
The background of the slide features a blue gradient with a network of white lines and dots. On the left, a hand is shown reaching towards a cluster of glowing, translucent gears. Another hand is shown at the bottom, palm up, as if presenting or supporting the gears. The overall theme is technology, engineering, and cloud development.

# CS 470 Project Two Conference Presentation: Cloud Development

Ethan Harrell  
[August/2023]



# Overview

- Hello class, my name is Ethan Harrell and I am 22 years old and this is my last term here at SNHU. I am graduating with a Bachelors degree in Computer Science!
- The purpose of this presentation  is to effectively communicate and explain the complexities and nuances of cloud development to a diverse audience, encompassing both technical and non-technical individuals. Throughout this presentation I will provide a comprehensive overview that balances technical depth with accessible language, I also intend to empower our audience with a clear understanding of how cloud development works, its benefits, challenges, and its pivotal role in shaping the digital world we inhabit today.

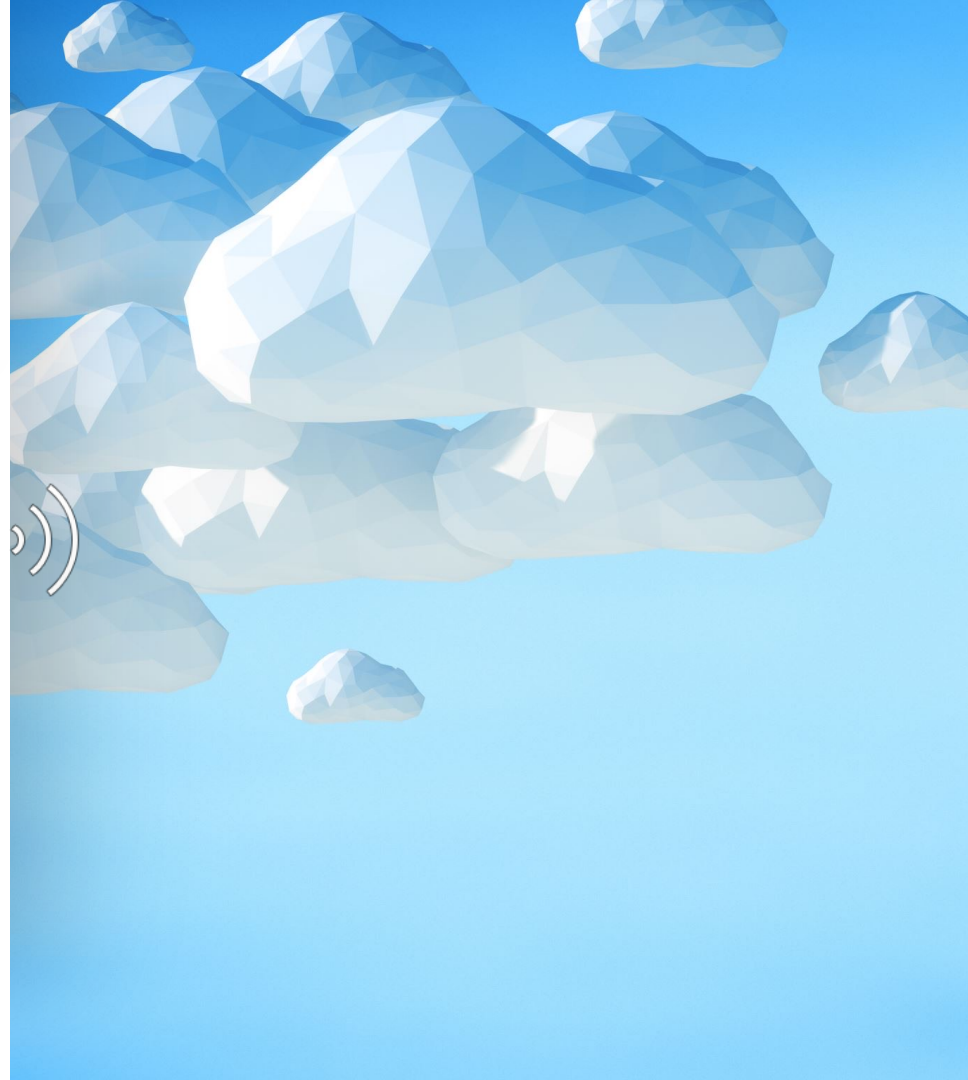
# Containerization

Migrating a full stack application to the cloud involves several models and strategies that ensure a smooth transition while maximizing the benefits of cloud computing. Here are some common models used for migrating full stack applications to the cloud:

- **Lift and Shift (Rehosting)**
- **Replatforming**
- **Refactoring (Re-architecting)**
- **Rearchitecting**
- **Rebuilding**
- **Hybrid Approach**

Containerization involves packaging an application and its dependencies into a standardized unit called a container, which can then be deployed consistently across different environments. Several tools are essential for effectively implementing containerization:

- **Docker**
- **Container Registry**



# Orchestration

Docker Compose is a valuable tool for simplifying the development, deployment, and management of multi-container applications. It offers several benefits that make it an attractive choice for orchestrating containerized applications:

- **Simplified Configuration**
- **Multi-Container Management**
- **Easy Development Environment**
- **Simplified Deployment**
- **Dependency Management**
- **Scaling**



# The Serverless Cloud

---

In a serverless architecture, developers focus on writing code to implement the business logic of their applications, while the cloud provider handles the provisioning, scaling, and management of the servers and resources needed to execute that code.

Here are some key advantages of the serverless architecture:

- **Reduced Operational Overhead**
- **Fast Time-to-Market**
- **Automatic High Availability**
- **Event-Driven Architecture**
- **Microbilling**



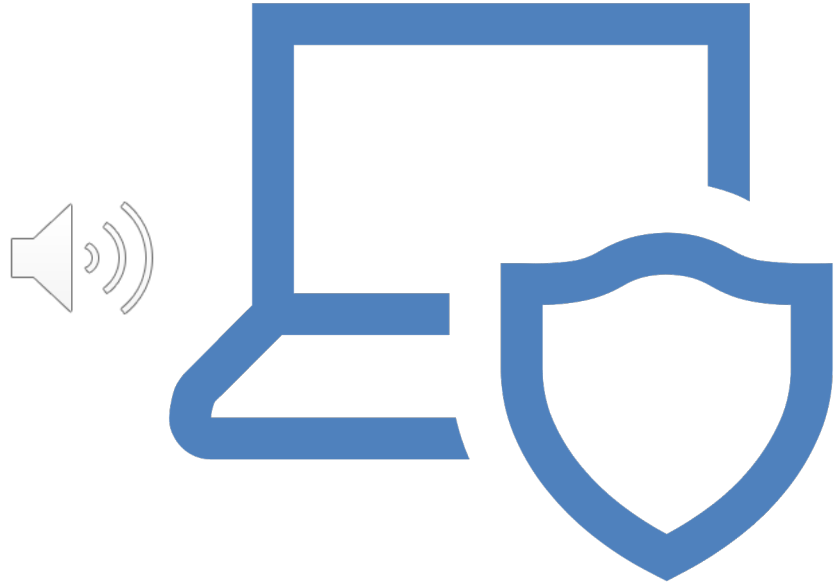
Amazon S3 (Simple Storage Service) is a scalable, cloud-based object storage service provided by Amazon Web Services (AWS). It's designed to store and retrieve large amounts of data, such as documents, images, videos, backups, and more. S3 is widely used for various purposes, including data storage, content distribution, data backup and archiving, and serving static website content.

Here's a comparison between Amazon S3 storage and local storage:

- **Amazon S3 Storage:**
- **Scalability**
- **Durability and Redundancy**
- **Accessibility**
- **Data Management**
- **Security**
- **Cost**
- **Consistency**

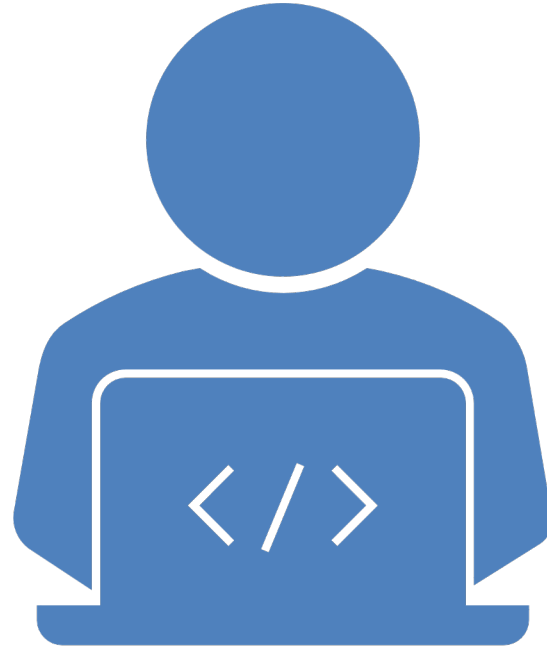
# Advantages of Using a Serverless API:

- Scalability
- Cost-Efficiency
- Reduced Management Overhead
- Automatic High Availability



## Lambda API Logic:

- AWS Lambda is a serverless compute service that allows you to run code without provisioning or managing servers. Lambda functions can be used to implement the logic of your serverless API. Each Lambda function represents a specific API endpoint or action.

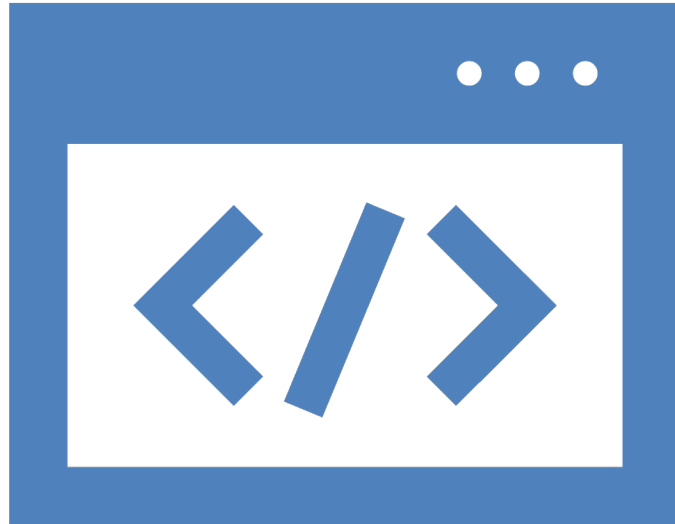




# Scripts Produced for Lambda API Logic:

For creating a serverless API with AWS Lambda, you generally write your function logic using a supported programming language, package it into a deployment package (ZIP file), and then upload it to AWS Lambda. The specific steps depend on the programming language you're using and the tooling you prefer. Commonly, the following scripts or files are involved:

- **Function Code**
- **Dependencies**
- **Deployment Configuration**





# Integrating Frontend with Backend (Serverless API):.

- **Authentication and Authorization**
  - API Gateway
  - Testing
  - Deployment
  - Usage and Monitoring
  - Continuous Integration/Continuous Deployment (CI/CD)
- Integrating the frontend with a serverless API involves creating a smooth connection between the user interface and the backend functions, enabling seamless communication and interaction between the two components.



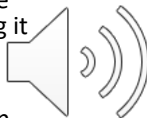
# The Serverless Cloud

## Data-Model Differences Between MongoDB and DynamoDB:

MongoDB and Amazon DynamoDB are both NoSQL databases, but they have different data models and use cases.

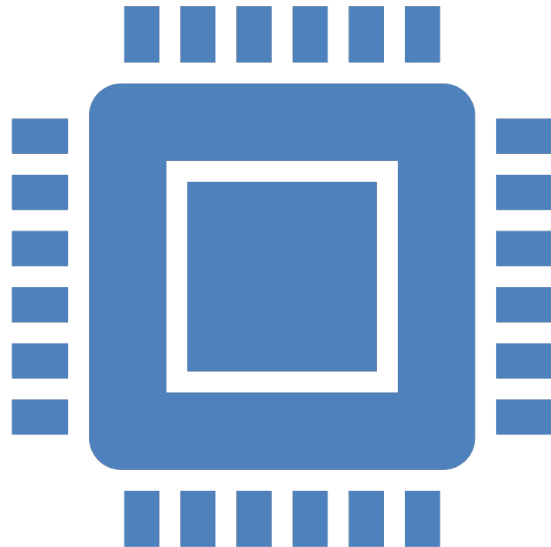
### MongoDB:

- **Document-Based:** MongoDB stores data in flexible, JSON-like documents. Each document can have different fields, making it suitable for dynamic and complex data structures.
- **Rich Query Language:** MongoDB provides a powerful query language that supports rich filtering, sorting, and aggregation operations.
- **Indexing:** MongoDB supports various indexing options to optimize query performance.

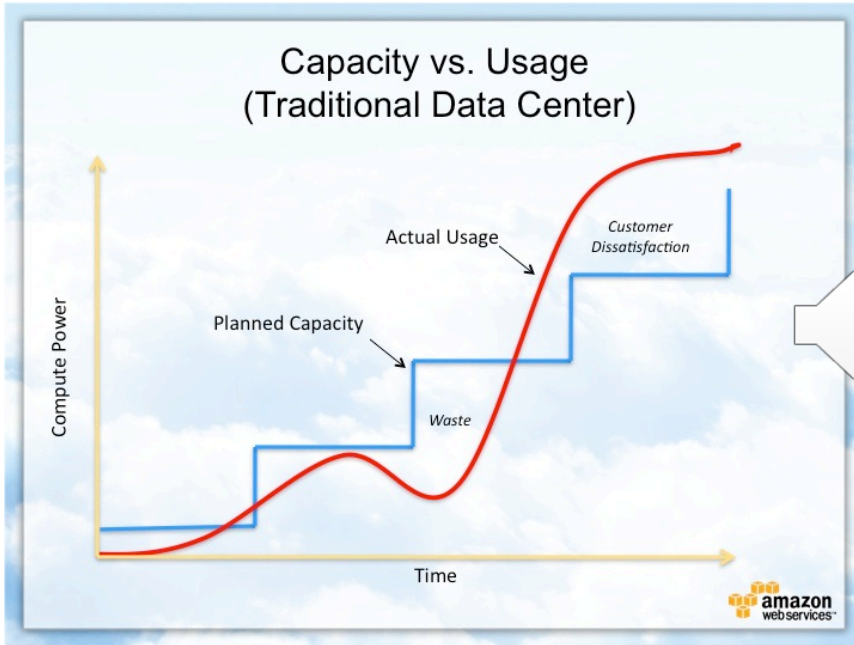


### DynamoDB:

- **Key-Value and Document Hybrid:** DynamoDB is primarily a key-value store, but it supports document-like structures within its values.
- **Limited Querying:** Querying in DynamoDB is based on primary key and secondary indexes. While it supports filtering, it doesn't provide the same rich querying capabilities as MongoDB.
- **Managed Service:** DynamoDB is a fully managed service provided by AWS, offering automatic scaling and high availability.



# Cloud-Based Development Principles



- **Elasticity:**
- Elasticity in the context of cloud computing refers to the ability of a system or application to dynamically scale its resources up or down based on demand. It allows your infrastructure to handle varying workloads without manual intervention, ensuring optimal performance during periods of high activity and cost savings during periods of low activity. Elasticity is made possible by cloud providers' ability to rapidly provision or de-provision resources in response to changing demands.
- 
- **Pay-for-Use Model:**
- The pay-for-use model, also known as consumption-based or utility-based pricing, is a billing approach where you're charged based on the resources you consume. This model contrasts with traditional upfront purchasing of hardware and software licenses, where you pay for capacity regardless of how much you actually use.

# Securing Your Cloud App

## Access

- To prevent unauthorized access to your cloud application use these three things:
  - **Authentication**
  - **Authorization**
  - **Encryption**



Optional: Break this into three separate slides.

# Roles:

- In the context of cloud security, roles and policies are key components of access management that work together to control and regulate user access to resources within a cloud environment.
- A role defines a set of permissions and access rights that can be granted to entities like users, applications, or services. There is more to roles than just this.



# Policies:

- Policies are attached to identities, such as users, groups, or roles, to control their access to various resources. There is more to policies than just this.
- **Relationship between Roles and Policies:**
- Roles and policies work together to grant specific access rights to users, applications, or services while maintaining security and compliance.



# Custom Policies:

- Custom policies are policies that you create to define specific permissions for your organization's requirements. They can be tailored to grant permissions beyond the built-in policies provided by cloud service providers. For example, in AWS IAM, you might create custom policies to grant access to specific resources or actions that aren't covered by existing policies.



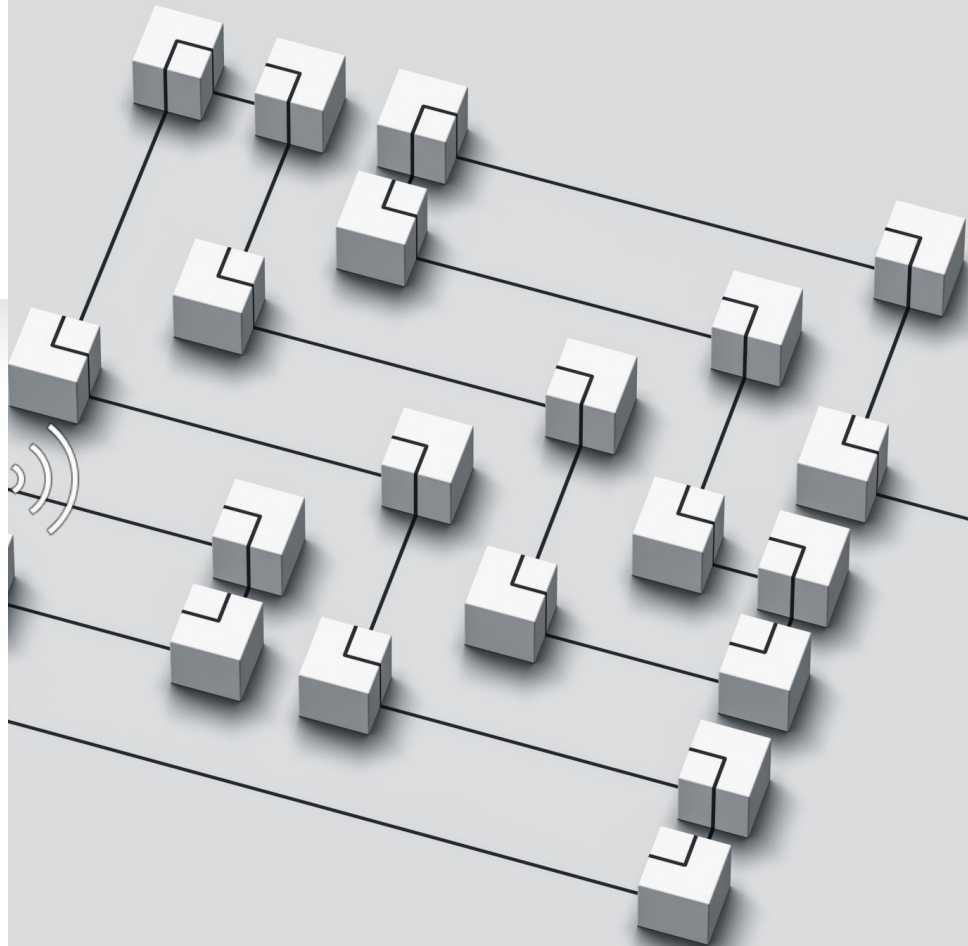


# API Security

- Securing the connections between AWS Lambda and API Gateway, Lambda and databases, and Lambda and S3 buckets involves implementing appropriate security measures to protect data and ensure secure communication. Here's how you can secure these connections:


## Lambda and API Gateway:

- **Encryption:** Configure HTTPS (TLS) for API Gateway to ensure encrypted communication between clients and the API. API Gateway provides SSL certificates by default.
- **Input Validation:** Implement proper input validation and sanitization in both the API Gateway and Lambda to prevent injection attacks and malicious inputs.
- **Resource Policies:** Set resource policies to restrict access to specific API endpoints based on IP ranges, AWS accounts, or other conditions.





## Lambda and Databases:

- 
- **IAM Roles:** Use IAM roles to grant the Lambda function permissions to access the database. Avoid using static credentials or hardcoding credentials in your code.
  - **VPC and Subnet:** Place your Lambda function within a Virtual Private Cloud (VPC) and specific subnets. Use security groups and Network Access Control Lists (NACLs) to control inbound and outbound traffic.
  - **Database Security:** Implement security measures at the database level, such as using database user accounts with the least privilege necessary and enabling encryption for data at rest and in transit.



## API Security Wrap Up

- By applying these security practices, you can ensure that the connections between AWS Lambda, API Gateway, databases, and S3 buckets are properly secured, protecting your data and maintaining the integrity of your applications.

# CONCLUSION

Here are three main points that highlight my understanding of cloud development:

- Elastic Scalability:** Cloud development emphasizes the ability to scale resources dynamically based on demand. This ensures optimal performance during traffic spikes and cost efficiency during low activity, enhancing user experience and resource utilization.
- Security by Design:** Cloud development prioritizes security through features like authentication, authorization, encryption, and access controls. Implementing these measures safeguards data and prevents unauthorized access, ensuring compliance with privacy regulations.
- Serverless Architecture:** Cloud development embraces serverless architecture, where developers focus on code logic while cloud providers handle infrastructure management. This streamlines development, improves time-to-market, and enables cost-effective, event-driven applications.

These points underscore my understanding of key concepts in cloud development, showcasing the importance of scalability, security, and modern architectural approaches.