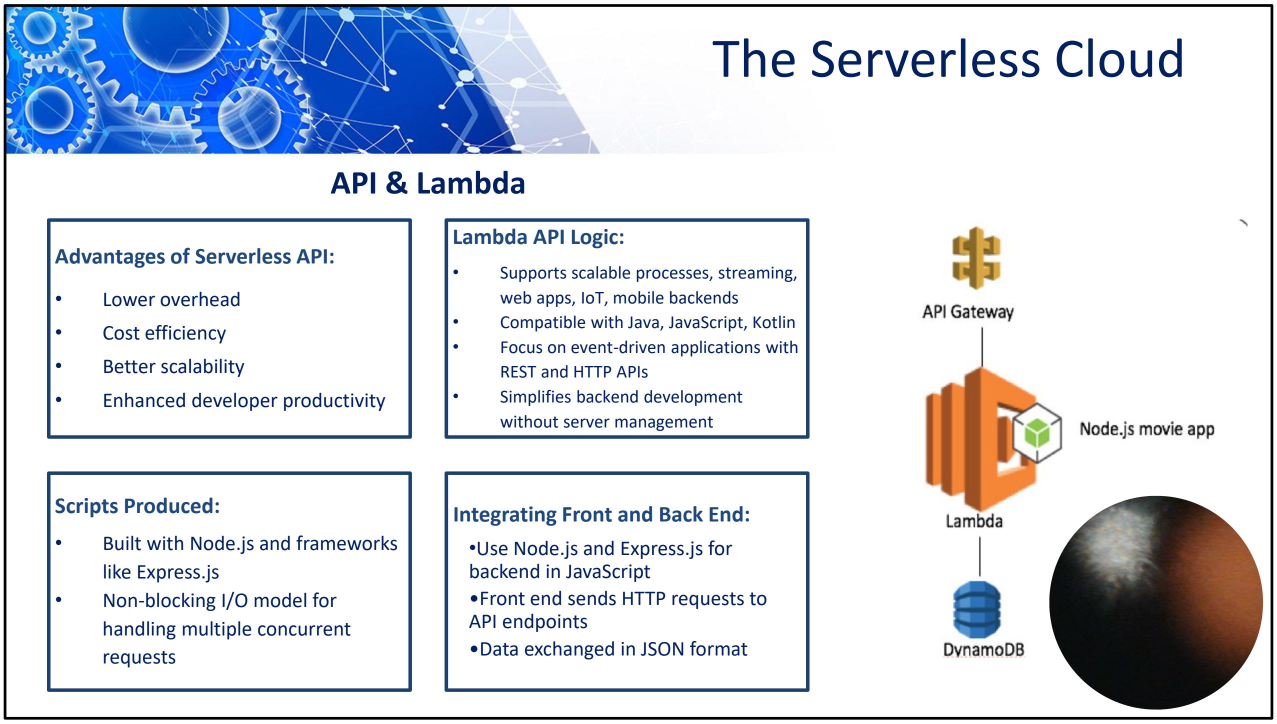
amazon S3
S3 Standard
S3 Standard IA
S3 one zone - infrequent access
S3 Glacier
amazon web services
S3

The Serverless Cloud

S3 Storage Advantage

- Scalable, expanding as required, with multiple levels of accessibility.
- Data is replicated across a region.
- You only pay for the st you use.

S3 is a scalable storage service offered by AWS that adjusts to your needs, unlike **local disk storage**, which is limited by physical space. **S3** automatically scales and replicates data across regions for high durability, while **local storage** confines data to the disk's physical location. With S3, you only pay for what you use, avoiding the upfront hardware costs and risks associated with local storage. Additionally, **S3** features Amazon **S3 Intelligent Tiering**, which automatically moves data to the most cost-effective storage tier based on access patterns, optimizing costs without affecting performance.



The Serverless Cloud

API & Lambda

Advantages of Serverless API:

- Lower overhead
- Cost efficiency
- Better scalability
- Enhanced developer productivity

Lambda API Logic:

- Supports scalable processes, streaming, web apps, IoT, mobile backends
- Compatible with Java, JavaScript, Kotlin
- Focus on event-driven applications with REST and HTTP APIs
- Simplifies backend development without server management

Scripts Produced:

- Built with Node.js and frameworks like Express.js
- Non-blocking I/O model for handling multiple concurrent requests

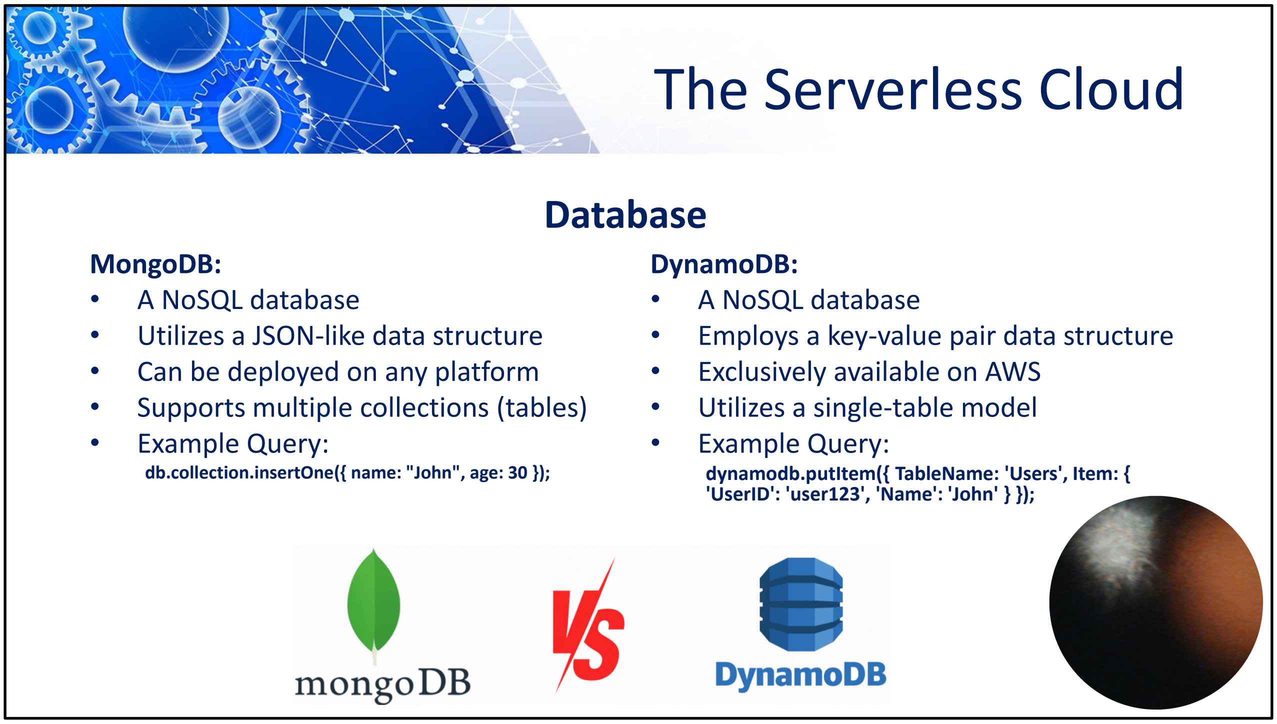
Integrating Front and Back End:

•Use Node.js and Express.js for backend in JavaScript

•Front end sends HTTP requests to API endpoints

•Data exchanged in JSON format

Amazon API Gateway and **AWS Lambda** are crucial components of a serverless architecture, as they provide scalable, event-driven services. **API Gateway** enables the creation and management of API endpoints for **CRUD** operations **(Create, Read, Update, Delete)** using **HTTP** methods such as GET, POST, PUT, and DELETE. These methods trigger corresponding **Lambda functions** that interact with the database. We also set up endpoints for data retrieval, with **Lambda functions** processing the necessary logic. **CORS (Cross-Origin Resource Sharing**) was enabled through the OPTIONS method to facilitate communication between the front-end and back-end. Additionally, **access policies** were established for each Lambda function to restrict data access to authorized users.



The Serverless Cloud

Database

MongoDB:

- A NoSQL database
- Utilizes a JSON-like data structure
- Can be deployed on any platform
- Supports multiple collections (tables)
- Example Query:

```
db.collection.insertOne({ name: "John", age: 30 });
```

DynamoDB:

- A NoSQL database
- Employs a key-value pair data structure
- Exclusively available on AWS
- Utilizes a single-table model
- Example Query:

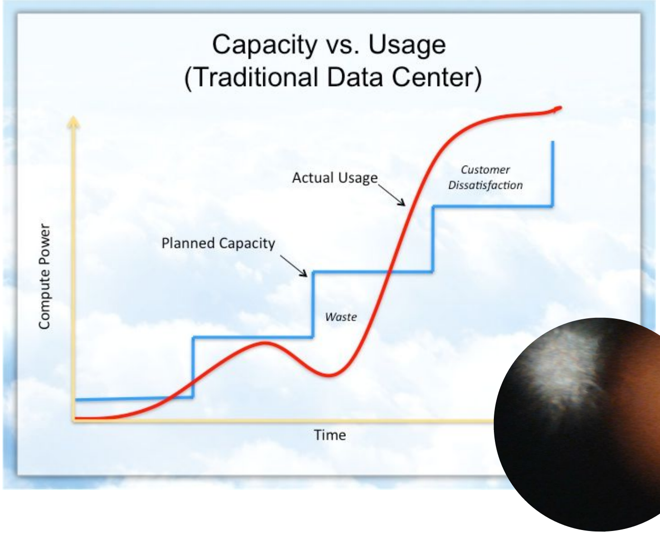
```
dynamodb.putItem({ TableName: 'Users', Item: {
'UserID': 'user123', 'Name': 'John' } });
```

MongoDB and **DynamoDB** are both NoSQL databases, but they have key differences—**MongoDB** stores data in flexible, JSON-like documents organized across multiple collections. In contrast, **DynamoDB** utilizes key-value pairs and is fully managed by Amazon Web Services (AWS). With **DynamoDB**, AWS takes care of updates and security, allowing you to avoid the management tasks required by **MongoDB**. Both databases are highly scalable, making them suitable for growing data needs. For this application, we implemented AWS **Lambda** functions to execute queries for finding, creating, updating, and deleting records triggered by **API endpoints** through Amazon **API Gateway.**

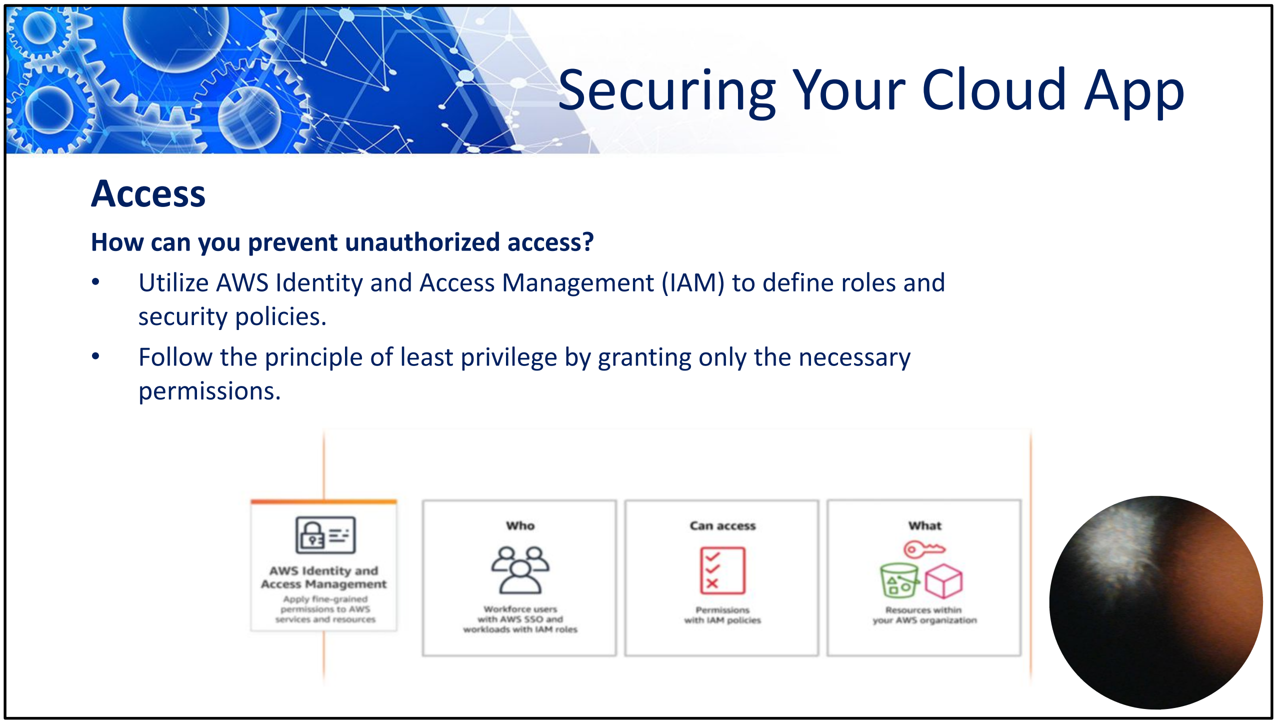
Cloud-Based Development Principles

Elasticity: The capability to scale resources up or down according to current demands.

Pay-for-use model: You only pay for the resources you actually use.



Elasticity in the cloud refers to the automatic scaling of resources in response to an application's current needs. Unlike traditional data centers, where you must predict your resource requirements and invest in infrastructure upfront, cloud computing operates on a **pay-as-you-go mode**l. This means you only pay for what you use, such as storage or function runtime. This approach helps eliminate wasted spending on unnecessary capacity and prevents performance issues from insufficient resources.



Securing Your Cloud App

Access

How can you prevent unauthorized access?

- Utilize AWS Identity and Access Management (IAM) to define roles and security policies.
- Follow the principle of least privilege by granting only the necessary permissions.

AWS Identity and Access Management

Apply fine-grained permissions to AWS services and resources

Who

Workforce users with AWS SSO and workloads with IAM roles

Can access

Permissions with IAM policies

What

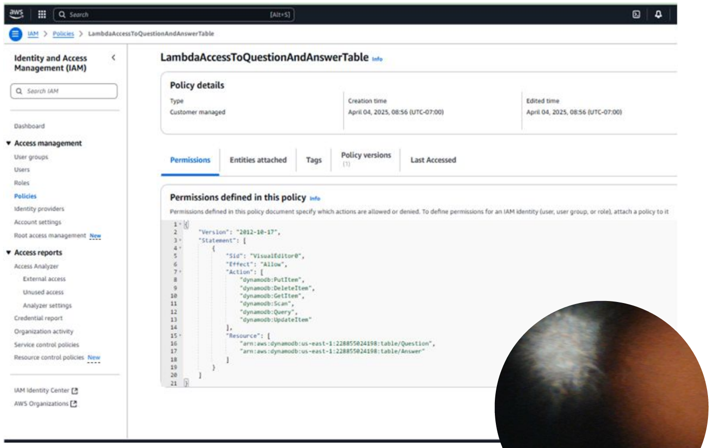
Resources within your AWS organization

Create **AWS IAM roles** and policies to protect your application from unauthorized access. Implementing these policies helps adhere to the **principle of least privilege**, granting users access only to necessary resources.

Securing Your Cloud App

Policies:

- Each role is associated with at least one policy, but it can have multiple policies.
- Users are assigned roles according to the level of access they require.



LambdaAccessToQuestionAndAnswerTable

Policy details

Type: Customer managed

Creation time: April 04, 2025, 08:56 (UTC-07:00)

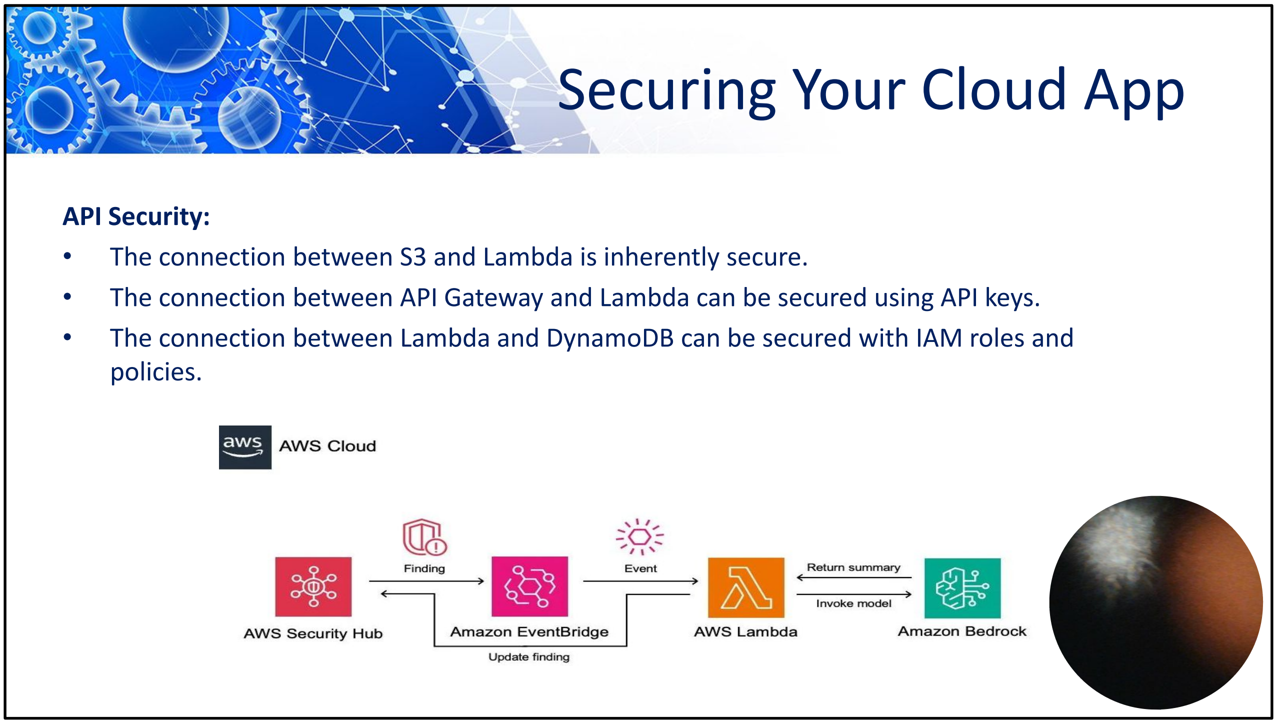
Edited time: April 04, 2025, 08:56 (UTC-07:00)

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:228855024198:table/Question",
                "arn:aws:dynamodb:us-east-1:228855024198:table/Answer"
            ]
        }
    ]
}
```

To manage access efficiently, a **role** is created to assign multiple access policies. Users are assigned roles based on the access they need to resources. Updating a policy for a group of users requires modifying the shared role, streamlining access management, and ensuring consistency across all users. For example, a custom policy named **LambdaAccessToQuestionAndAnswerTable** grants Lambda functions permissions to read, write, update, and delete data from a **DynamoDB table**.



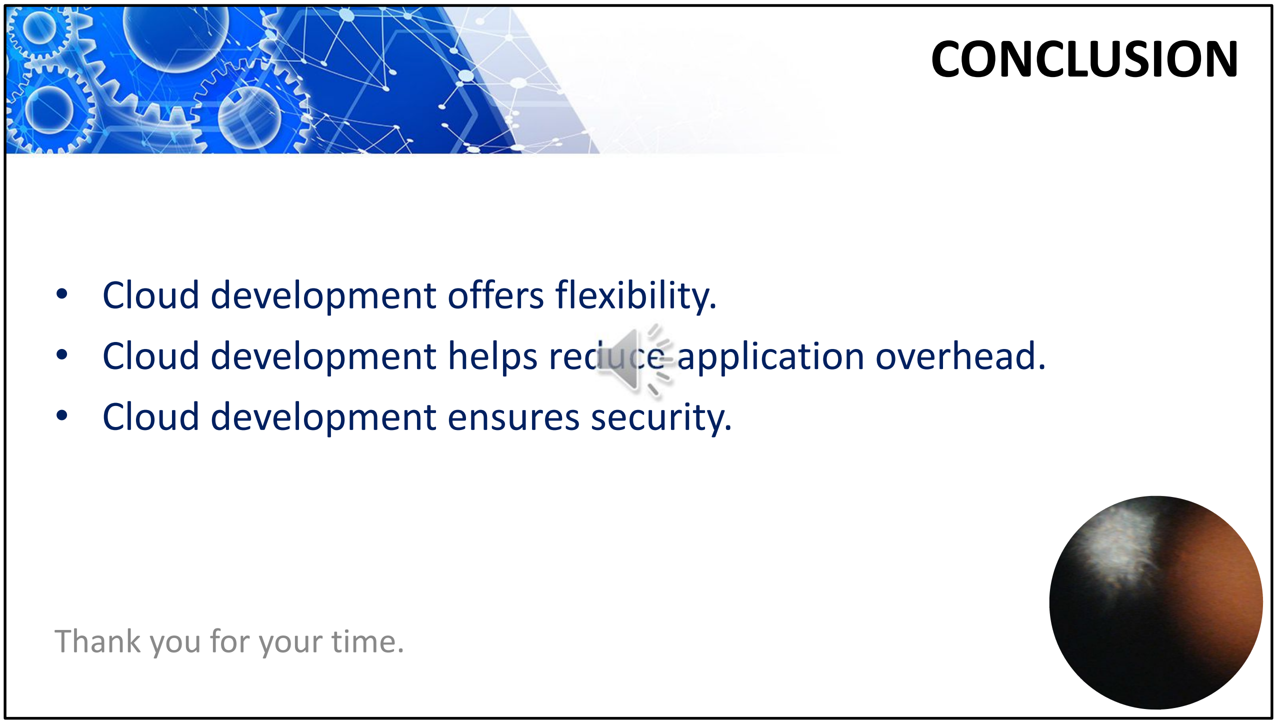
Securing Your Cloud App

API Security:

- The connection between S3 and Lambda is inherently secure.
- The connection between API Gateway and Lambda can be secured using API keys.
- The connection between Lambda and DynamoDB can be secured with IAM roles and policies.

aws
AWS Cloud
Finding
Event
Return summary
Invoke model
AWS Security Hub
Amazon EventBridge
AWS Lambda
Amazon Bedrock
Update finding

The S3 bucket must be public for website visibility. **AWS secures** the connection between **S3 and Lambda**, while API keys enhance security between **API Gateway and Lambda**, allowing only authorized users to access endpoints. A custom **role and policy** secure the connection between **Lambda and DynamoDB**, restricting database access to specific Lambda functions.



CONCLUSION

- Cloud development offers flexibility.
- Cloud development helps reduce application overhead.
- Cloud development ensures security.

Thank you for your time.

In conclusion, cloud development offers exceptional flexibility and scalability with a pay-as-you-go model, reducing overhead costs by eliminating significant upfront infrastructure investments. Serverless architecture minimizes hardware management, further decreasing operational expenses. Additionally, cloud development enhances security as the provider manages infrastructure and applies the latest security updates. Well-defined security policies also help protect your application in the cloud.