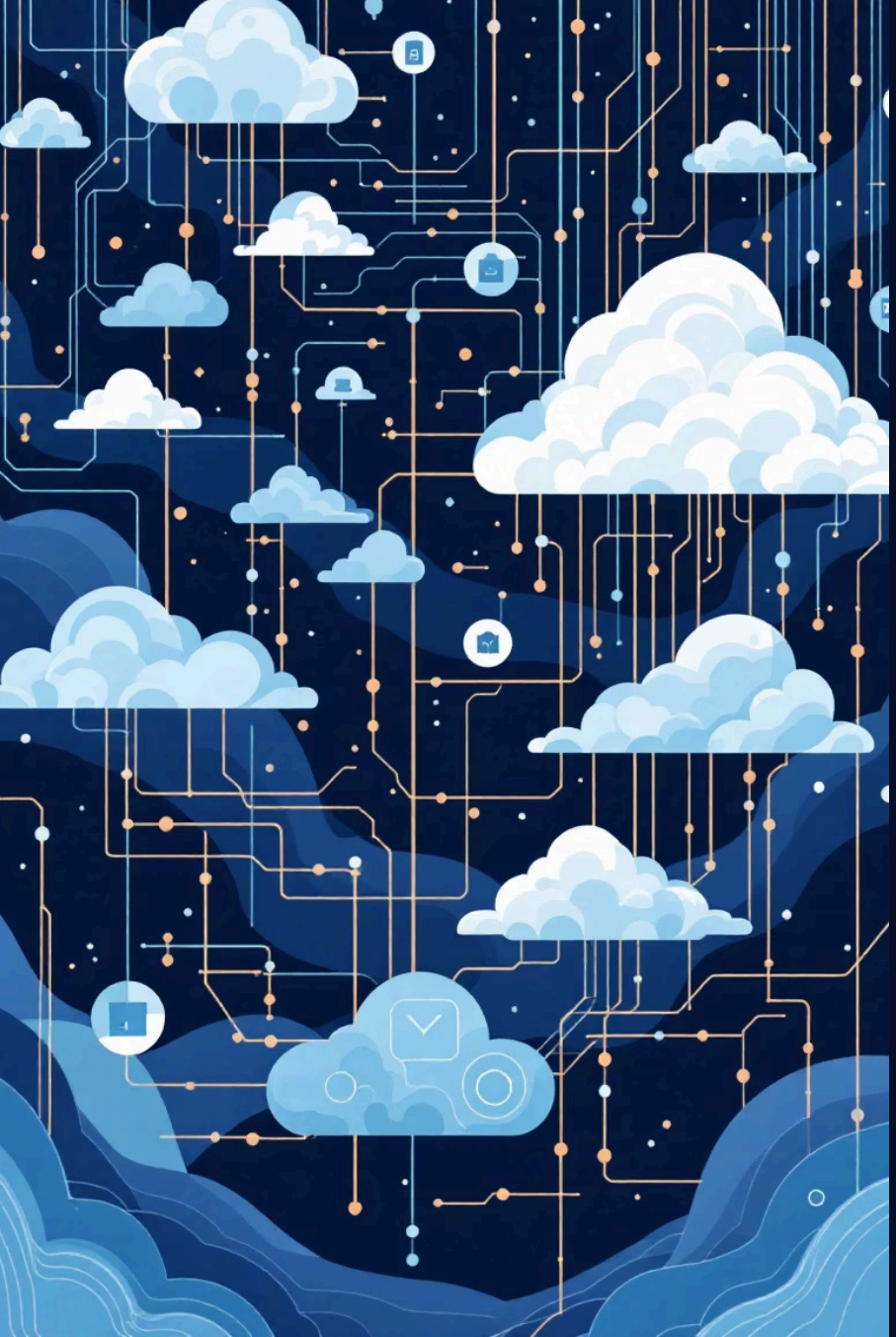


Serverless Sentiment Analysis System using AWS Lambda and API Gateway

A Cloud-Native Approach to Real-Time Sentiment Analysis

Course: Cloud Architecture Design

Shruthi Reddy, Yazhisai G, Akshay SM

Introduction to Sentiment Analysis

Sentiment analysis, also known as opinion mining, is the process of computationally identifying and categorising opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc., is positive, negative, or neutral.

Our primary objective is to develop a robust sentiment analysis system leveraging AWS cloud services. This project aims to demonstrate how a serverless approach can provide unparalleled scalability and cost-efficiency for processing textual data.

Problem Statement: The Need for Automation

1

Time-Consuming Manual Analysis

Manually sifting through vast amounts of text to gauge sentiment is incredibly inefficient and prone to human error, particularly for large datasets.

2

Traditional Server Management

Conventional systems demand continuous provisioning, maintenance, and scaling of servers, leading to significant operational overheads and resource wastage.

3

Automated, Scalable Solution

There is a critical need for an automated, scalable, and fully serverless solution that can efficiently process and analyse sentiment without constant human intervention.

Project Objectives

Build Cloud-Native Web App

Develop a fully functional web application deployed entirely on AWS cloud infrastructure, ensuring a modern and accessible user interface.

Leverage AWS Services

Utilise AWS Lambda for serverless backend processing and Amazon Comprehend for advanced Natural Language Processing (NLP) capabilities.

Frontend Hosting on S3

Host the static website frontend securely and cost-effectively using Amazon S3, ensuring high availability and rapid content delivery.

Minimal Operational Overhead

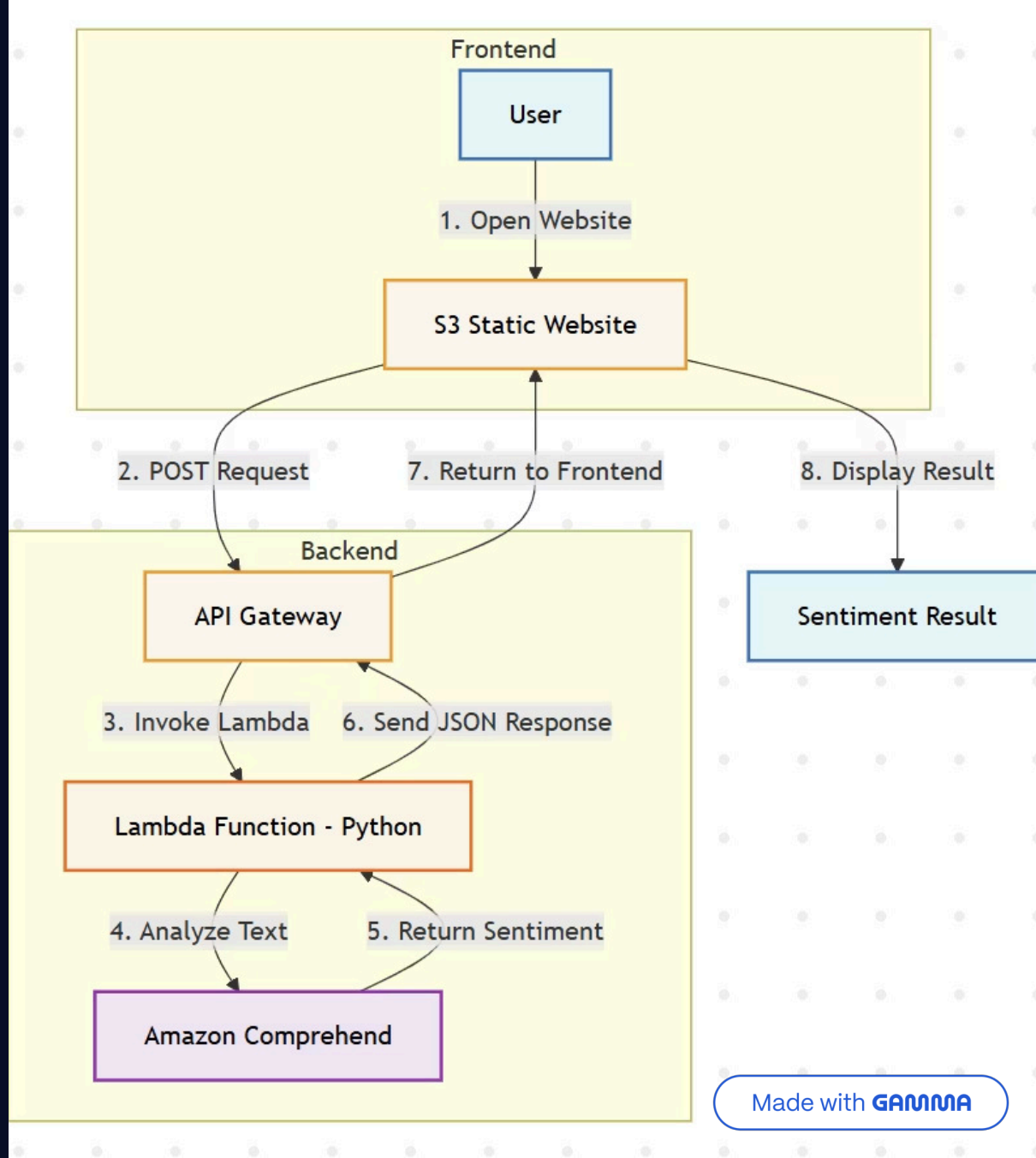
Design the system to operate with minimal manual intervention, reducing operational costs and freeing up resources.

Core AWS Services Utilised

AWS Lambda (Python Runtime)	Executes backend code serverlessly, processing requests without managing servers.
Amazon API Gateway	Manages HTTP requests, routing frontend calls securely to the appropriate Lambda function.
Amazon Comprehend	Performs NLP-based sentiment detection, analysing text for positive, negative, or neutral tones.
Amazon S3	Hosts the static website frontend, offering scalable and durable storage for web assets.
AWS CLI & AWS SAM	Tools used for automating deployment, testing, and managing serverless applications efficiently.

System Architecture Diagram

This diagram illustrates the seamless flow of data from the user interface through various AWS services, culminating in real-time sentiment analysis.



Workflow Explanation

01

User Input on Web App

The user interacts with the S3-hosted web application, entering text for sentiment analysis.

02

API Gateway Receives Request

The user's request is securely transmitted to Amazon API Gateway, which acts as the entry point for all API calls.

03

Lambda Processes Input

API Gateway triggers the AWS Lambda function, which then processes the input text and prepares it for analysis.

04

Comprehend Detects Sentiment

The Lambda function calls Amazon Comprehend, which performs the core NLP task to identify the sentiment of the text.

05

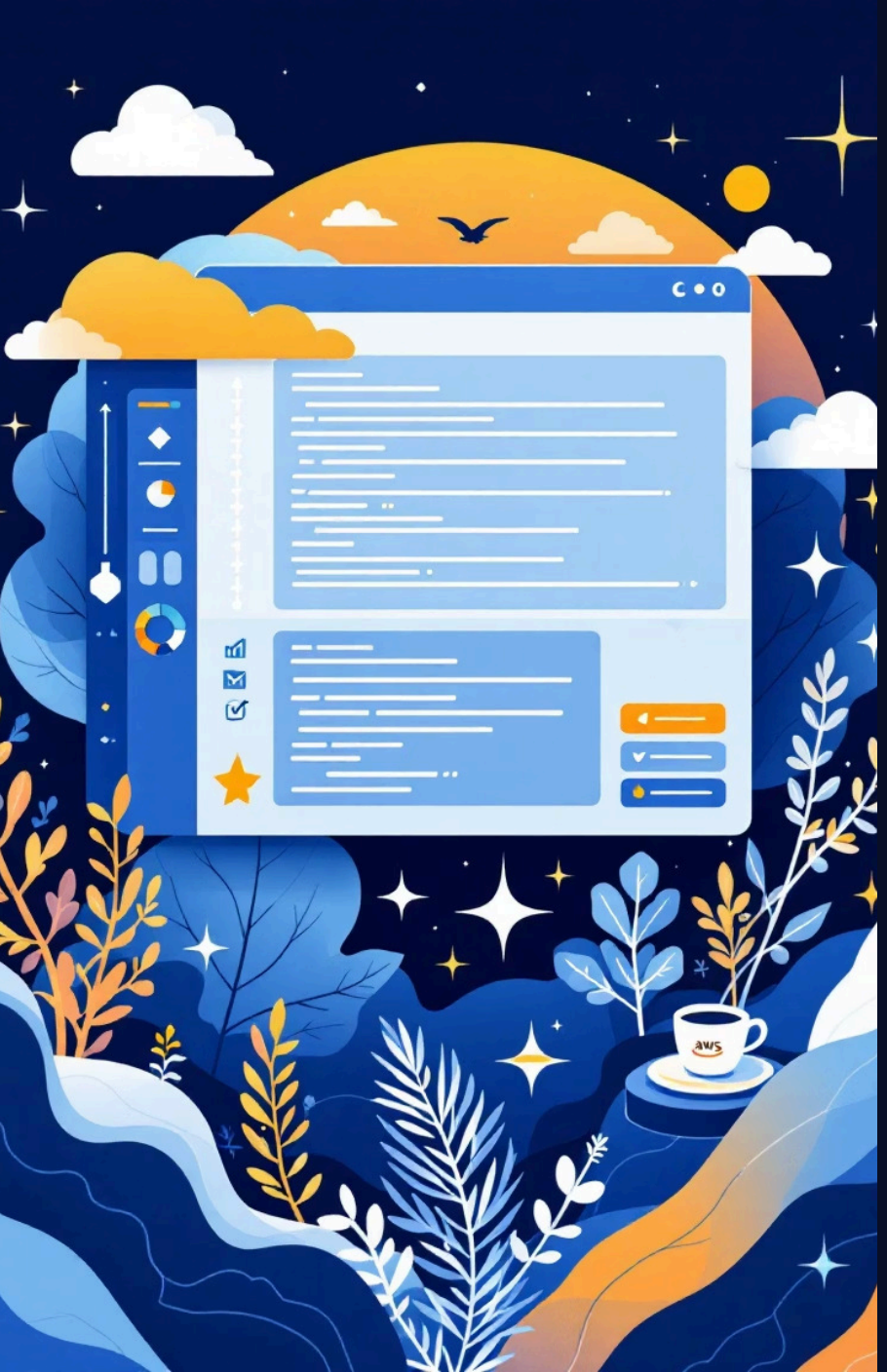
Lambda Formats Output

Comprehend returns the sentiment results to Lambda, which then formats the output for display on the frontend.

06

Result Displayed to User

The formatted sentiment analysis result is sent back to the S3 web app and dynamically displayed to the user.



Implementation Details

Frontend Development

- Built using standard web technologies: HTML and JavaScript.
- Hosted securely and efficiently on Amazon S3 for static content delivery.

Backend Logic

- Implemented as a Python Lambda function.
- Utilises the Boto3 library to interact with AWS services, specifically Amazon Comprehend.

Deployment & Automation

- AWS Serverless Application Model (SAM) and AWS CLI were employed.
- Ensures streamlined packaging and deployment of all serverless resources.

Cross-Origin Resource Sharing (CORS)

- Carefully configured in API Gateway.
- Essential for enabling secure communication between the browser-based frontend and the API Gateway.

Results and Observations

Example Input

> "I love building solutions with AWS! It offers incredible flexibility and powerful tools."

Example Output

> Sentiment: Positive
>
> Confidence Score: 99.5%

Key Observations

- The system demonstrated efficient handling of multiple concurrent inputs.
- Achieved low-latency responses, crucial for real-time applications.
- Successfully delivered a fully serverless, cost-effective sentiment analysis solution.

Conclusion & Future Scope

We have successfully designed and implemented a functional, scalable, and serverless sentiment analysis system, demonstrating seamless integration across multiple AWS services.

1

Real-time Analytics Dashboard

Integrate a dashboard for visualising sentiment trends and metrics in real-time, enhancing data interpretation.

2

Multilingual Support

Extend the system to support various languages by leveraging Amazon Comprehend's multilingual capabilities, broadening its applicability.

3

Enhanced UI with Visual Indicators

Improve the user interface with richer visual sentiment indicators and interactive elements for a more engaging user experience.

Thank You