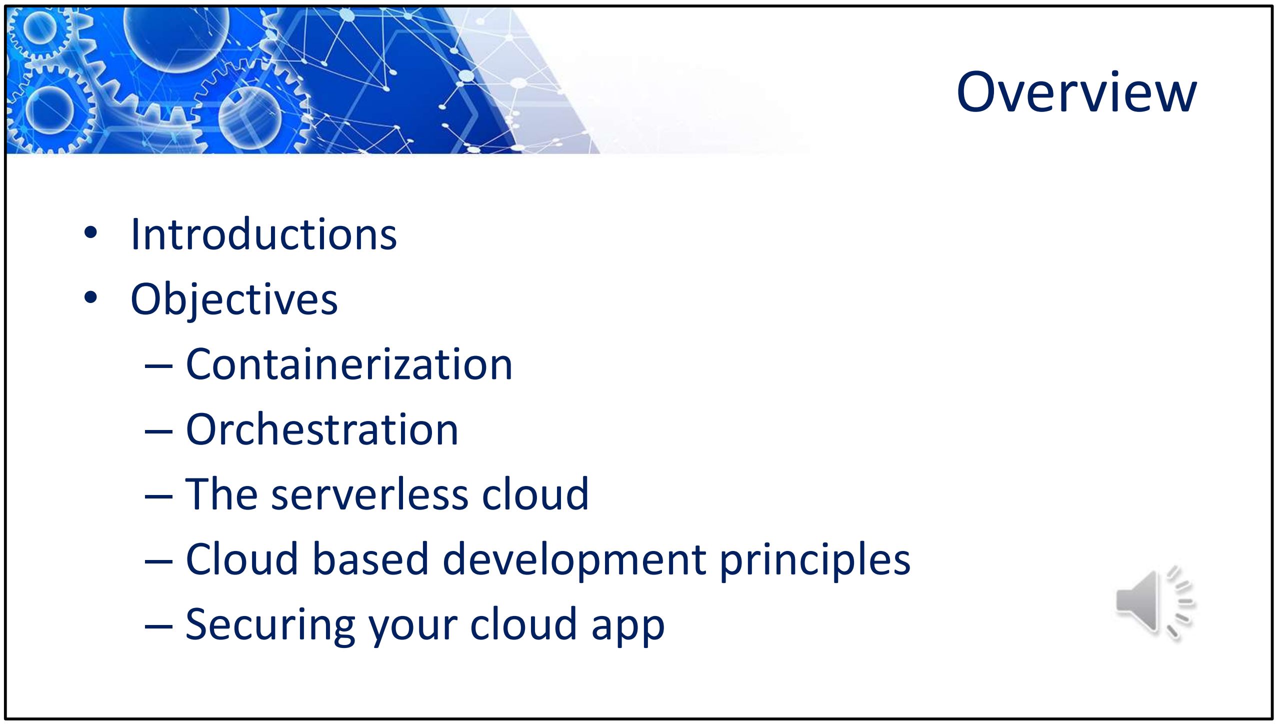
CS 470 Project Two Conference Presentation: Cloud Development

Shawn W. Way

December 2024

Project Two Conference Presentation: Cloud Development
Shawn Way
December 2024
Link: https://youtu.be/32WZrY0QV2k

Welcome to my conference presentation on cloud development.



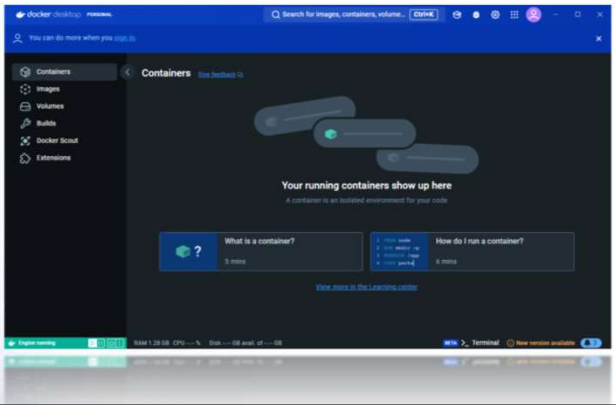
Overview

- Introductions
- Objectives
  - Containerization
  - Orchestration
  - The serverless cloud
  - Cloud based development principles
  - Securing your cloud app

I’m Shawn Way. I’m working towards a Bachelors of Computer Science with concentration in software engineering. I’m here today to talk about cloud development and the involved technologies. We will be covering the objectives you see there on the slide. Let’s get into it.

Containerization

- Virtual Machine (VM) vs Containerization
- Docker for Containerization



Your running containers show up here

A container is an isolated environment for your code

What is a container?

5 mins

How do I run a container?

6 mins

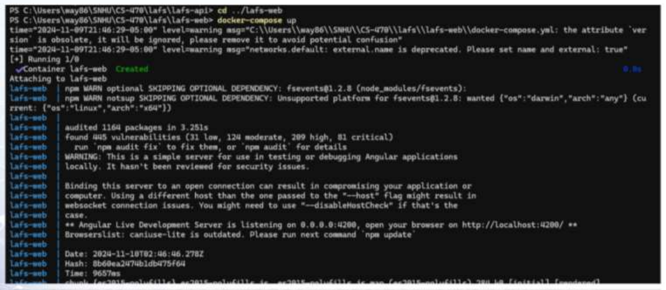
View more in the Learning center

Engine running

There are a couple main options when it comes to how applications are hosted on the cloud. We can use virtual machines, or VMs for short. These are full operating systems that only exist on the server. The other option is the use of containers, or containerization. Containers break up an application into self-contained units that can run on their own without needing to worry about dependencies, or software that is used to build the application. This is the main benefit of using containers over VMs. That isolated nature increases the flexibility and portability of the application and its individual parts. Docker is one of the most well-known tools for containerization. This is a free application that allows you to create and manage containers for your applications. Pictured here is the Docker Desktop: an application that provides a Graphical User Interface, or GUI, for container management.

Orchestration

- What is the value of using Docker Compose?



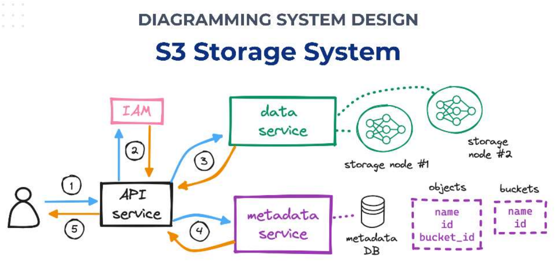
```
PS C:\Users\way86\SNHU\CS-470\lafs\lafs-api> cd ../lafs-web
PS C:\Users\way86\SNHU\CS-470\lafs\lafs-web> docker-compose up
time="2024-11-09T21:46:29-05:00" level=warning msg="C:\\Users\\way86\\SNHU\\CS-470\\lafs\\lafs-web\\docker-compose.yml: the attribute `ver
sion` is obsolete, it will be ignored, please remove it to avoid potential confusion"
time="2024-11-09T21:46:29-05:00" level=warning msg="networks.default: external.name is deprecated. Please set name and external: true"
[+] Running 1/0
 ✓Container lafs-web  Created                                                                                              0.0s
Attaching to lafs-web
lafs-web  | npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.8 (node_modules/fsevents):
lafs-web  | npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.8: wanted {"os":"darwin","arch":"any"} (cu
rrent: {"os":"linux","arch":"x64"})
lafs-web  |
lafs-web  | audited 1164 packages in 3.251s
lafs-web  | found 445 vulnerabilities (31 low, 124 moderate, 209 high, 81 critical)
lafs-web  |   run `npm audit fix` to fix them, or `npm audit` for details
lafs-web  | WARNING: This is a simple server for use in testing or debugging Angular applications
lafs-web  | locally. It hasn't been reviewed for security issues.
lafs-web  |
lafs-web  | Binding this server to an open connection can result in compromising your application or
lafs-web  | computer. Using a different host than the one passed to the "--host" flag might result in
lafs-web  | websocket connection issues. You might need to use "--disableHostCheck" if that's the
lafs-web  | case.
lafs-web  | ** Angular Live Development Server is listening on 0.0.0.0:4200, open your browser on http://localhost:4200/ **
lafs-web  | Browserslist: caniuse-lite is outdated. Please run next command `npm update`
lafs-web  |
lafs-web  | Date: 2024-11-10T02:46:46.278Z
lafs-web  | Hash: 8b60ea2474b1db475f64
lafs-web  | Time: 9657ms
```

Docker Compose is an application for container orchestration. It is made by Docker and is best used in a development environment. A multi-container application can be configured using only a single YAML file. Also, all containers can be started with only a single command. This can save a lot of time in development and testing when the application is constantly changing.

The Serverless Cloud

Serverless

- Serverless and its advantages
- S3 storage vs local storage

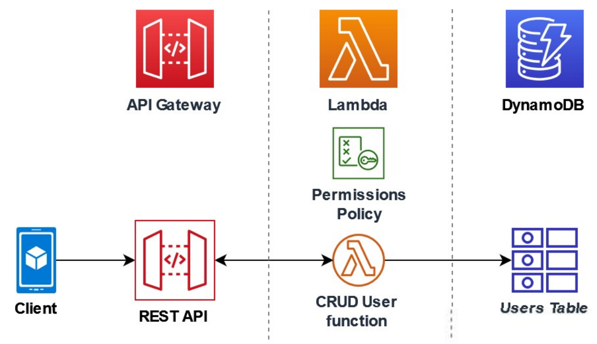


When an application is “serverless”, it means that the details of the server have been abstracted away to the cloud. There are still physical servers that house and run the data necessary for the application, but all the details have been removed from the concern of the application. This has the advantage of making development easier, and it allows developers to focus on the features of the application instead of server implementation details. S3 storage is a service provided by AWS for cloud storage. It is based on an object model, and objects are stored within buckets. These objects and buckets are similar to files and directories on a computer. Speaking of the differences between S3 and local storage, it is very difficult to size the amount of storage that one will need to manage an application for its life. This can lead to purchasing too much storage and underutilizing it or not purchasing enough storage and suffering when the application needs to scale. Local storage also must be maintained. This requires people and equipment, which can be costly. S3 storage does all of this for you.

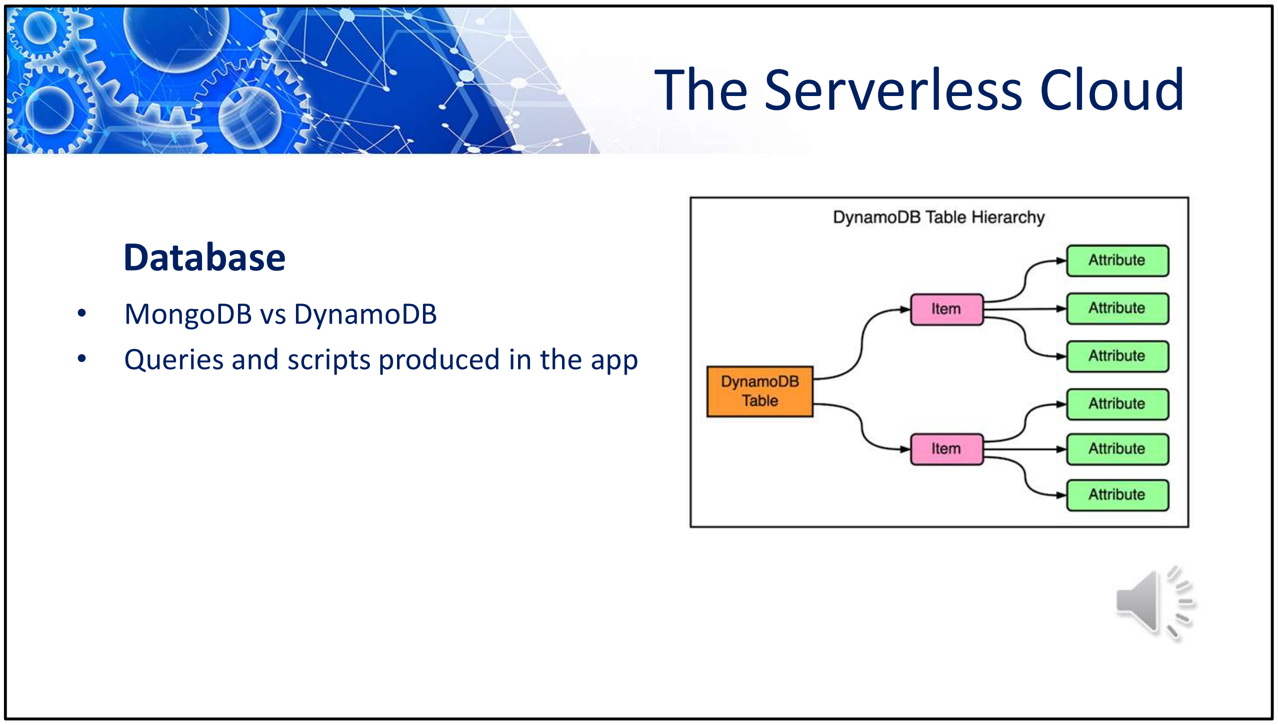
The Serverless Cloud

API & Lambda

- Advantages of using a serverless API
- Lambda API Logic and scripts
- Connecting the front to the back end



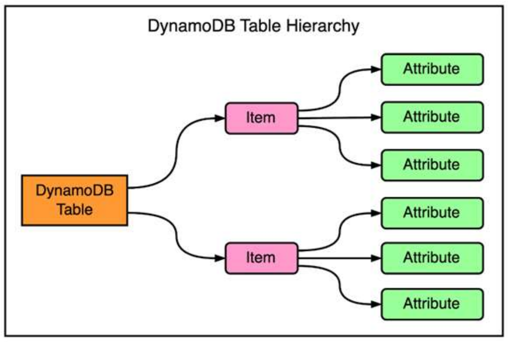
A serverless API helps to abstract much of the implementation details of the server away from the development team. This can speed up development and help the team to focus on the features that they want to ship. Also, a serverless API is a service that is managed by an external vendor. This means that they can make resources available as traffic to the site and API calls are made. This makes it simple to scale the application while only using exactly what you need. Lambda is the service that allows you to build the functions that your application will use. These scripts are also managed by Amazon, and they are used only when necessary. In our application, Lambda was used to get the data that the user would need in whatever form that was necessary. The front end displayed the application while the back end stored the data for questions and answers. The Lambda scripts held the functions that would query the database and pass data back and forth. API Gateway provided the functionality to host the API so that these calls could be made.



The Serverless Cloud

Database

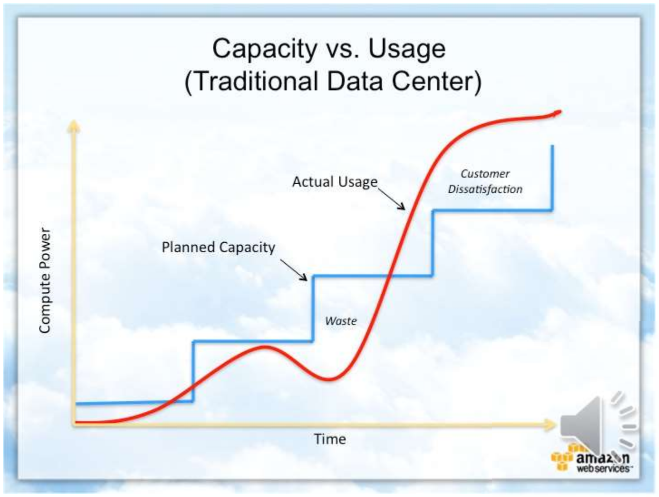
- MongoDB vs DynamoDB
- Queries and scripts produced in the app



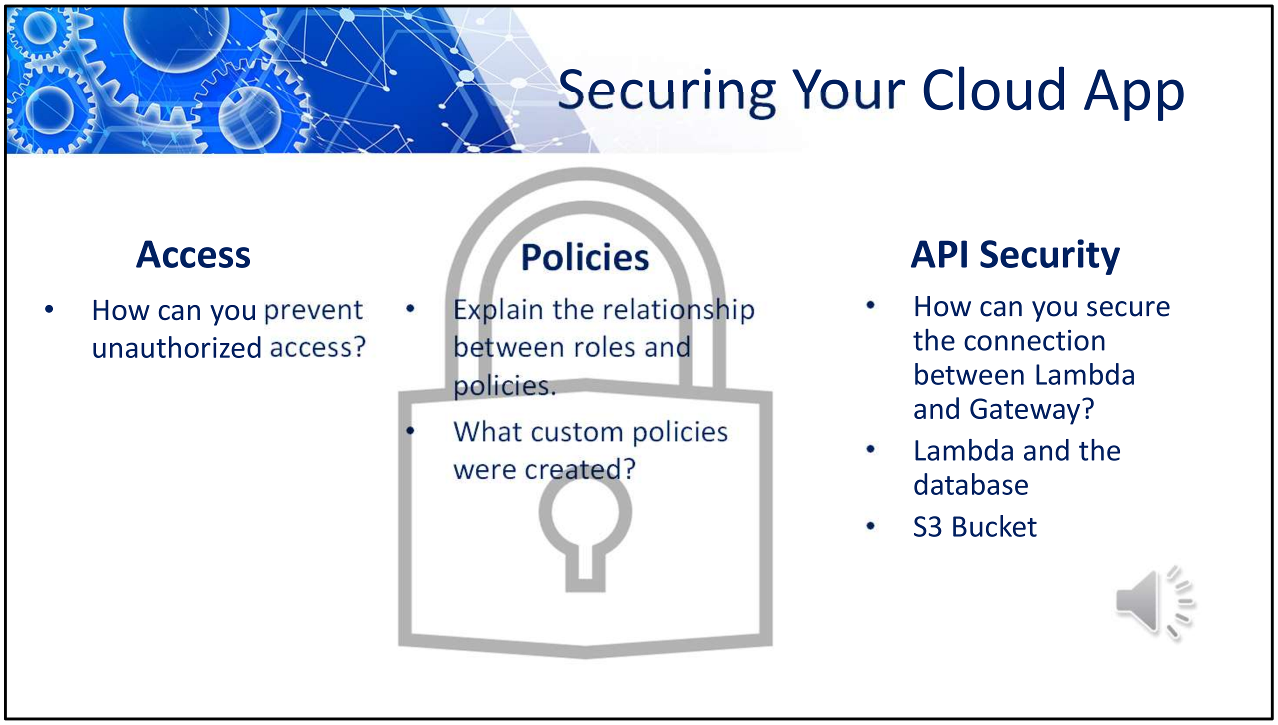
MongoDB and DynamoDB are NoSQL databases. DynamoDB proves storage as a service offered by AWS. MongoDB is an open-source database that the user must scale and maintain. DynamoDB organizes data into tables and is thus less flexible with the number of datatypes that can be stored. MongoDB is more flexible in terms of what can be stored, but it is not as performant as a result of this flexibility. In our application, we used queries to show the questions and answers on the site. There were queries to pull the entire table, just a single entry, and to update, create or delete an item. This allowed questions and answers to be added, changed, and deleted. There was also a system of rating questions and answers that was recorded with the update function.

Cloud-Based Development Principles

- Elasticity
- Pay-for-use model



One of the greatest advantages of using AWS is the elastic pricing model. You are only charged for the resources you use. With traditional hardware, you would have to add capacity to meet demand. This rarely follows the true demand closely and can lead to customer dissatisfaction, as shown in the graph. On the other side, if you have more capacity than you need you will end up wasting resources. The elasticity of AWS alleviates this issue and makes scaling much simpler and more cost effective.



Securing Your Cloud App

Access

- How can you prevent unauthorized access?

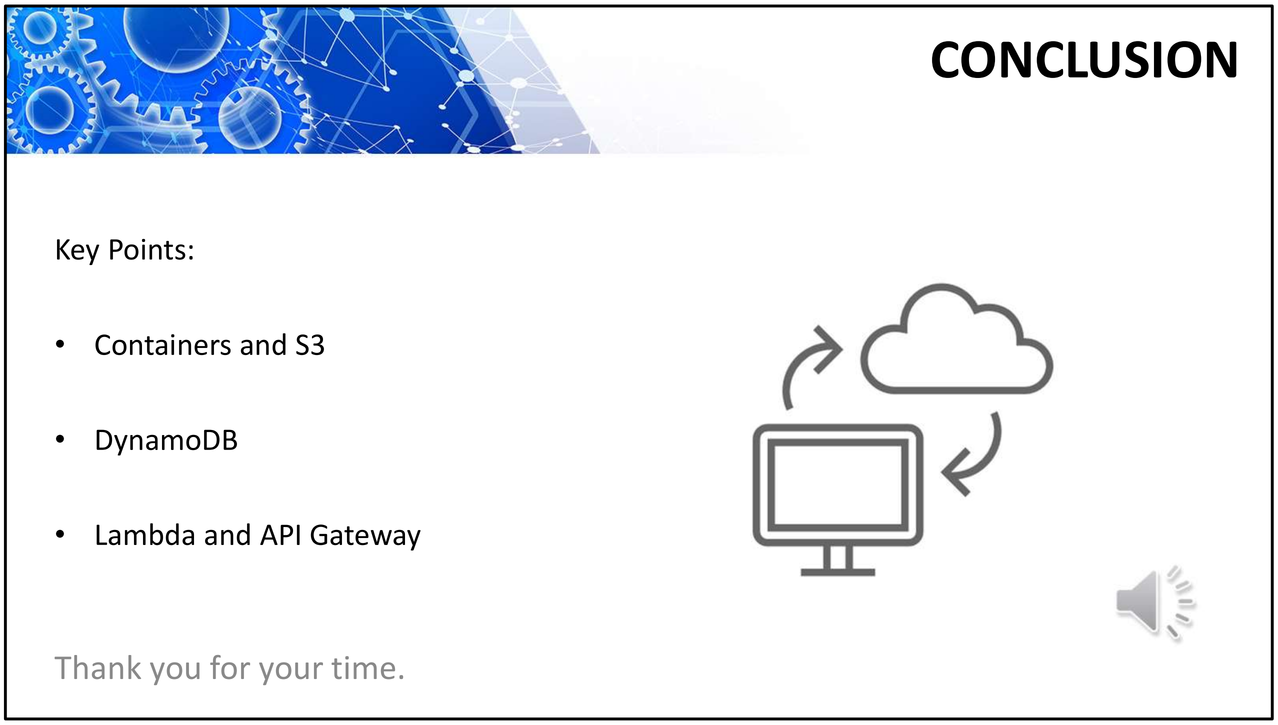
Policies

- Explain the relationship between roles and policies.
- What custom policies were created?

API Security

- How can you secure the connection between Lambda and Gateway?
- Lambda and the database
- S3 Bucket

AWS also provides a lot of security features to make sure that sensitive and confidential data does not make it into the wrong hands. AWS provides authentication and authorization functionality in the form of accounts. These accounts can be created, removed, and monitored for suspicious activity. Policies can be made to allow access to be allocated to the features that you wish to provide. Roles are similar to “jobs” or groups that can be defined. These roles are then given the polices they need to have the functionality they require. Using the roles and policies effectively guarantees that the principle of least privilege is followed. For our application, we had a custom role defined for the LabRole. This role had the necessary policies to define and link the Lambda functions, use API gateway, S3, and DynamoDB. API security is provided by letting you handle how the data is accessed, where it can be accessed from, and having policies defined around these functions. Each layer of the application (API, Lambda, and S3 buckets) were abstracted away from the user. Each of these had policies to ensure that data was only used by those who were authorized to do so, and that this data was handled in only the way that was allowed.



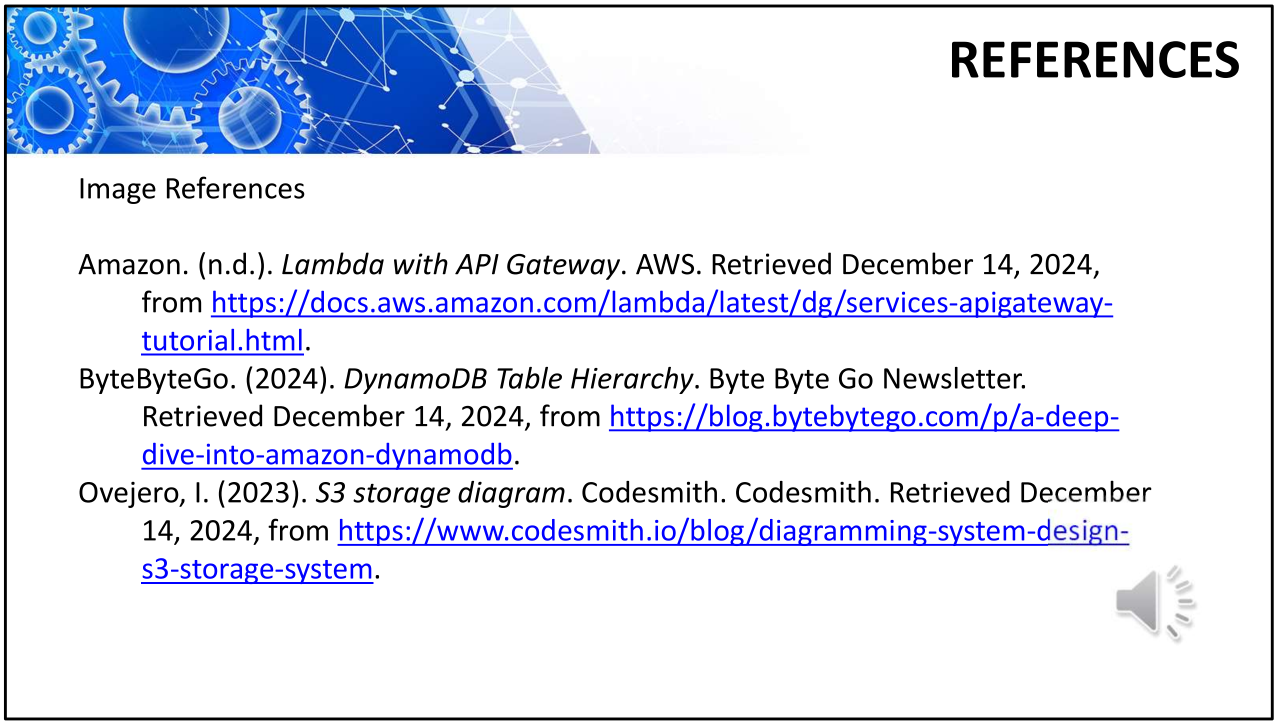
CONCLUSION

Key Points:

- Containers and S3
- DynamoDB
- Lambda and API Gateway

Thank you for your time.

The key points behind cloud development were those you see here. Containers are used to hold the individual units of the program in modular, flexible units. These containers are then stored in S3 as objects in a bucket. S3 backs up these objects to ensure that version changes are archived, and so that data is less likely to be lost. The front end of the application is hosted from these containers in the S3. The back end of the application uses DynamoDB to hold the data that will be displayed on the front end. This data is stored in tables that have items with attributes to store all necessary information. Lambda and API gateway are used to query this database and serve the data to the front end for display. All of these functions are used dynamically and only when they are required. This allows the application to scale to any size, and it saves money by ensuring that you only have to pay for the resources that you actually use. This has been my presentation on cloud infrastructure and development. Thank you for your time.



REFERENCES

Image References

Amazon. (n.d.). *Lambda with API Gateway*. AWS. Retrieved December 14, 2024, from https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway-tutorial.html.

ByteByteGo. (2024). *DynamoDB Table Hierarchy*. Byte Byte Go Newsletter. Retrieved December 14, 2024, from https://blog.bytebytego.com/p/a-deep-dive-into-amazon-dynamodb.

Ovejero, I. (2023). *S3 storage diagram*. Codesmith. Codesmith. Retrieved December 14, 2024, from https://www.codesmith.io/blog/diagramming-system-design-s3-storage-system.

Here are the sources for the images that you saw in this application.