EXPERIMENT NO. 2

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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 :

AWS Elastic Beanstalk

AWS Elastic Beanstalk is a Platform as a Service (PaaS) offering from Amazon Web Services (AWS) that allows developers to deploy and manage applications in the AWS Cloud without needing to manage the underlying infrastructure. It automates the deployment process, including provisioning resources like EC2 instances, load balancers, and databases, making it easier to manage web applications and services.

Key Features of Elastic Beanstalk:

1. **Ease of Use:**
   Elastic Beanstalk simplifies the deployment process by automatically handling the infrastructure setup, deployment, monitoring, and scaling of your application. Developers can focus on writing code without worrying about the underlying infrastructure.
2. **Support for Multiple Languages and Frameworks:**
   Elastic Beanstalk supports a wide range of programming languages and frameworks, including Java, .NET, Node.js, PHP, Python, Ruby, Go, and Docker.
3. **Automatic Scaling:**
   Elastic Beanstalk automatically scales your application up or down based on the demand. It adjusts the number of instances running your application to meet traffic demands, ensuring optimal performance and cost-efficiency.
4. **Health Monitoring:**
   Elastic Beanstalk monitors the health of your applications and provides detailed logs and metrics. It automatically replaces any failed instances to maintain application availability.
5. **Environment Management:**
   Elastic Beanstalk allows you to manage multiple environments (such as development, testing, and production) for your application. You can easily deploy updates to specific environments without affecting others.

AWS CodeBuild

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages ready for deployment. It allows you to automate the build process, ensuring that your code is compiled and tested consistently across all development environments.

Key Features of AWS CodeBuild:

1. **Fully Managed Build Service:**
   CodeBuild eliminates the need to set up, patch, update, and manage your own build servers. AWS manages the infrastructure, allowing you to focus on developing and testing your code.
2. **Scalability:**
   CodeBuild scales automatically to handle multiple builds concurrently, ensuring that your builds are processed quickly, even during peak times.
3. **Custom Build Environments:**
   CodeBuild allows you to define custom build environments using Docker images. This flexibility enables you to tailor the build environment to meet the specific needs of your application.
4. **Integration with Other AWS Services:**
   CodeBuild integrates seamlessly with other AWS services, such as CodePipeline, CodeCommit, and S3, allowing you to create a complete CI/CD pipeline.
5. **Pay-As-You-Go Pricing:**
   CodeBuild charges you only for the build time you use, making it a cost-effective solution for automating your build process.

Deploying on S3 Using AWS CodePipeline

AWS CodePipeline is a fully managed continuous delivery service that automates the build, test, and deployment phases of your release process. CodePipeline integrates with other AWS services, including CodeBuild and S3, to automate the entire application release process.

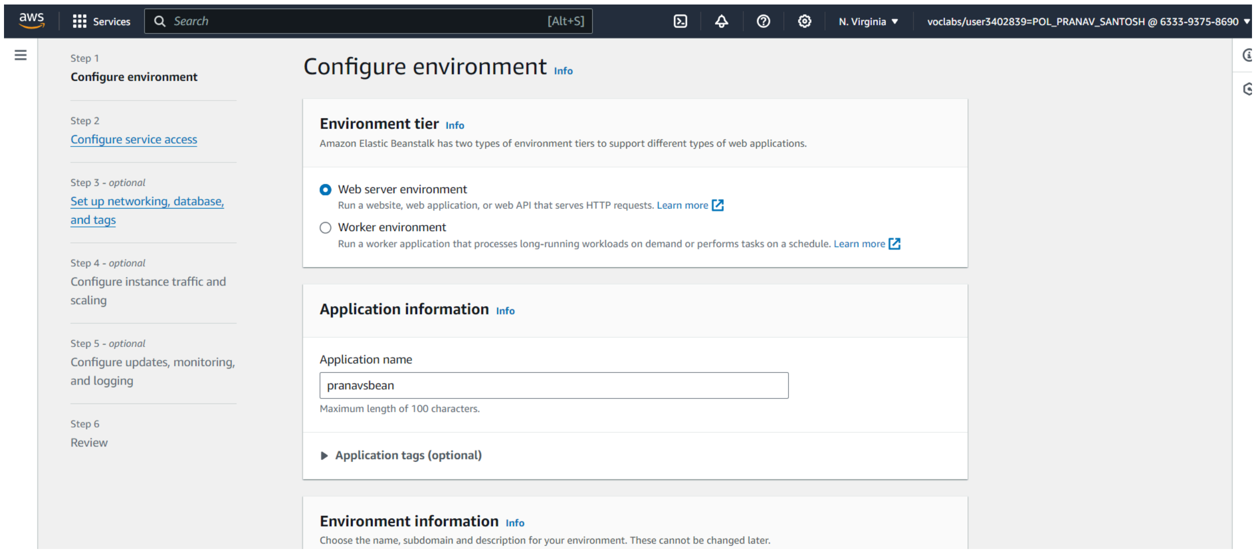
Key Steps to Deploy on S3 Using AWS CodePipeline:

1. **Source Stage:**
   The first stage in the pipeline is typically the source stage, where the source code is retrieved from a repository, such as AWS CodeCommit, GitHub, or S3. This code serves as the input for the subsequent build and deployment stages.

2. **Build Stage (Using AWS CodeBuild):**
   In the build stage, CodePipeline triggers a build process using AWS CodeBuild. CodeBuild compiles the source code, runs tests, and packages the application. The output is an artifact that is stored in an S3 bucket, ready for deployment.
3. **Deploy Stage (Deploying to S3):**
   The final stage in the pipeline is the deploy stage. In this stage, CodePipeline automatically deploys the artifacts generated in the build stage to an S3 bucket. The S3 bucket can serve as a static website hosting service or store files that are accessed by your application.
4. **Automation and Notifications:**
   CodePipeline can be configured to trigger automatic builds and deployments based on changes to the source code repository. It can also be integrated with Amazon SNS to send notifications about the status of the pipeline, allowing you to monitor the release process.
5. **Pipeline Monitoring:**
   CodePipeline provides visual monitoring and logging of each stage in the pipeline, making it easier to track the status of your builds and deployments. Any failures in the pipeline can be quickly identified and addressed.

Implementation ;
Elastic Beanstalk

Step 1: create environment


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3402839=POL_PRANAV_SANTOSH @ 6333-9375-8690
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

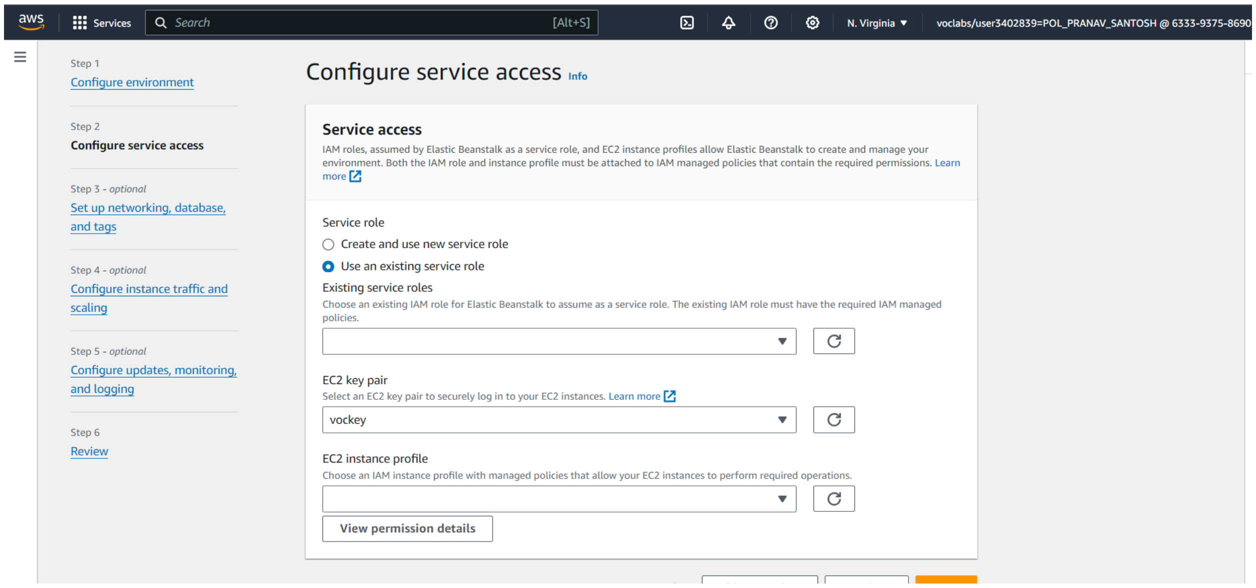
Application information Info

Application name
pranavsbean
Maximum length of 100 characters.

▶ Application tags (optional)

Environment information Info
Choose the name, subdomain and description for your environment. These cannot be changed later.

Step 2 : add your Ec2 key pair and instance profile



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 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]

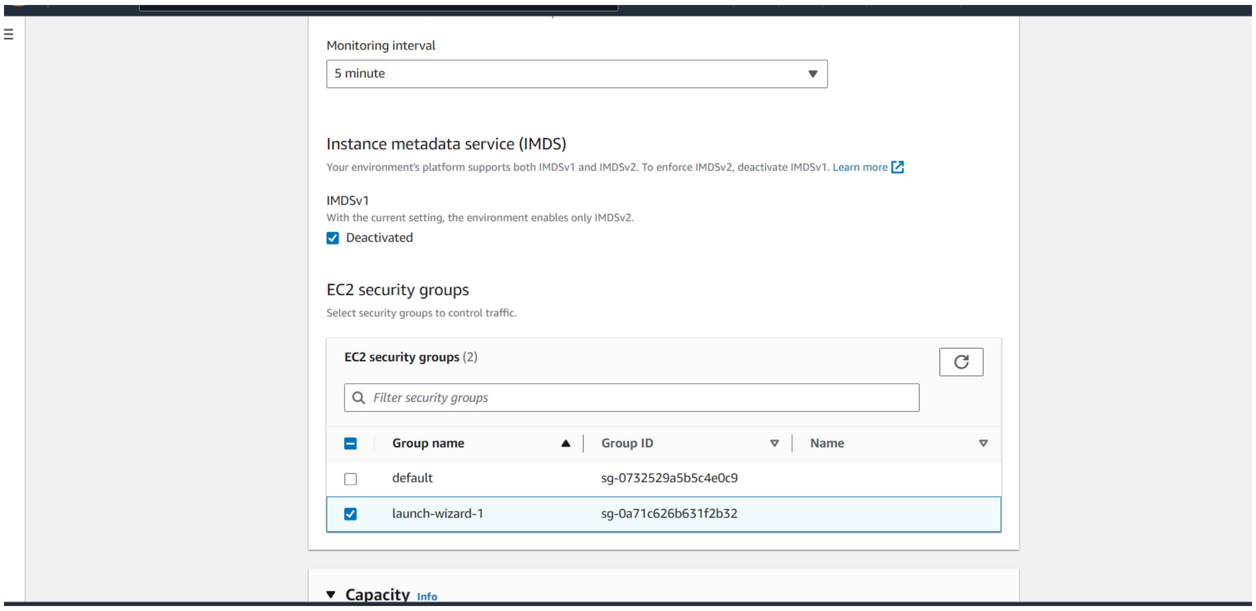
Existing service roles
Choose an existing IAM role for Elastic Beanstalk to assume as a service role. The existing IAM role must have the required IAM managed policies.

EC2 key pair
Select an EC2 key pair to securely log in to your EC2 instances. Learn more
vockey

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

View permission details

Step 3 : add security config and review all settings



Monitoring interval
5 minute

Instance metadata service (IMDS)
Your environment's platform supports both IMDSv1 and IMDSv2. To enforce IMDSv2, deactivate IMDSv1. Learn more

IMDSv1
With the current setting, the environment enables only IMDSv2.
✓ Deactivated

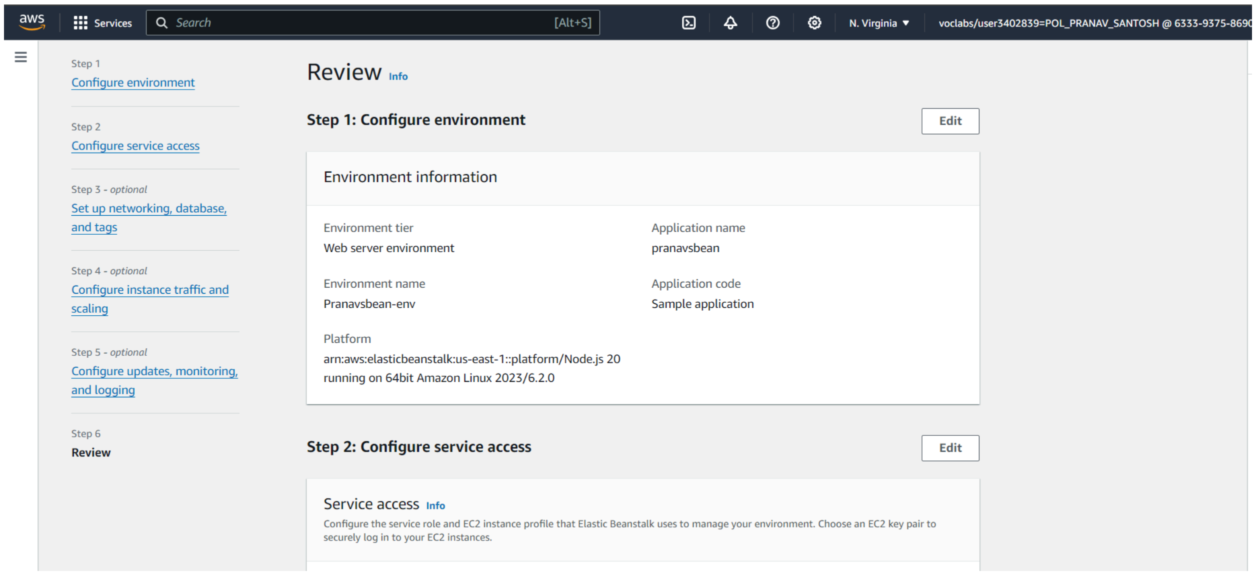
EC2 security groups
Select security groups to control traffic.

EC2 security groups (2)

Filter security groups

	Group name	Group ID	Name
☐	default	sg-0732529a5b5c4e0c9	
✓	launch-wizard-1	sg-0a71c626b631f2b32	

Capacity Info



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

Review Info

Step 1: Configure environment Edit

Environment information

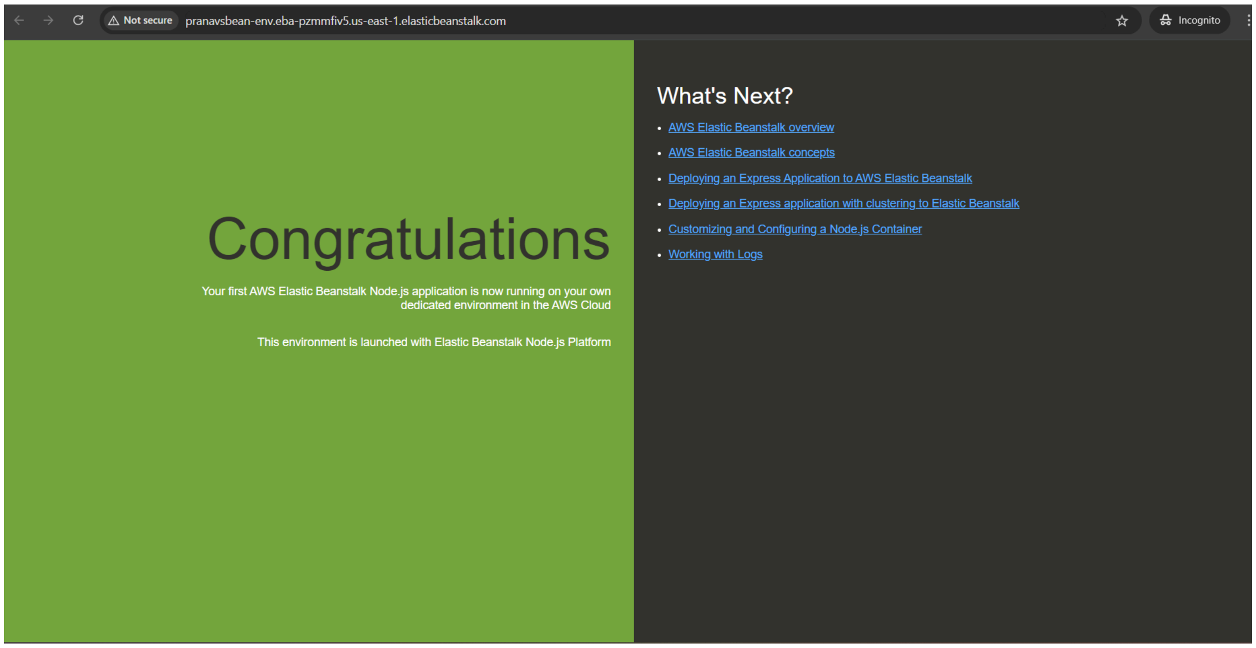
Environment tier Web server environment	Application name pranavsbean
Environment name Pranavsbean-env	Application code Sample application
Platform arn:aws:elasticbeanstalk:us-east-1::platform/Node.js 20 running on 64bit Amazon Linux 2023/6.2.0	

Step 2: Configure service access Edit

Service access Info

Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.

Step 4 : Beanstalk environment is created



Congratulations

Your first AWS Elastic Beanstalk Node.js application is now running on your own dedicated environment in the AWS Cloud

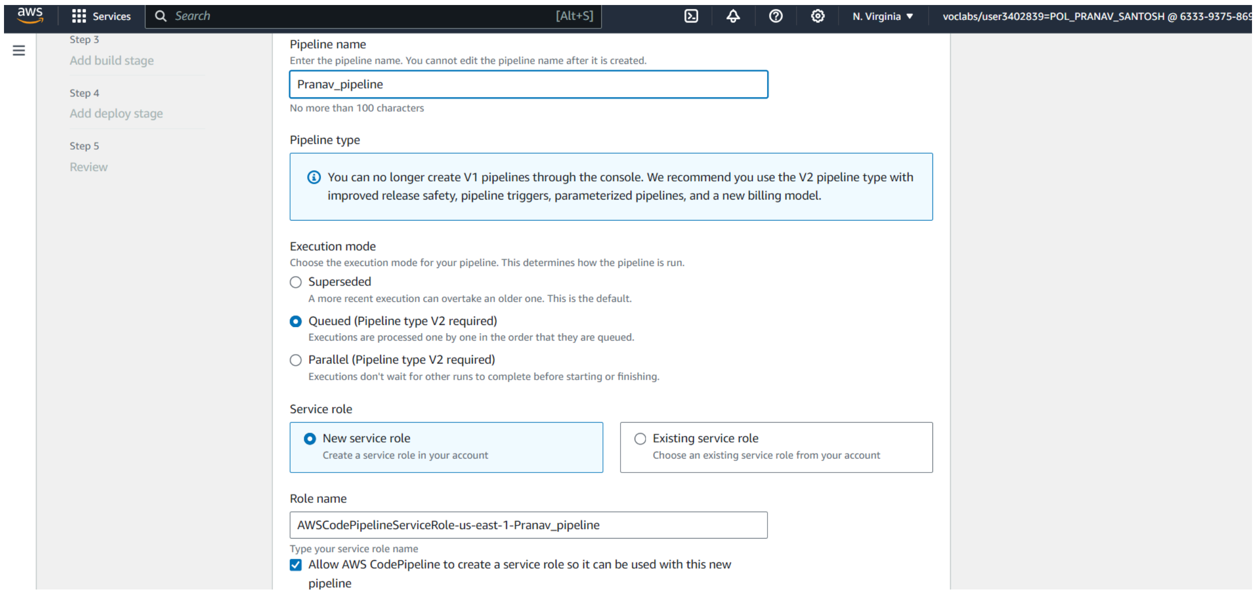
This environment is launched with Elastic Beanstalk Node.js Platform

What's Next?

- AWS Elastic Beanstalk overview
- AWS Elastic Beanstalk concepts
- Deploying an Express Application to AWS Elastic Beanstalk
- Deploying an Express application with clustering to Elastic Beanstalk
- Customizing and Configuring a Node.js Container
- Working with Logs

Pipeline Creation

Step 1 : click on create pipeline and give name



Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
Pranav_pipeline
No more than 100 characters

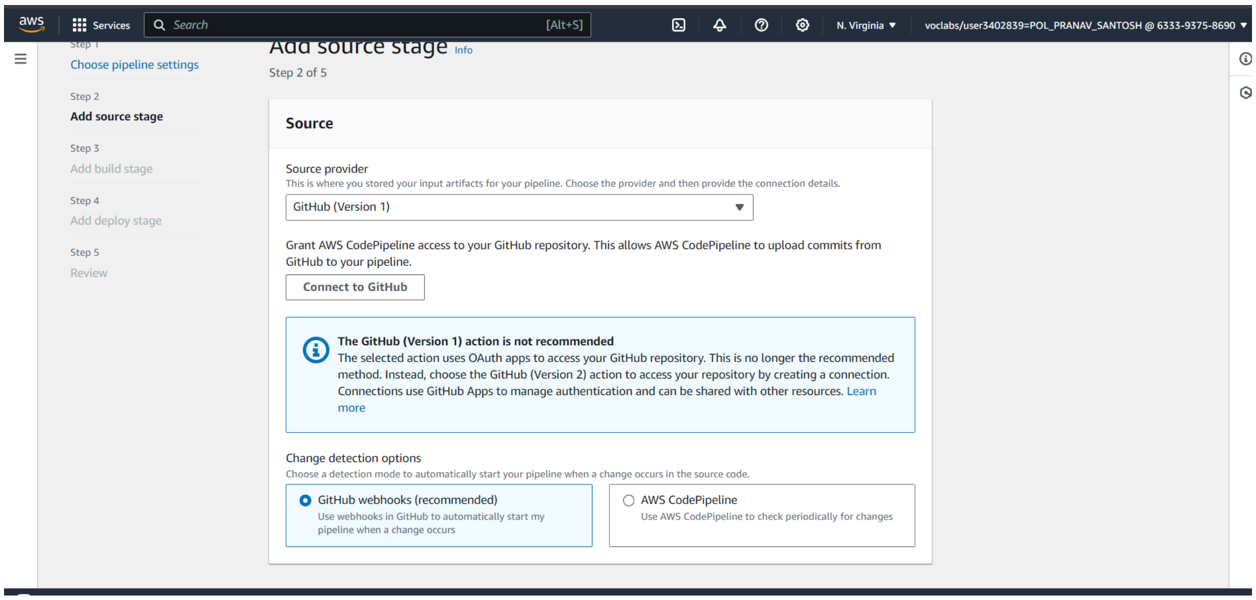
Pipeline type
You can no longer create V1 pipelines through the console. We recommend you use the V2 pipeline type with improved release safety, pipeline triggers, parameterized pipelines, and a new billing model.

Execution mode
Choose the execution mode for your pipeline. This determines how the pipeline is run.
- Superseded
  A more recent execution can overtake an older one. This is the default.
- [✓ Queued (Pipeline type V2 required)]
  Executions are processed one by one in the order that they are queued.
- Parallel (Pipeline type V2 required)
  Executions don't wait for other runs to complete before starting or finishing.

Service role
- [✓ New service role] Create a service role in your account
- Existing service role Choose an existing service role from your account

Role name
AWSCodePipelineServiceRole-us-east-1-Pranav_pipeline
Type your service role name
[✓] Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

Step 2 : Add Your github account and add the file to add to pipeline deployment



Add source stage Info
Step 2 of 5

Source

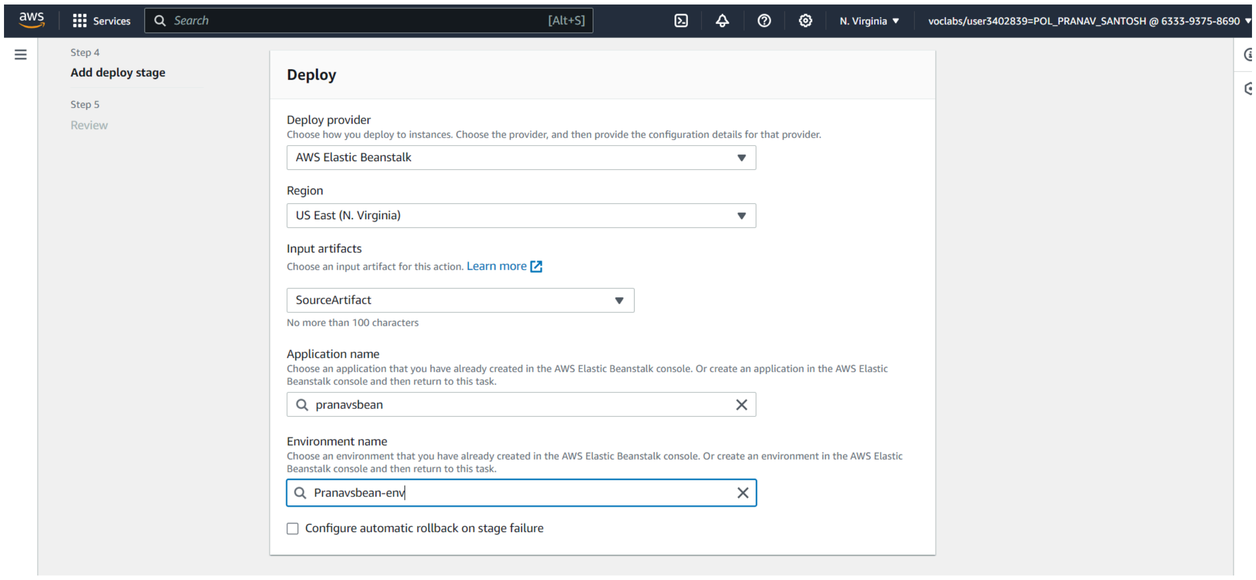
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (Version 1)

Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.
Connect to GitHub

The GitHub (Version 1) action is not recommended
The selected action uses OAuth apps to access your GitHub repository. This is no longer the recommended method. Instead, choose the GitHub (Version 2) action to access your repository by creating a connection. Connections use GitHub Apps to manage authentication and can be shared with other resources. Learn more

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.
- [✓ GitHub webhooks (recommended)] Use webhooks in GitHub to automatically start my pipeline when a change occurs
- AWS CodePipeline Use AWS CodePipeline to check periodically for changes

Step 3 : Add deploy config choosing the elastic beanstalk



Deploy

Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.
AWS Elastic Beanstalk

Region
US East (N. Virginia)

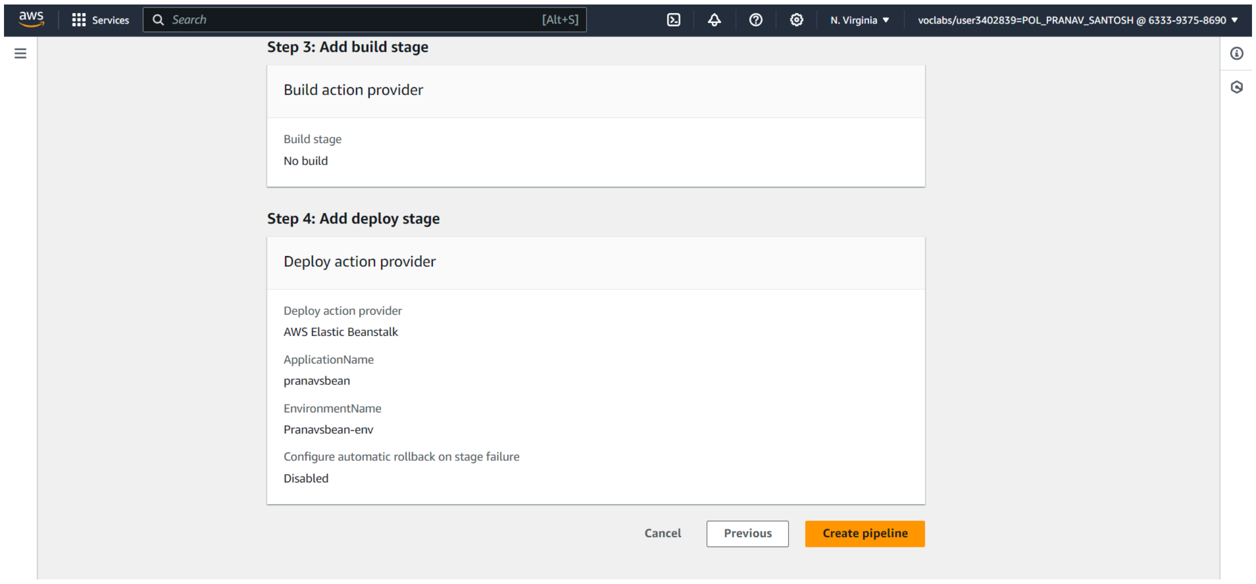
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
No more than 100 characters

Application name
Choose an application that you have already created in the AWS Elastic Beanstalk console. Or create an application in the AWS Elastic Beanstalk console and then return to this task.
pranavsbean

Environment name
Choose an environment that you have already created in the AWS Elastic Beanstalk console. Or create an environment in the AWS Elastic Beanstalk console and then return to this task.
Pranavsbean-env

☐ Configure automatic rollback on stage failure

Step 4 : review changes and submit



Step 3: Add build stage

Build action provider

Build stage
No build

Step 4: Add deploy stage

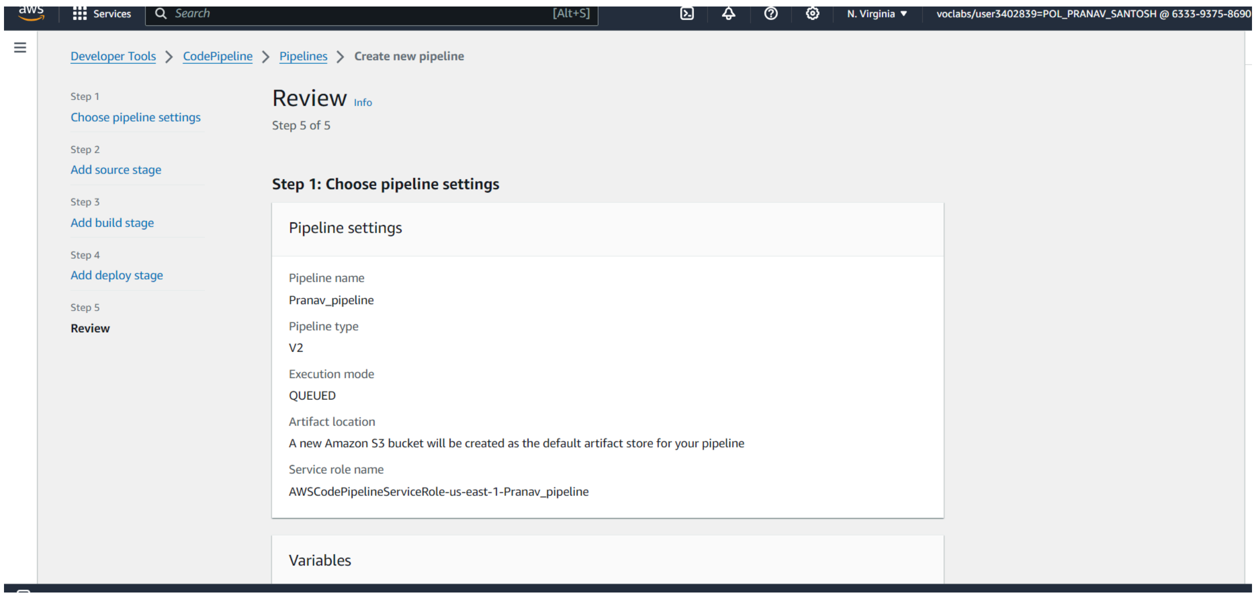
Deploy action provider

Deploy action provider
AWS Elastic Beanstalk

ApplicationName
pranavsbean

EnvironmentName
Pranavsbean-env

Configure automatic rollback on stage failure
Disabled



Review Info

Step 5 of 5

Step 1: Choose pipeline settings

Pipeline settings

Pipeline name
Pranav_pipeline

Pipeline type
V2

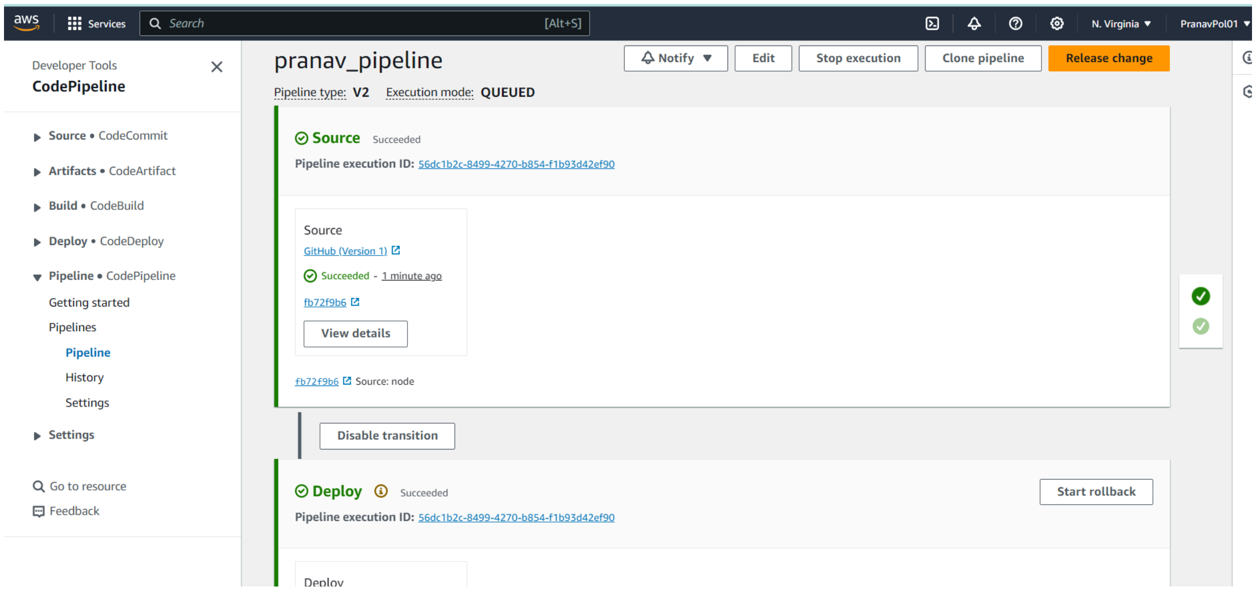
Execution mode
QUEUED

Artifact location
A new Amazon S3 bucket will be created as the default artifact store for your pipeline

Service role name
AWSCodePipelineServiceRole-us-east-1-Pranav_pipeline

Variables

Step 5 : view the pipeline build and deployment



pranav_pipeline

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded
Pipeline execution ID: 56dc1b2c-8499-4270-b854-f1b93d42ef90

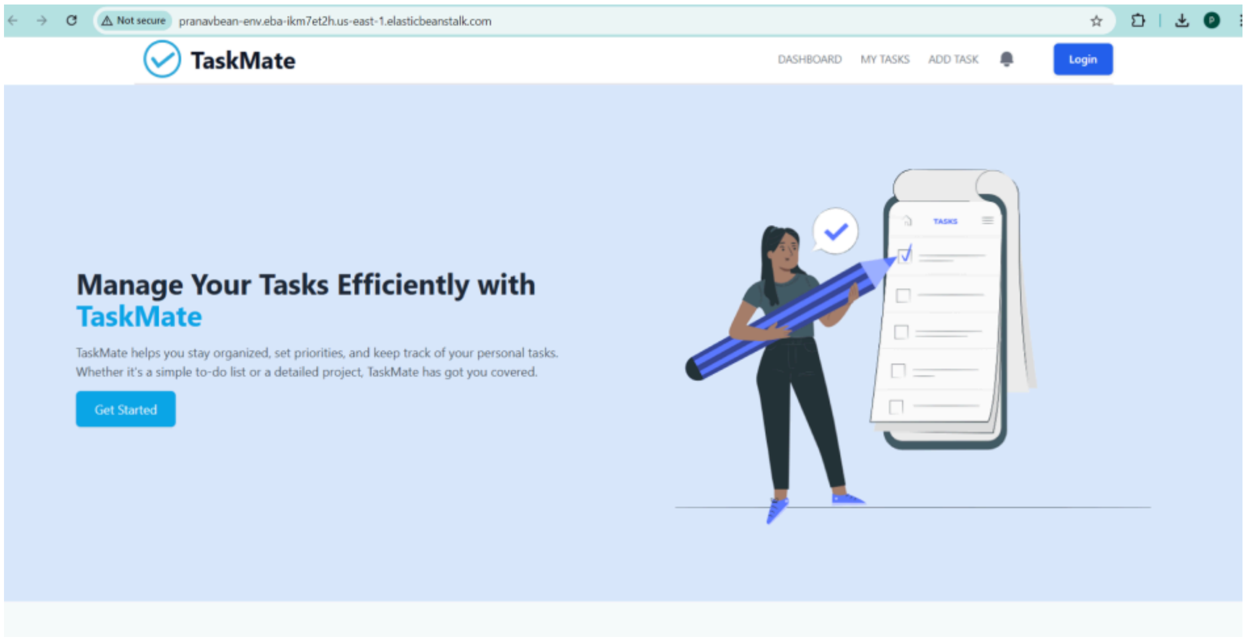
Source
GitHub (Version 1)
Succeeded - 1 minute ago
fb72f9b6

fb72f9b6 Source: node

Deploy Succeeded
Pipeline execution ID: 56dc1b2c-8499-4270-b854-f1b93d42ef90

Deploy

Step 6 : Check the deployed website at beanstalk link



Manage Your Tasks Efficiently with TaskMate

TaskMate helps you stay organized, set priorities, and keep track of your personal tasks. Whether it's a simple to-do list or a detailed project, TaskMate has got you covered.