Shravani Anil Patil **D15A:38**

Advanced DevOps Experiment-2

Aim: Using AWS CodePipeline, deploy Sample Application on Elastic BeanStalk using AWS CodeDeploy.

Theory:-

Elastic Beanstalk simplifies the process of deploying and managing applications on the AWS cloud, allowing developers to focus on building applications without needing to worry about the underlying infrastructure. AWS offers over 100 services, which can sometimes make it challenging to know exactly how to configure and provision the right resources. Elastic Beanstalk removes this complexity by automatically managing the necessary AWS resources—such as Amazon EC2 instances, load balancing, and scaling—while still giving you control and flexibility over your setup.

Elastic Beanstalk supports a variety of programming languages including Go, Java, .NET, Node.js, PHP, Python, Ruby, and Docker platforms. This means that if your app is built in one of these languages or platforms, Elastic Beanstalk can handle its deployment. For Docker containers, you can even use your own custom configurations and dependencies, expanding the potential use cases beyond those natively supported languages.

Using Elastic Beanstalk is straightforward. You can interact with it via the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or a dedicated high-level CLI tool called eb. Once your app is ready, all you need to do is upload it as an application version (like a .war file for Java applications), and Elastic Beanstalk takes care of the rest—provisioning resources, setting up your environment, and ensuring your app runs smoothly.

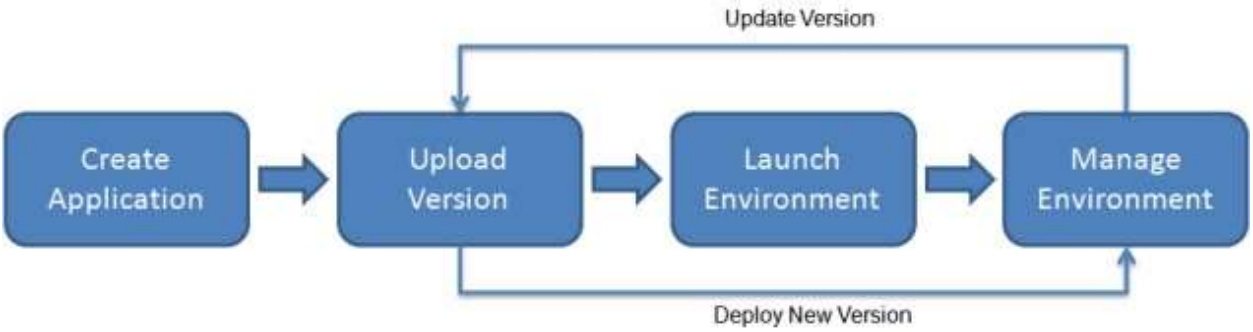
After your environment is set up, you can continue to manage and monitor it through the web interface or the CLI tools. You can scale your application by adjusting the number of EC2 instances or deploying new versions of your app with ease. The end result is a fully automated workflow that takes care of the technical heavy lifting, allowing you to deliver updates and maintain your app without diving deep into infrastructure management.

Here's a simplified breakdown of how Elastic Beanstalk works:

Create the application: Upload your code and define an application version.Elastic Beanstalk provisions resources: It automatically sets up the required AWS services and resources (EC2, load balancers, etc.).

Deploy and manage: You can monitor the application's performance, scale resources, and deploy new versions as needed.

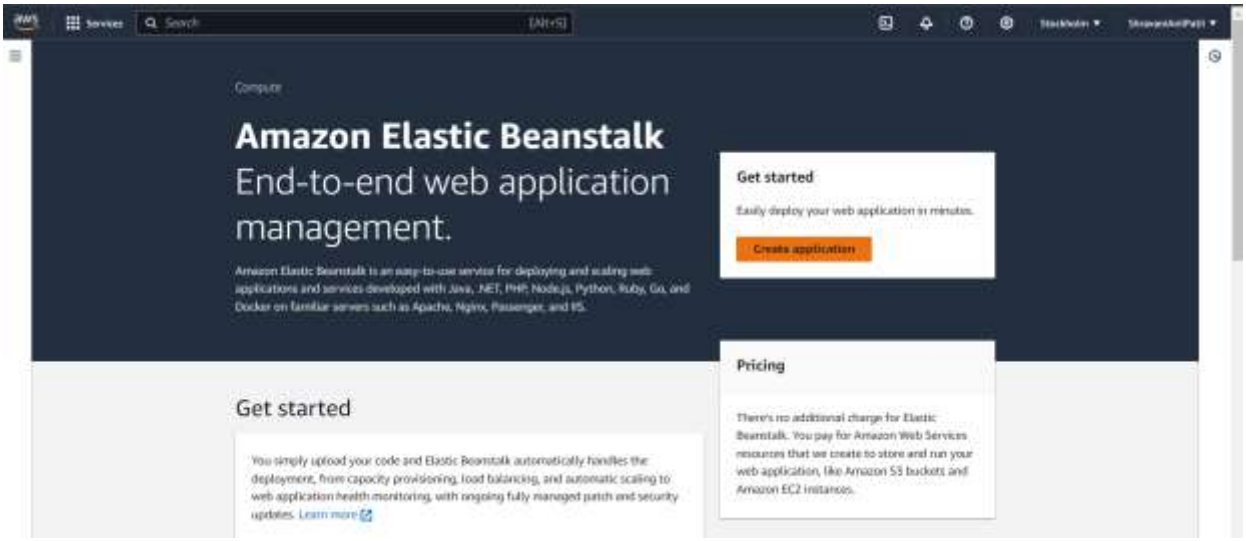
Elastic Beanstalk makes AWS cloud management simple, while keeping flexibility intact. It's like having a smart assistant for infrastructure that automates the complexities of cloud provisioning but still allows you to customize when needed.

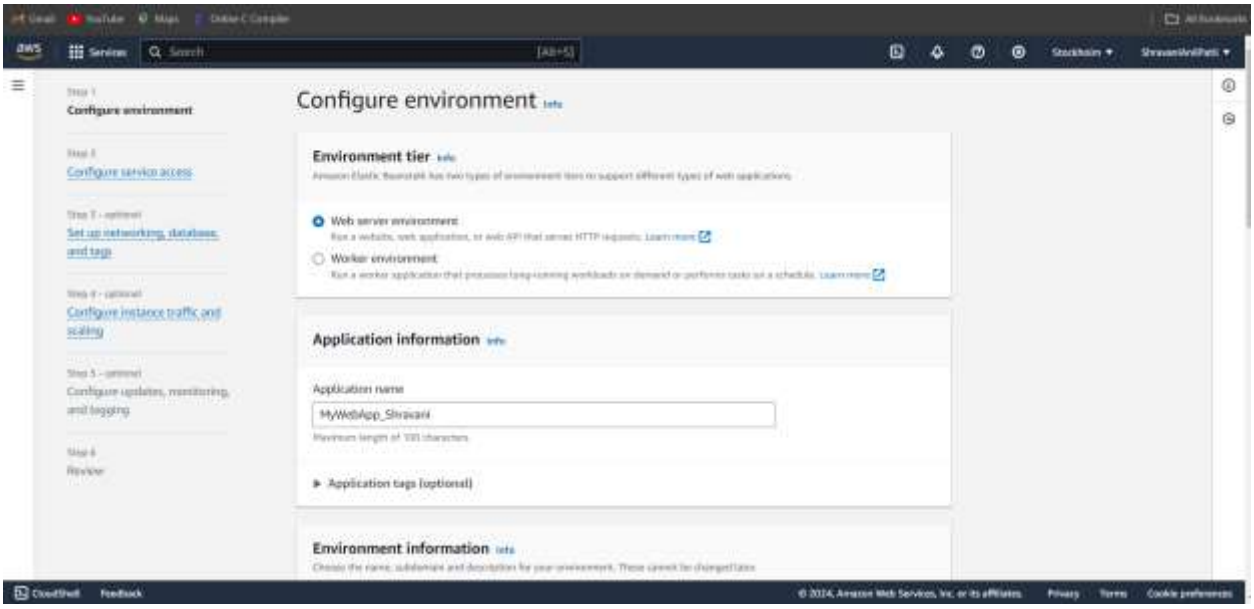


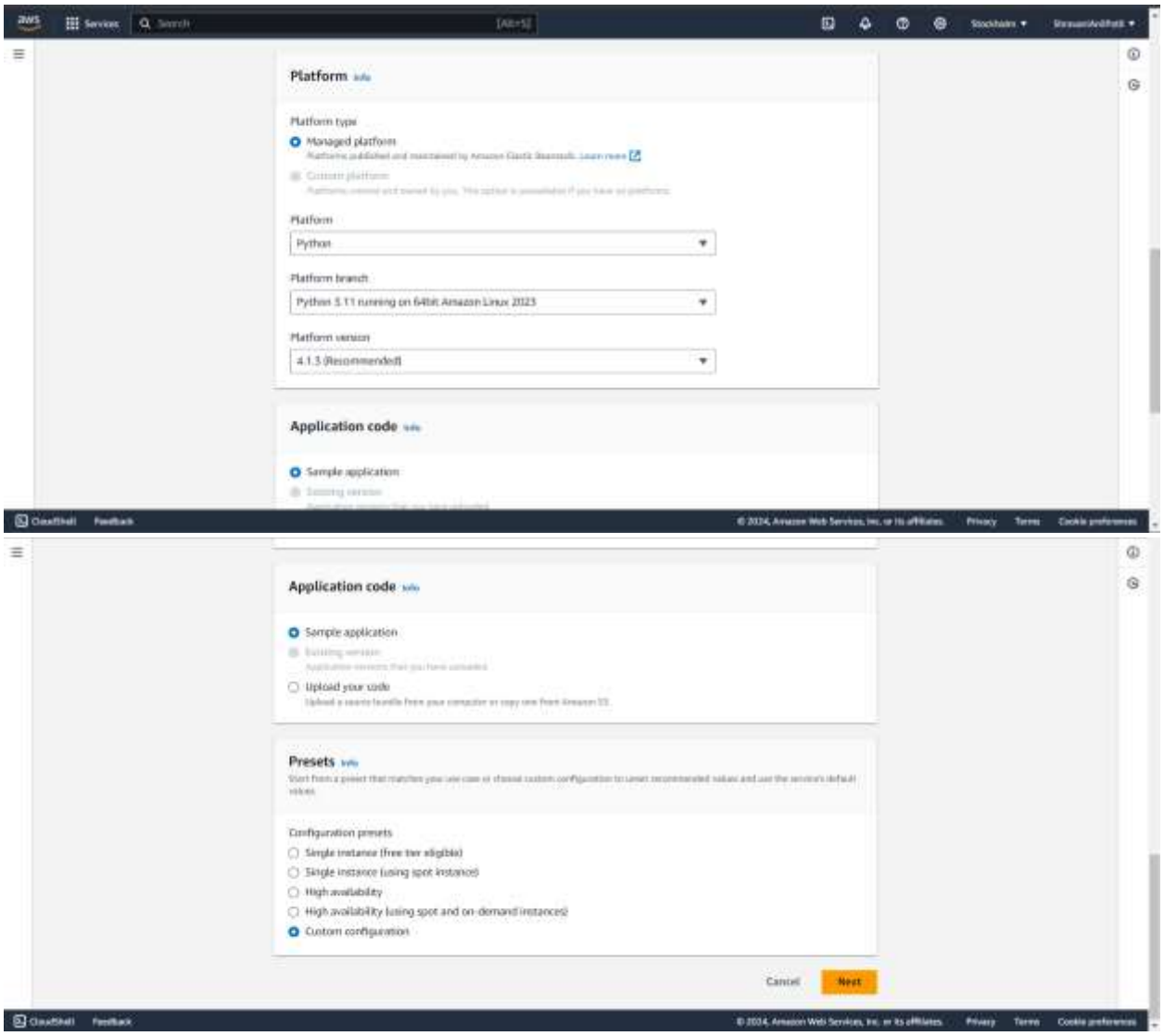
After you create and deploy your application, information about the application—including metrics, events, and environment status—is available through the Elastic Beanstalk console, APIs, or Command Line Interfaces, including the unified AWS CLI.

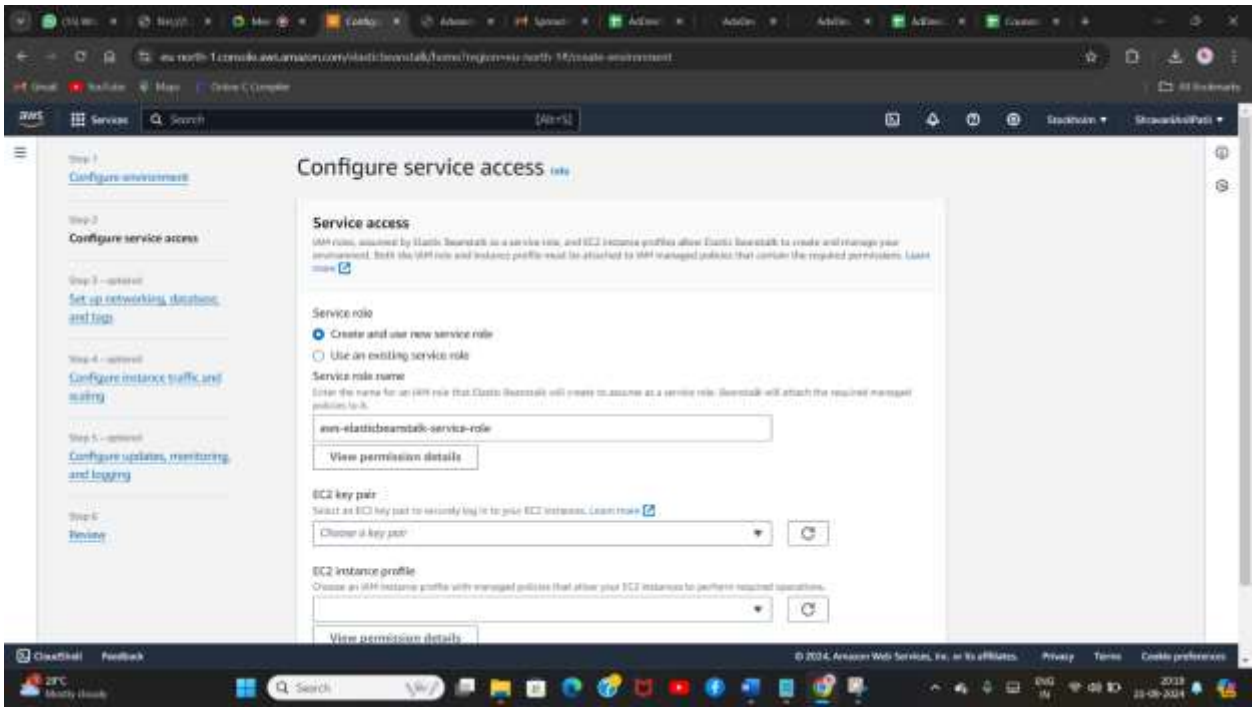
Implementation:-

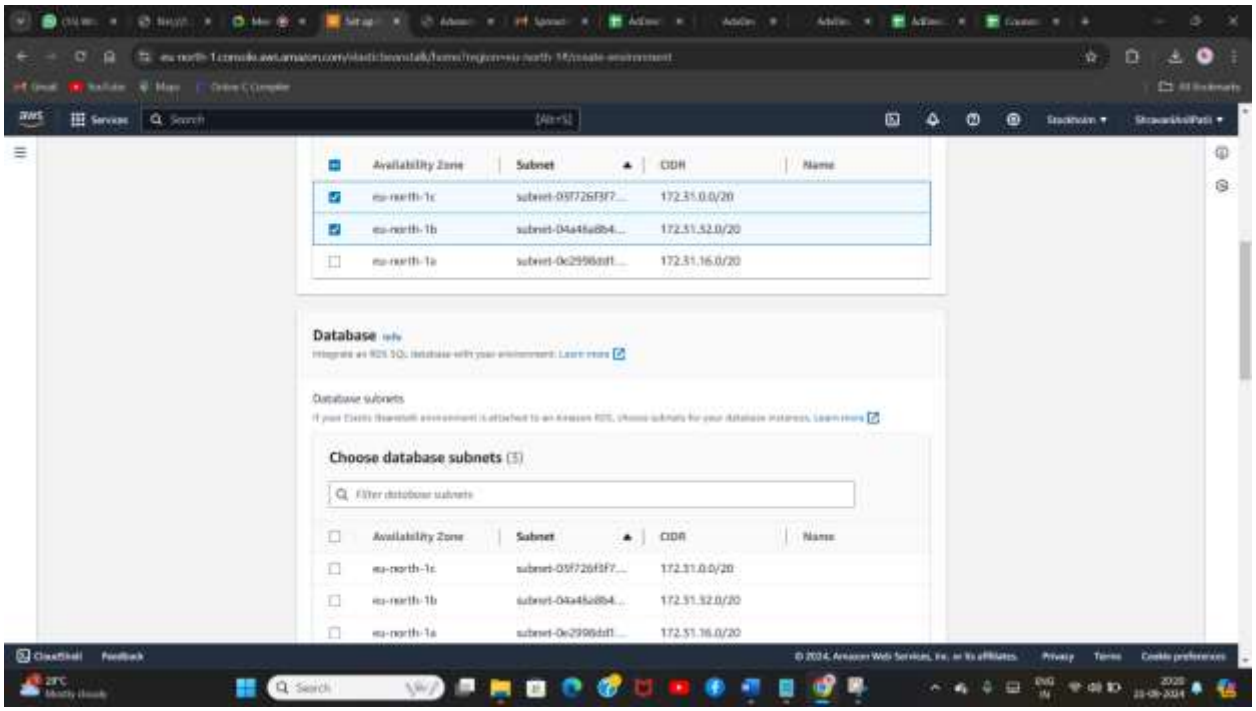
Deploying basic web page on Elastic Beanstalk

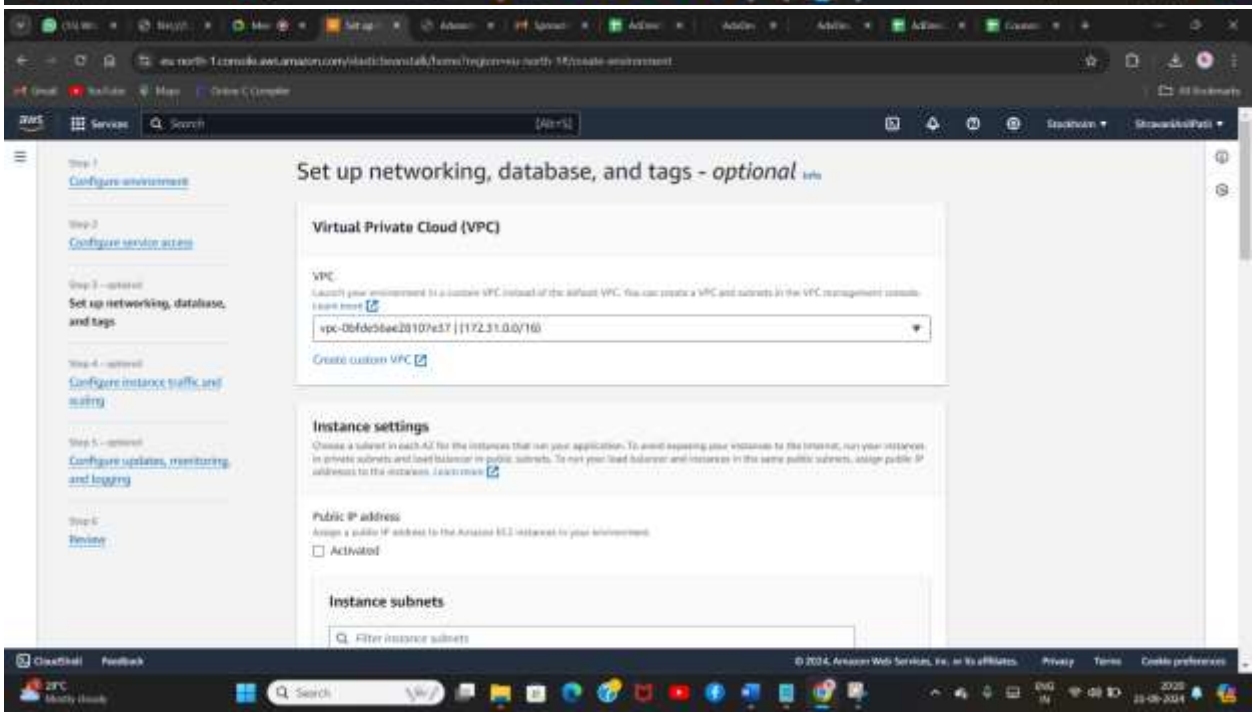


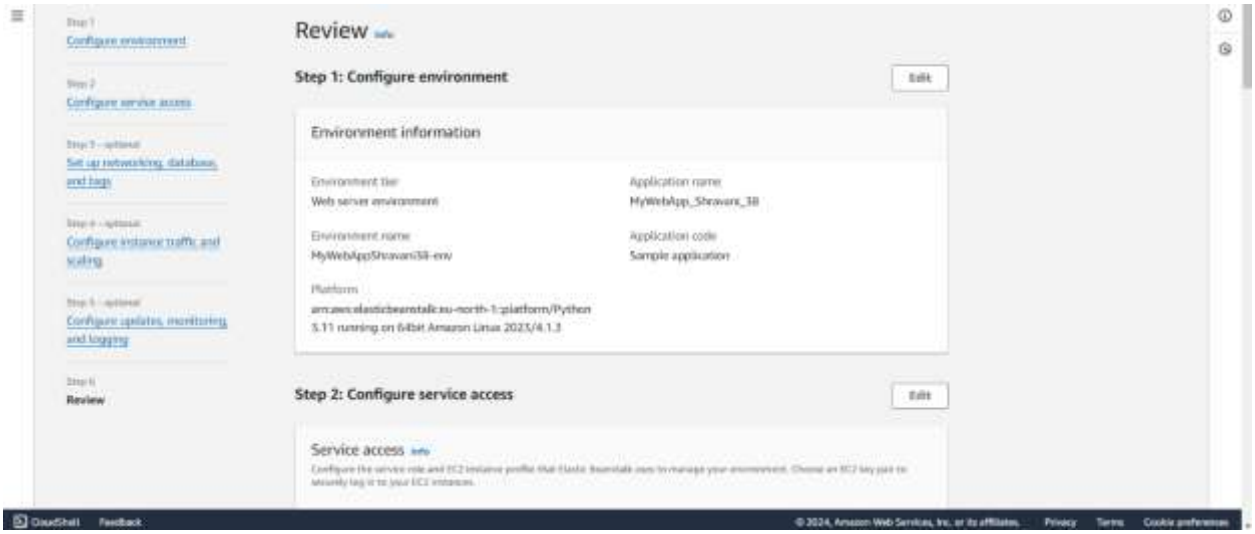


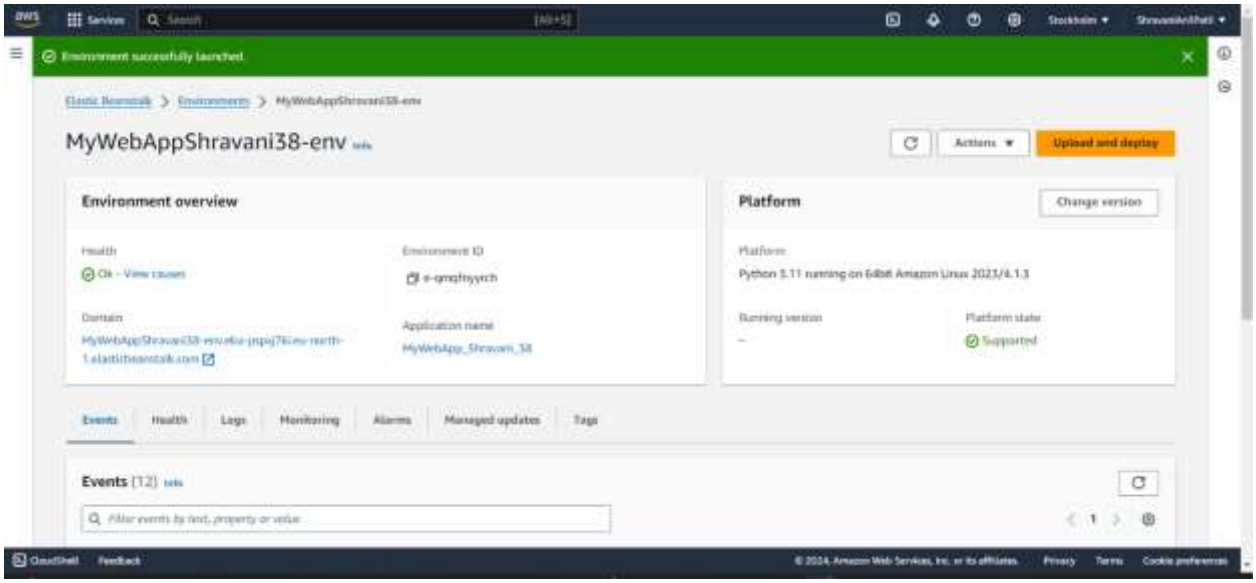

Platform
Platform type
Managed platform
Platform
Python
Platform branch
Python 3.11 running on 64bit Amazon Linux 2023
Platform version
4.1.3 (Recommended)
Application code
Sample application
Application code
Sample application
Upload your code
Presets
Configuration presets
Single instance (free tier eligible)
Single instance (using spot instance)
High availability
High availability (using spot and on-demand instances)
Custom configuration
Cancel
Next

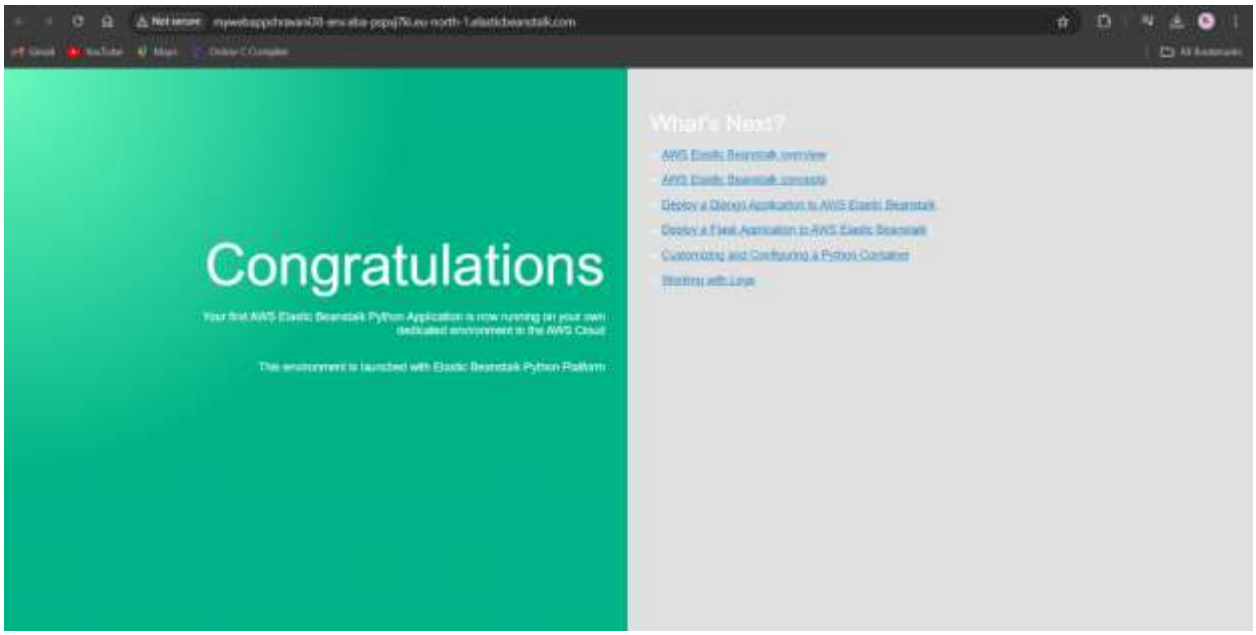

Configure service access
Service access
Service role
Create and use new service role
Use an existing service role
Service role name
aws-elasticbeanstalk-service-role
View permission details
EC2 key pair
Choose a key pair
EC2 instance profile
View permission details

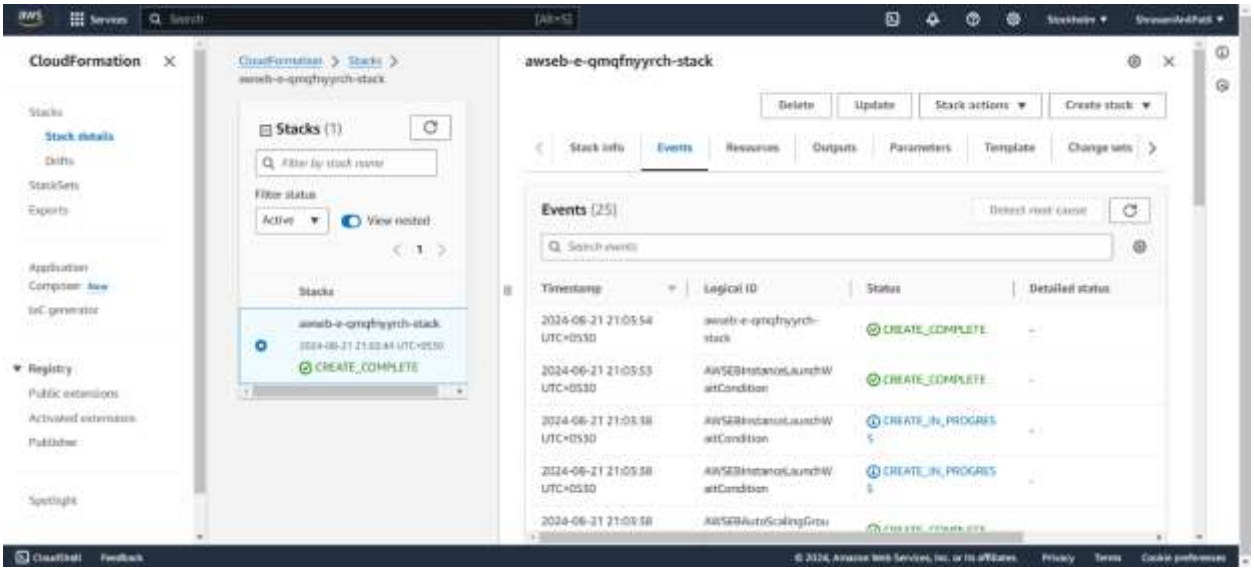

Availability Zone
Subnet
CIDR
Name
eu-north-1c
172.31.0.0/20
eu-north-1b
172.31.32.0/20
eu-north-1a
172.31.16.0/20
Database
Choose database subnets (3)
Availability Zone
Subnet
CIDR
Name
eu-north-1c
172.31.0.0/20
eu-north-1b
172.31.32.0/20
eu-north-1a
172.31.16.0/20

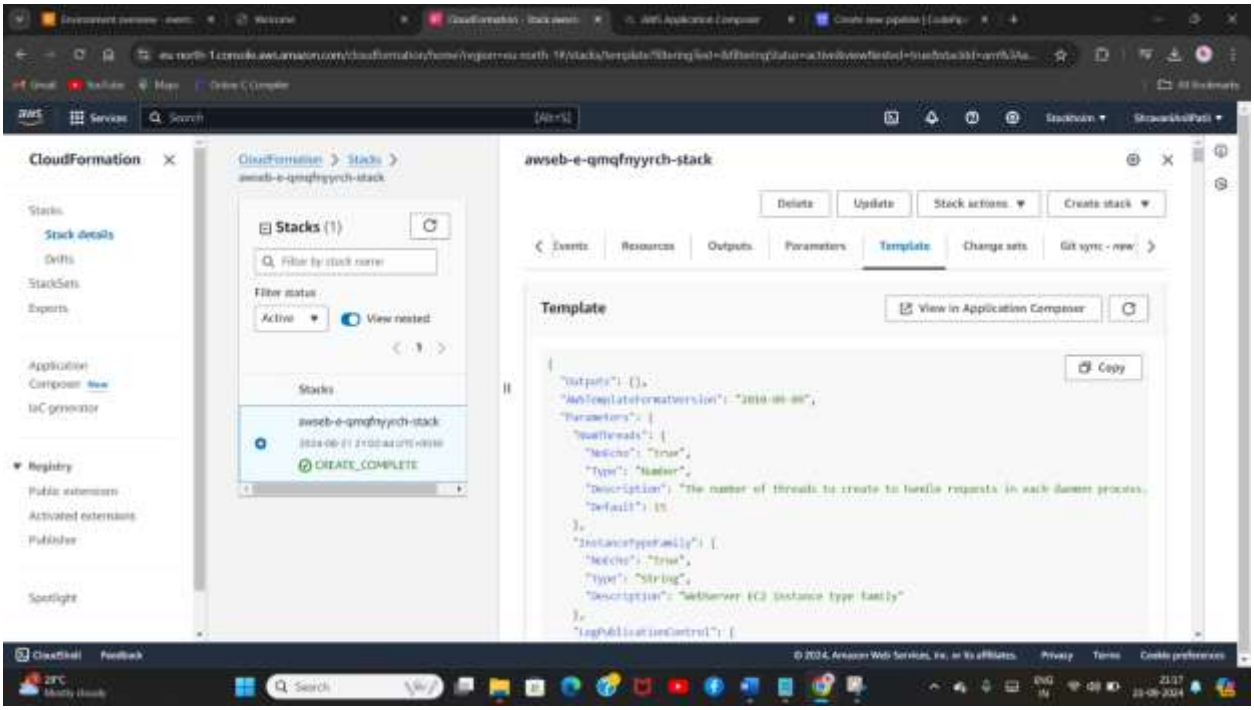

Set up networking, database, and tags - optional
Configure environment
Configure service access
Set up networking, database, and tags
Configure instance traffic and scaling
Configure updates, monitoring, and logging
Review
Virtual Private Cloud (VPC)
VPC
Create custom VPC
Instance settings
Public IP address
Activated
Instance subnets

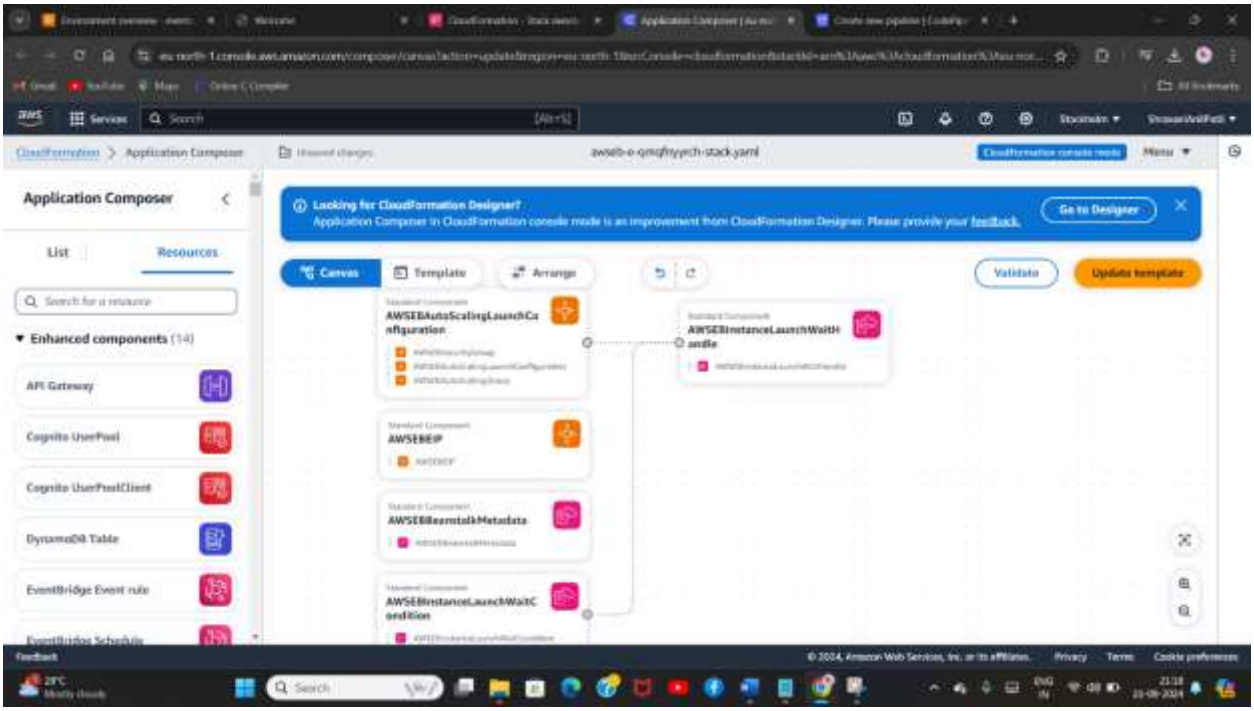

Review
Step 1: Configure environment
Edit
Environment information
Environment tier
Web server environment
Application name
MyWebApp_Shravani_38
Environment name
MyWebApp Shravani38-env
Application code
Sample application
Platform
arn:aws:elasticbeanstalk:eu-north-1::platform/Python
3.11 running on 64bit Amazon Linux 2023/4.1.3
Step 2: Configure service access
Edit
Service access
Review
CloudShell
Feedback
Privacy
Terms
Cookie preferences


Environment successfully launched.
MyWebAppShravani38-env
Actions
Upload and deploy
Environment overview
Platform
Change version
Health
Environment ID
Platform
Python 3.11 running on 64bit Amazon Linux 2023/4.1.3
Domain
Application name
Running version
Platform state
Supported
Events
Health
Logs
Monitoring
Alarms
Managed updates
Tags
Events (12)

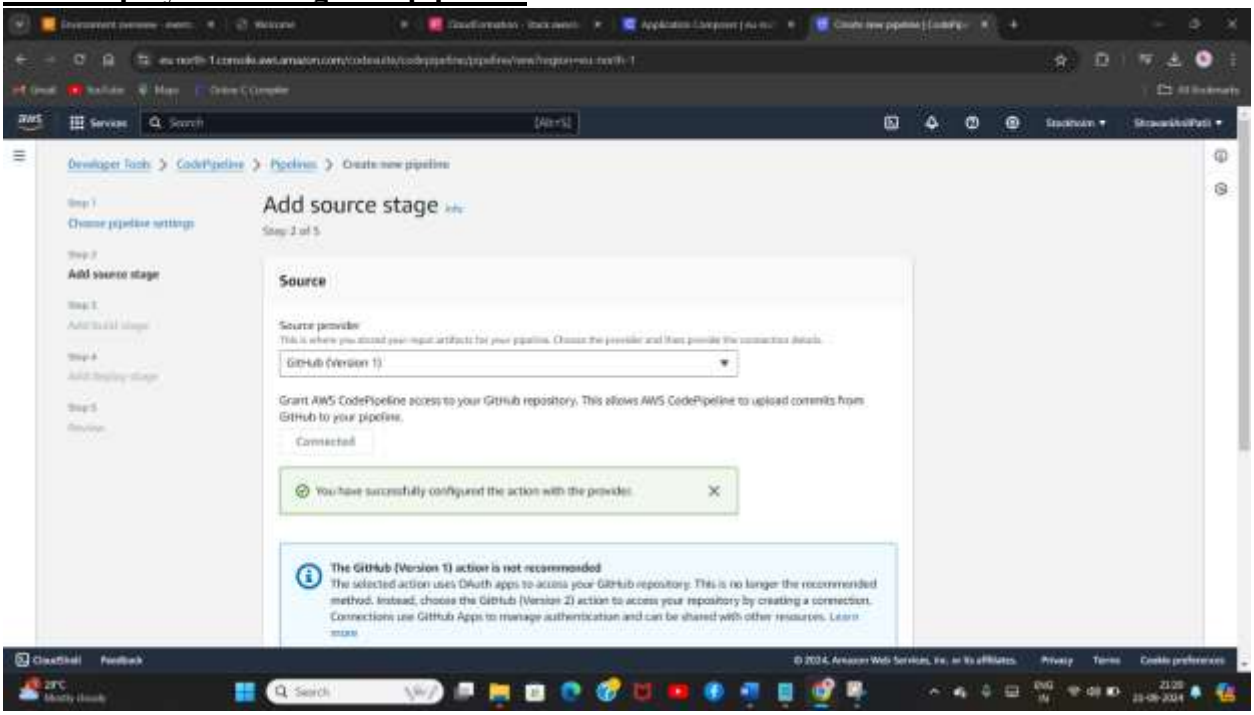

Congratulations
What's Next?
AWS Elastic Beanstalk overview
AWS Elastic Beanstalk concepts
Deploy a Django Application to AWS Elastic Beanstalk
Deploy a Flask Application to AWS Elastic Beanstalk
Customizing and Configuring a Python Container
Working with Logs

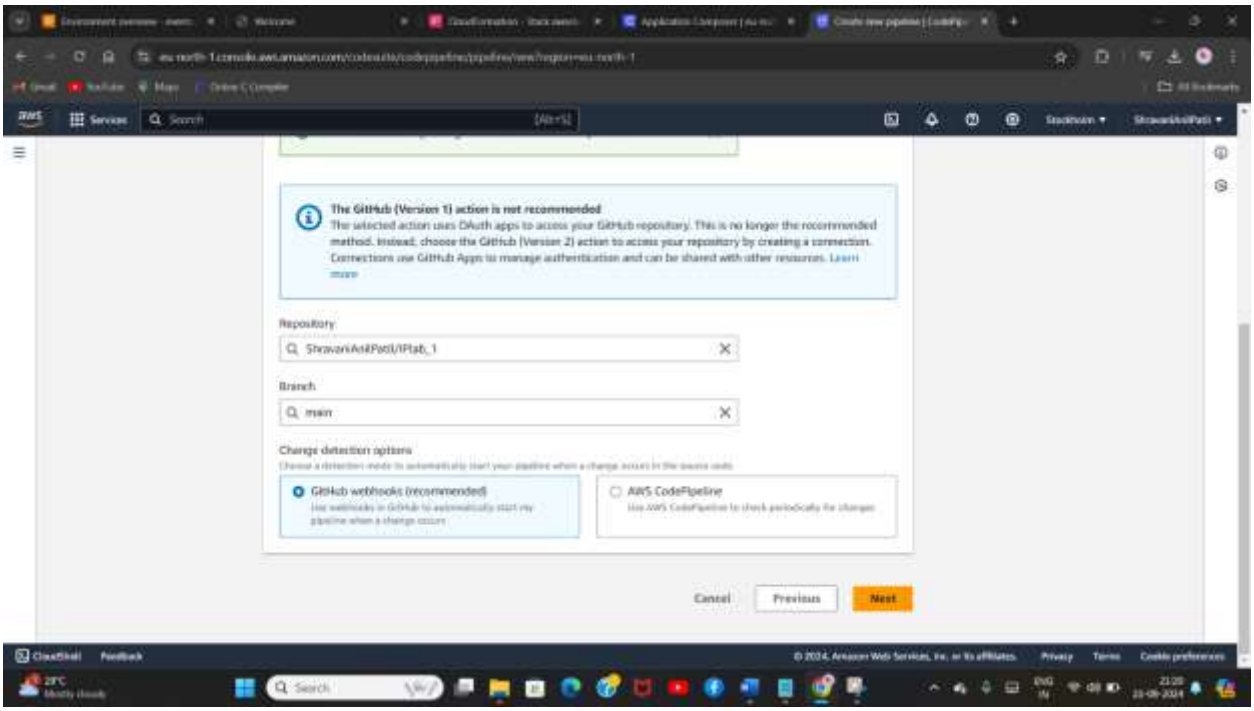

CloudFormation
Stacks (1)
awseb-e-qmqfnyyrch-stack
CREATE_COMPLETE
Events (25)
Timestamp
Logical ID
Status
Detailed status

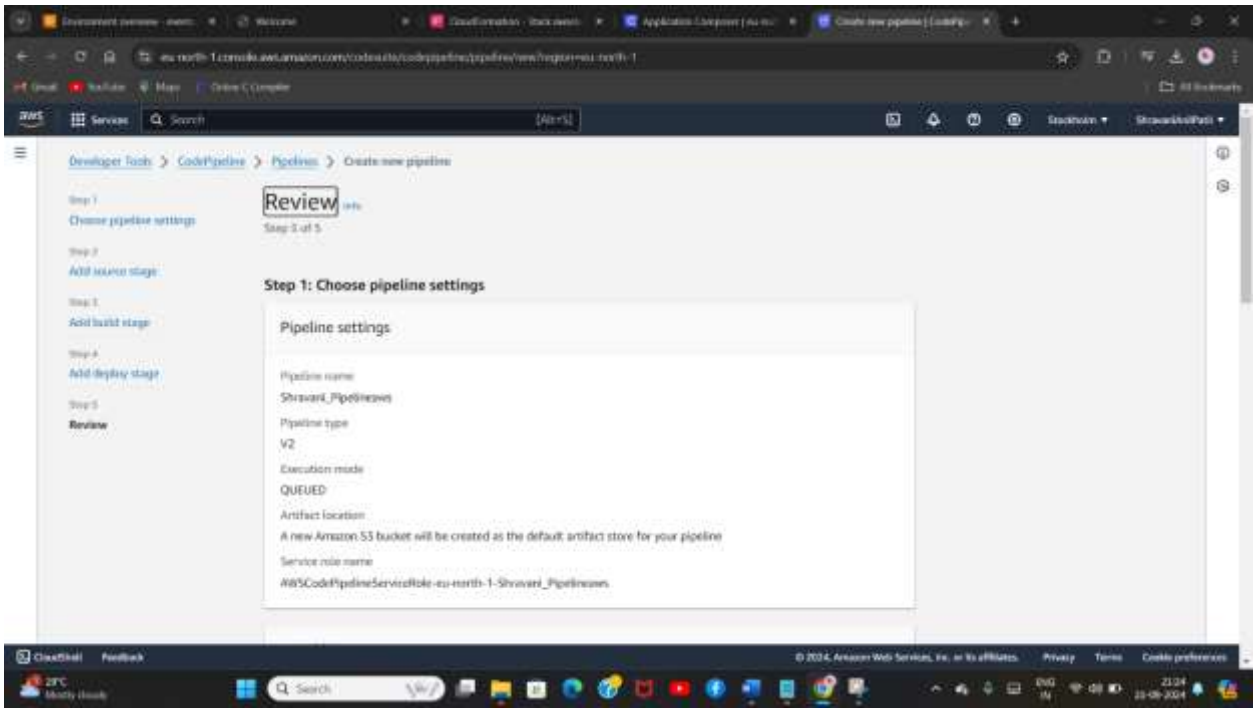

CloudFormation
Stacks (1)
awseb-e-qmqfnyyrch-stack
CREATE_COMPLETE
Template
View in Application Composer
Copy

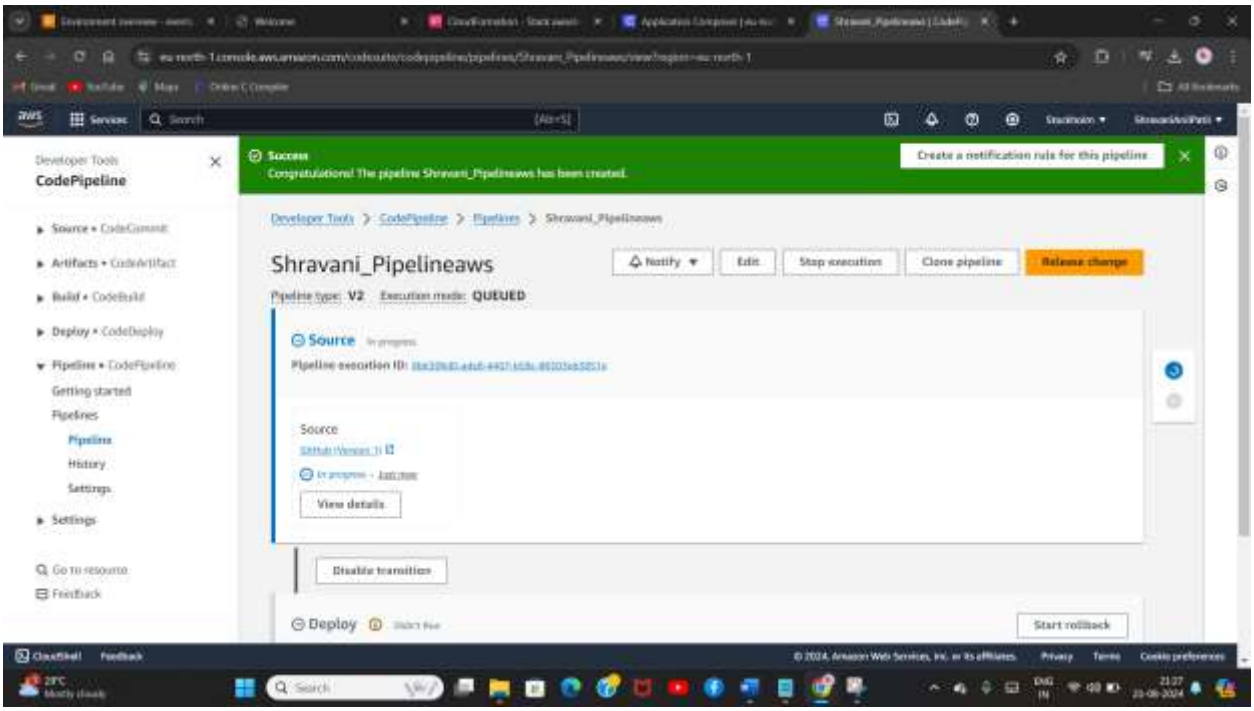


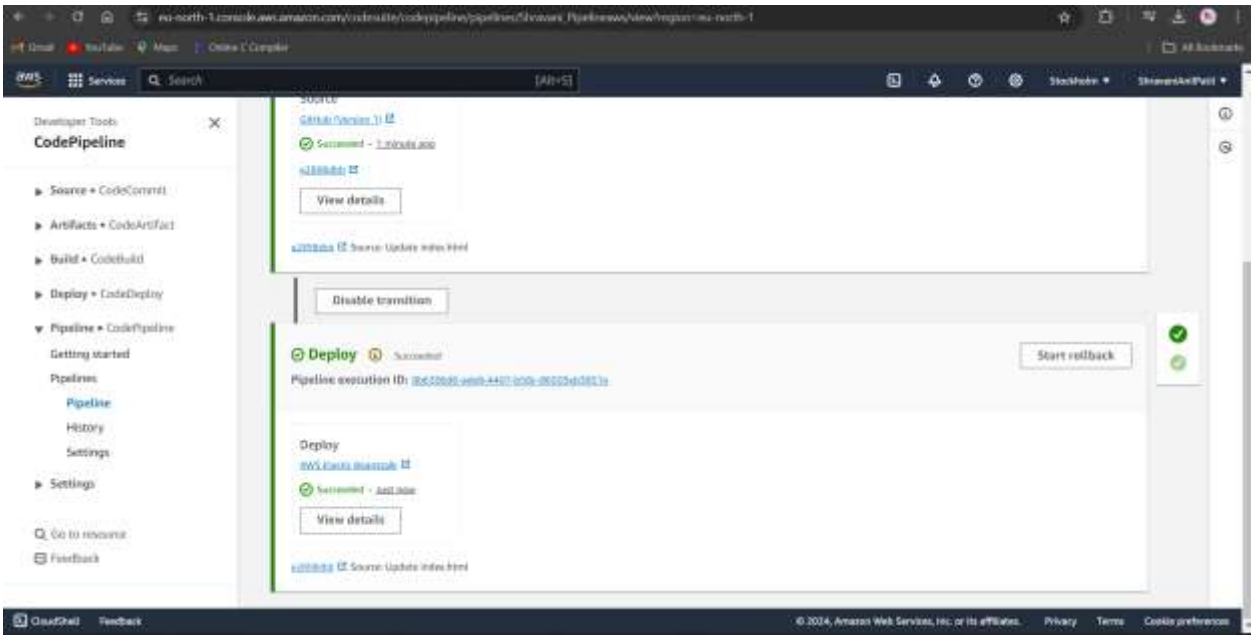
Code Deployment using Codepipeline:

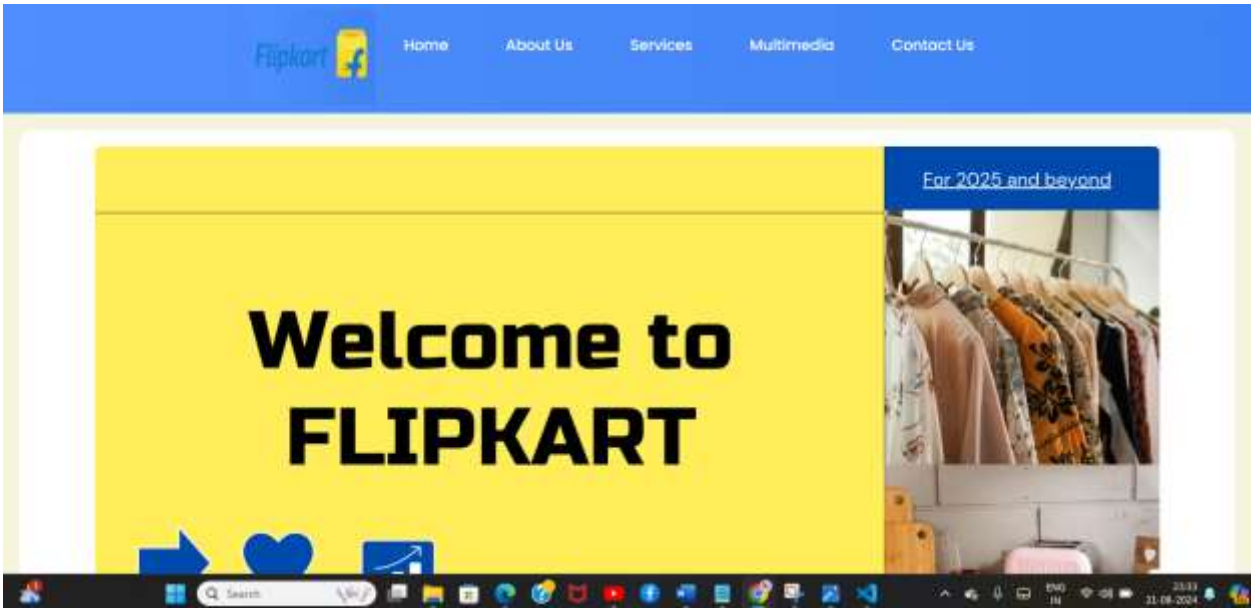



The GitHub (Version 1) action is not recommended
Repository
ShravaniAnilPatil/IPlab_1
Branch
main
Change detection options
GitHub webhooks (recommended)
AWS CodePipeline
Cancel
Previous
Next


Developer Tools > CodePipeline > Pipelines > Create new pipeline
Review
Step 1: Choose pipeline settings
Pipeline settings
Pipeline name
Shravani_Pipelinenew
Pipeline type
V2
Execution mode
QUEUED
Artifact location
A new Amazon S3 bucket will be created as the default artifact store for your pipeline
Service role name
AWSCodePipelineServiceRole-eu-north-1-Shravani_Pipelinenew


Success
Congratulations! The pipeline Shravani_Pipelineaws has been created.
Create a notification rule for this pipeline
Developer Tools
CodePipeline
Shravani_Pipelineaws
Notify
Edit
Stop execution
Clone pipeline
Release change
Pipeline type: V2
Execution mode: QUEUED
Source
Source
View details
Disable transition
Deploy
Start rollback


Developer Tools
CodePipeline
View details
Disable transition
Deploy
Start rollback
Pipeline execution ID:
Deploy
View details


Flipkart
Home
About Us
Services
Multimedia
Contact Us
Welcome to
FLIPKART
For 2025 and beyond

-