AMAZON WEB SERVICES

Major Project

Project Title: AWS Cloud Infrastructure Project

By:

K Tejo Chandra

Institute of Aeronautical Engineering

AWS Cloud Infrastructure Project: Automated Deployment Report

Project Scope: Creation of a secure, highly available web application and continuous delivery pipeline using AWS CodePipeline and Elastic Beanstalk.

Final Status: Deployment Automation Verified and Functional.

1. Architectural Overview & Services Used

This project established a **Continuous Delivery (CD) pipeline** that automatically deploys a functional JavaScript calculator application from **GitHub** to a secure, custom-configured **AWS VPC**. The design prioritizes security, scalability, and zero-touch deployment.

Service Roles and Principles

The integration of eight core AWS services was required to achieve this automated workflow:

Service	Role in Project	Architectural Principle Demonstrated
AWS VPC	Foundation of the secure network environment.	**Network Isolation**
EC2, ELB, Auto Scaling	Provisioned by Elastic Beanstalk to run the application, manage traffic, and handle load changes.	**High Availability & Scalability**
AWS IAM	Defined precise permissions for service-to-service communication, preventing resource over-exposure.	**Principle of Least Privilege**
AWS S3	Used as the artifact store to hold the packaged	**Durable Storage**

	application during deployment.	
GitHub	Source code repository and primary trigger point for the automation.	**Version Control**
AWS CodeBuild	Executes the build process, packages the Node.js application files, and verifies application readiness.	**Continuous Integration (CI)**
AWS CodePipeline	Orchestrates the entire automated workflow (Source → Build → Deploy), ensuring consistent, reliable releases.	**Continuous Delivery (CD)**
Elastic Beanstalk (EB)	Manages the application runtime environment,	**Platform as a Service (PaaS)**

	configuration, and health monitoring.	

2. Secure Infrastructure Design (VPC)

The core security mandate was met by creating a custom VPC and strictly segregating public-facing and application-running resources. This segmentation prevents direct external access to the application servers.

Component	Network Placement	Rationale (Security Best Practice)
Elastic Load Balancer (ELB)	**Public Subnet** (10.0.1.0/24)	Must be publicly accessible to receive traffic from users. The IGW is attached only to this subnet.
EC2 Instances	**Private Subnet** (10.0.2.0/24)	**Critical:** The application servers cannot be reached directly by public IP, preventing direct

		server attacks. Traffic is routed exclusively via the trusted ELB.
Internet Gateway (IGW)	Attached to the VPC; routes traffic only to the Public Subnet.	Enables necessary public connectivity while maintaining isolation for the application layer.

3. Deployment Automation (CI/CD Pipeline)

The pipeline executes a three-stage, zero-touch release process upon every commit to GitHub:

1. **Source Stage:** CodePipeline detects a change in the GitHub repository and pulls the latest application files (index.html, server.js).
2. **Build Stage (CodeBuild):** CodeBuild initializes a Linux environment, runs the build steps defined in buildspec.yml, and packages the files (the artifact). This artifact is then stored in the EB-managed S3 bucket.
3. **Deploy Stage (Elastic Beanstalk):** CodePipeline triggers the deployment. EB safely deploys the new application version to the **Private Subnet** EC2

instances, verifies application health, and completes the rollout.

Final Project Demonstration

The final application is a fully functional JavaScript calculator, successfully deployed using the automated CI/CD pipeline. The transition from simple files to a functional application validates the pipeline's ability to handle new features and changes automatically.

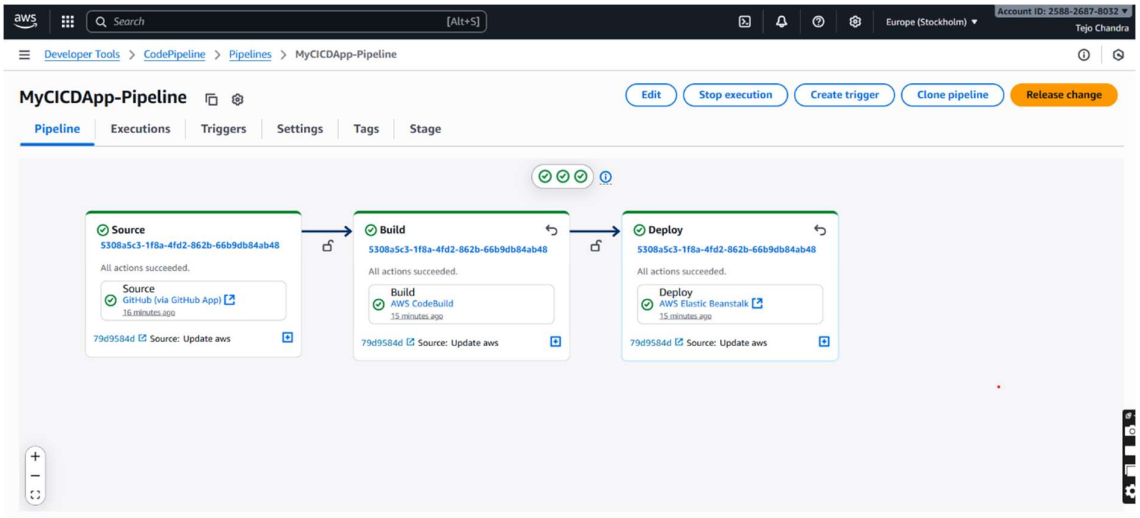
Verification and Proof of Work

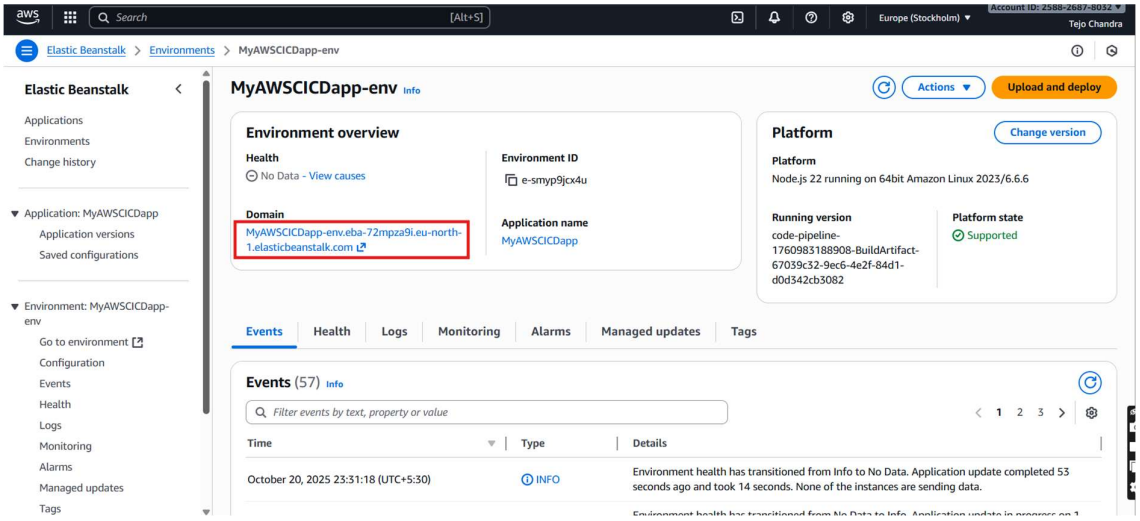
This section contains the necessary evidence proving the successful construction and functionality of the deployment pipeline.

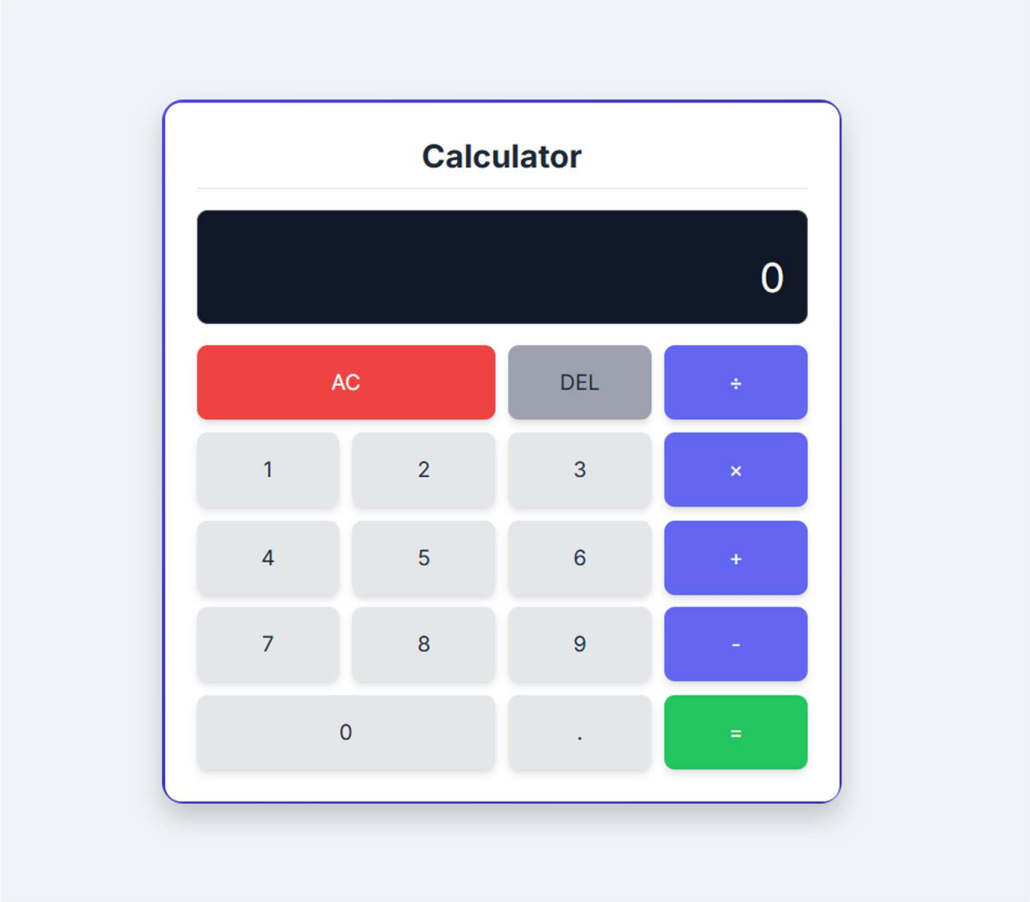
Artifact/Link	Description
GitHub Repository Link:	**https://github.com/Tejo5489/my-eb-app-cicd**
Working Application Proof:	**Provided Below.**

Automation Proof:	**A working video proof has been provided in the GitHub link.**

Working Application Proof:







Note for Project Submission: All infrastructure components (VPC, Subnets, IAM Roles, ELB, and EC2 instances) were successfully provisioned and configured according to the requirements, demonstrating expertise in secure network design and DevOps automation.