**Reflection on Lab Work: Building a Decoupled and Serverless Architecture with AWS**

**Implementing a Serverless Architecture on AWS**

**Breaking a Monolithic Node.js Application into Microservices**

In this lab, I explored two different ways to design a system using AWS services. The first approach, in Phase 1, involved tightly connecting the web server and the application server. In Phase 2, I changed the design to make the system loosely connected, or "decoupled," using Amazon SQS and Amazon SNS .

In Phase 1, the web server directly communicated with the application server whenever an image was uploaded.

In Phase 2, we made the web server and application server work independently.

Independent Components: The web server updated a DynamoDB table without needing to constantly communicate with the application server. The application server processed data only when needed.

In lab work  **Implementing a Serverless Architecture on AWS**

In this lab, I learned how to use AWS Lambda to create a serverless architecture, eliminating the need to manage and provision servers. With AWS Lambda, you only need to provide the code and a trigger. The function runs as needed, from once a week to hundreds of times per second, and you pay only for what you use.

The lab demonstrated the process of invoking a Lambda function when a file is uploaded to Amazon S3. The file's data is then loaded into a DynamoDB table, which can be viewed on a dashboard. This serverless solution scales automatically and incurs minimal costs when idle, only charging for data storage.

In lab work **Breaking a Monolithic Node.js Application into Microservices**

During this lab, I experienced firsthand the complexities and advantages of transitioning from a monolithic architecture to a microservices-based design. Traditional monolithic architectures, while simpler to begin with, present significant challenges as applications grow. The increasing codebase complexity makes updates and maintenance difficult, hindering the integration of new features and technologies. This ultimately limits innovation and responsiveness to changing requirements.

By adopting a microservices architecture, we decomposed the monolithic Node.js application into discrete, independently running services, each performing a single function

**Conclusion**

This lab showed me the clear benefits of a decoupled and serverless system. By using AWS services like SQS, SNS, and Lambda, I created a system that was more reliable, scalable, and easier to maintain than the tightly connected system in Phase 1. This experience taught me the importance of designing systems where parts can work independently to handle larger loads and recover from issues more smoothly, while also leveraging serverless technology to improve cost efficiency and scalability.

Also the lab work of module 14 was really useful because, overall, this lab showed me how microservices make applications easier to manage, more flexible, and more resilient. It also highlighted the importance of tools like Docker and Amazon ECS for deploying and managing these services.