Cloud-Based User Feedback Web Application

Introduction

Cloud computing and DevOps practices have become essential in modern application development due to their ability to deliver scalable, reliable, and cost-effective solutions. Organizations increasingly rely on cloud platforms to deploy applications that can handle varying user demands while ensuring high availability and security.

This project, “Cloud-Based User Feedback Web Application,” demonstrates the practical implementation of DevOps concepts using Amazon Web Services (AWS). The application is designed to allow users to access a static website, submit feedback through a web form, and store that feedback securely in a cloud-managed database. The system follows a cloud-native architecture by separating the frontend, backend, and database layers. The frontend of the application is hosted on Amazon S3 as a static website, providing high durability and global accessibility. The backend application runs on Amazon EC2, where it processes user requests and communicates with Amazon RDS (MySQL) to store feedback data. To ensure scalability and reliability, the architecture supports load balancing, auto scaling, and continuous monitoring using Amazon CloudWatch and SNS.

By completing this project, hands-on experience was gained in deploying real-world cloud infrastructure, configuring networking and security, integrating multiple AWS services, and applying DevOps best practices. This project serves as a comprehensive demonstration of building, deploying, monitoring, and managing a cloud-based web application using AWS.

Problem Statement

Many traditional web applications lack scalability, reliability, and proper monitoring when handling user feedback. This project aims to build a **cloud-based user feedback system** using AWS that allows users to submit feedback through a web interface, securely store the data in a managed database, and ensure high availability, scalability, and monitoring by following DevOps best practices.

Objective

The objective of this project is to **design, deploy, and manage a cloud-based web application** using AWS services and DevOps best practices. The application allows users to:

- Access a **static frontend website**
- Submit feedback through a **form**
- Store feedback securely in a **database**
- Monitor application health and availability

This project demonstrates hands-on experience with:

- AWS infrastructure
- Backend–frontend integration
- Cloud security
- Monitoring and cost optimization

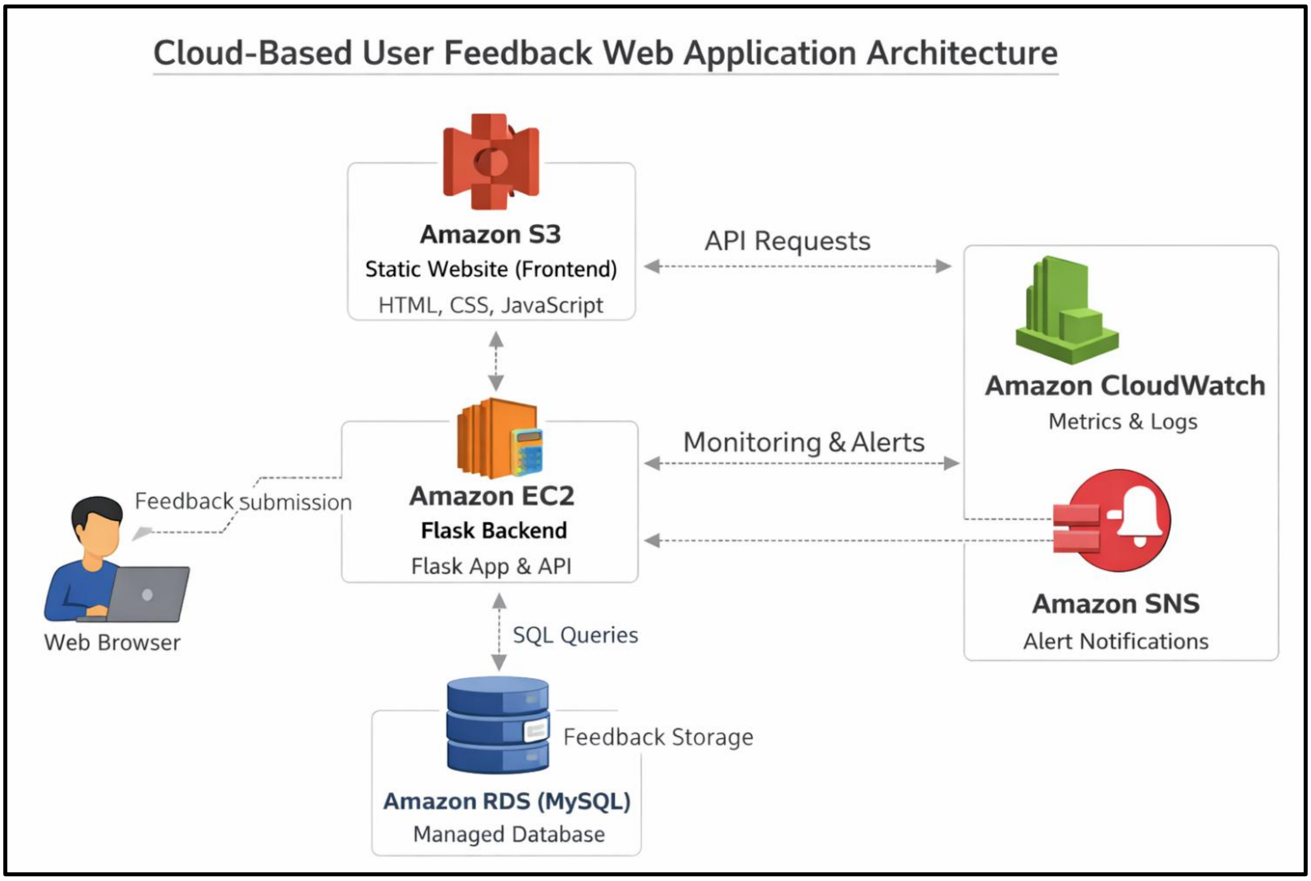
Use Case Description

Users access a static website hosted on Amazon S3. The website contains a feedback form where users submit their details. When the form is submitted:

1. The request is sent to a backend application running on EC2
2. The backend processes the request
3. Feedback data is stored in Amazon RDS (MySQL)
4. Application health is monitored using CloudWatch
5. Alerts can be sent using SNS (optional/extendable)

Architecture Overview

The application follows a three-tier architecture. The frontend is hosted on **Amazon S3** as a static website. User requests are processed by a **Flask backend running on Amazon EC2**, and feedback data is stored securely in **Amazon RDS (MySQL)**. Monitoring is enabled using **Amazon CloudWatch** to ensure application health and reliability.



Components Used

The following components were used in the implementation of this project:

1. **GitHub Repository**
   - Stores frontend and backend source code
   - Enables version control and future CI/CD integration
2. **Amazon S3 (Static Website Hosting)**
   - Hosts:
     1. index.html
     2. style.css
     3. script.js
     4. images/
   - Configured for **public read access**
   - Serves as the frontend layer
3. **Application Load Balancer (ALB)**
   - Routes HTTP requests from users to backend EC2 instances
   - Enables scalability and high availability
   - Health checks backend endpoints
4. **Amazon EC2 (Backend – Flask Application)**
   - Runs a Python Flask application
   - Exposes REST APIs:
   - /health
   - /feedback
   - Processes user requests and connects to RDS
5. **Amazon RDS (MySQL)**
   - Stores user feedback data
   - Managed database service
   - Ensures durability, backups, and security
6. **Amazon CloudWatch & SNS**
   - Monitors EC2 and RDS metrics
   - Logs application events
   - Can send alerts via SNS (email/SMS) on failures

Technologies Used

Layer	Technology
Frontend	HTML, CSS, JavaScript
Backend	Python, Flask
Database	Amazon RDS (MySQL)
Hosting	Amazon S3
Compute	Amazon EC2
Networking	VPC, Security Groups
Monitoring	CloudWatch
Alerts	SNS
Version Control	GitHub

Step-by-Step Implementation

Step 1: Frontend Development

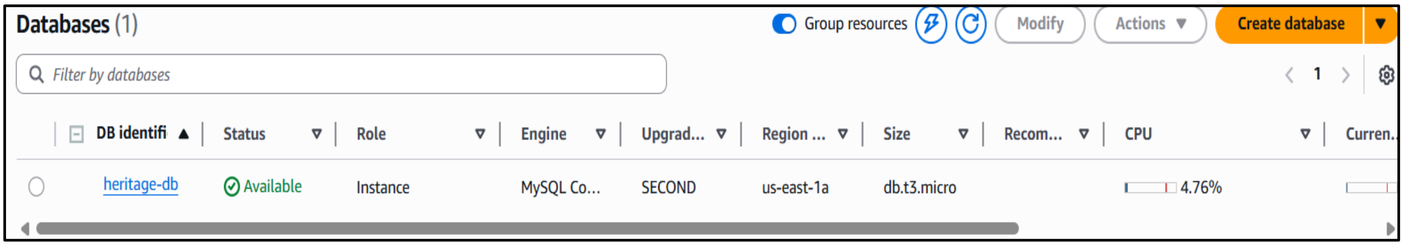
- Created a static website using HTML, CSS, and JavaScript
- Added a feedback form with validation
- Used JavaScript fetch() to call backend API

Step 2: Backend Development

- Developed Flask application with:
  - /health endpoint for monitoring
  - /feedback endpoint to accept POST requests
- Connected Flask app to RDS using pymysql
- Enabled CORS for frontend-backend communication

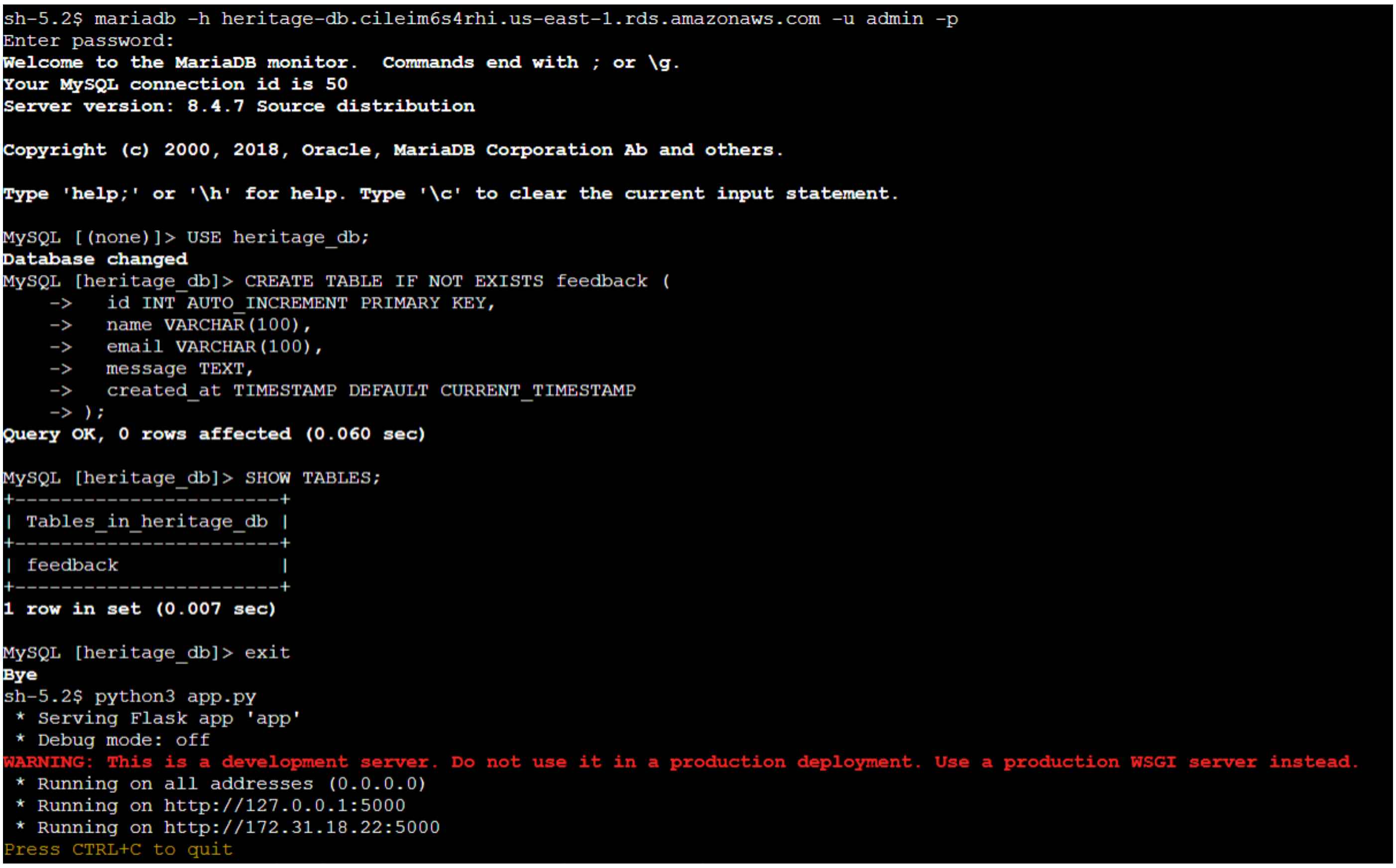
Step 3: Database Setup (RDS)

- Created MySQL RDS instance
- Created database: heritage_db
- Created table: feedback
- Configured security group to allow EC2 access



Databases (1)

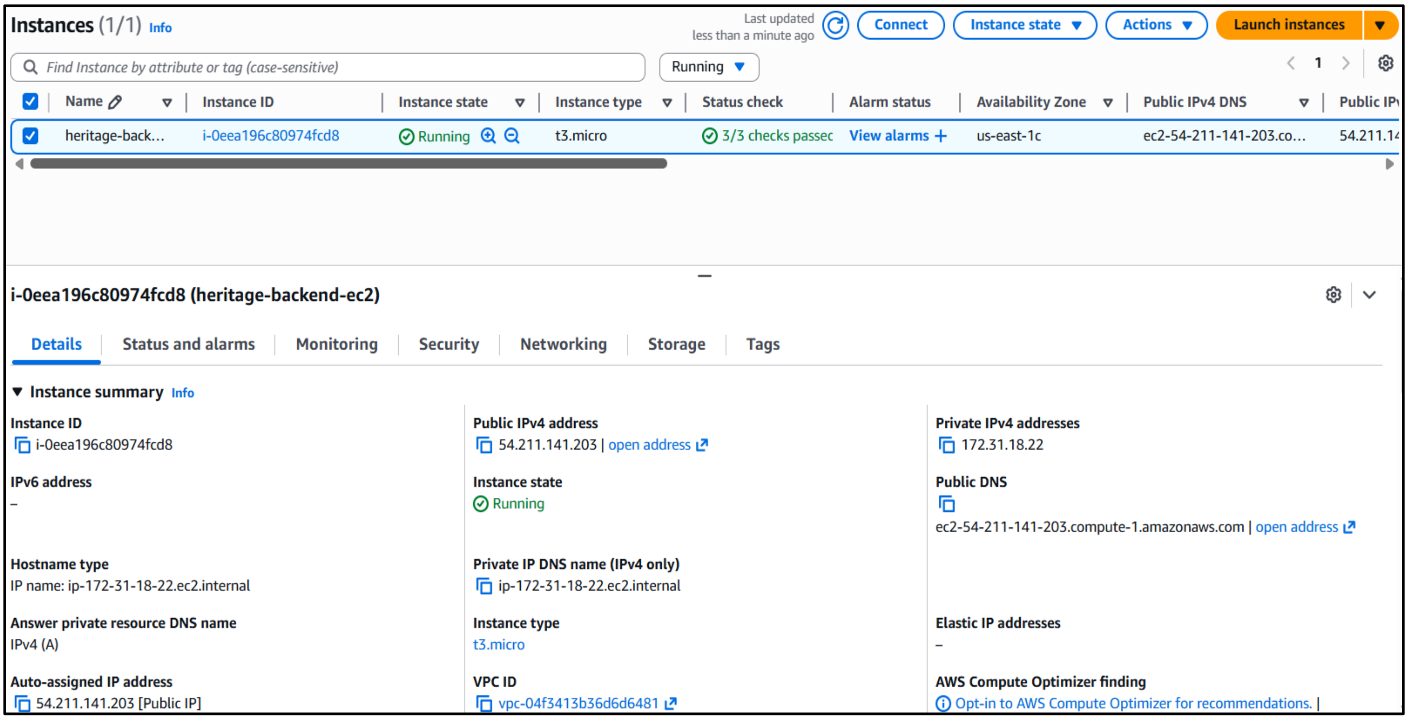
DB identifi ▲	Status	Role	Engine	Upgrad...	Region ...	Size	Recom...	CPU	Curren...
heritage-db	Available	Instance	MySQL Co...	SECOND	us-east-1a	db.t3.micro		4.76%	



```
sh-5.2$ mariadb -h heritage-db.cileim6s4rhi.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 50
Server version: 8.4.7 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> USE heritage_db;
Database changed
MySQL [heritage_db]> CREATE TABLE IF NOT EXISTS feedback (
    ->   id INT AUTO_INCREMENT PRIMARY KEY,
    ->   name VARCHAR(100),
    ->   email VARCHAR(100),
    ->   message TEXT,
    ->   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.060 sec)

MySQL [heritage_db]> SHOW TABLES;
+-----------------------+
| Tables_in_heritage_db |
+-----------------------+
| feedback              |
+-----------------------+
1 row in set (0.007 sec)

MySQL [heritage_db]> exit
Bye
sh-5.2$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.18.22:5000
Press CTRL+C to quit
```

Step 4: EC2 Deployment

- Launched EC2 instance
- Installed:
  - Python
  - Flask
  - MariaDB client
- Deployed backend code
- Tested API endpoints successfully



Instances (1/1) Info

Last updated less than a minute ago

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
☑	heritage-back...	i-0eea196c80974fcd8	✓ Running	t3.micro	✓ 3/3 checks passec	View alarms +	us-east-1c	ec2-54-211-141-203.co...	54.211.14

i-0eea196c80974fcd8 (heritage-backend-ec2)

▼ Instance summary Info

Instance ID
i-0eea196c80974fcd8

Public IPv4 address
54.211.141.203 | open address

Private IPv4 addresses
172.31.18.22

IPv6 address
–

Instance state
✓ Running

Public DNS
ec2-54-211-141-203.compute-1.amazonaws.com | open address

Hostname type
IP name: ip-172-31-18-22.ec2.internal

Private IP DNS name (IPv4 only)
ip-172-31-18-22.ec2.internal

Answer private resource DNS name
IPv4 (A)

Instance type
t3.micro

Elastic IP addresses
–

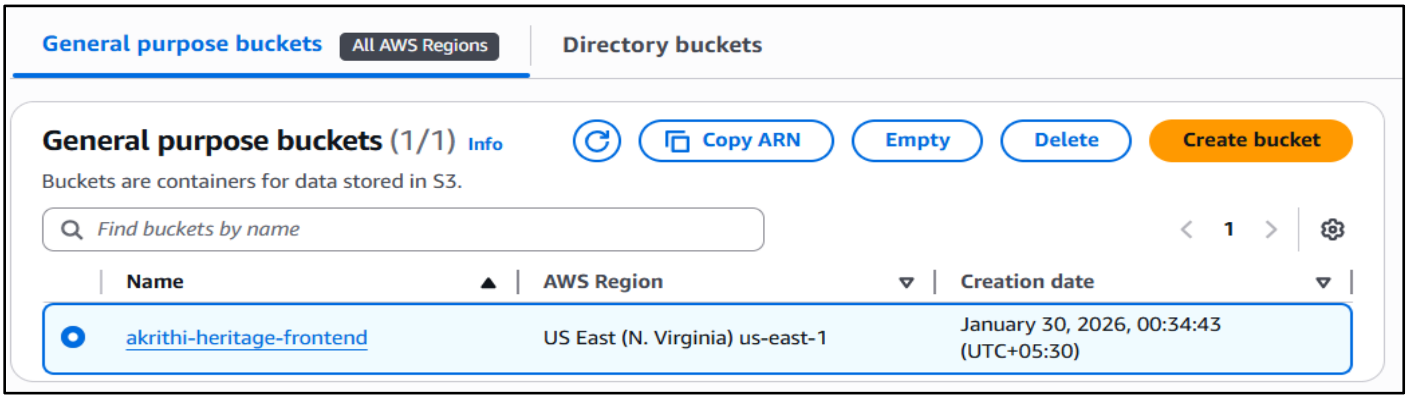
Auto-assigned IP address
54.211.141.203 [Public IP]

VPC ID
vpc-04f3413b36d6d6481

AWS Compute Optimizer finding
ⓘ Opt-in to AWS Compute Optimizer for recommendations. |

Step 5: S3 Static Website Hosting

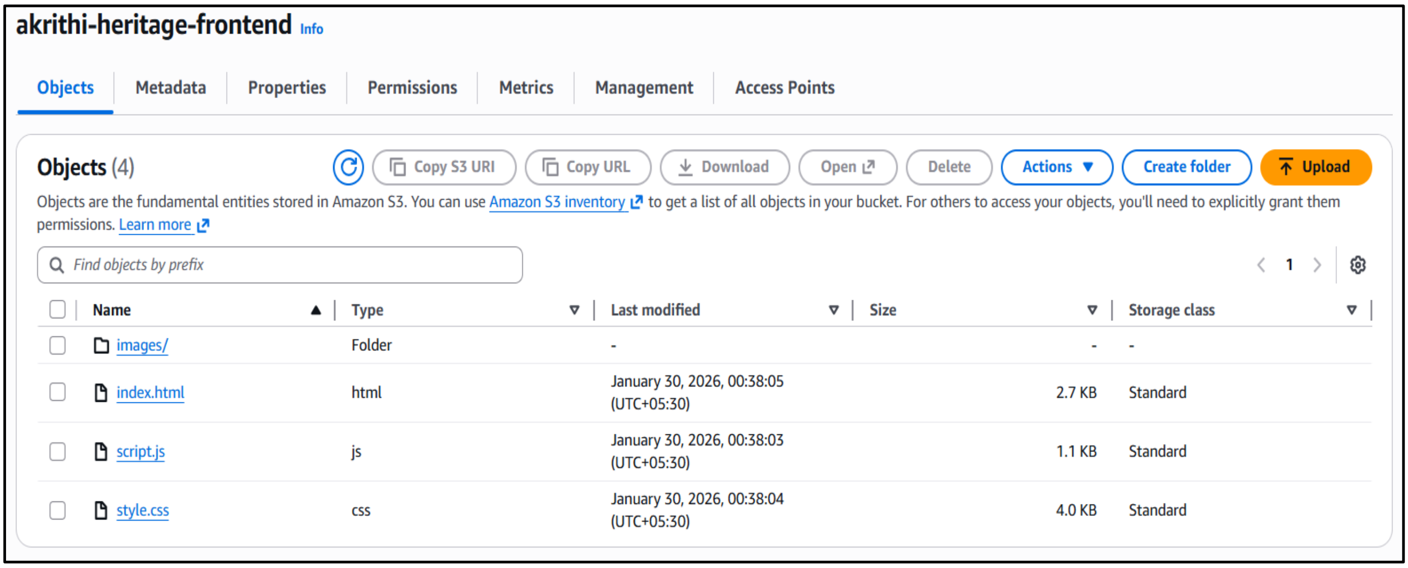
- Created S3 bucket
- Uploaded frontend files
- Disabled block public access
- Added bucket policy for public read
- Enabled static website hosting



General purpose buckets (1/1)

Buckets are containers for data stored in S3.

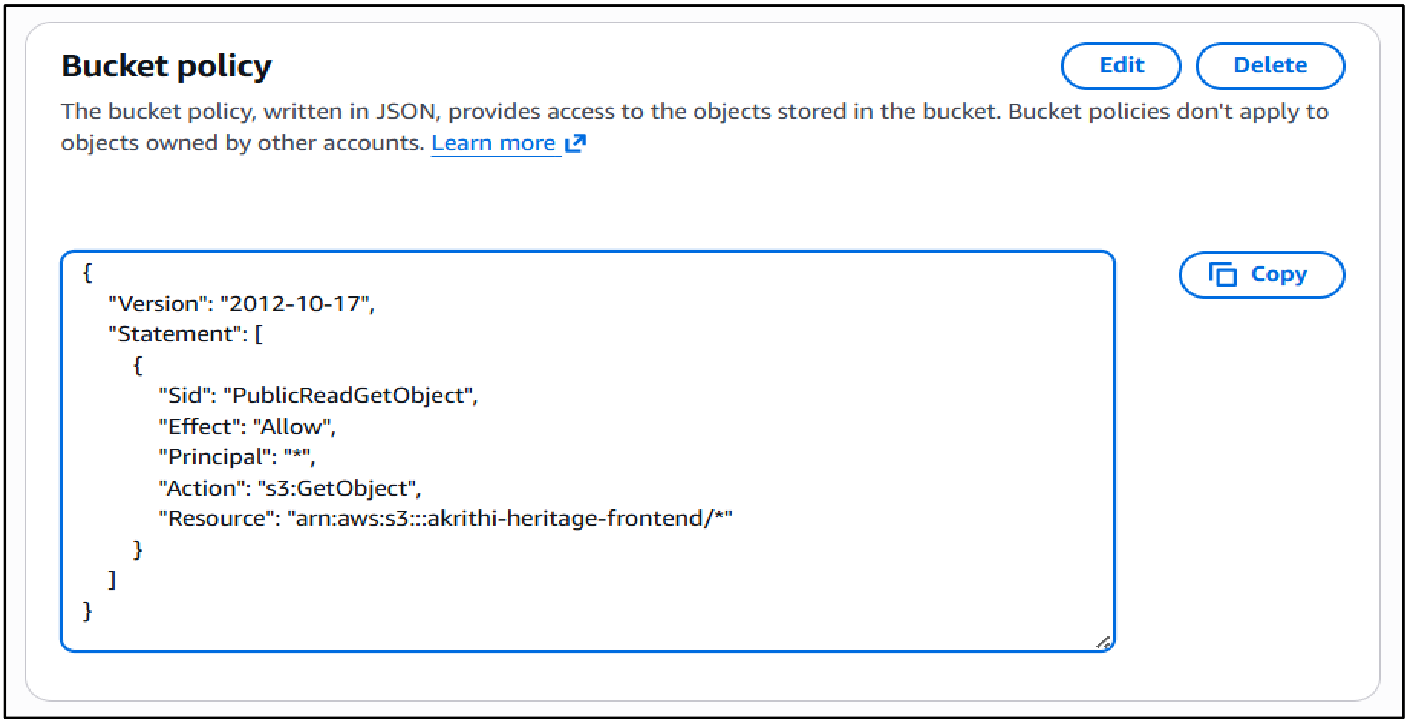
	Name	AWS Region	Creation date
●	akrithi-heritage-frontend	US East (N. Virginia) us-east-1	January 30, 2026, 00:34:43 (UTC+05:30)



Objects (4)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

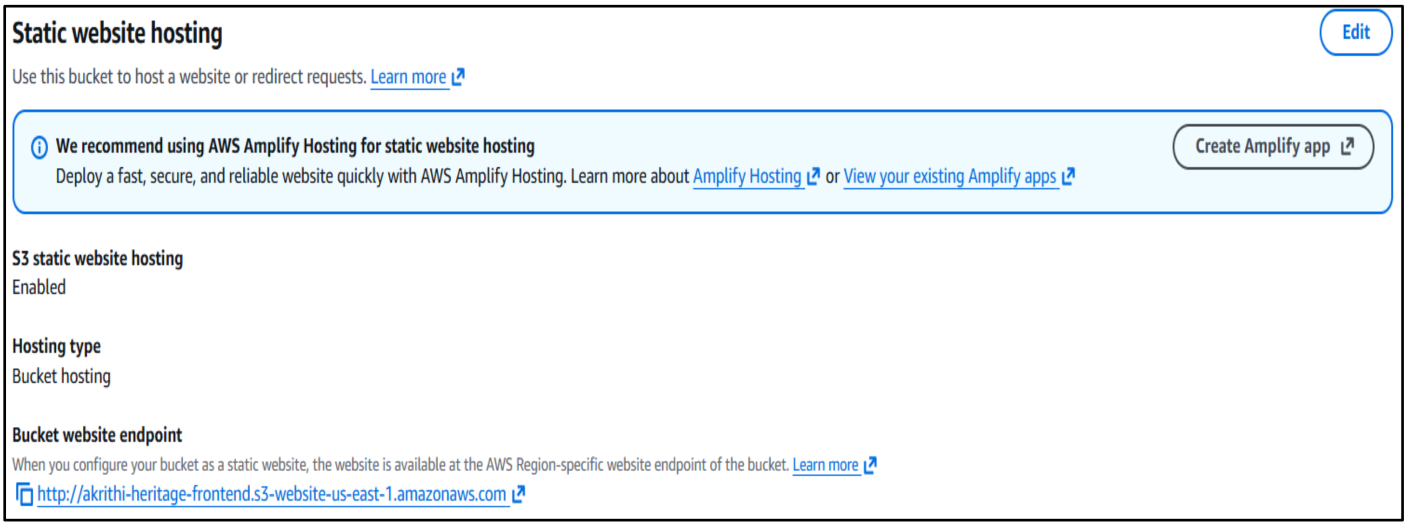
□	Name	Type	Last modified	Size	Storage class
□	images/	Folder	-	-	-
□	index.html	html	January 30, 2026, 00:38:05 (UTC+05:30)	2.7 KB	Standard
□	script.js	js	January 30, 2026, 00:38:03 (UTC+05:30)	1.1 KB	Standard
□	style.css	css	January 30, 2026, 00:38:04 (UTC+05:30)	4.0 KB	Standard



Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::akrithi-heritage-frontend/*"
    }
  ]
}
```



Static website hosting

Use this bucket to host a website or redirect requests. Learn more

We recommend using AWS Amplify Hosting for static website hosting

Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. Learn more about Amplify Hosting or View your existing Amplify apps

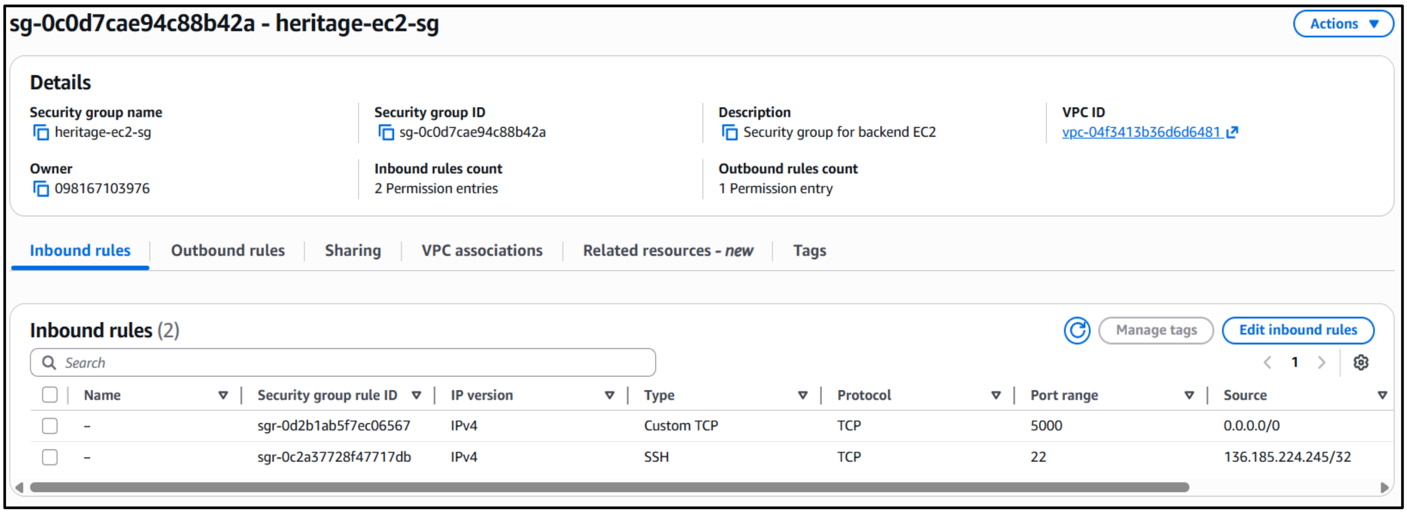
S3 static website hosting
Enabled

Hosting type
Bucket hosting

Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Learn more
http://akrithi-heritage-frontend.s3-website-us-east-1.amazonaws.com

Step 6: Networking & Security

- Configured:
  - Security groups
  - Inbound rules for HTTP (5000)
  - Restricted SSH access
- Ensured EC2 → RDS connectivity



sg-0c0d7cae94c88b42a - heritage-ec2-sg

Details

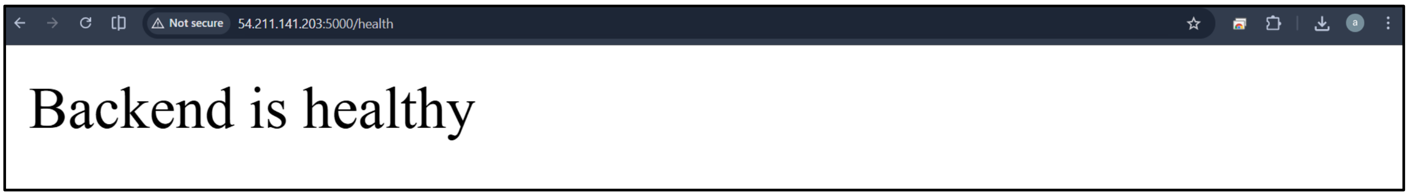
Security group name	Security group ID	Description	VPC ID
heritage-ec2-sg	sg-0c0d7cae94c88b42a	Security group for backend EC2	vpc-04f3413b36d6d6481
Owner	**Inbound rules count**	**Outbound rules count**	
098167103976	2 Permission entries	1 Permission entry	

Inbound rules (2)

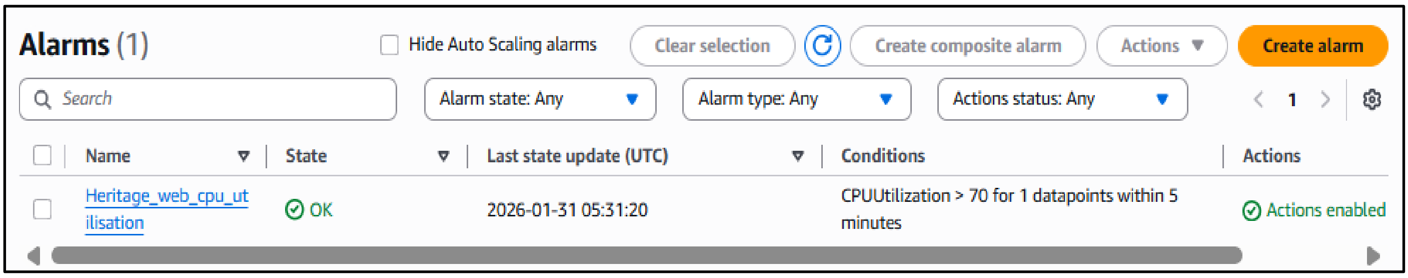
□	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
□	-	sgr-0d2b1ab5f7ec06567	IPv4	Custom TCP	TCP	5000	0.0.0.0/0
□	-	sgr-0c2a37728f47717db	IPv4	SSH	TCP	22	136.185.224.245/32

Step 7: Monitoring & Cost Optimization

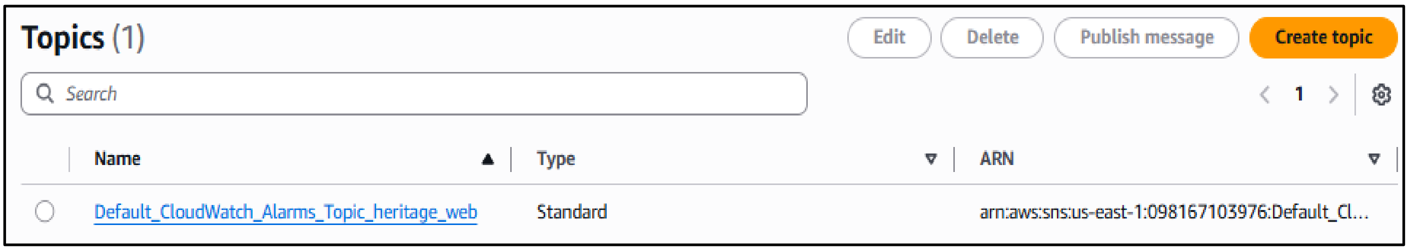
- Enabled CloudWatch metrics
- Monitored EC2 instance health and application availability
- Created CloudWatch alarms for resource monitoring
- Integrated Amazon SNS to send alert notifications
- Verified application health using /health endpoint
- Stopped EC2 and RDS when not in use
- Took RDS snapshot for safety



Backend is healthy



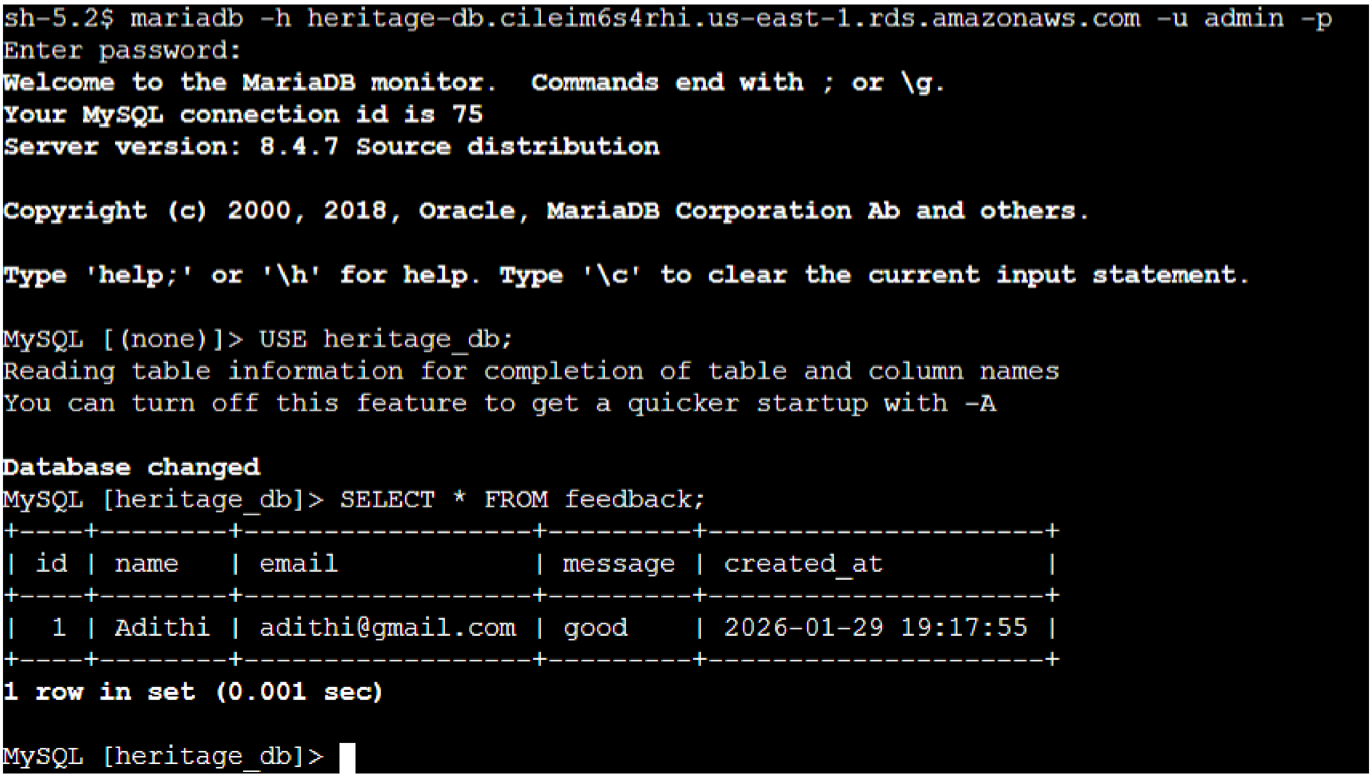
□	Name	State	Last state update (UTC)	Conditions	Actions
□	Heritage_web_cpu_utilisation	✓ OK	2026-01-31 05:31:20	CPUUtilization > 70 for 1 datapoints within 5 minutes	✓ Actions enabled



	Name	Type	ARN
○	Default_CloudWatch_Alarms_Topic_heritage_web	Standard	arn:aws:sns:us-east-1:098167103976:Default_Cl…

Testing Validation

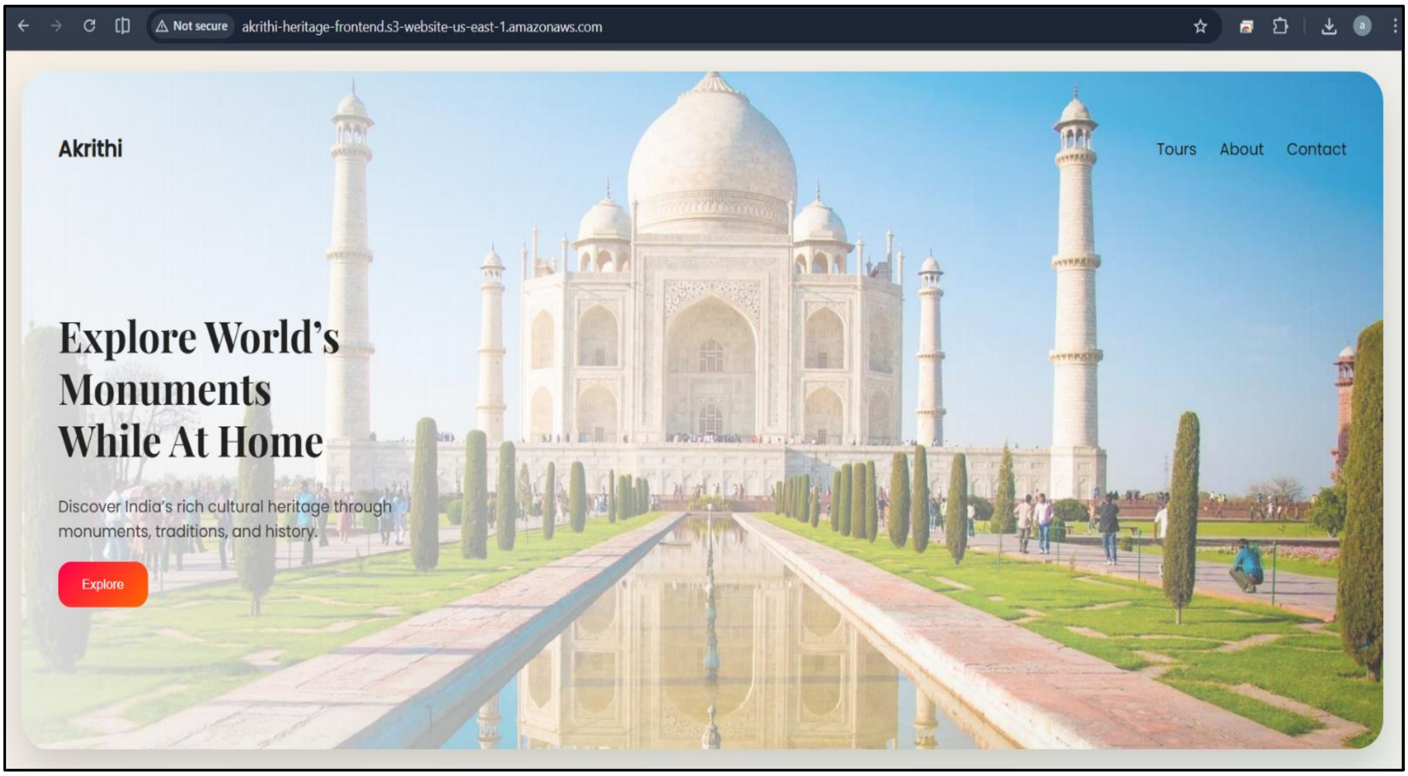
- Frontend loads from S3
- Feedback form submits data
- Backend API responds correctly
- Data stored in RDS
- Health endpoint returns success
- Architecture works end-to-end



```
sh-5.2$ mariadb -h heritage-db.cileim6s4rhi.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 75
Server version: 8.4.7 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> USE heritage_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [heritage_db]> SELECT * FROM feedback;
+----+--------+------------------+---------+---------------------+
| id | name   | email            | message | created_at          |
+----+--------+------------------+---------+---------------------+
|  1 | Adithi | adithi@gmail.com | good    | 2026-01-29 19:17:55 |
+----+--------+------------------+---------+---------------------+
1 row in set (0.001 sec)

MySQL [heritage_db]>
```

Project Outcome

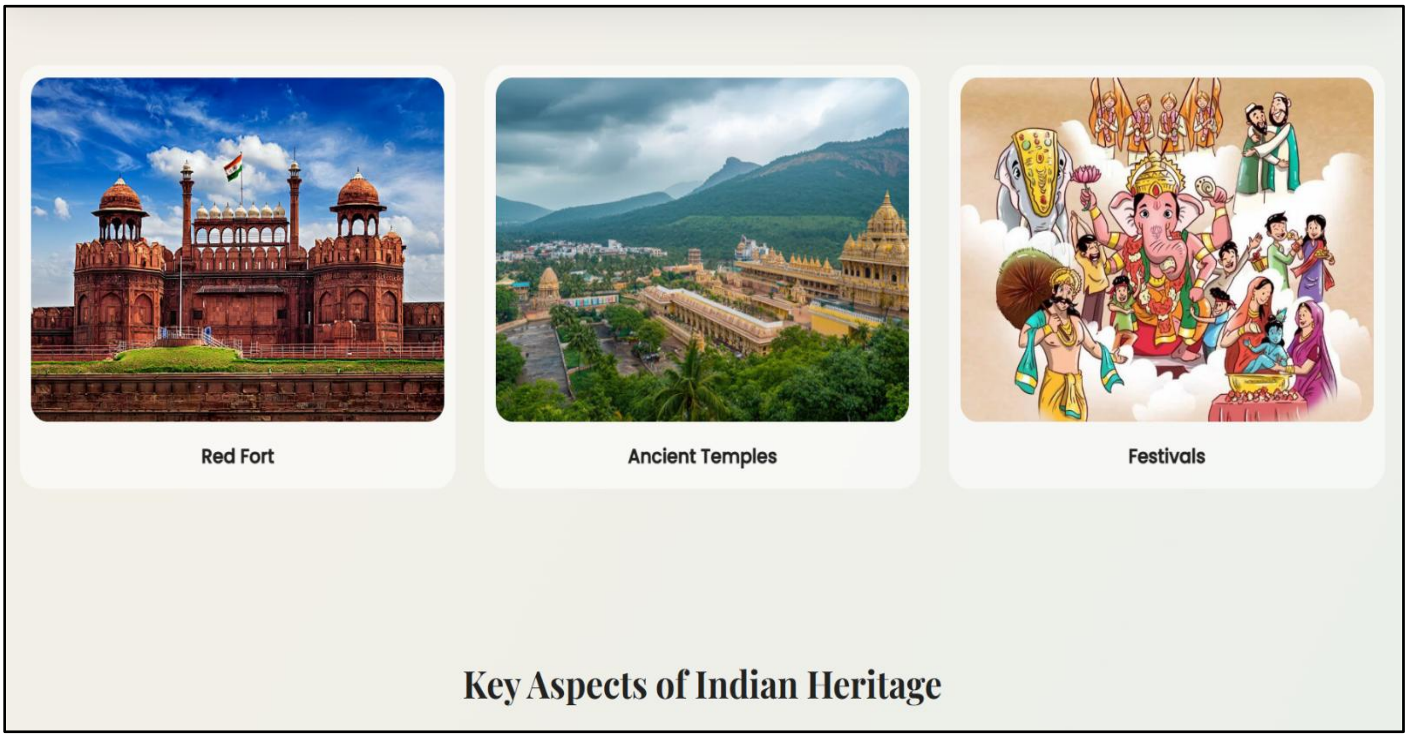
The project successfully delivered a scalable, cloud-based web application using AWS services, following DevOps best practices throughout the deployment process. The application was securely deployed with proper access control, networking, and service integration, ensuring reliable communication between the frontend, backend, and database layers. Monitoring, security, and cost-optimization measures were implemented to maintain performance and operational efficiency. Overall, the project fulfils all capstone requirements and demonstrates a practical understanding of cloud-native architecture and DevOps principles.



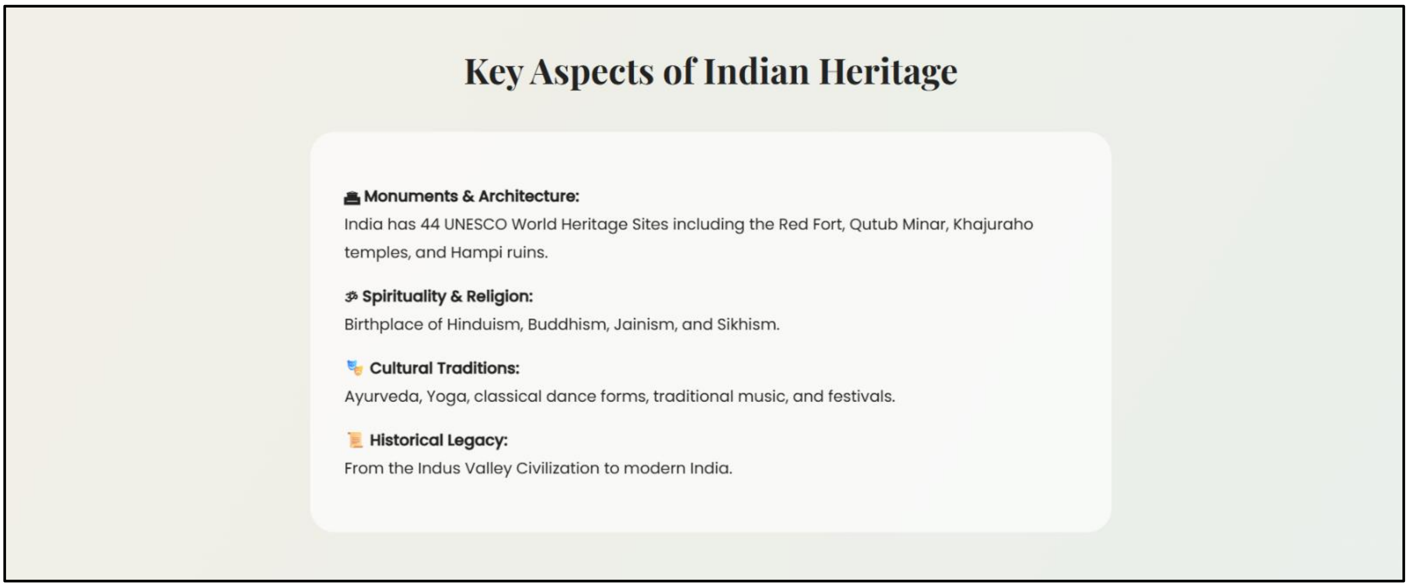
Not secure akrithi-heritage-frontend.s3-website-us-east-1.amazonaws.com

Explore World's Monuments While At Home

Discover India's rich cultural heritage through monuments, traditions, and history.



Key Aspects of Indian Heritage



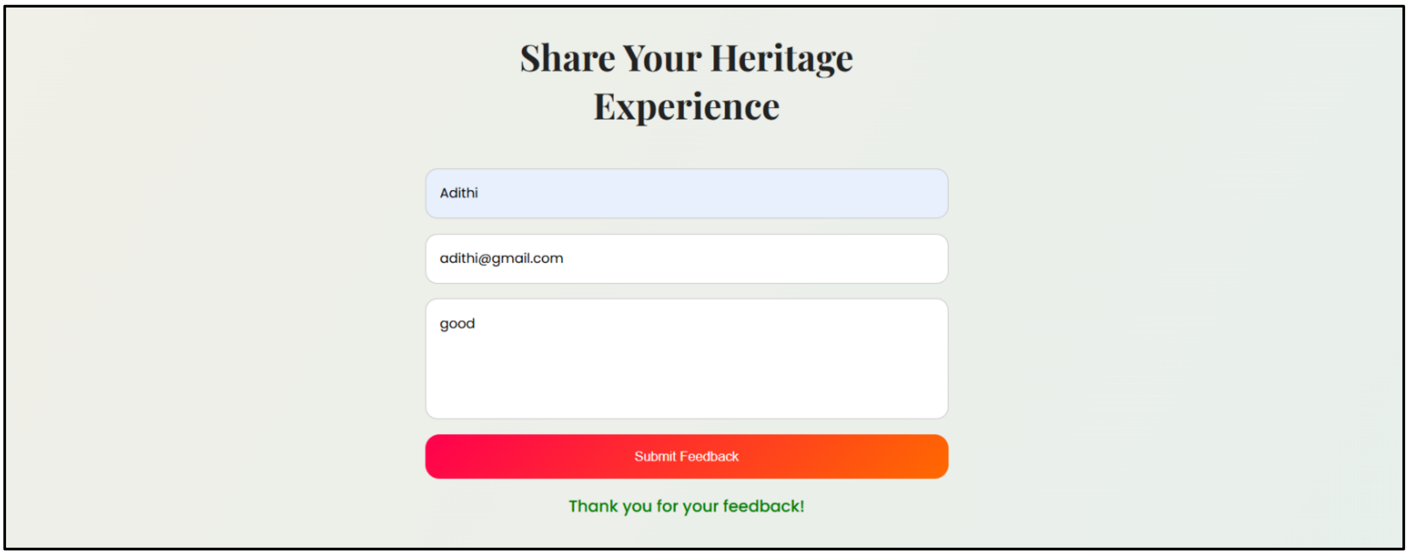
Key Aspects of Indian Heritage

🏛 **Monuments & Architecture:**
India has 44 UNESCO World Heritage Sites including the Red Fort, Qutub Minar, Khajuraho temples, and Hampi ruins.

🕉 **Spirituality & Religion:**
Birthplace of Hinduism, Buddhism, Jainism, and Sikhism.

🎭 **Cultural Traditions:**
Ayurveda, Yoga, classical dance forms, traditional music, and festivals.

📜 **Historical Legacy:**
From the Indus Valley Civilization to modern India.



Share Your Heritage Experience

Adithi

adithi@gmail.com

good

Thank you for your feedback!

Conclusion

This project successfully demonstrates the deployment of a cloud-based user feedback web application using AWS services. Amazon S3 is used for hosting the frontend, Amazon EC2 runs the Flask backend, and Amazon RDS (MySQL) securely stores user feedback data. Amazon CloudWatch is used to monitor application health and performance, while Amazon SNS enables alert notifications for system events. Overall, the project highlights the effective use of AWS and DevOps best practices to build a scalable, reliable, and monitored cloud-native application.

By -
R. Adithi
Aspiring DevOps Engineer
Batch - 06