**PRESENTATION**

For this project, I built a **Weather Data Collection System** using **AWS S3** and the **OpenWeather API**.

The goal was to collect real-time weather data and store it as JSON files in AWS S3.

Along the way, I applied several DevOps practices such as API integration, cloud storage management, Python development, and version control with Git.

**Consultant:**  
Can you explain the project in more detail? What were your objectives and deliverables?

**Me:**  
Of course. The project consisted of the following steps:

* **API Integration:** I fetched real-time weather data for multiple cities using the OpenWeather API.
* **Cloud Storage:** I stored time-stamped historical weather data securely in an **AWS S3 bucket**.
* **Python Backend:** I used Python to manage API calls, process data, and handle errors gracefully.
* **Version Control:** I used Git for versioning and collaboration, ensuring a clean and maintainable codebase.

**Consultant:**  
How did you prepare before starting the project?

**Me:**  
Before starting, I made sure to set up the necessary environment and tools:

* **AWS S3 and IAM:** I created IAM users for secure access to AWS resources and configured the access key and secret access key.
* **OpenWeather API Key:** I obtained an API key and securely stored it in a .env file.
* **Project Structure:** I organized the project with a clean folder structure, creating directories like src, and data. I also used a .gitignore file to exclude sensitive or unnecessary files from version control.

**Consultant:**  
What technologies and tools did you use for the project?

**Me:**  
For this project, I used the following tools and libraries:

* **boto3:** The AWS SDK for Python to interact with S3 buckets programmatically.
* **requests:** To fetch real-time weather data from the OpenWeather API. HTPP request
* **python-dotenv:** To securely manage sensitive information like API keys and bucket names.

I listed all dependencies in a requirements.txt file and installed them using the command:

bash

pip install -r requirements.txt

**Consultant:**  
How did you run and test the system?

**Me:**  
After setting everything up, I executed the Python script using the following command:

bash

python3 src/weather\_dashboard.py

As a result, the system successfully created an S3 bucket and began storing live weather data in JSON format.

I also implemented error-handling mechanisms to ensure the script could gracefully manage unexpected issues, like API failures or network errors.

**Consultant:**  
What did you learn from this project?

**Me:**  
Here are the key takeaways from this project:

1. **AWS S3 Management:** I gained hands-on experience managing S3 buckets programmatically using boto3.
2. **API Integration:** I developed practical skills in integrating external APIs into a Python application and handling dynamic data.
3. **Secure Credential Management:** I learned to manage sensitive information securely using .env files and the python-dotenv library.
4. **Version Control:** I improved my Git skills by maintaining a clean, well-documented, and collaborative codebase.

**Consultant:**  
What would you add if you wanted to improve or expand this project?

**Me:**  
If I were to take this project further, I would consider the following improvements:

* **AWS Lambda:** Move the script to a serverless architecture to reduce operational overhead and costs.
* **CloudWatch Monitoring:** Set up monitoring and alerting mechanisms to ensure the system runs reliably and to track any errors or anomalies.
* **CI/CD Pipelines:** Implement a continuous integration and deployment process using GitHub Actions or a similar tool to automate testing and deployment.

**What is an API Key?**

An **API key** is a unique identifier that acts as a credential for authenticating and authorizing access to an **API** (Application Programming Interface).

Think of it as a password that allows an application or user to interact with an API securely. It helps the API provider know:

1. **Who is accessing the API** (authentication).
2. **What permissions the user or application has** (authorization).
3. **How much of the API they are using** (tracking usage).

**What Does an API Key Do?**

1. **Authentication:**  
   The API key ensures that the request comes from a verified source. Only authorized users or systems with a valid API key can access the API.
2. **Authorization:**  
   API keys often define what actions or resources are available to the user. For instance, some keys may grant **read-only** access, while others allow **write** or **delete** operations.
3. **Usage Tracking and Rate Limiting:**  
   API providers monitor API key usage to ensure fair use and prevent abuse. For example, a free-tier API key might have limits like "100 requests per day."
4. **Billing and Account Association:**  
   The key helps tie API usage to your account, making it easy to track billing and usage statistics.

**Example in Action**

Imagine you're using the OpenWeather API to fetch weather data. To access the data, you'll need to include your API key in each API request like this:

https://api.openweathermap.org/data/2.5/weather?q=London&appid=YOUR\_API\_KEY

Here:

* **appid=YOUR\_API\_KEY** authenticates you to the API.
* Without this key, the API would reject your request.

**Why Is It Important?**

* **Security:** It restricts access to authorized users or applications.
* **Control:** API providers can revoke or limit an API key if it's misused.
* **Customization:** Different API keys can have different permission levels or usage limits.

By using API keys, APIs ensure a secure and controlled way for developers to access their services.