Project Report on

Serverless Chat Application Using AWS Services

1. Introduction

The objective of this project was to design and implement a real-time chat application using serverless architecture with AWS CloudFormation, AWS Lambda, WebSocket API, and DynamoDB. The project aimed to showcase the capabilities of AWS services for building scalable and cost-effective chat applications.

2. Objectives

* Develop a real-time chat application using serverless architecture.
* Utilize AWS CloudFormation for infrastructure provisioning and management.
* Use AWS Lambda for serverless compute to handle chat functionality.
* Implement WebSocket API for real-time communication between clients and servers.
* Utilize Amazon DynamoDB for storing chat messages.

3. Architecture Overview

The architecture of the serverless chat application consisted of the following components:

* AWS Lambda: Functions responsible for handling chat operations such as sending and receiving messages.
* WebSocket API Gateway: Establishes real-time communication between clients and Lambda functions.
* Amazon DynamoDB: We’ll store client IDs in a DynamoDB table.

4. Implementation

4.1 Infrastructure Setup with AWS CloudFormation:

* Created AWS CloudFormation templates to define the infrastructure components required for the chat application.
* Provisioned resources including Lambda functions, WebSocket API Gateway, and DynamoDB tables using CloudFormation stacks.

4.2 Developing Lambda Functions:

* Developed Lambda functions using Node.js to implement chat functionality.
* Implemented functions for sending messages, retrieving messages, and managing connections.
* Integrated AWS SDK for Node.js to interact with DynamoDB for storing and retrieving chat messages.

4.3 Configuring WebSocket API Gateway:

* Configured WebSocket API Gateway to establish bidirectional communication channels between clients and servers.
* Defined WebSocket routes for sending and receiving messages.
* Configured API Gateway to trigger Lambda functions for message handling.

5. Deployment and Testing

* Deployed the serverless chat application using AWS CloudFormation stacks.
* Tested the application functionality including real-time message sending and receiving.
* Conducted load testing to evaluate scalability and performance under different loads.

6. Conclusion

The Serverless Chat Application successfully demonstrated the utilization of AWS services for building real-time chat applications. By leveraging AWS CloudFormation, Lambda, WebSocket API, and DynamoDB, the application achieved scalability, cost-effectiveness, and high performance. The project highlighted the benefits of serverless architecture for developing modern chat applications and provided insights into the implementation process using AWS services.