VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi- 590018

An Industry Internship(21INT82) Report on

"SERVERLESS WEATHER API USING AWS LAMBDA AND OPENWEATHERMAP"

Submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

By

ABHIRAJ M L **1EP21CS003**

Internal guide:

Prof. Prasanth

Assistant Professor

Dept. of CSE, EPCET

Company:

Amazon Web Services (AWS)

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

Approved by AICTE New Delhi | Affiliated to VTU, Belagavi,

Virgo Nagar, Bengaluru-560049

2024-2025

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

Approved by AICTE New Delhi | Affiliated to VTU, Belagavi,
Virgo Nagar, Bengaluru-560049

CERTIFICATE

This is to certify that the Industry Internship(21INT82) entitled "Serverless Weather API using AWS Lambda and OpenWeatherMap" is a bonafide work carried out by **ABHIRAJ M L [1EP21CS003]**, in partial fulfillment of the requirements of BACHELOR OF ENGINEERING in COMPUTER SCIENCE AND ENGINEERING in VISVESVARAYA TECHNOLOGICAL UNIVERSITY,

Belgaum, during the year 2024-2025. It is certified that corrections/suggestions recommended have been incorporated in the Internship report.

Internal Guide **Prof. Prasanth** **Assistant Professor** **Dept. of CSE, EPCET, Bengaluru**	**Company Guide** **Amazon Web Services(AWS)**	**Signature of HOD** **Dr. I Manimozhi** **HOD, Dept. of CSE, EPCET, Bengaluru**	**Signature of Principal** **Dr. Mrityunjaya V Latte** **Principal** **EPCET, Bengaluru**

Reviewers:

Reviewer 1	**Reviewer 2**
Name: Mrs. Divyashree	**Name: Mrs. Neha Gopal**
Signature with date:	**Signature with date:**

Examiners:

Examiner 1	**Examiner 2**
Name:	**Name:**
Signature with date:	**Signature with date:**

COMPANY CERTIFICATE

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere regards and thanks to **Management** of **East Point Group of Institutions, Bengaluru** for providing me an opportunity to work on this Internship.

I am ineffably indebted to **Dr. S Prakash, Senior Vice President**, **East Point Group of Institutions,** for his conscientious guidance and encouragement to accomplish this Internship.

I would like to express my humble and sincere thanks to **Dr. Mrityunjaya V Latte, Principal, East Point College of Engineering and Technology** for his suggestions

motivated me for the successful completion of my internship.

I would like to express my heartfelt thanks to **Dr. I Manimozhi**, **Professor and Head of Department** of Computer Science and Engineering, EPCET for her valuable advice and suggestions to do my best in this Internship.

I am obliged to guide **Prof. Prasanth** Assistant Professor, Dept. of CSE, Internship Coordinators **Prof. Madhushree**, Assistant Professor, Dept. of CSE and **Prof. Nithyananda C R,** Associate Professor, Dept. of CSE who have rendered valuable assistance and guidance for the Internship.

I would like to thank **Amazon Web Services** for providing with technical support and knowledge.

I would like to thank my **Parents** and **Friends** for their support and encouragement during the course of my internship. Finally, I offer my regards to all the **faculty members of the CSE Department** and all those who supported me in any respect during the Internship.

Abhiraj M L[1EP21CS003]

CONTENTS

List of Figures

CHAPTER 1

COMPANY PROFILE

Company Overview

Amazon Web Services (AWS) is a subsidiary of Amazon providing on-demand cloud computing platforms and APIs to individuals, companies, and governments. Launched in 2006, AWS is the world's most comprehensive and widely adopted cloud platform, offering over 200 fully featured services from data centres globally.

Vision and Mission

- **Vision:** To be Earth's most customer-centric company where customers can find and discover anything they want to buy online, and to provide the best cloud computing services that empower businesses worldwide.
- **Mission:** To enable businesses and developers to use web services to build scalable, reliable, and low-cost applications with the flexibility of pay-as-you-go pricing.

CHAPTER 2

ABOUT COMPANY

Amazon Web Services (AWS) is the world's leading cloud computing platform, offering a broad and deep set of on-demand services to individuals, businesses, and governments. Launched in 2006 by Amazon, AWS pioneered the cloud infrastructure industry and continues to dominate the market due to its robust features, scalability, and flexibility. It provides a comprehensive suite of over 200 fully featured services across computing, storage, databases, networking, artificial intelligence, machine learning, analytics, security, and more.

One of the biggest strengths of AWS is its global infrastructure. It operates in over 32 geographic regions, with more than 100 Availability Zones (AZs) around the world. This allows customers to deploy applications in multiple regions for better performance, fault tolerance, and disaster recovery. Whether it's a startup, an enterprise, or a government agency, AWS enables users to build scalable and highly available applications with low latency and global reach.

A major reason for AWS's popularity is its pay-as-you-go pricing model, which allows users to pay only for the resources they consume, with no upfront investment or long-term commitments. This is especially beneficial for small businesses and developers, as it lowers the barrier to entry for using enterprise-grade infrastructure.

AWS provides a wide range of computing services, including Amazon EC2 (virtual servers), AWS Lambda (serverless computing), and ECS/EKS for containerized applications. In terms of storage, Amazon S3 is a highly durable and scalable object storage service used globally for backups, web content, and media storage. Databases are another core offering, with services like Amazon RDS, DynamoDB, and Aurora supporting relational and NoSQL database needs.

Security is a top priority for AWS. It offers features such as Identity and Access Management (IAM) for access control, encryption for data in transit and at rest, and compliance with international standards like ISO, HIPAA, and GDPR. AWS's shared responsibility model ensures that while AWS secures the infrastructure, customers are responsible for securing their data and applications.

Services Offered

AWS provides a wide range of cloud-based products including:

- **Compute** – Amazon EC2, AWS Lambda
- **Storage** – Amazon S3, Amazon EBS
- **Databases** – Amazon RDS, DynamoDB
- **Networking** – Amazon VPC, Route 53
- **Machine Learning & AI** – Amazon SageMaker, Rekognition, Comprehend
- **Analytics** – Amazon Athena, Redshift, Glue
- **Security & Identity** – IAM, Cognito, KMS

Global Infrastructure

AWS operates in 32 geographic regions with over 100 Availability Zones across the globe, enabling high availability, fault tolerance, and scalability. Each region is a separate geographic area that contains multiple isolated locations known as Availability Zones.

Key Clients

Some of the most recognized brands using AWS include:

- Netflix
- LinkedIn
- NASA
- Samsung
- Airbnb
- General Electric
- Spotify

Why AWS?

- **Pay-as-you-go pricing**: You only pay for the services you use, with no upfront cost. This helps reduce expenses and avoids over-provisioning.
- **Scalability and flexibility**: AWS automatically adjusts resources based on demand. It supports both vertical and horizontal scaling effortlessly.
- **High availability and reliability**: AWS offers data centres in multiple regions and zones. This ensures minimal downtime and smooth disaster recovery.
- **Secure and Compliant Architecture**: AWS provides strong data encryption and access control. It meets major global security and compliance standards.
- **Global reach with low latency**: With global data centres, AWS delivers fast performance worldwide. It ensures low latency and a better user experience for global users.

CHAPTER 3

TASK PERFORMED

3.1 Introduction to AWS & Cloud

Cloud computing has revolutionized how businesses and developers build, deploy, and scale applications. Among the leading cloud service providers, Amazon Web Services (AWS) stands out as a comprehensive and widely adopted platform, offering over 200 fully featured services from data centers globally.

This project, "Smart Support Ticket Analyzer for Helpdesk Teams," leverages AWS to automate and enhance the processing of customer support tickets. Traditionally, helpdesk systems involve manual categorization and assignment of tickets, which leads to delays and errors. By integrating services like AWS Lambda, Amazon Comprehend, and Amazon S3, this project automates ticket analysis, detects sentiment, and extracts relevant entities to prioritize and route queries effectively.

AWS's cloud-native tools enable scalable, serverless processing and real-time analysis, making it an ideal choice for intelligent customer support systems. The flexibility, reliability, and ease of integration offered by AWS empower developers to build smart applications without worrying about underlying infrastructure.

Features of AWS

Amazon Web Services (AWS) is a leading cloud computing platform that offers a broad set of global cloud-based products and services. Here are some key features of AWS:

- **Scalability**: AWS can automatically scale resources up or down based on demand using like Auto Scaling and Elastic Load Balancer.
- **Flexibility**: It supports multiple programming languages, operating systems, databases, and architectures.
- **Cost-Effectiveness**: AWS offers a pay-as-you-go pricing model, reducing upfront investment.
- **High Availability and Reliability**: With data centers across multiple regions and availability zones, AWS ensures minimal downtime.

Applications of AWS

Amazon Web Services (AWS) powers a vast number of use cases across industries due to its flexibility, scalability, and vast service offerings. Here are some of the major applications of AWS:

- **Web and Application Hosting**

  AWS provides infrastructure and tools for hosting dynamic websites, web applications, and APIs using services like Amazon EC2, Elastic Beanstalk, and Elastic Load Balancing.

- **Data Storage and Backup**

  AWS offers secure, durable, and scalable storage solutions such as **Amazon S3**, **Glacier**, and **EBS**, used for storing static assets, backups, and large data sets.

Advantages of AWS

- **Cost-Effective**

  AWS operates on a pay-as-you-go pricing model, which means users only pay for the resources they consume. This eliminates the need for large upfront capital investments in hardware.

- **Scalability and Flexibility**

  AWS allows automatic scaling of resources based on demand using services like Auto Scaling and Elastic Load Balancing, which helps manage variable workloads.

3.2 Introduction to Serverless Weather API using AWS Lambda and OpenWeatherMap

Weather data is essential for a wide range of applications, from travel planning and logistics to agriculture and smart city solutions. In the context of modern cloud computing, building a scalable, real-time weather API can be efficiently achieved using serverless architecture on platforms like Amazon Web Services (AWS).

This project focuses on developing a serverless weather data service using AWS Lambda and the OpenWeatherMap API. The solution allows users to query weather information for any city through an API endpoint. This endpoint, exposed via Amazon API Gateway, triggers a Lambda function which then calls the OpenWeatherMap service to retrieve current weather data in real-time.

With the increasing demand for fast, scalable, and cost-effective solutions, AWS provides a perfect environment to build such APIs without the burden of managing servers or infrastructure. AWS Lambda handles compute execution on demand, while API Gateway acts as the public-facing entry point, and CloudWatch provides visibility into system performance and logging.

This architecture ensures that the system is highly available, automatically scalable, and extremely cost-efficient, charging only for the actual compute time used during API calls.

Why Use AWS for Serverless Weather API

Amazon Web Services (AWS) is a leading cloud platform offering powerful tools to build serverless APIs. Here's why AWS is ideal for a weather API solution:

- **Managed, Scalable Infrastructure:** AWS Lambda automatically scales based on the number of incoming API requests. This eliminates the need for provisioning or managing servers, making it ideal for bursty or unpredictable workloads like weather queries.
- **Seamless API Integration:** With Amazon API Gateway, developers can expose HTTP endpoints without managing a backend server. This makes it easy to connect client applications to the weather-fetching service.
- **Security and Compliance:** Using AWS IAM, the solution can be secured with fine-grained roles and policies, ensuring only authorized access to services. Encryption, access control, and compliance with standards like SOC, GDPR, and ISO are built in.
- **Cost-Efficiency:** The serverless model ensures that costs are incurred only during actual usage, making it a budget-friendly solution for both prototypes and production deployments.
- **Real-Time Monitoring:** AWS CloudWatch helps track logs, execution times, and errors for Lambda functions, ensuring complete visibility and easy debugging**.**

Project Objective

The primary objective of this project is to design and implement a cloud-based, serverless solution that delivers real-time weather information using Amazon Web Services (AWS) and the OpenWeatherMap API. The system uses key AWS services such as API Gateway to receive HTTP requests, AWS Lambda for serverless processing, and IAM for secure role-based access

This project aims to:

- Enable users to query weather data for any location via an HTTP API endpoint.
- Trigger AWS Lambda functions upon API request, avoiding traditional server setups.
- Use OpenWeatherMap API to fetch accurate weather data, including temperature, humidity, and conditions.
- Return structured weather details in JSON format for easy integration with web/mobile applications.
- Demonstrate a secure, scalable, and cost-effective serverless architecture with high availability and low maintenance.

Benefits of Serverless Weather APIs

Using serverless architecture to build weather APIs offers numerous benefits, both technically and economically:

- **On-Demand Execution**: Lambda runs only when triggered, which is ideal for irregular or user-driven weather data requests.
- **Zero Server Management**: Developers focus on logic instead of worrying about provisioning, scaling, or patching servers.
- **24/7 Accessibility**: The API is always available and can handle global traffic at any time.
- **Scalability**: Automatically scales to handle multiple requests simultaneously without latency issues.
- **Security**: IAM ensures fine-grained access control, minimizing risks of unauthorized access.
- **Cost-Effectiveness**: Pay-per-use billing significantly reduces costs, especially for low to medium traffic applications.
- **Seamless Integration**: The weather API can be easily consumed by websites, mobile apps, or IoT devices for real-time environmental insights.

3.3 Amazon Services Used

For the Serverless Weather API project using AWS, the following services were used in combination to implement an efficient, event-driven architecture:

- **Amazon API Gateway**: Acts as the entry point for HTTP requests. It receives city- based queries from users and triggers the Lambda function.
- **AWS Lambda**: Executes the backend logic for fetching weather data from the OpenWeatherMap API. It runs only when triggered, making it a perfect fit for serverless APIs.
- **OpenWeatherMap API**: A third-party weather data service that provides current weather conditions based on the city name input.

Amazon API Gateway

Amazon API Gateway is a fully managed service that enables developers to create and publish RESTful APIs. In this project, it serves as the interface through which external users access the serverless weather data endpoint.

Users send requests with a city name as a query parameter, and API Gateway securely forwards this request to the corresponding AWS Lambda function. API Gateway is highly scalable and supports traffic throttling, CORS, authentication, and logging features.

Its integration with Lambda eliminates the need for a traditional backend server, making it ideal for lightweight, real-time applications like this weather API.

OpenWeatherMap API

OpenWeatherMap is a public API that provides accurate and up-to-date weather data for any location around the globe. It supports various endpoints, including:

- **Current Weather Data**: Returns temperature, humidity, pressure, and weather conditions.
- **Forecast**: Offers hourly and daily forecasts.
- **Weather Icons**: Displays weather conditions visually using icons.

AWS Lambda

AWS Lambda is a core component of the serverless architecture used in this project. It executes backend logic on-demand in response to API Gateway triggers.

Key Features:

- **Languages Supported**: Python, Node.js, Java, Go, Ruby, .NET, etc.
- **Event-Driven Execution**: Automatically runs when triggered by API Gateway.
- **Scalability**: Lambda functions scale automatically based on incoming traffic.

3.4 Project Modules

The Serverless Weather API project is modularized into distinct components that work together in a seamless, event-driven pipeline. Each module is designed to handle specific tasks efficiently, ensuring better maintainability, scalability, and integration with external services like dashboards or alerting systems.

API Trigger & Input Handling

This module is responsible for receiving user requests via the API Gateway and extracting relevant query parameters (such as city name or geographical coordinates).

Key tasks include:

- Accepting HTTP GET requests through Amazon API Gateway.
- Parsing input parameters (e.g., city, state, or country).
- Validating the input to ensure proper formatting and non-empty values.
- Triggering the AWS Lambda function to initiate weather data retrieval.

Weather Data Fetching & Processing

This is the core module where the Lambda function integrates with the OpenWeatherMap API to fetch real-time weather information.

Key functionalities:

- Calling the OpenWeatherMap REST API using the provided location details.
- Extracting relevant data such as temperature, humidity, weather conditions, and wind speed.
- Handling different response formats and error codes from the external API.

Result Handling & Response Generation

After obtaining and processing the weather data, this module is responsible for returning a clean and meaningful response to the client.

Key steps:

- Structuring the response in a JSON format with weather details.
- Returning appropriate HTTP status codes (200 OK, 400 Bad Request, 500 Internal Error).
- Logging successful responses for auditing and future reference.
- Optionally storing historical request and response data in Amazon S3 or DynamoDB for analytics.

Error Handling & Monitoring

This module ensures robust functioning of the system by capturing and addressing potential issues during API execution.

Key tasks:

- Detecting empty or malformed input queries.
- Catching errors from the OpenWeatherMap API (e.g., network issues, API limits).
- Logging all errors and Lambda execution traces using AWS CloudWatch Logs.
- Setting up CloudWatch Alarms or SNS Notifications to alert in case of system failures.

3.5 Implementation Flow

System Architecture

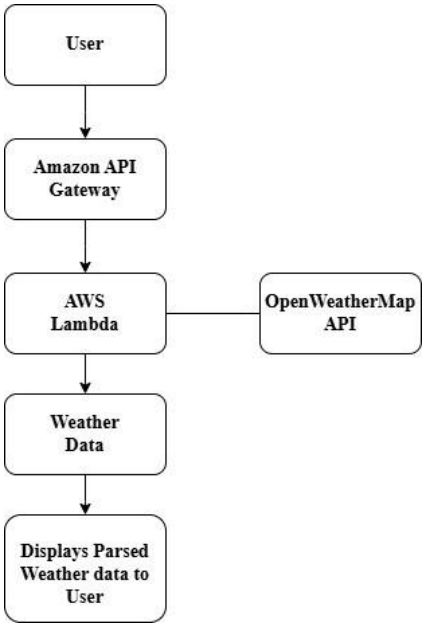

Figure 3.5: System Architecture

The system architecture for the Serverless Weather API is based on a **cloud-native, event-driven design** using AWS services. The components work together as follows:

- **User:**

  Interacts with a web or mobile app to input a **city name** for which weather data is requested.

- **Amazon API Gateway:**

  Acts as the **entry point** for incoming HTTP requests. It securely **routes requests to AWS Lambda** and handles tasks like **authorization, throttling, and request validation**.

- **AWS Lambda:**

  The Lambda function is triggered by API Gateway and serves as the **main compute unit**. It:

  1. Parses the input city name.
  2. Constructs a request to the **OpenWeatherMap API** processes the received weather data.

- **OpenWeatherMap API:**

  An external weather service that provides real-time weather information such as temperature, humidity, and wind speed when queried with a city name.

- **Weather Data Processing:**

  Inside the Lambda function, the response from OpenWeatherMap is:

  1. **Parsed** to extract useful data.
  2. **Formatted** into a user-friendly structure (JSON).
  3. **Filtered** to remove unnecessary information.

- **Response Delivery:**

  The **formatted weather data** is returned to API Gateway, which sends it back to the **user interface** for display.

Workflow Diagram

The workflow for the serverless weather API can be illustrated in the following sequence:

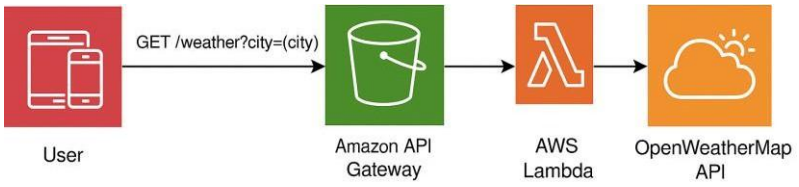

Figure 3.6 Workflow diagram

- **User Input**: A city name is passed via a query string to the API Gateway.
- **Triggering Lambda**: The API Gateway triggers a Lambda function mapped to this endpoint.
- **Weather API Integration**: Lambda fetches weather details by making a secure HTTP request to OpenWeatherMap.

Step by step execution

- **User Sends API Request**

  The process begins when a user sends GET request to the API Gateway endpoint, providing location parameters to fetch weather data.

- **Trigger Lambda Function**

  The API Gateway automatically triggers a pre-configured AWS Lambda function that is linked to the API Gateway.

- **OpenWeatherMap API**

  The Lambda function makes an API call to the OpenWeatherMap API, passing the location parameters from the user's request to retrieve weather data for specified location.

- **Process and Format Data**

  The Lambda function processes the raw weather data received from OpenWeatherMap, which mayinclude:

  - Extracting relevant weather information (e.g., temperature, humidity, weather conditions).
  - Formatting the data for the response, such as converting temperature units or rounding values for readability.

CHAPTER 4

REFLECTION

4. Reflection / Outcome Analysis

This internship provided hands-on experience with designing and implementing a **serverless REST API** using AWS services. The project aimed to deliver a **real-time weather forecasting system** using AWS Lambda, API Gateway, and the OpenWeatherMap API. The video walkthrough highlighted key steps—from configuration to testing—validating the practicality and efficiency of serverless cloud-based systems.

4.1 OpenWeatherMap API Key Generation

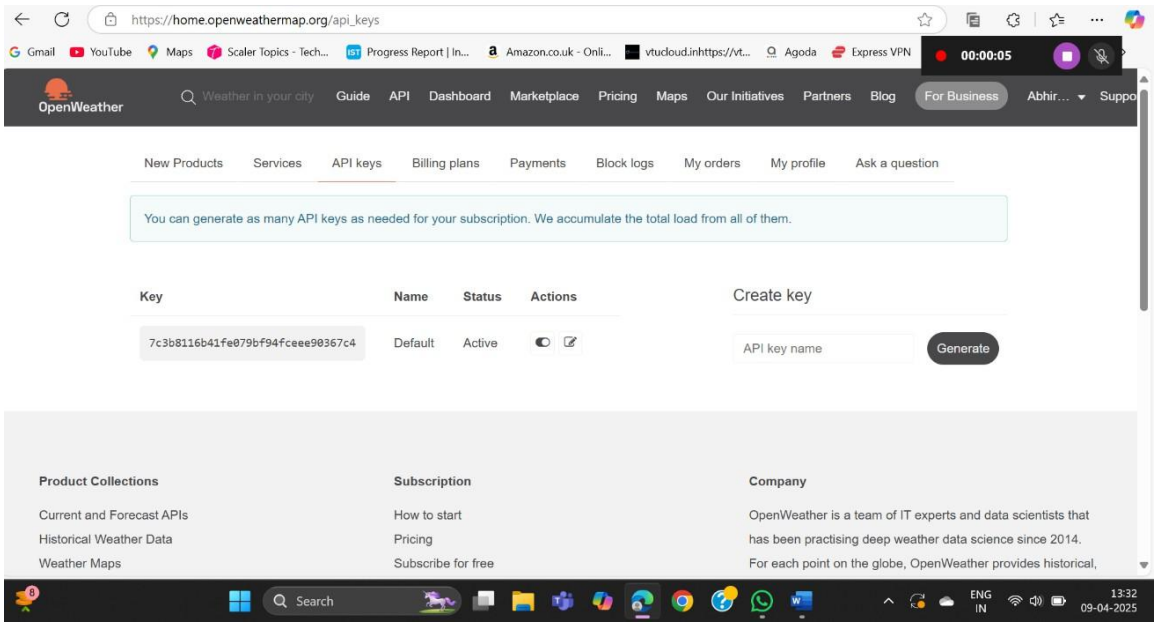

Figure 4.1 Home Page

- The user logs in and navigates to the API Keys tab.
- The dashboard lists all existing keys along with their status.
- A new API key can be created by entering a name and clicking "Generate".
- The generated key is later used to authenticate weather data requests in the AWS Lambda.

4.2 Creation of Lambda Function

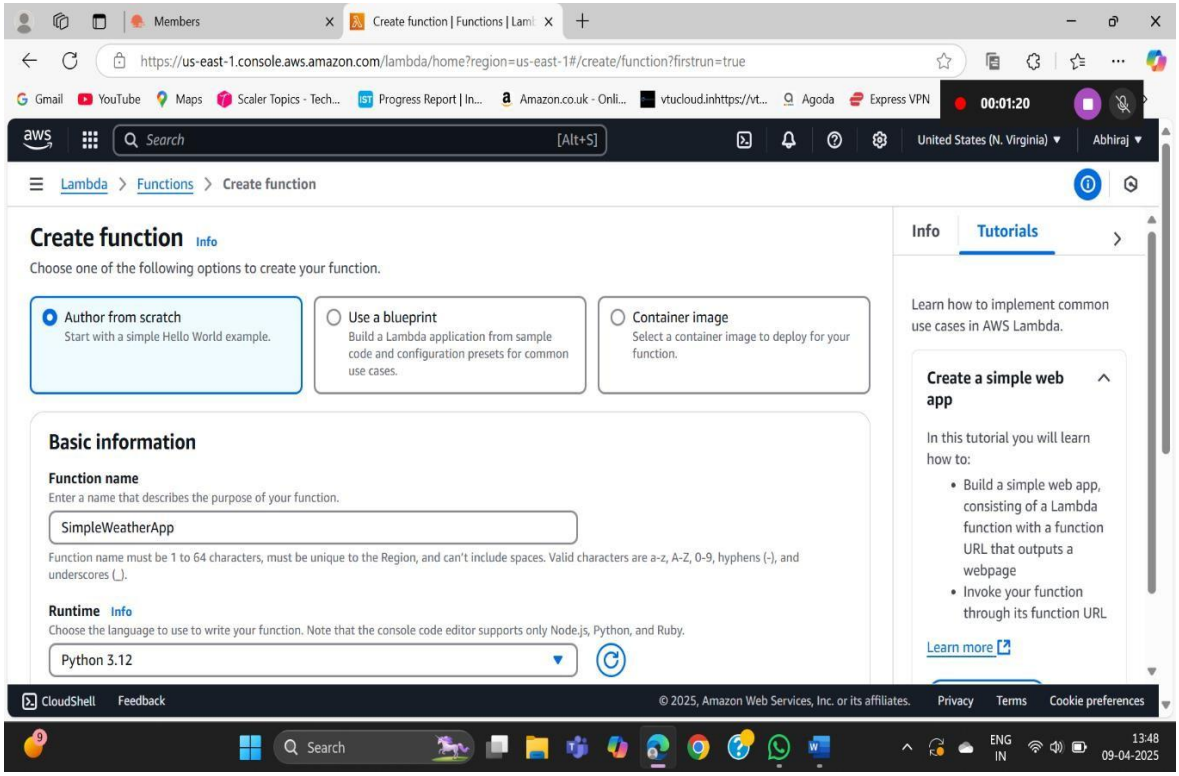

Figure 4.2.1 AWS Lambda Console

- The Lambda function is the core compute resource that processes requestsand returns weather data fetched from OpenWeatherMap.
- After creation, AWS confirms with a success message and displays options to configure triggers, destinations, and access the function URL.
- The **Function ARN (Amazon Resource Name)** is also provided, which uniquely identifies the Lambda function across AWS.
- The setup is essential for building a serverless weather app where code execution is handled without managing any servers.

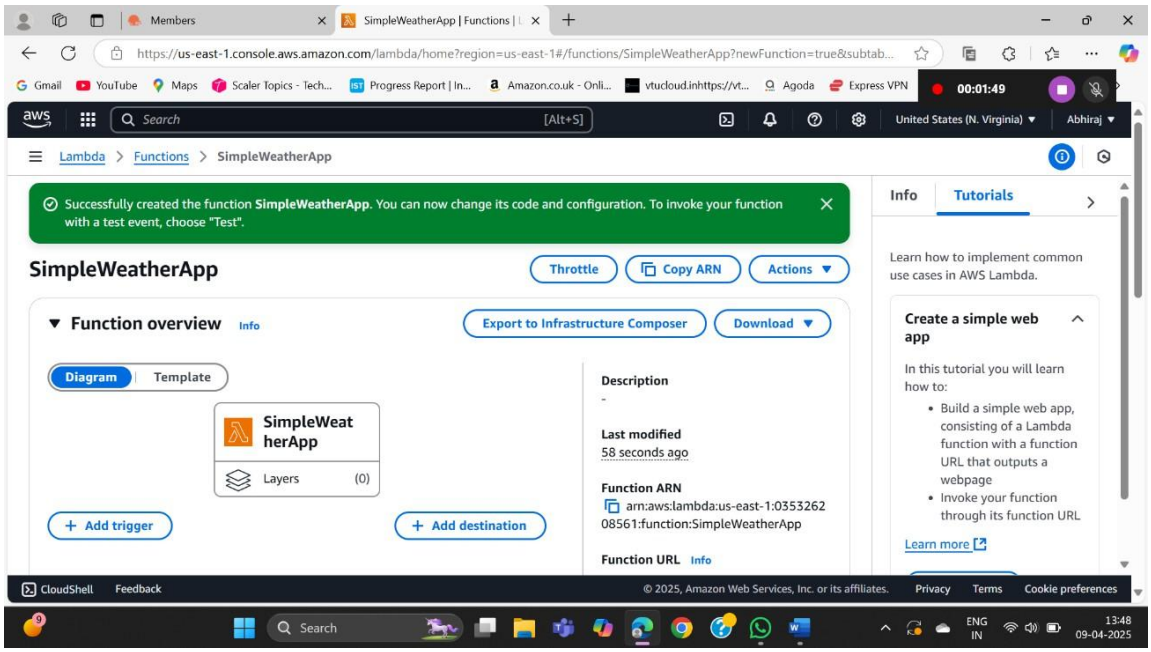

Figure 4.2.2 SimpleWeatherApp Lambda function

4.3 Updating Lambda Function Code with API Call

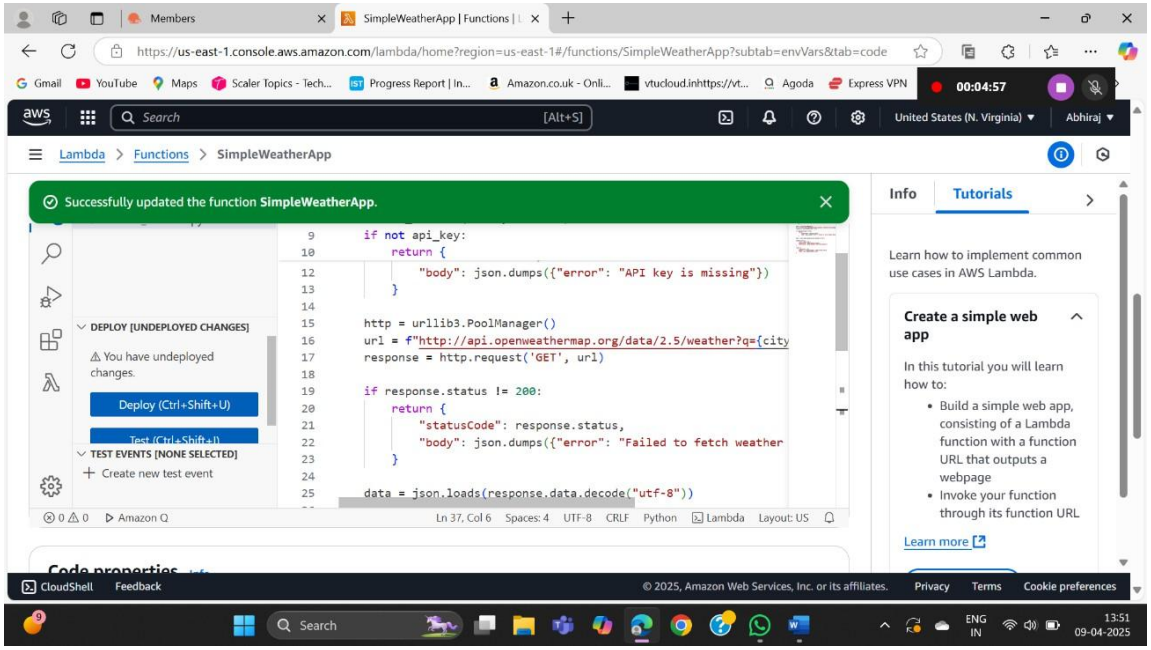

Figure 4.3 Configuring Lambda Function

Add the lambda function code

```
import json
import os
import urllib3

def lambda_handler(event, context):
    city = event.get("queryStringParameters", {}).get("city", "Bangalore")
    api_key = os.environ.get("WEATHER_API_KEY")

    if not api_key:
        return {
            "statusCode": 500,
            "body": json.dumps({"error": "API key is missing"})
        }

    http = urllib3.PoolManager()
    url =
f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&units=metric"
    response = http.request('GET', url)

    if response.status != 200:
        return {
            "statusCode": response.status,
```

```
"body": json.dumps({"error": "Failed to fetch weather data"})
    }
data = json.loads(response.data.decode("utf-8"))
weather_info = {
    "city": data["name"],
    "temperature": data["main"]["temp"],
    "description": data["weather"][0]["description"],
}
return {
    "statusCode": 200,
    "headers": {"Content-Type": "application/json"},
    "body": json.dumps(weather_info)
}
```

4.4 Configuring API Gateway Route for Weather API

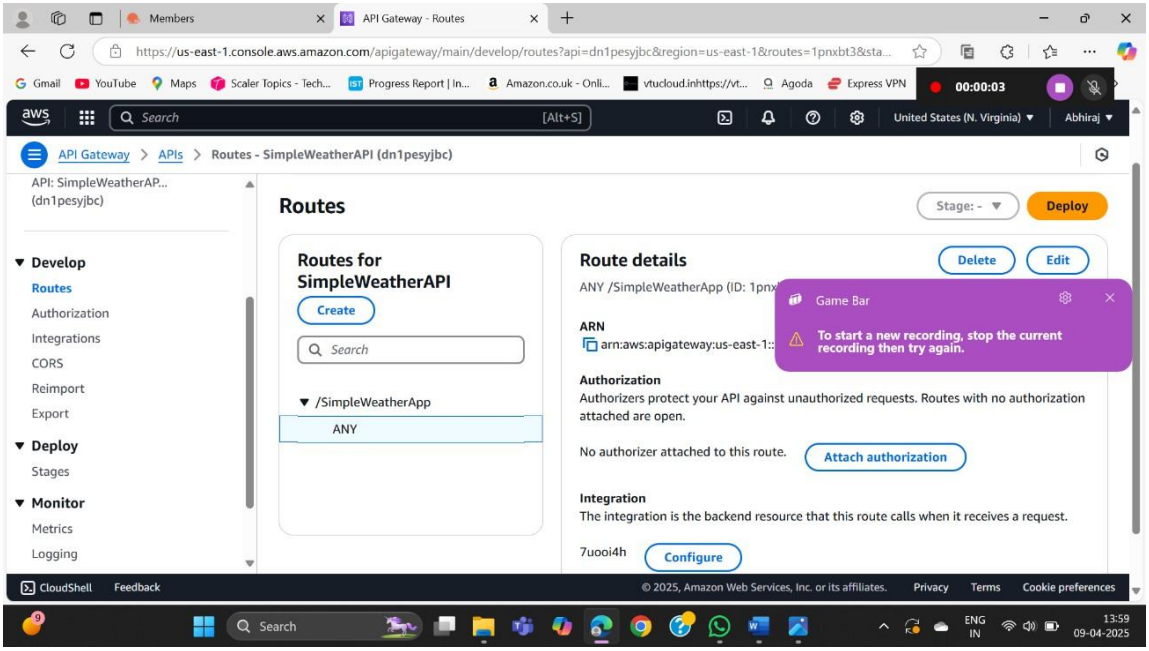

Figure 4.4 Route Configuration in API Gateway

- The **ARN** uniquely identifies the route within AWS infrastructure.
- No authorization has been attached to the route, meaning it is currently open to public access.
- The backend integration is set to the previously created Lambda function, ensuring that all incoming requests are processed through it.

4.5 Testing Weather API Response in Browser

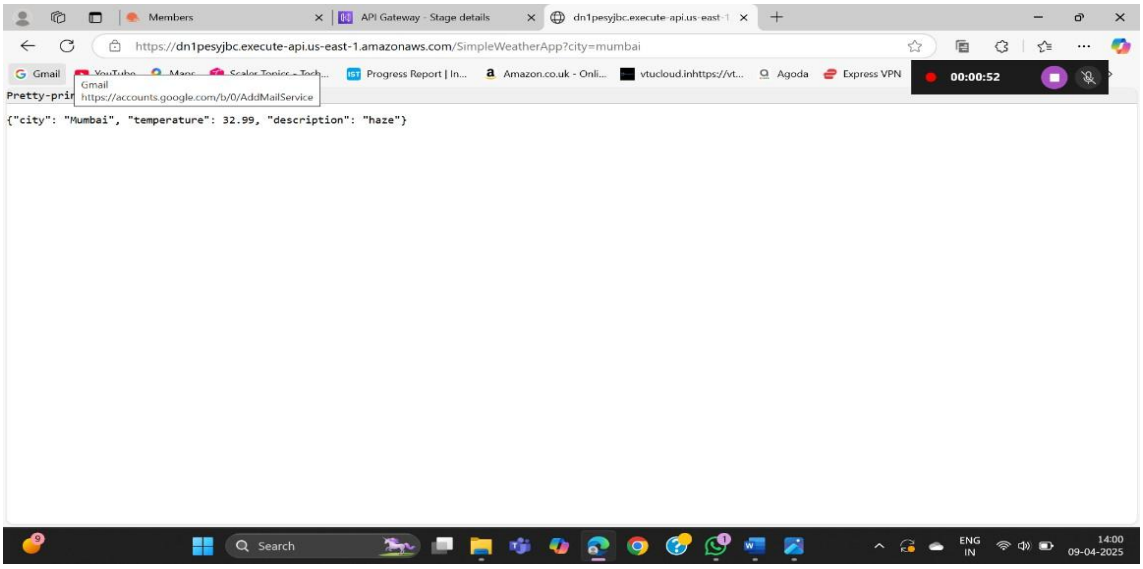

Figure 4.5.1 Weather API Response Output

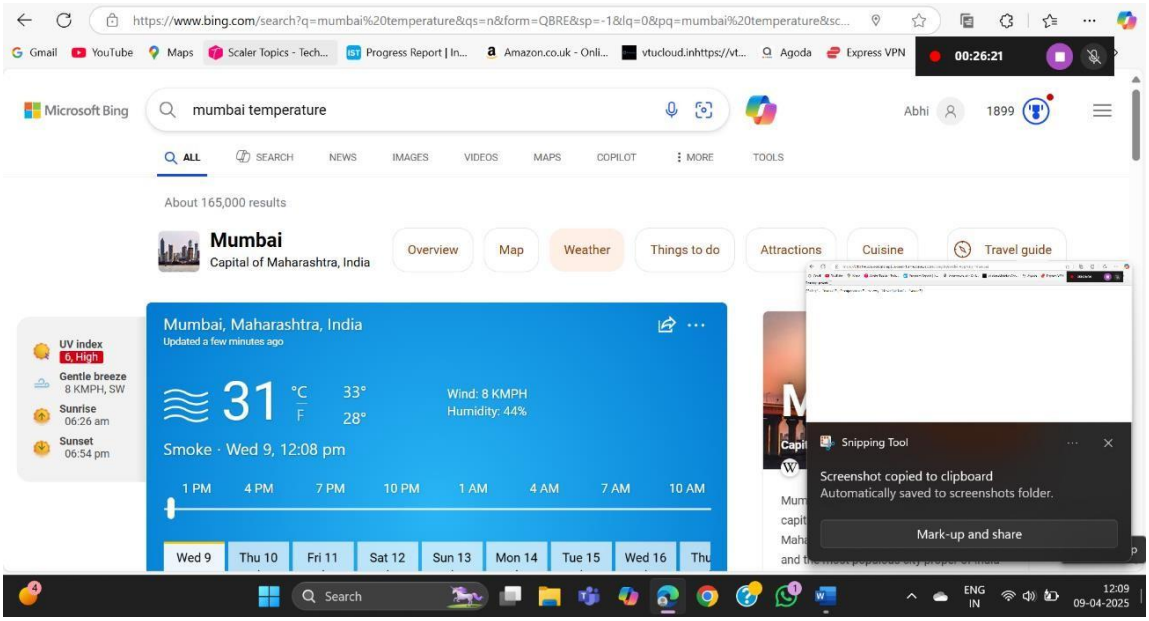

Figure 4.5.2 Comparison with Live Weather Source

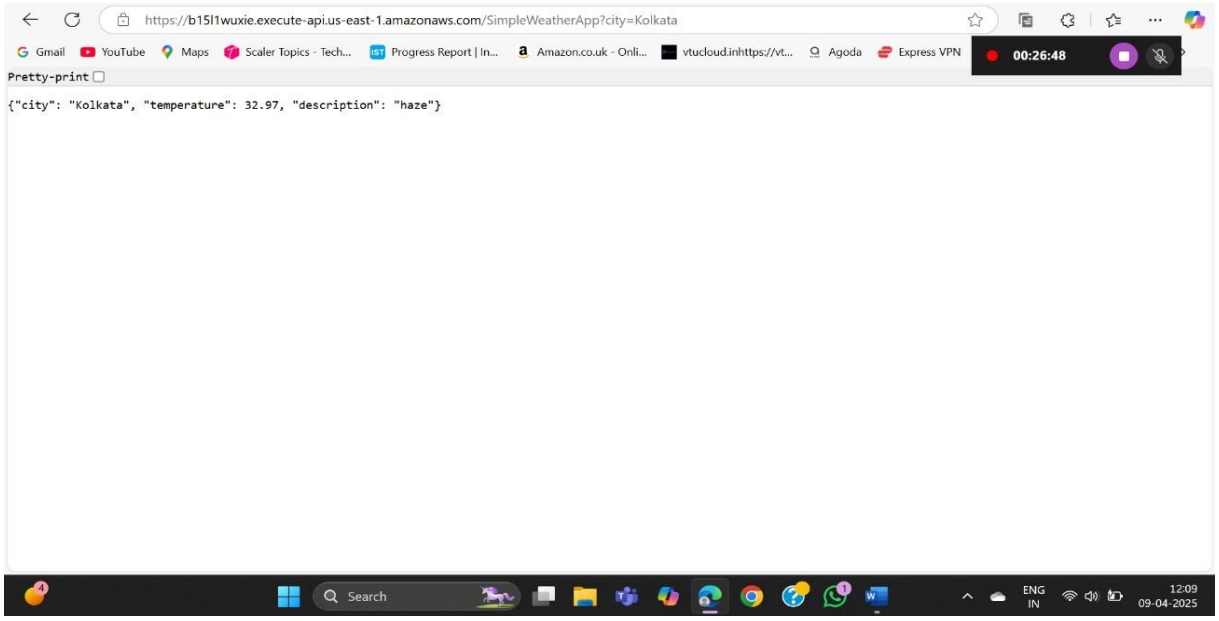

Figure 4.5.3 Weather API Response Output for Kolkata

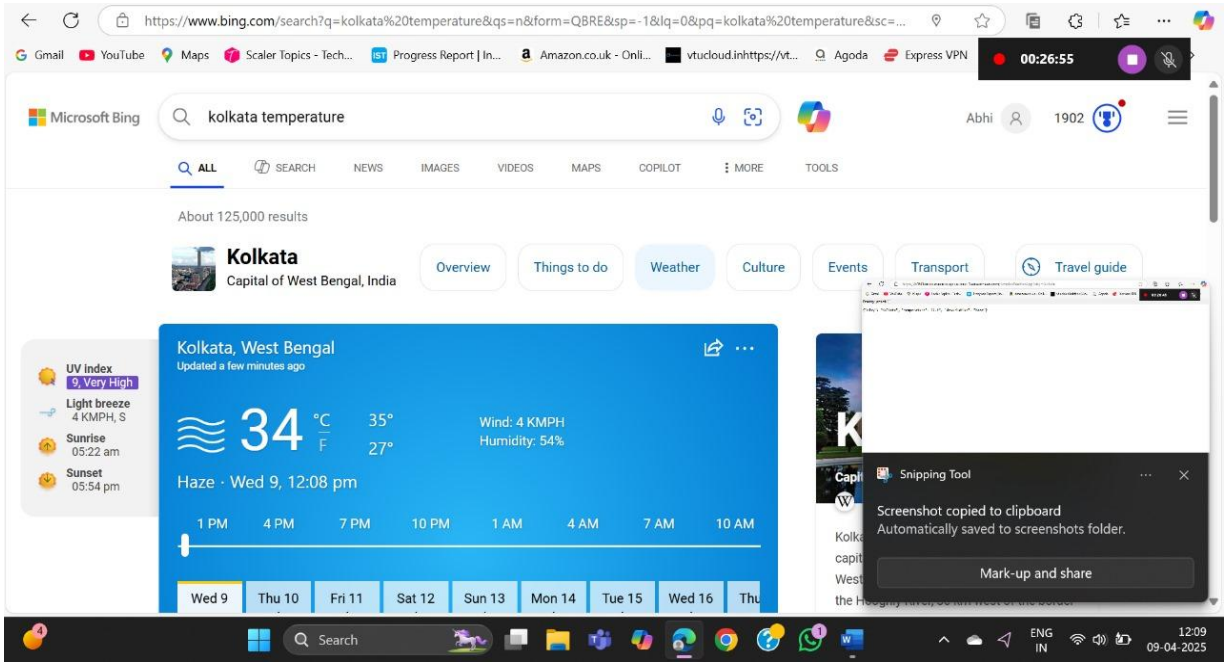

Figure 4.5.4 Comparison with Live Weather Source

CHAPTER 5

CONCLUSION

The Serverless Weather API using AWS Lambda and OpenWeatherMap project successfully demonstrates the efficiency and scalability of serverless architecture for delivering real-time weather information. By integrating AWS Lambda with the OpenWeatherMap API and managing requests through Amazon API Gateway, a lightweight and cost-effective solution was developed that eliminates the need for traditional server infrastructure.

This project highlights the strength of AWS services in building event-driven, on-demand applications that are both highly available and easy to maintain. It also emphasizes the importance of IAM role configuration for secure execution, API Gateway for request handling, and CloudWatch for logging and monitoring system performance.

The system offers a flexible foundation for further enhancements such as location-based forecasts, historical weather analysis, or mobile integration. Overall, the project reinforces the power of combining serverless computing with external APIs to create practical, real-world solutions in a modern cloud environment.

The knowledge gained from this project significantly contributes to building skills in AWS development, serverless architecture, API integration, and real-time data processing.

Amazon CloudWatch is a monitoring and observability service that provides data and actionable insights for AWS, applications, and services. It helps monitor logs, set alarms, and track metrics.

REFERENCES

[1] Amazon API Gateway Documentation Amazon Web Services. (n.d.). Amazon API Gateway. Retrieved from https://docs.aws.amazon.com/apigateway.

[2] AWS Lambda Documentation Amazon Web Services. (n.d.). AWS Lambda. Retrieved from https://docs.aws.amazon.com/lambda.

[3] OpenWeatherMap API Documentation OpenWeatherMap. (n.d.). API Documentation. Retrieved from https://openweathermap.org/api.

[4] AWS Identity and Access Management (IAM) Documentation Amazon Web Services. (n.d.). AWS IAM. Retrieved from https://docs.aws.amazon.com/iam.

[5] AWS Serverless Application Model (SAM) Documentation Amazon Web Services. (n.d.). AWS Serverless Application Model (SAM). Retrieved from https://docs.aws.amazon.com/serverless-application-model/.

[6] AWS Well-Architected Framework Amazon Web Services. (n.d.). AWS Well-Architected Framework. Retrieved from https://aws.amazon.com/architecture/well-architected/.

[7] OpenWeatherMap API Example Using AWS Lambda Amazon Web Services. (n.d.). Tutorial: Building a Weather API with AWS Lambda and OpenWeatherMap. Retrieved from https://aws.amazon.com/blogs/compute/weather-api-lambda-openweathermap/