

Case Study Report: Continuous Deployment using AWS CodePipeline and Elastic Beanstalk

Name: Shreyas Nitin Naik

Roll No.: 41

Class: D15B

Subject: Advance DevOps Lab

Email ID: shreyasnai1039@gmail.com

1. Introduction

Continuous deployment is a pivotal practice in modern software development, enabling automated and rapid delivery of code changes to production environments. In this case study, we explore AWS CodePipeline in combination with AWS Elastic Beanstalk to achieve a fully automated CI/CD pipeline. The report will walk through each step in deploying a sample application, and highlight how AWS services streamline this process.

2. Key Feature & Application

Key Feature:

The unique feature of this case study is the integration of **AWS CodePipeline with Elastic Beanstalk**, which allows for automatic deployment of application updates based on changes detected in a GitHub repository. This integration ensures rapid and reliable code delivery with minimal manual intervention.

Application:

This setup is ideal for organizations looking to automate their deployment pipelines, reduce the risk of manual errors, and improve their time-to-market. It is applicable in environments where continuous integration, continuous testing, and continuous deployment (CI/CD) are critical for development efficiency.

3. Tools Used

- **AWS CodePipeline:** For creating the CI/CD pipeline.
- **AWS CodeBuild:** (Optional) For compiling and testing the code.
- **AWS Elastic Beanstalk:** For deploying the application to an EC2 instance.
- **IAM (Identity and Access Management):** For creating and managing roles and permissions.
- **GitHub:** As the source repository for code.

You may access these tools through your AWS Academy portal or personal AWS login.

4. Detailed Step-by-Step Process

Step 1: Setting Up IAM Roles

- Go to the **IAM Console** and create a new IAM role with the necessary permissions for EC2, S3, and CodePipeline. This role will be assigned to the EC2 instance later.

The screenshot shows the 'Create role' wizard at Step 1: 'Select trusted entity'. It displays a 'Trusted entity type' section with five options: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Below this is a 'Use case' section for EC2 services. The sidebar shows steps 2 and 3: 'Add permissions' and 'Name, review, and create'.

The screenshot shows the 'Create role' wizard at Step 3: 'Name, review, and create'. It includes 'Role details' fields for 'Role name' (Shreyas-CaseStudy) and 'Description' (Allows EC2 instances to call AWS services on your behalf). At the bottom, it shows the 'Step 1: Select trusted entities' section with a trust policy JSON code:

```
1 < [ { "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [
```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AWSCodeDeployFullAccess	AWS managed	Permissions policy
AWSCodeDeployRole	AWS managed	Permissions policy
AWSElasticBeanstalkMulticontainerDocker	AWS managed	Permissions policy
AWSElasticBeanstalkWebTier	AWS managed	Permissions policy
AWSElasticBeanstalkWorkerTier	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional [Info](#)
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#)
You can add up to 50 more tags.

Role Shreyas-CaseStudy created.

IAM > Roles

Roles (4) [Info](#)

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linker)	2 days ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linker)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linker)	-
Shreyas-CaseStudy	AWS Service: ec2	-

Roles Anywhere [Info](#)

Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads

X.509 Standard

Temporary credentials

Step 2: Creating an Elastic Beanstalk Application

- Navigate to the **Elastic Beanstalk** console.
- Create a new application and give it a meaningful name (e.g., "MySampleApp").
- Select a platform like **PHP** or **Node.js** based on your application's requirements.
- Assign the IAM role created in Step 1 to the EC2 instance profile in the Elastic Beanstalk settings.

Configure environment Info

Environment tier Info
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.

Web server environment
Run a website, web application, or web API that serves HTTP requests. [Learn more](#)

Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. [Learn more](#)

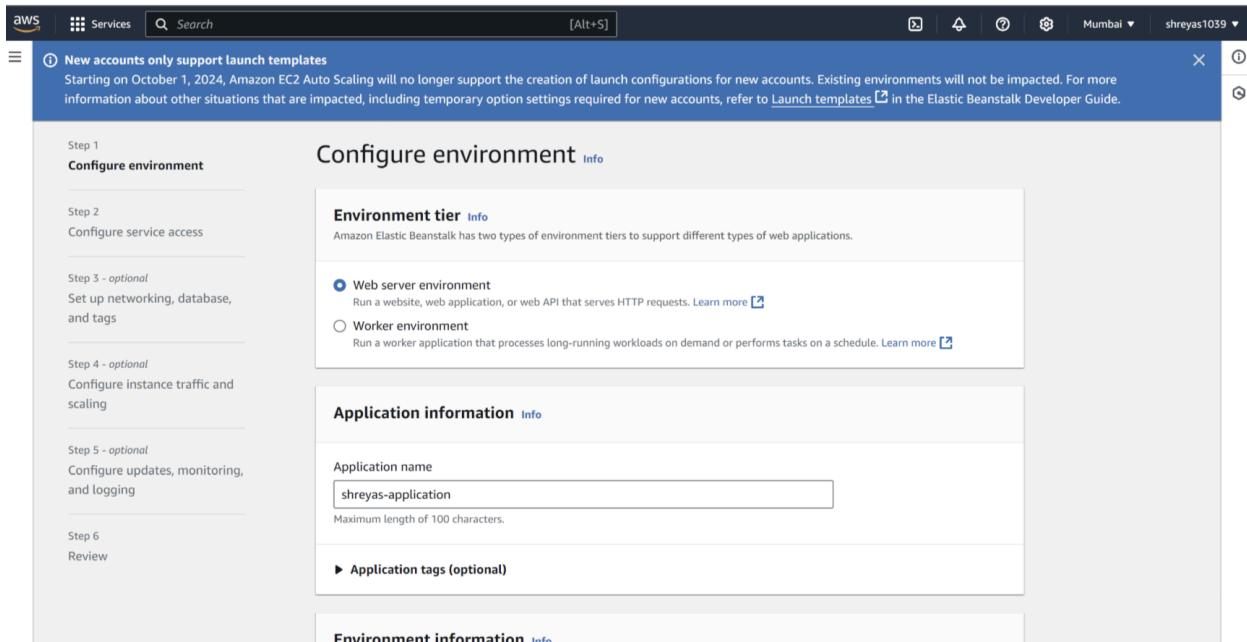
Application information Info

Application name

Maximum length of 100 characters.

► Application tags (optional)

Environment information Info



Platform type

Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#)

Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.

Platform

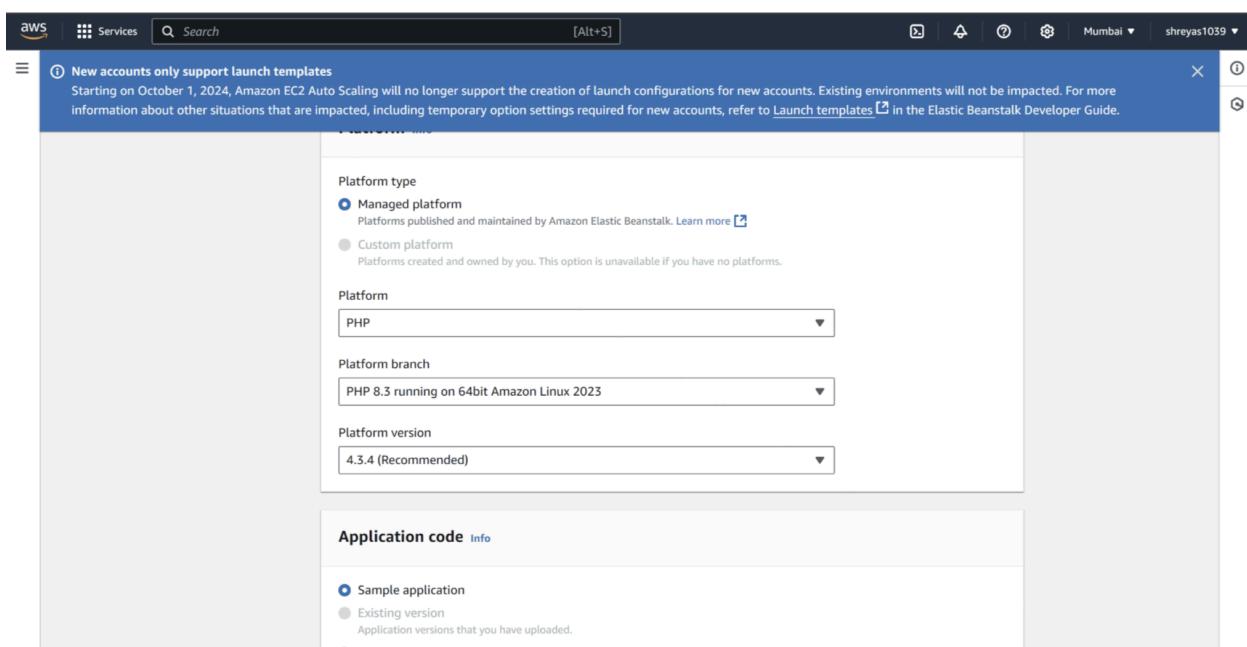
Platform branch

Platform version

Application code Info

Sample application

Existing version
Application versions that you have uploaded.



Configure service access Info

Service access

IAM roles, assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. [Learn more](#)

Service role

Create and use new service role
 Use an existing service role

Existing service roles

Choose an existing IAM role for Elastic Beanstalk to assume as a service role. The existing IAM role must have the required IAM managed policies.

Shreyas-CaseStudy C

EC2 key pair

Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)

Choose a key pair C

EC2 instance profile

Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.

Shreyas-CaseStudy C

[View permission details](#)

Cancel Skip to review Previous Next

Step 3: Configuring VPC and EC2 Instance

- Choose the appropriate **Virtual Private Cloud (VPC)** and ensure the EC2 instance has a public IP.
- Confirm that the application environment is successfully launched.

Set up networking, database, and tags - optional Info

Virtual Private Cloud (VPC)

VPC

Launch your environment in a custom VPC instead of the default VPC. You can create a VPC and subnets in the VPC management console. [Learn more](#)

vpc-0e02654175d6b189b | (172.16.0.0/16) | ajay-vpc C

[Create custom VPC](#)

Instance settings

Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances. [Learn more](#)

Public IP address

Assign a public IP address to the Amazon EC2 instances in your environment.

Activated

Instance subnets

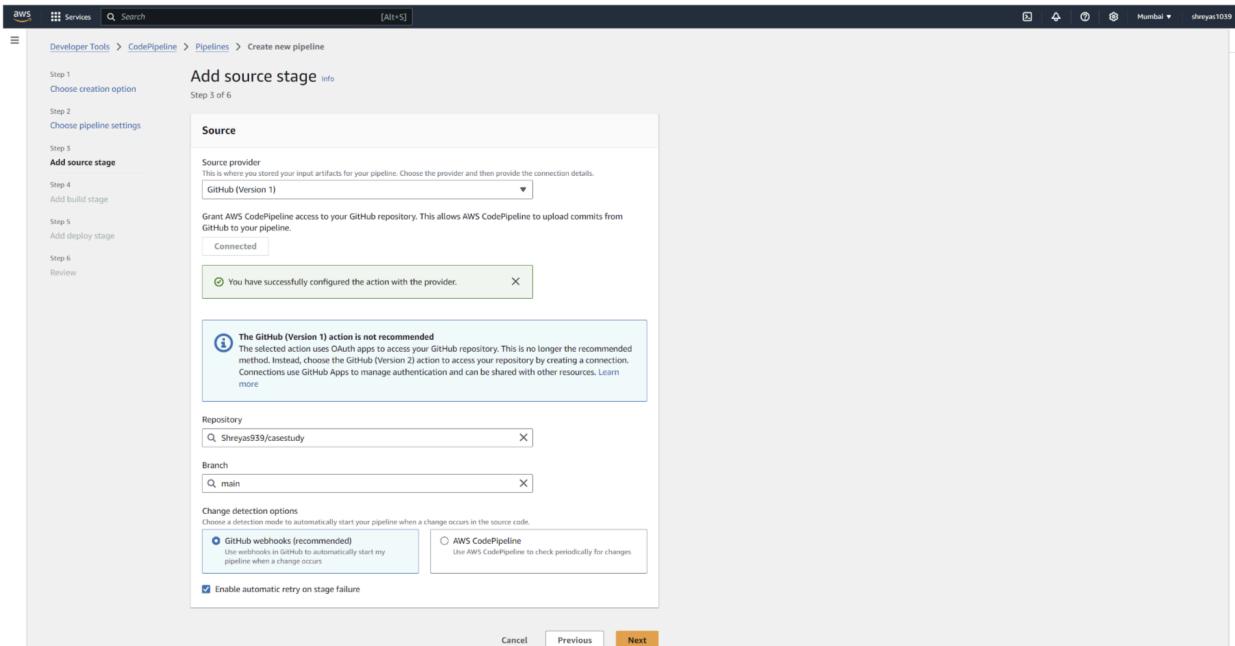
Availability Zone	Subnet	CIDR	Name
ap-south-1b	subnet-0ab0c24a0...	172.16.1.0/24	ajaypuppetsubnet

The screenshot shows the AWS Elastic Beanstalk console. On the left, a sidebar navigation includes 'Applications', 'Environments', 'Change history', 'Application: shreyas-application' (with 'Application versions' and 'Saved configurations'), and 'Environment: Shreyas-application-env' (with 'Go to environment', 'Configuration', 'Events', 'Health', 'Logs', 'Monitoring', 'Alarms', 'Managed updates', and 'Tags'). The main content area displays the 'Shreyas-application-env' environment. A green banner at the top says 'Environment successfully launched.' Below it, the 'Shreyas-application-env' page has tabs for 'Info', 'Actions', and 'Upload and deploy'. The 'Environment overview' section shows 'Health' (Warning), 'Domain' (Shreyas-application-env.eba-bmvshahw.ap-south-1.elasticbeanstalk.com), 'Environment ID' (e-ybcmpmx3iz), and 'Application name' (shreyas-application). The 'Platform' section shows 'Platform' (PHP 8.3 running on 64bit Amazon Linux 2023/4.3.4), 'Running version' (–), and 'Platform state' (Supported). Below these are tabs for 'Events', 'Health', 'Logs', 'Monitoring', 'Alarms', 'Managed updates', and 'Tags'. The 'Events' tab shows '(10)' events with a search bar and a filter icon.

Step 4: Creating a Pipeline with AWS CodePipeline

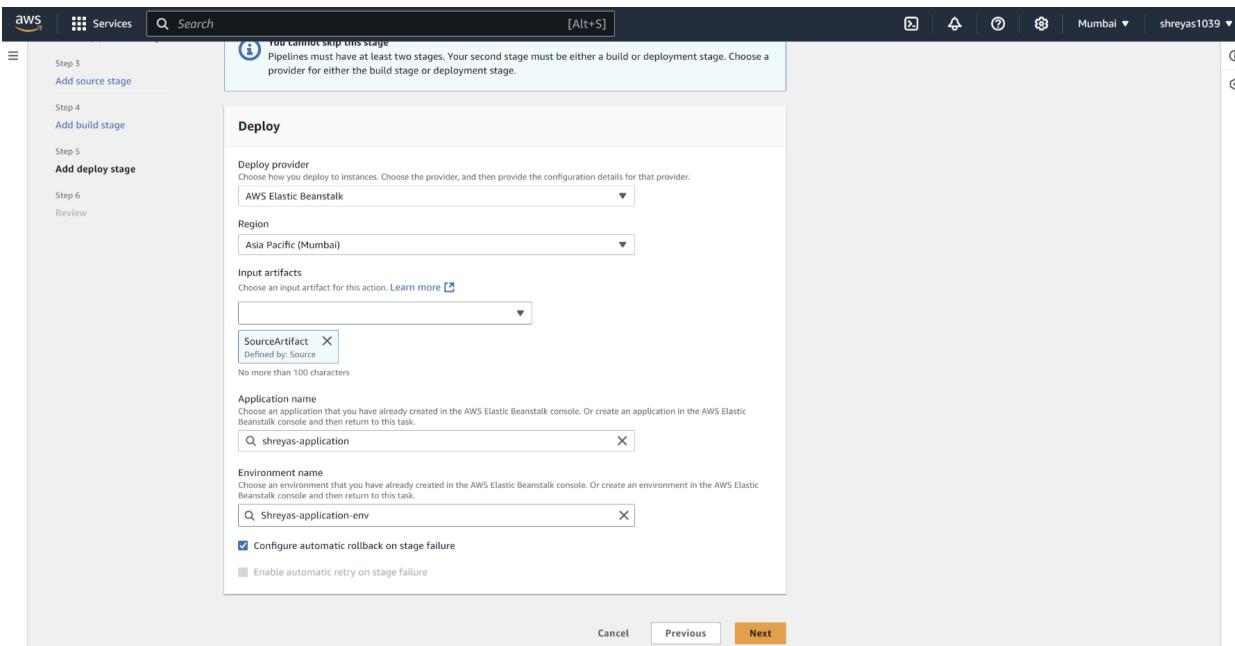
- Open the **CodePipeline** console.
- Create a new pipeline and name it appropriately (e.g., "MyAppPipeline").
- For the source stage, select **GitHub (Version 1)** as the source provider instead of an S3 bucket. This ensures smooth integration and avoids complications from **auto-scaling policy changes affecting S3**.

The screenshot shows the AWS CodePipeline console. The left sidebar lists steps: Step 1 ('Choose creation option'), Step 2 ('Choose pipeline settings'), Step 3 ('Add source stage'), Step 4 ('Add build stage'), Step 5 ('Add deploy stage'), Step 6 ('Review'). The main content area is titled 'Choose pipeline settings' (Info) and is Step 2 of 6. It contains a 'Pipeline settings' section with a 'Pipeline name' field containing 'shreyas-pipeline' (with a note: 'No more than 100 characters') and a 'Pipeline type' note: 'You can no longer create V1 pipelines through the console. We recommend you use the V2 pipeline type with improved release safety, pipeline triggers, parameterized pipelines, and a new billing model.' Below this are sections for 'Execution mode' (radio buttons for 'Superseded', 'Queued (Pipeline type V2 required)', and 'Parallel (Pipeline type V2 required)') and 'Service role' (a dropdown menu).



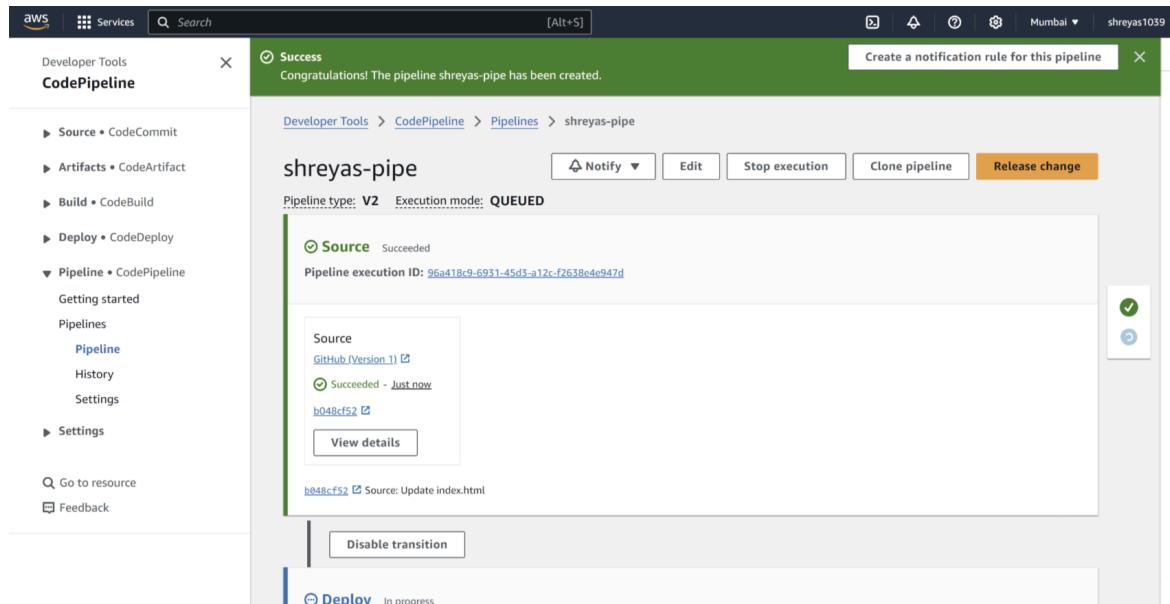
Step 5: Configuring the Deployment Stage

- Skip the build stage if unnecessary.
- Select **AWS Elastic Beanstalk** as the deployment provider.
- Specify the application and environment created earlier in Elastic Beanstalk.



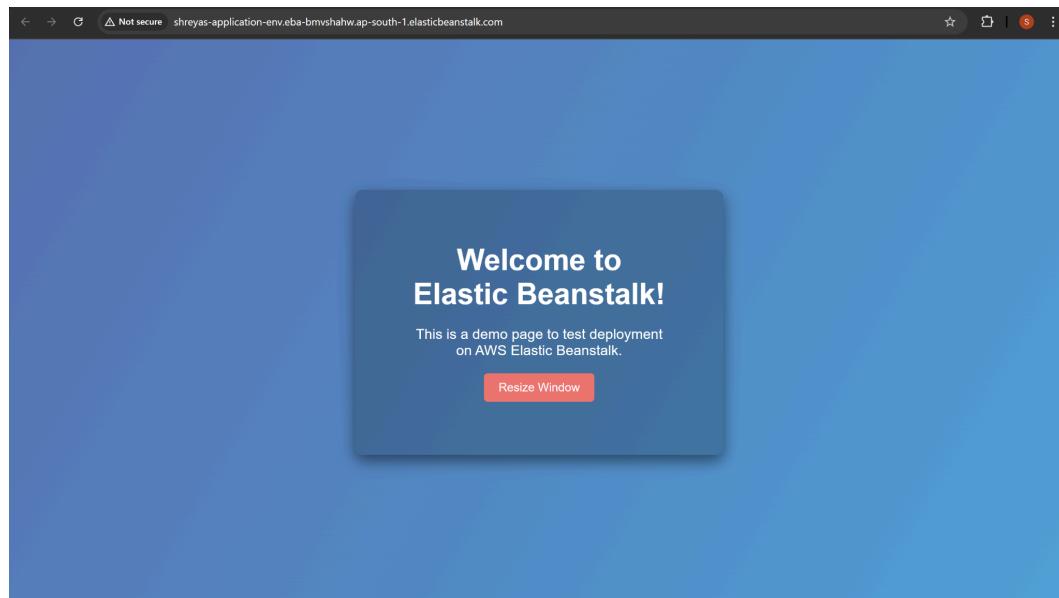
Step 6: Executing and Monitoring the Pipeline

- Once the pipeline is set up, it will automatically trigger whenever changes are made in the GitHub repository.
- Monitor the status of the pipeline stages (Source, Deploy) in the CodePipeline console.



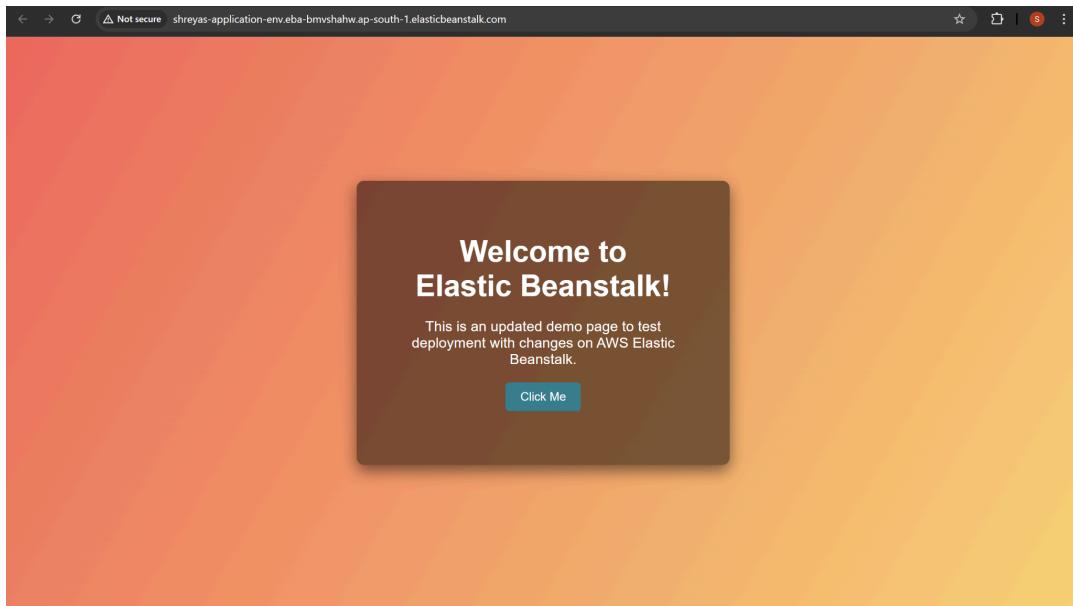
Step 7: Verifying Deployment

- After the pipeline runs successfully, go to the **Elastic Beanstalk** console and access the environment's domain to verify that the application is deployed.



Step 8: Updating the Application Code

- Make a change to the code in your GitHub repository (e.g., update the HTML or CSS file).
- CodePipeline will automatically detect the change and redeploy the updated application.



5. Key Learnings and Future Enhancements

- **Key Learnings:** The importance of IAM roles and permissions, seamless integration between GitHub and CodePipeline, and the reliability of Elastic Beanstalk for application deployment.
- **Future Enhancements:** Incorporating **AWS CodeBuild** to automatically compile and test the code, integrating monitoring tools like **CloudWatch** for performance metrics, and extending this solution to handle multi-environment deployments (e.g., staging and production).

6. Conclusion

In this case study, we demonstrated the power of continuous deployment using AWS CodePipeline and Elastic Beanstalk. By automating the pipeline from code push to production deployment, we were able to minimize manual effort and ensure faster releases. This approach is crucial for companies looking to adopt agile practices, improve code quality, and maintain continuous delivery.