Student Data Management with AWS

Launch a Serverless Web App with AWS

PREPARED BY:
KENZA ELHAMCHI
ZAKARIA BENLAMKADAM

SUPERVISED BY:
HAYAT ROUTAIB

Content

Project Overview

- This project focuses on developing a web application for recording and retrieving student data using **AWS serverless technologies**.
- The goal was to create a scalable, cost-effective, and easy-to-maintain solution by leveraging serverless architecture.

Key Objectives :

- Simplify backend infrastructure management.
- Enable automatic scaling to handle traffic surges.
- Use a pay-as-you-go model to reduce costs.

Serverless??

Importancfe of Serverless Application

A serverless deployment is **not actually serverless**—it still involves a server, but that server is managed by cloud providers. This approach allows us to focus more on writing code rather than dealing with the headache of creating and managing infrastructure for better scalability

Importancfe of Serverless Application

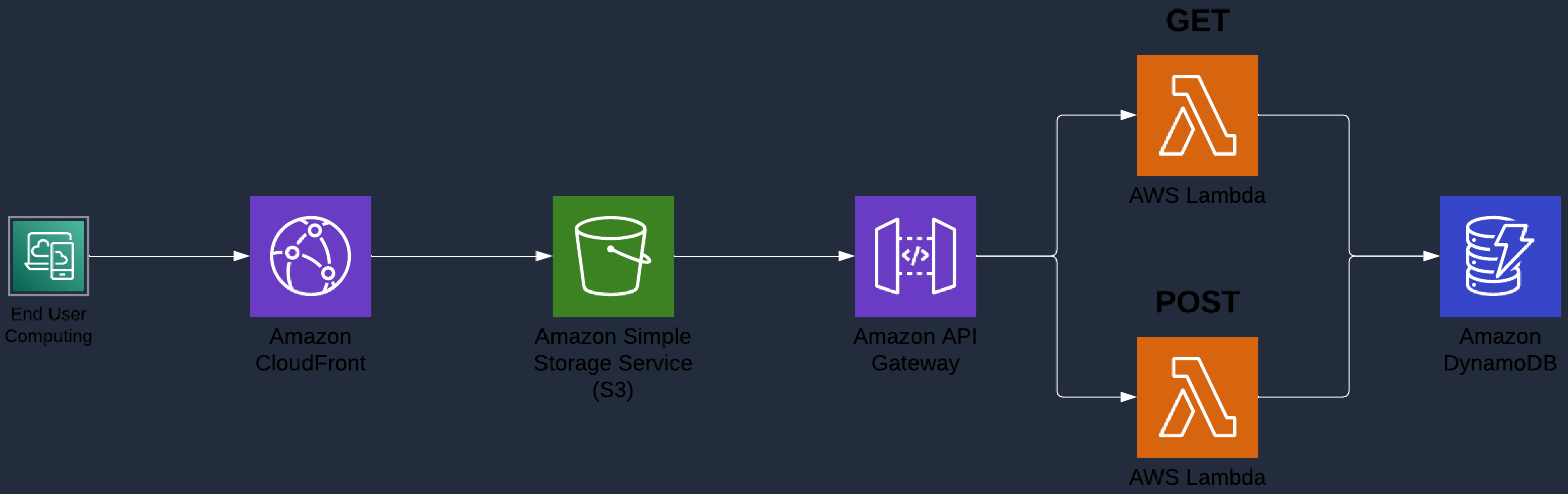
With serverless deployment, features like auto-provisioning and management are handled automatically, and it is especially beneficial for cost optimization.

For example, if you deploy your application on an EC2 instance with specific specifications like CPU, RAM, and storage, you are responsible for managing the entire infrastructure. Even if the traffic to your application is low throughout the day and the instance is underutilized, you still need to pay for the entire server, regardless of its usage time.

Importancfe of Serverless Application

In contrast, with serverless deployment, you only pay for the compute time your application uses. For our application deployment, we are leveraging AWS as our cloud provider.

Architecture



Amazon Simple Storage Service (S3)

Hosts static files like HTML, CSS, and JavaScript.

Benefits:

- Scalable storage with minimal costs.
- Static hosting ensures fast and reliable access to the website.

Amazon CloudFront

Delivers static content globally, caching it at edge locations.

Benefits:

- Low latency and fast load times.
- Automatic scaling for high traffic.

Amazon API Gateway

Handles HTTP requests and routes them to Lambda functions.

Benefits:

- Built-in security with request validation.
- Simplifies API management with minimal setup.

AWS Lambda

Executes code for API requests without server setup.

Functions Used:

- GET Function: Retrieves student data from DynamoDB.
- POST Function: Saves new student data in DynamoDB.

Benefits:

- Cost-effective pay-per-use model.
- Scales automatically to meet demand.

Amazon DynamoDB

Stores student records in a NoSQL format.

Benefits:

- High throughput with low latency.
- Auto-scaling to handle large datasets.

AWS PRICING

- **FLEXIBLE PAYMENT MODEL: WITH AWS, YOU ONLY PAY FOR THE SERVICES YOU USE, AS LONG AS YOU USE THEM, WITH NO LONG-TERM CONTRACTS OR COMPLEX LICENSING MANAGEMENT.**
- **AWS FREE TIER: OFFERS INCLUDE:**

SERVICES THAT ARE FREE FOREVER.
12-MONTH FREE TRIAL SERVICES AFTER SIGNING UP.
SHORT-TERM FREE TRIAL OFFERS.

- **AWS PRICING CALCULATOR: ESTIMATE THE COSTS OF YOUR ARCHITECTURAL SOLUTIONS.**
- **OPTIMIZE YOUR COSTS: LEARN THE STEPS TO EFFICIENTLY REDUCE YOUR AWS EXPENSES**

Project Features

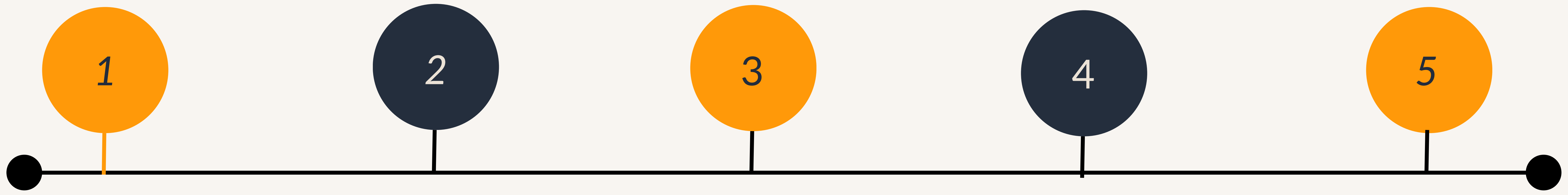
Data Entry

Data Retrieval

FEATURE 1: DATA ENTRY
USERS CAN INPUT STUDENT DETAILS SUCH AS ID, NAME, CLASS, AND AGE USING AN INTUITIVE FORM.
ONCE THE FORM IS FILLED, CLICKING THE "SAVE STUDENT DATA" BUTTON SENDS THE DATA TO THE BACKEND FOR VALIDATION AND SECURE STORAGE IN DYNAMODB.

FEATURE 2: DATA RETRIEVAL
USERS CAN RETRIEVE ALL STORED STUDENT INFORMATION BY CLICKING THE "VIEW ALL STUDENTS" BUTTON.
THE DATA IS FETCHED FROM DYNAMODB AND DISPLAYED IN A STRUCTURED TABLE FORMAT FOR EASY ACCESS AND REVIEW.
VISUAL PREVIEW:
THE ABOVE INTERFACE SHOWCASES BOTH THE DATA ENTRY FORM AND THE TABLE WHERE RETRIEVED DATA IS DISPLAYED.
THIS DEMONSTRATES A SEAMLESS USER EXPERIENCE FOR MANAGING STUDENT RECORDS.

Data Flow



EVENT 01

User enters data via the interface

EVENT 02

Request sent to API Gateway

EVENT 03

API Gateway triggers a Lambda function

EVENT 04

Lambda reads/writes data in DynamoDB

EVENT 05

Result returned to the user.

Technical Implementation

- **Frontend**

List important files like index.html, style.css, app.js.
Include snippets of relevant code if possible.

- **Backend:**

Add a snippet of a Lambda function (e.g., a POST function for saving data).

- **AWS Configuration:**

Highlight key configurations, such as:

1. **API Gateway with configured endpoints.**
2. **S3 bucket for hosting static files.**
3. **DynamoDB table schema for data storage.**

- ✓ **Frontend Setup**
- ✓ **Backend Development**
- ✓ **API Gateway Configuration**
- ✓ **AWS S3 for Static Hosting**
- ✓ **DynamoDB for Data Storage**

Backend development and API gateways ensure seamless communication across system components:frontend, backend, and database

Benefits and Limitations

BENEFITS

Scalability through AWS Lambda and DynamoDB.

Low cost: Pay-as-you-go model

Reduced maintenance with serverless architecture

LIMITATIONS

Dependency on AWS infrastructure

Learning curve for AWS services for new users

Latency issues: Network latency

Conclusion

Summary of the Project:

This project successfully implemented a scalable and cost-effective solution for managing student data using AWS cloud services. Key objectives were achieved, including automated data entry, seamless data retrieval, and a functional user interface supported by a serverless architecture with AWS Lambda, DynamoDB, API Gateway, and S3.

THANK YOU .

WELCOME TO ANY QUESTIONS .