



# Vivekanand Education Society's

## Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra  
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

### Department of Information Technology

A.Y. 2024-25

## Advance DevOps Lab

### Experiment 02

Aim: To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Roll No.	43
Name	Harsh Pramod Padyal
Class	D15B
Subject	Advance DevOps Lab
LO Mapped	LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements.
Grade:	

**AIM :** To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

**THEORY :**

**Continuous deployment** is a key practice in modern DevOps, enabling organizations to streamline their software release process by automating the deployment of application updates. It allows for the seamless delivery of code revisions to production environments without requiring explicit approval from a developer, thereby reducing time-to-market and enhancing the overall efficiency of the development lifecycle.

**AWS CodePipeline** is a continuous integration and continuous delivery (CI/CD) service that facilitates the building, testing, and deployment of code whenever there is a change in the source code repository. By automating these steps, CodePipeline ensures that new features, bug fixes, and updates are reliably and consistently delivered to users.

One of the critical components of a continuous deployment pipeline is the deployment environment, which is typically made up of virtual servers or containers that host the application.

**Amazon Elastic Beanstalk (EBS)** is a Platform as a Service (PaaS) offering that simplifies the deployment and management of applications in the cloud. It abstracts the underlying infrastructure, such as EC2 instances, load balancers, and scaling configurations, allowing developers to focus on writing code without worrying about provisioning and maintaining the infrastructure.

In a typical AWS CodePipeline workflow, the source code for an application is stored in a version control system like GitHub, an S3 bucket, or AWS CodeCommit. The pipeline monitors this source repository for changes and triggers a series of automated actions whenever a change is detected. These actions might include building the application, running automated tests, and finally deploying the code to a live environment.

The deployment target in this setup could be an Amazon EC2 instance managed by Elastic Beanstalk, which takes care of the deployment details like setting up the necessary resources, deploying the code, and ensuring that the application is running smoothly. This integration with Elastic Beanstalk offers an out-of-the-box deployment solution that is both scalable and resilient.

AWS CodePipeline's integration with Elastic Beanstalk ensures that every code change goes through a consistent deployment process, thereby minimizing human errors and ensuring that the application remains stable and reliable. This automated process not only accelerates the development cycle but also improves the quality of the software by providing immediate feedback on the code's performance in a production-like environment.

## Create a role in an IAM.

The screenshot shows the AWS IAM console in the 'Roles' section. The left sidebar contains the 'Identity and Access Management (IAM)' menu with options like Dashboard, Access management, Users, Policies, and Access reports. The main content area displays 'Roles (2)' with a list of existing roles: 'AWSServiceRoleForSupport' and 'AWSServiceRoleForTrustedAdvisor'. Below this, the 'Roles Anywhere' section is visible, featuring cards for 'Access AWS from your non AWS workloads', 'X.509 Standard', and 'Temporary credentials'. The top navigation bar includes the AWS logo, a search bar, and the user's name 'Harsh'.

## Add EC2 for a service or use case.

The screenshot shows the 'Create role' wizard in the AWS IAM console. The left sidebar indicates the current step is 'Step 2: Add permissions'. The main content area is titled 'Trusted entity type' and offers five options: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Below this, the 'Use case' section is shown, with 'EC2' selected in the 'Service or use case' dropdown menu. The bottom navigation bar includes the AWS logo, a search bar, and the user's name 'Harsh'.

## Give name to the role.

The screenshot shows the AWS IAM console 'Create role' page. The left sidebar indicates the current step is 'Step 3: Name, review, and create'. The main content area is titled 'Name, review, and create' and contains a 'Role details' section. In this section, the 'Role name' is 'Harsh\_Iam' and the 'Description' is 'Allows EC2 instances to call AWS services on your behalf.' Below this is the 'Step 1: Select trusted entities' section, which is currently empty. The bottom of the page shows the footer with '© 2024, Amazon Web Services, Inc. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

us-east-1.console.aws.amazon.com/iam/home?region=ap-southeast-2#/roles/create?trustedEntityType=AWS\_SERVICE&selectedService=EC2&...

aws Services Search [Alt+S]

Global Harsh

IAM > Roles > Create role

Step 1  
[Select trusted entity](#)

Step 2  
[Add permissions](#)

Step 3  
**Name, review, and create**

### Name, review, and create

**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
  
Maximum 64 characters. Use alphanumeric and '+,=, @, -, \_' characters.

**Description**  
Add a short explanation for this role.  
  
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: \_+=, @-/[]!#\$%^&\*()~'"

**Step 1: Select trusted entities** [Edit](#)

**Trust policy**

1 - {

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Required policies (permissions) to be added while creating IAM user.

The screenshot shows the AWS IAM console 'Create role' page, Step 2: Add permissions. The left sidebar indicates the current step is 'Step 2: Add permissions'. The main content area is titled 'Step 2: Add permissions' and contains a 'Permissions policy summary' section. This section displays a table with three rows, each representing a policy: 'AWSElasticBeanstalkMulticontainerDocker', 'AWSElasticBeanstalkWebTier', and 'AWSElasticBeanstalkWorkerTier'. All three policies are of type 'AWS managed' and are attached as 'Permissions policy'. Below the table is the 'Step 3: Add tags' section, which is currently empty. The bottom of the page shows the footer with '© 2024, Amazon Web Services, Inc. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

us-east-1.console.aws.amazon.com/iam/home?region=ap-southeast-2#/roles/create?trustedEntityType=AWS\_SERVICE&selectedService=EC2&...

aws Services Search [Alt+S]

Global Harsh

IAM > Roles > Create role

Step 1  
[Select trusted entity](#)

Step 2  
**Add permissions**

Step 3  
[Add tags](#)

### Step 2: Add permissions

**Permissions policy summary**

Policy name	Type	Attached as
<a href="#">AWSElasticBeanstalkMulticontainerDocker</a>	AWS managed	Permissions policy
<a href="#">AWSElasticBeanstalkWebTier</a>	AWS managed	Permissions policy
<a href="#">AWSElasticBeanstalkWorkerTier</a>	AWS managed	Permissions policy

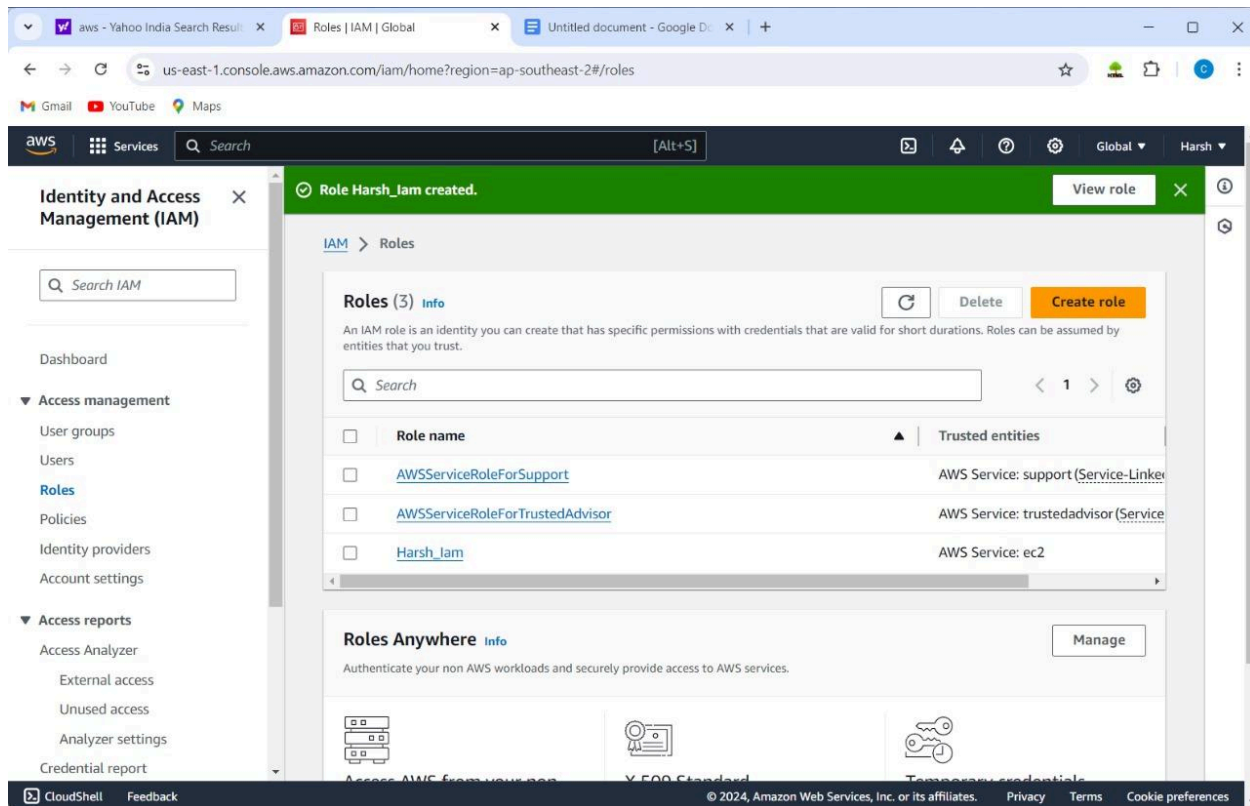
**Step 3: Add tags**

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

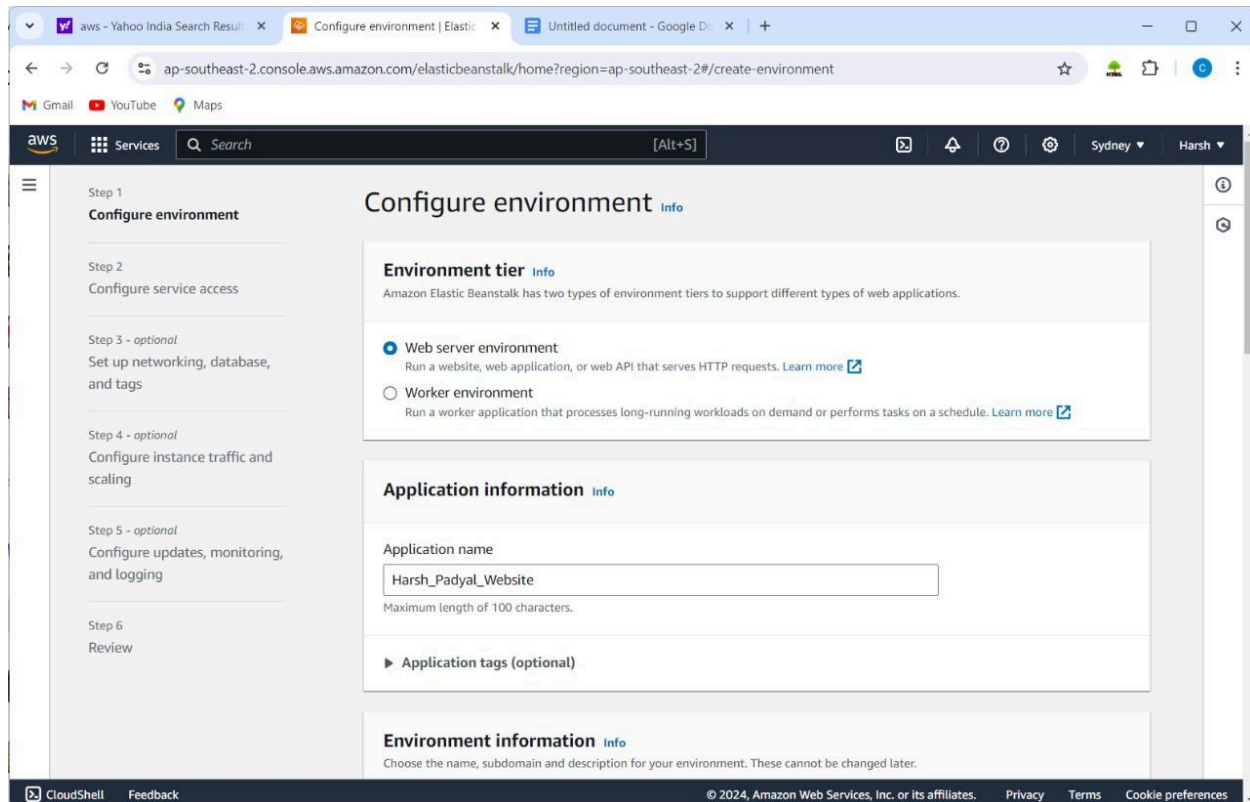
No tags associated with the resource.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## IAM role is being created



Go to the Elastic beanstalk and create an application. Give the appropriate name for the application.



Select the platform as PHP.

Environment description

**Platform** 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

PHP

Platform branch

PHP 8.3 running on 64bit Amazon Linux 2023

Platform version

4.3.2 (Recommended)

In an ec2 instance profile, select the created IAM role.

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](#)

**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.

Harsh\_Iam

EC2 key pair

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

Choose a key pair

EC2 instance profile

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

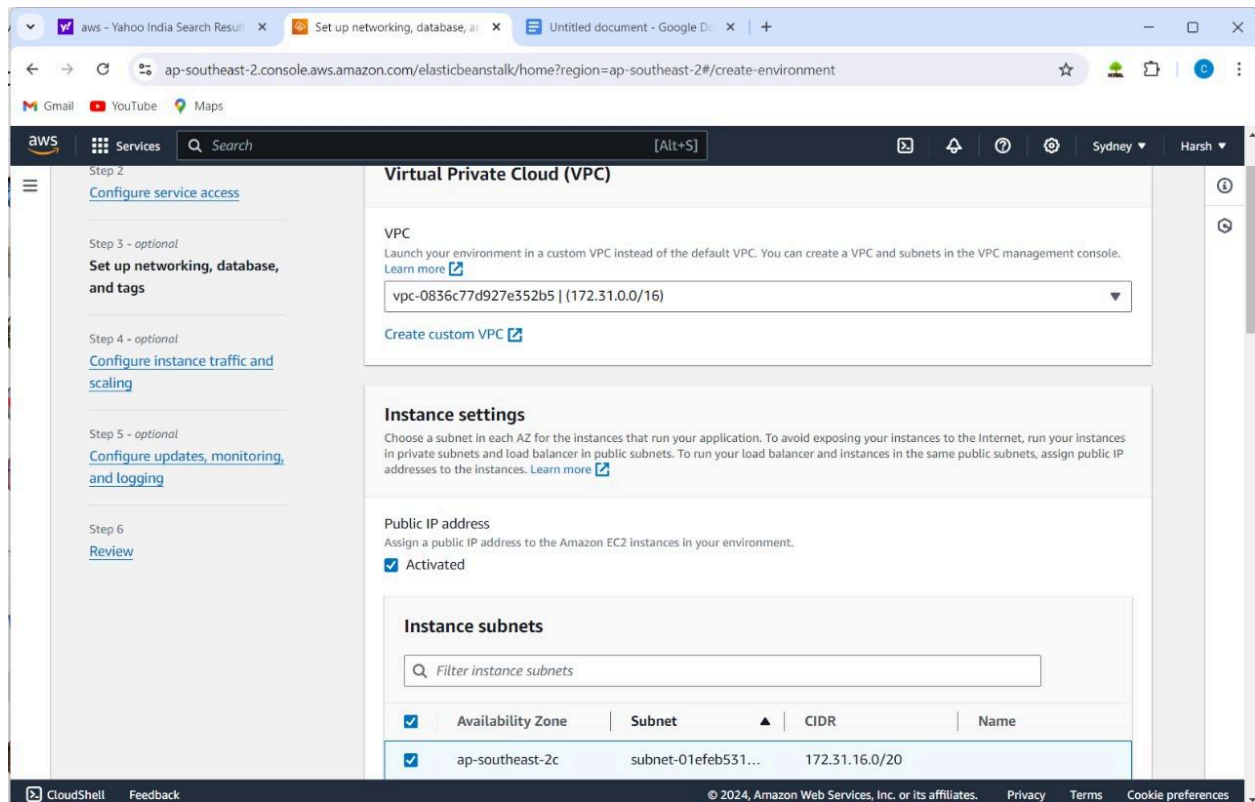
Harsh\_Iam

[View permission details](#)

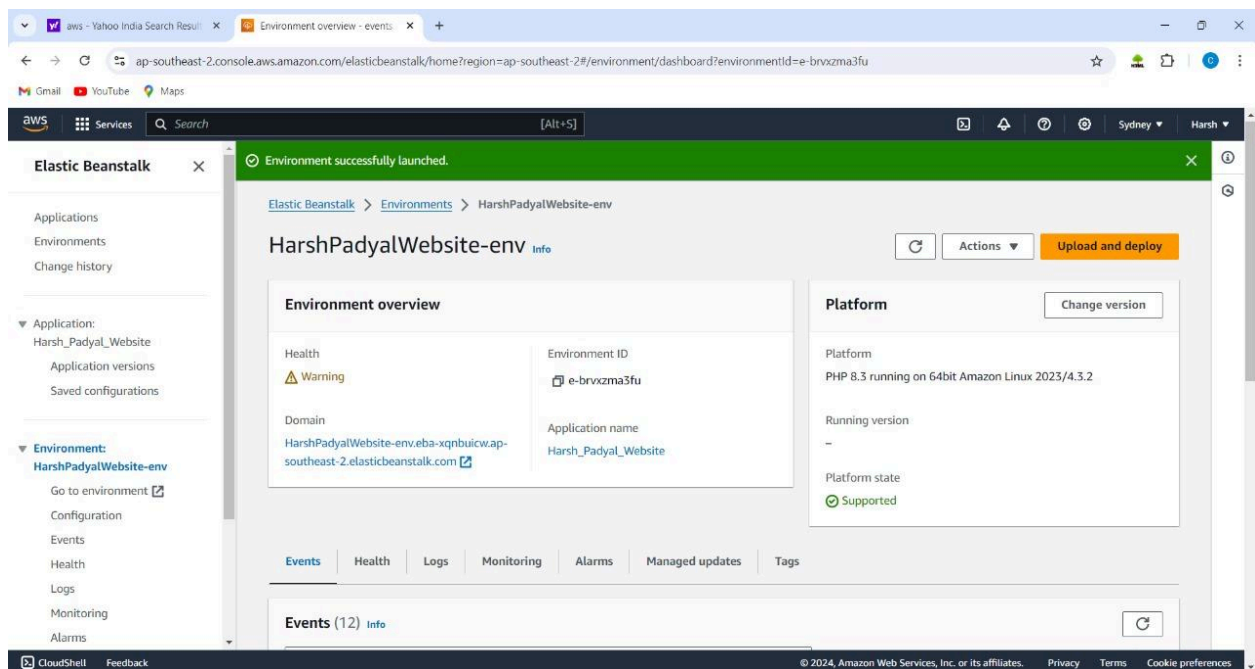
Cancel Skip to review Previous **Next**



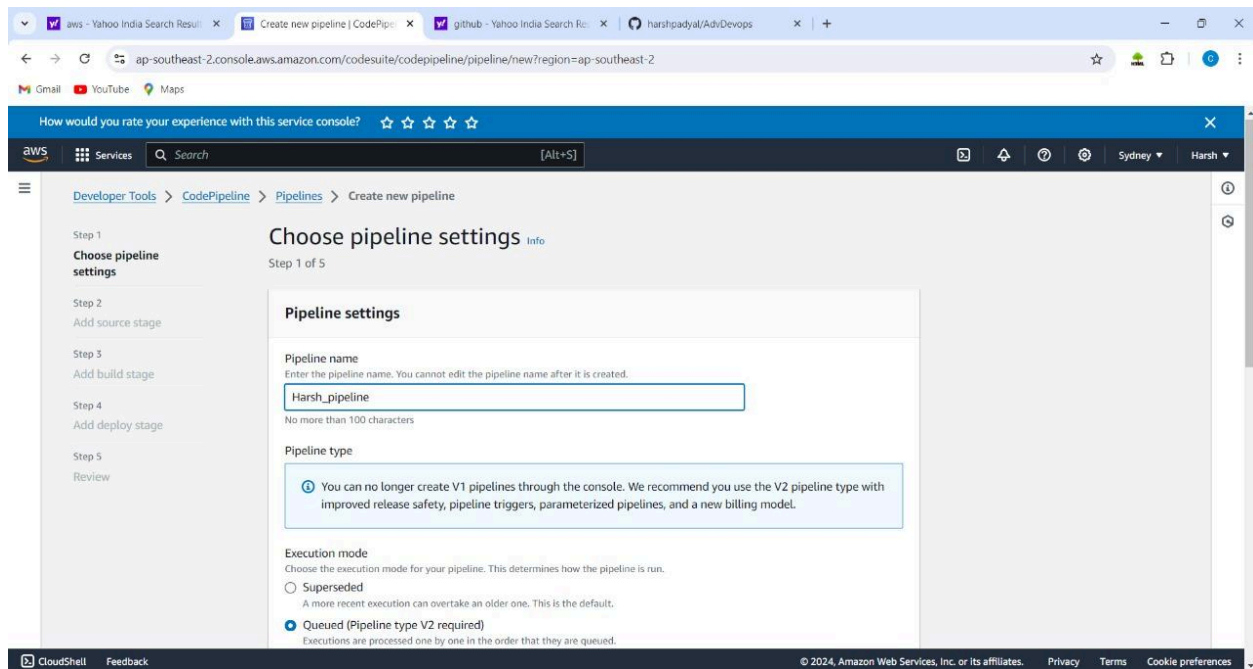
Vpc is to be selected. Public IP address and availability is to be checked.



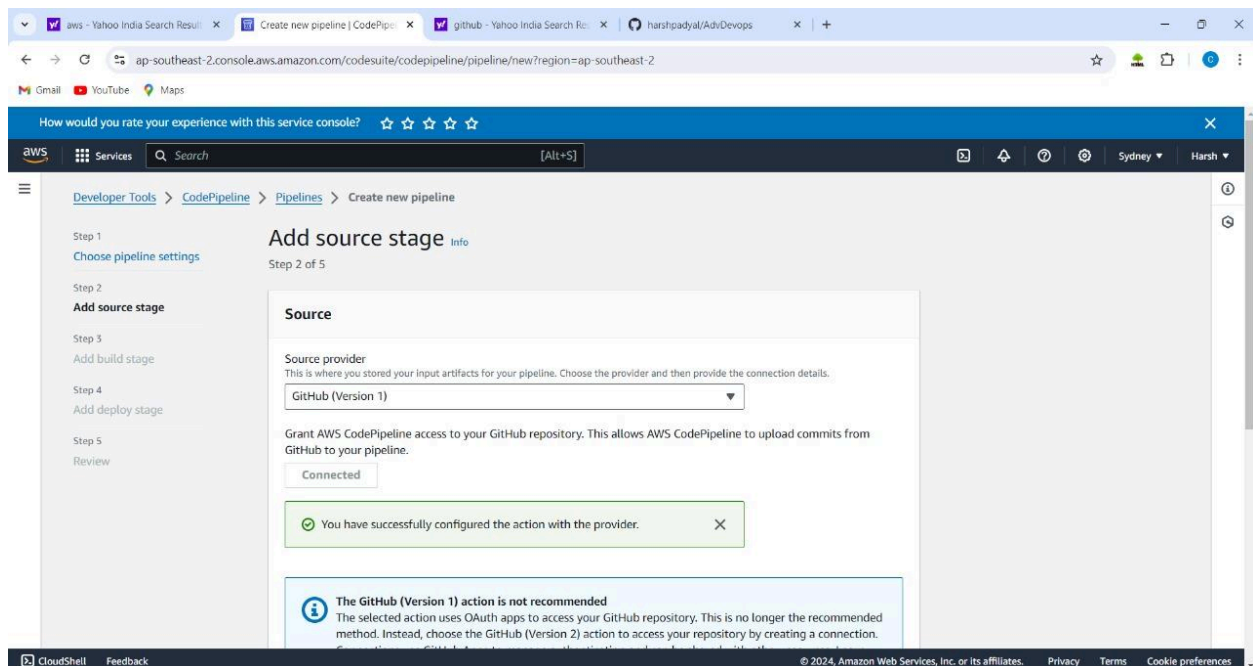
Environment is launched successfully.



Go to the CodePipeline and select the source as GitHub (version 1).



After skipping the build stage, AWS Elastic beanstalk is to be selected in the Deploy Provider. Select your recently created application name and environment name.





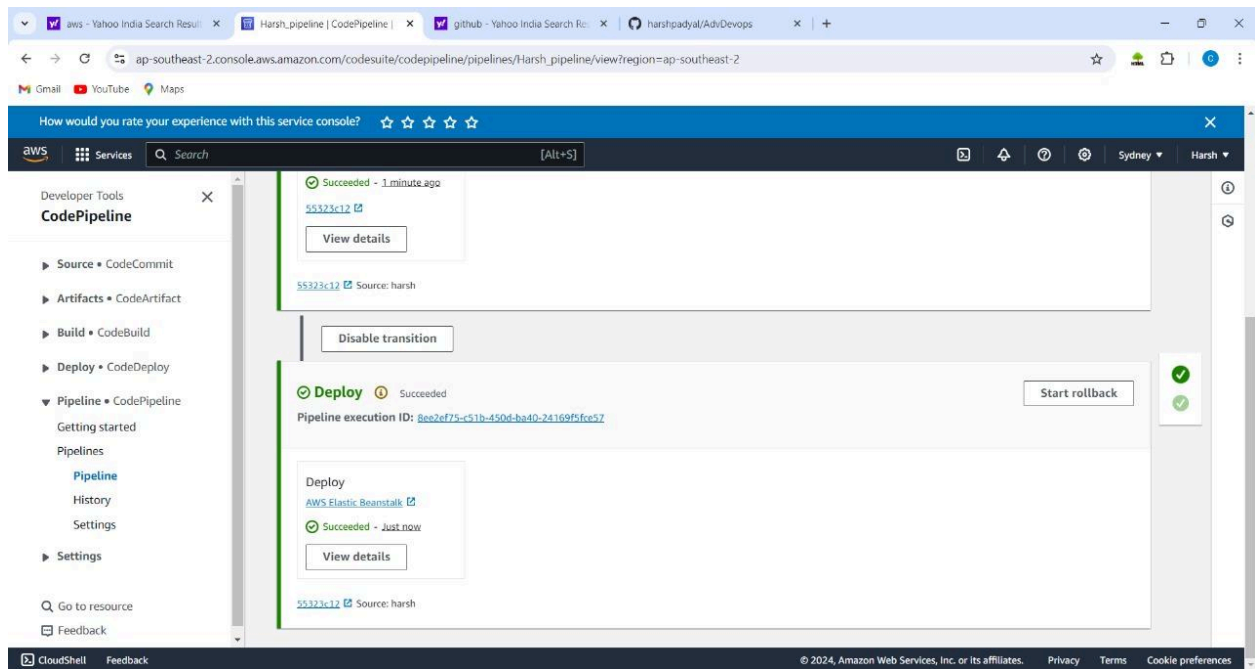
Pipeline is created. The Source and Deploy section is also successful.

The image consists of two screenshots from the AWS CodePipeline console.

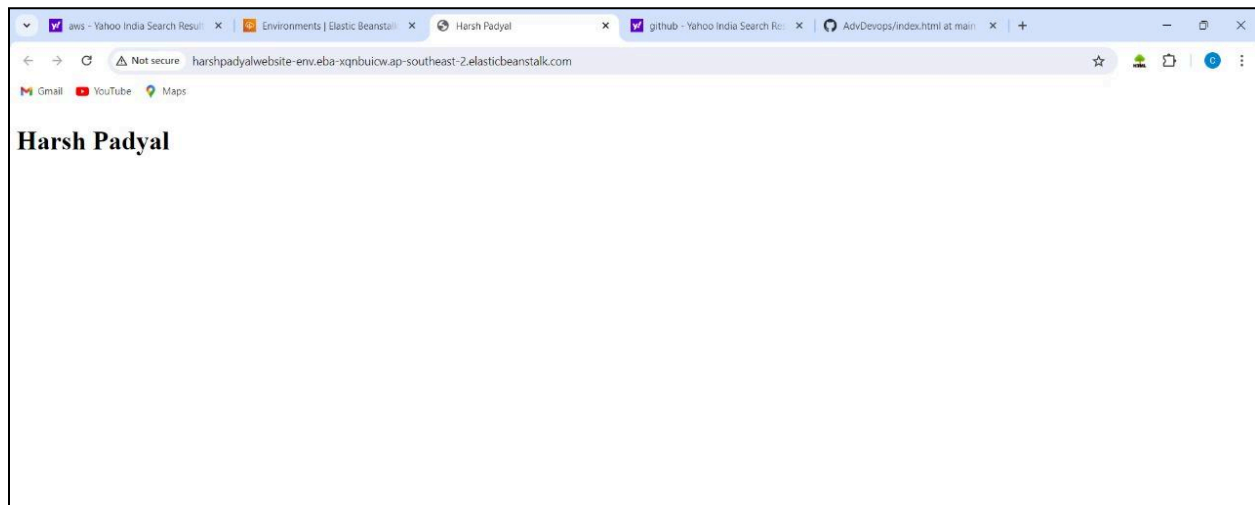
The top screenshot shows the 'Deploy' stage configuration. The 'Deploy provider' is set to 'AWS Elastic Beanstalk'. The 'Region' is 'Asia Pacific (Sydney)'. The 'Input artifacts' section is empty. The 'Application name' is 'Harsh\_Padyal\_Website'. The 'Environment name' is 'HarshPadyalWebsite-env'. There is a checkbox for 'Configure automatic rollback on stage failure' which is unchecked.

The bottom screenshot shows the 'Harsh\_pipeline' view. A green banner at the top indicates 'Success: Congratulations! The pipeline Harsh\_pipeline has been created.' The pipeline type is 'V2' and the execution mode is 'QUEUED'. The 'Source' stage is shown as 'In progress' with a 'View details' button. The 'Deploy' stage is also shown as 'In progress' with a 'View details' button. The 'Disable transition' button is visible at the bottom.

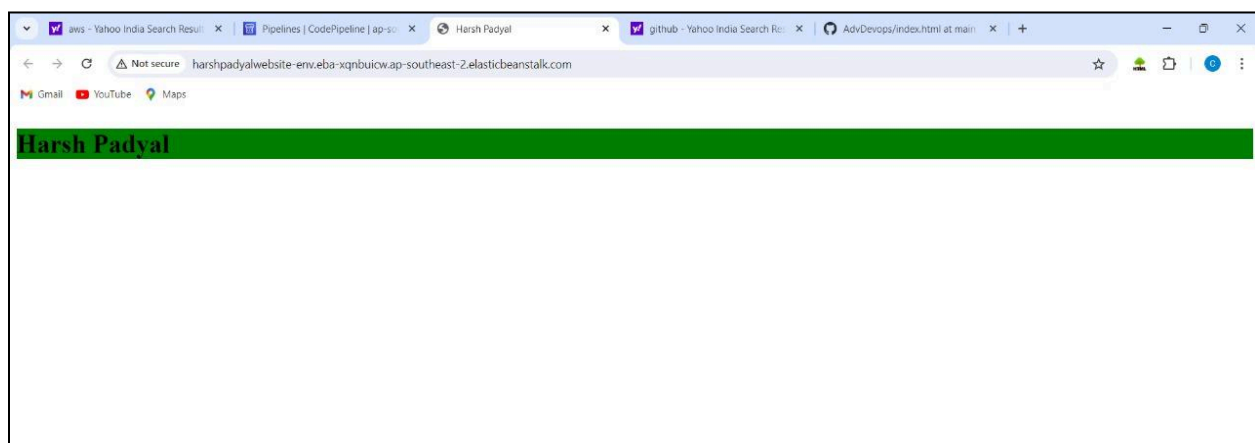
Go to the elastic beanstalk environment and click on domain.



Following output is to be generated of the code which is in the github repository.



Changes are done in the code of the Github repository and it is being directly deployed without any configurations.



**CONCLUSION :**

Continuous deployment using AWS CodePipeline and Elastic Beanstalk represents a powerful approach to modern software development, where automation plays a crucial role in delivering high-quality software quickly and efficiently. This method supports the agile methodology by enabling rapid iterations and continuous improvements, leading to more responsive and innovative applications.