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: shreyasnaik1039@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

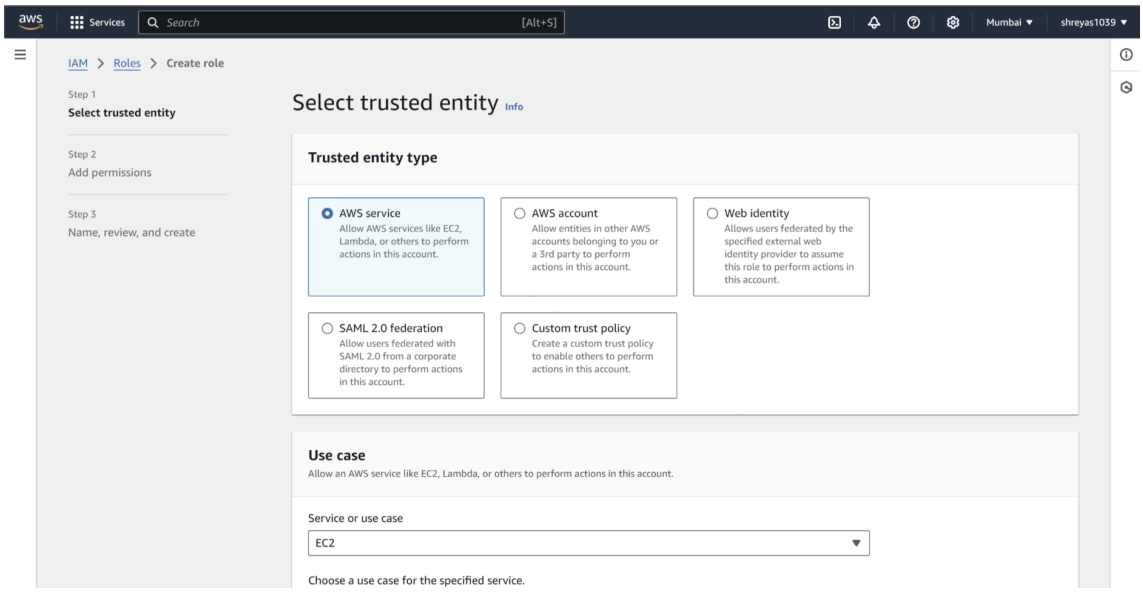
- **AWS CodePipeline**: For creating the CI/CD pipeline.
- **AWS CodeBuild**: (Optional) For compiling and testing the code.
- **AWS Elastic Beanstalk**: For deploying the application to an EC2 instance.
- **IAM (Identity and Access Management)**: For creating and managing roles and permissions.
- **GitHub**: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

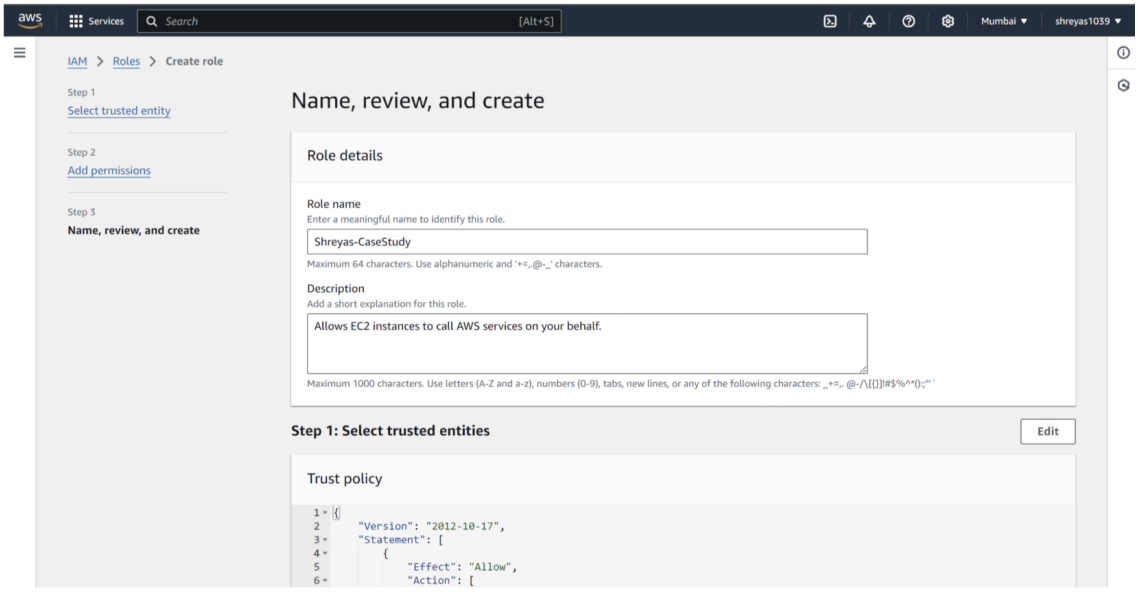
You may access these tools through your AWS Academy portal or personal AWS login.

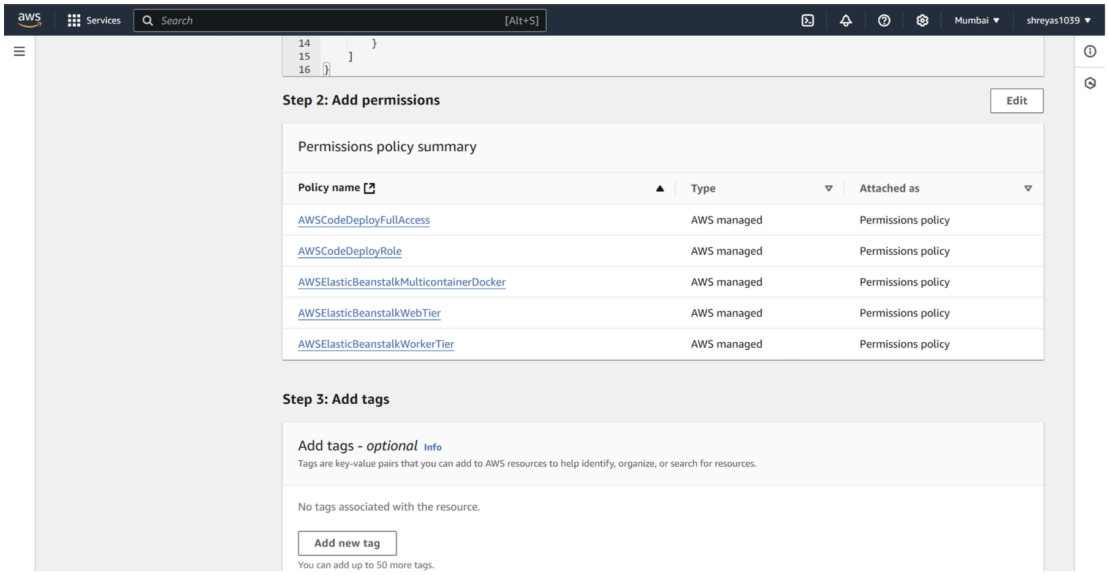
4. Detailed Step-by-Step Process

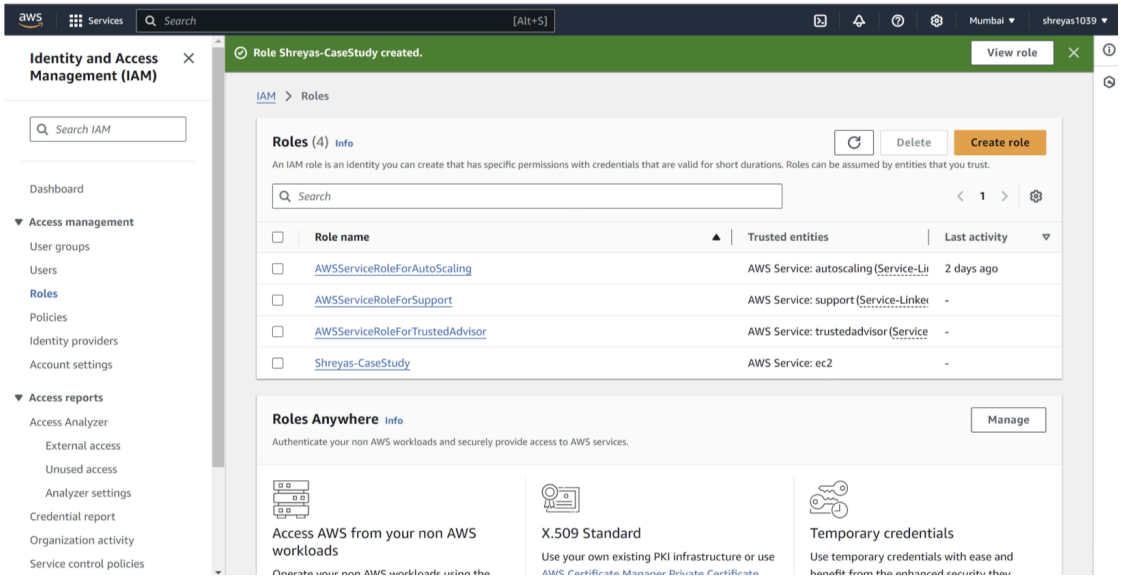
Step 1: Setting Up IAM Roles

- 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.



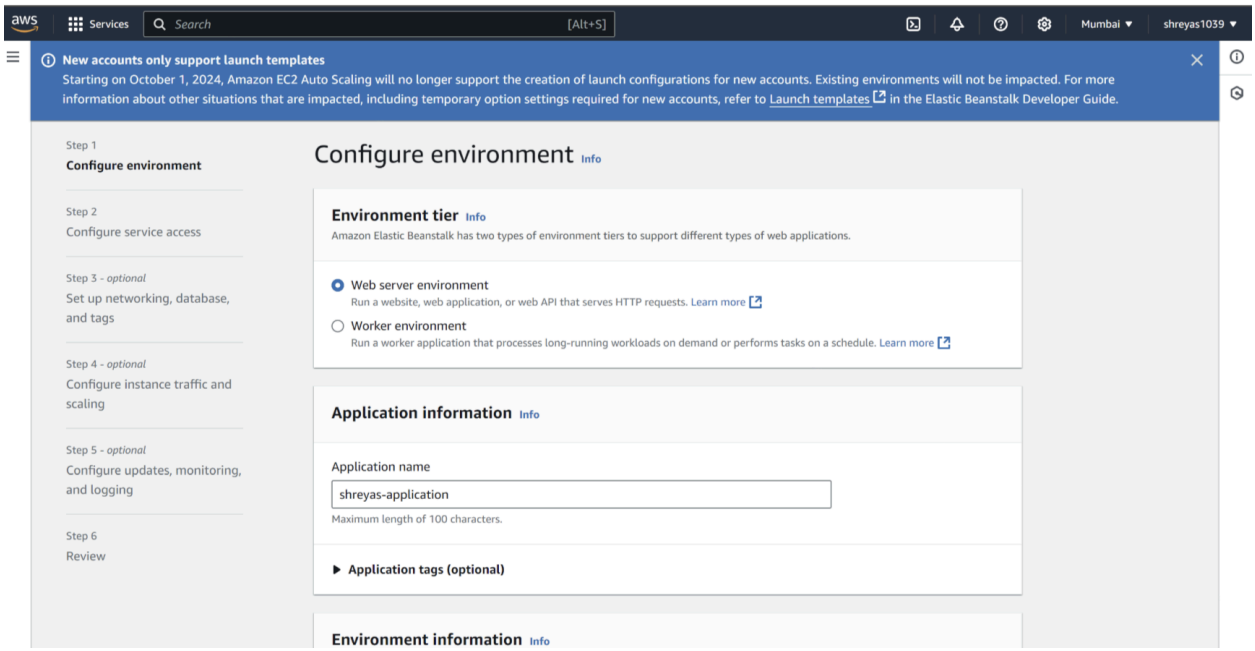


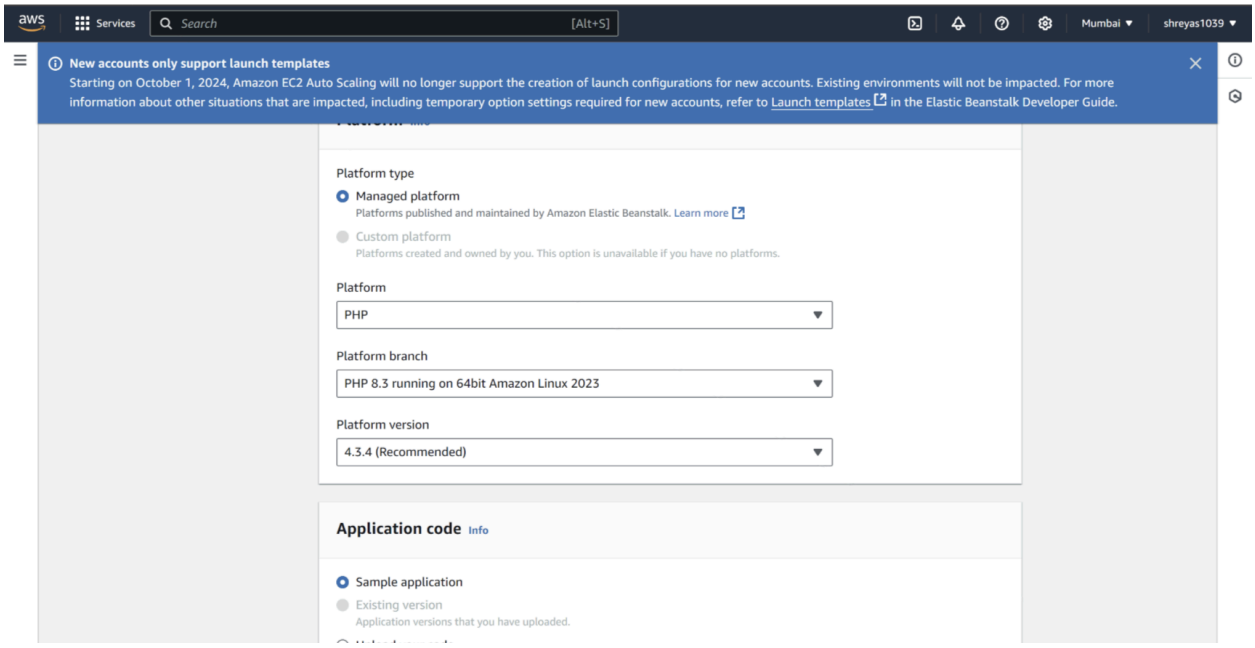


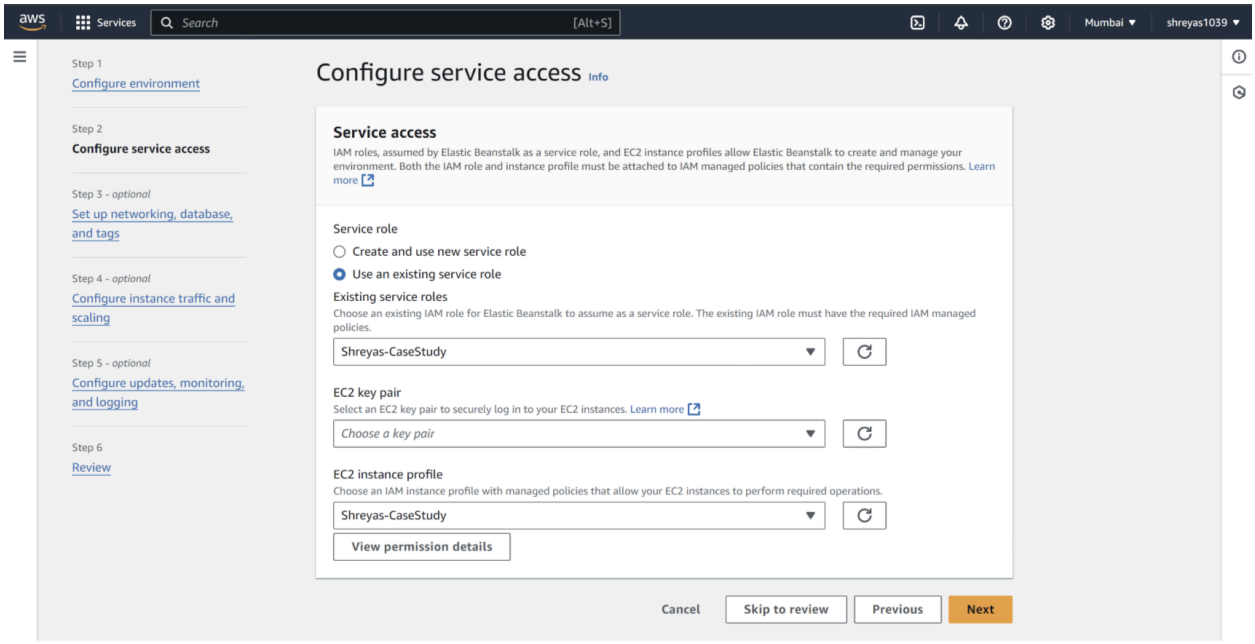


Step 2: Creating an Elastic Beanstalk Application

- 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.

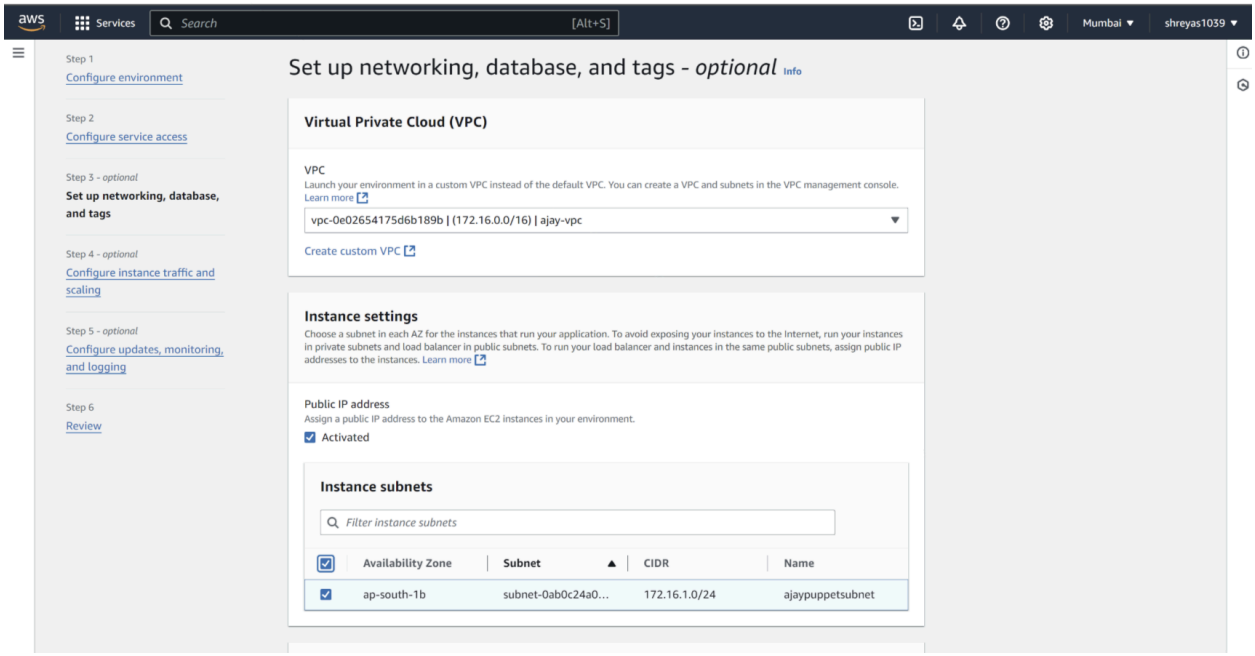

aws
Services
Search
[Alt+S]
Mumbai
shreyas1039
New accounts only support launch templates
Starting on October 1, 2024, Amazon EC2 Auto Scaling will no longer support the creation of launch configurations for new accounts. Existing environments will not be impacted. For more information about other situations that are impacted, including temporary option settings required for new accounts, refer to Launch templates in the Elastic Beanstalk Developer Guide.
Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Set up networking, database, and tags
Step 4 - optional
Configure instance traffic and scaling
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
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
shreyas-application
Maximum length of 100 characters.
Application tags (optional)
Environment information Info

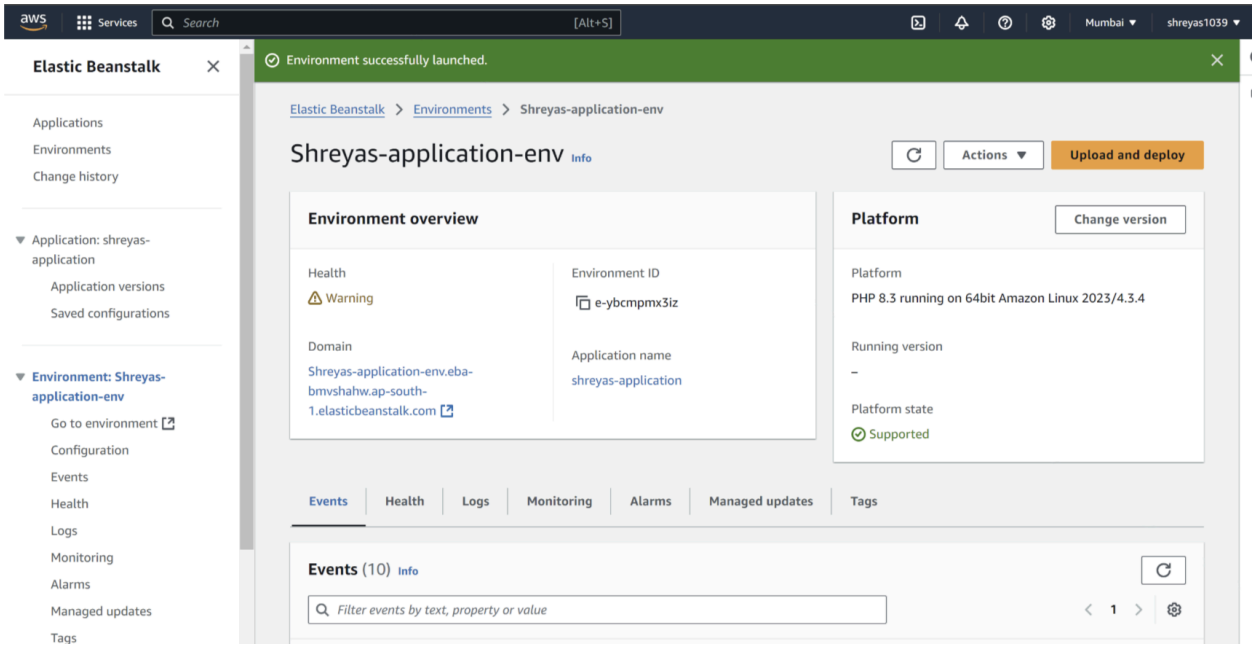

aws
Services
Search
[Alt+S]
Mumbai
shreyas1039
New accounts only support launch templates
Starting on October 1, 2024, Amazon EC2 Auto Scaling will no longer support the creation of launch configurations for new accounts. Existing environments will not be impacted. For more information about other situations that are impacted, including temporary option settings required for new accounts, refer to Launch templates in the Elastic Beanstalk Developer Guide.
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
PHP
Platform branch
PHP 8.3 running on 64bit Amazon Linux 2023
Platform version
4.3.4 (Recommended)
Application code Info
Sample application
Existing version
Application versions that you have uploaded.



Step 3: Configuring VPC and EC2 Instance

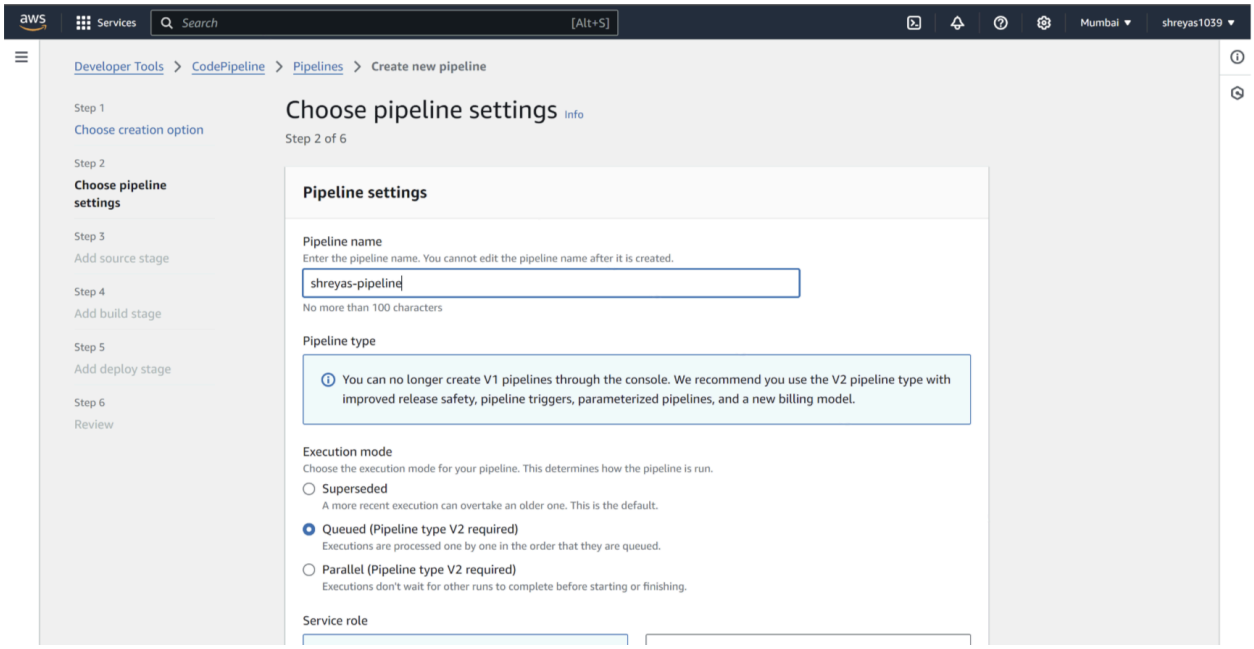
- Choose the appropriate **Virtual Private Cloud (VPC)** and ensure the EC2 instance has a public IP.
- Confirm that the application environment is successfully launched.

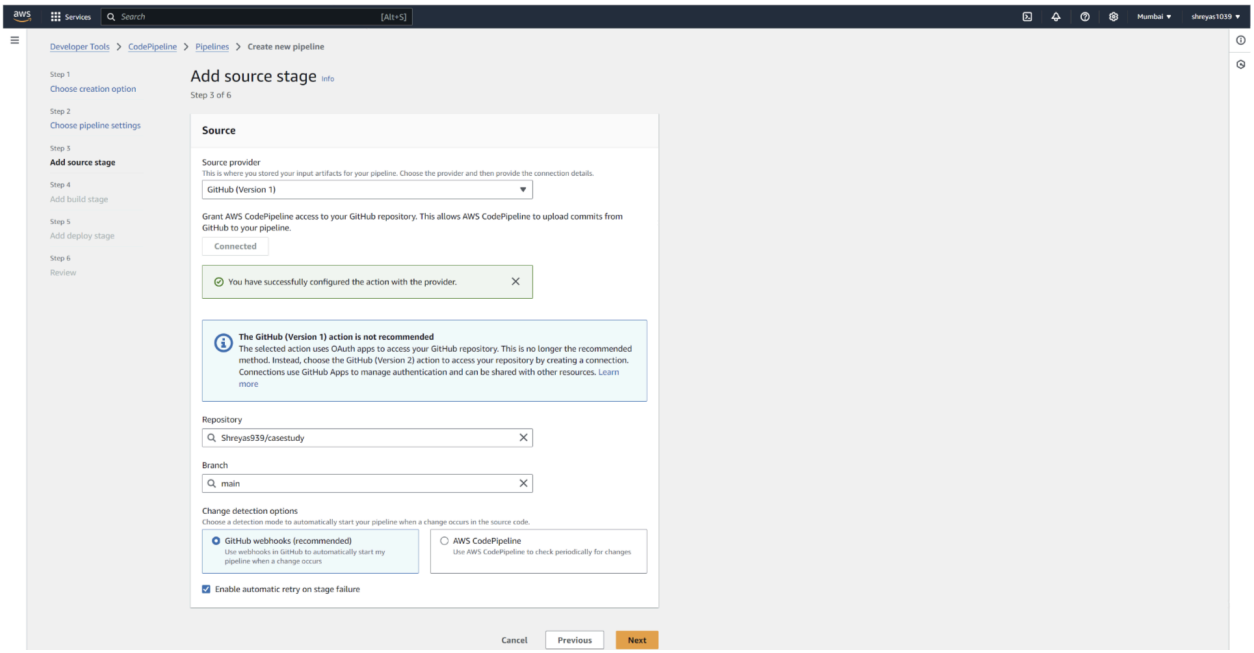




Step 4: Creating a Pipeline with AWS CodePipeline

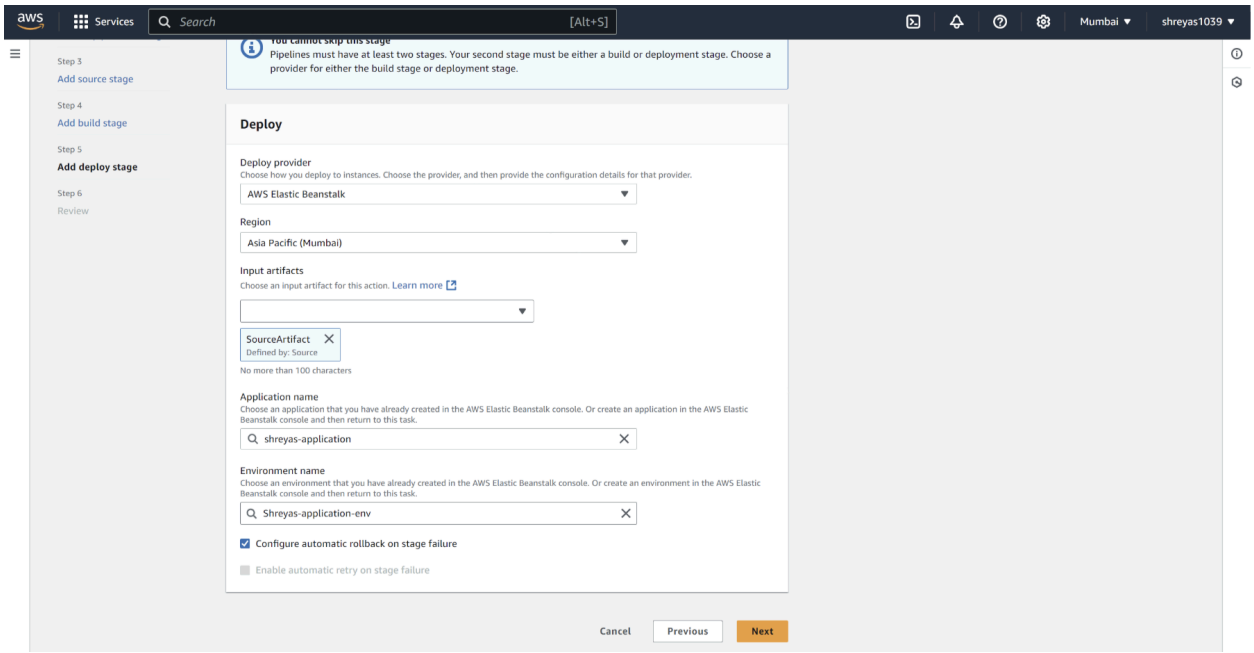
- Open the **CodePipeline** console.
- Create a new pipeline and name it appropriately (e.g., "MyAppPipeline").
- For the source stage, select **GitHub (Version 1)** as the source provider instead of an S3 bucket. This ensures smooth integration and avoids complications from **auto-scaling policy changes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





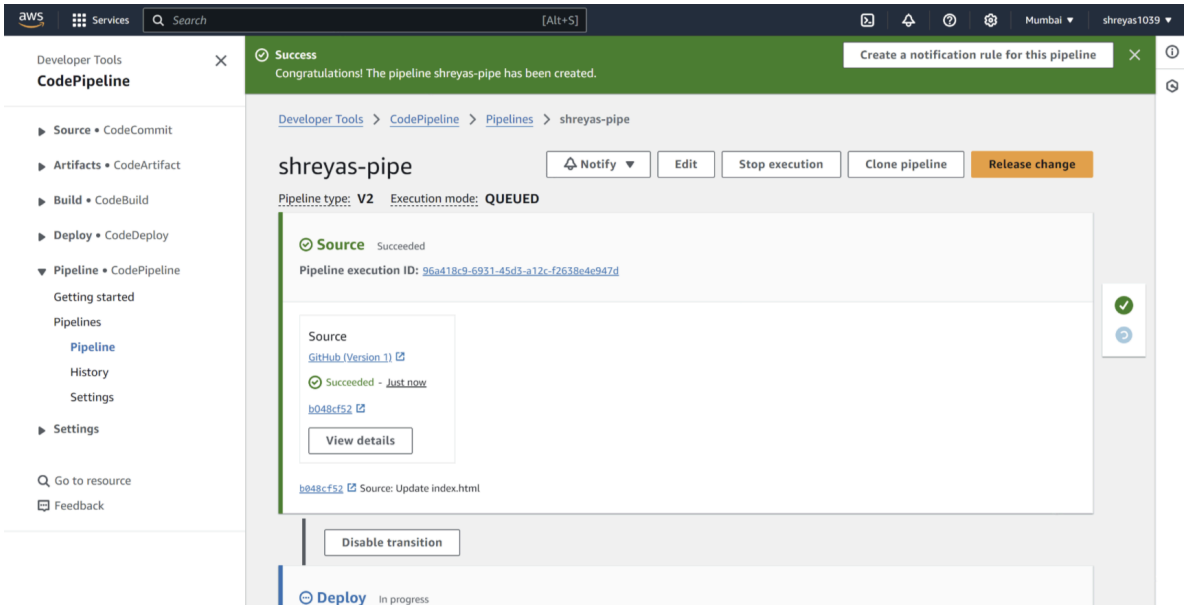
Step 5: Configuring the Deployment Stage

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



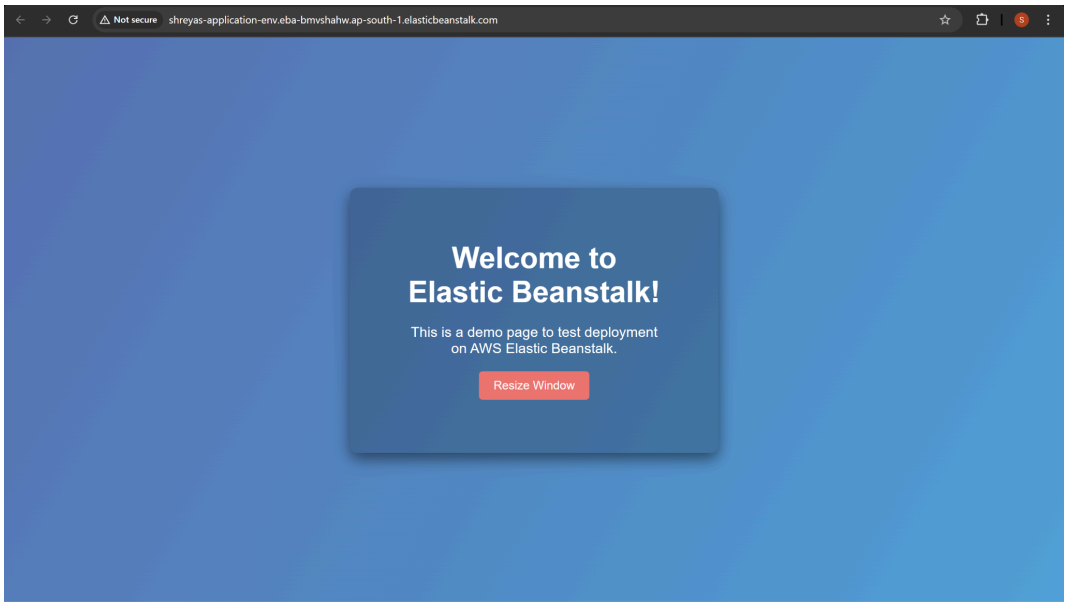
Step 6: Executing and Monitoring the Pipeline

- 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.



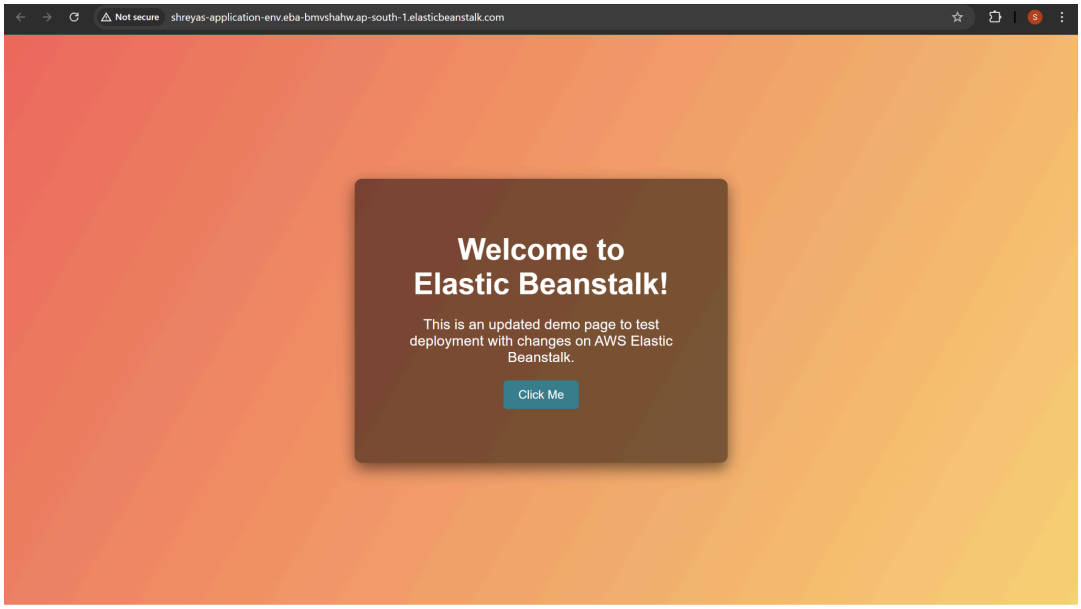
Step 7: Verifying Deployment

- 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.