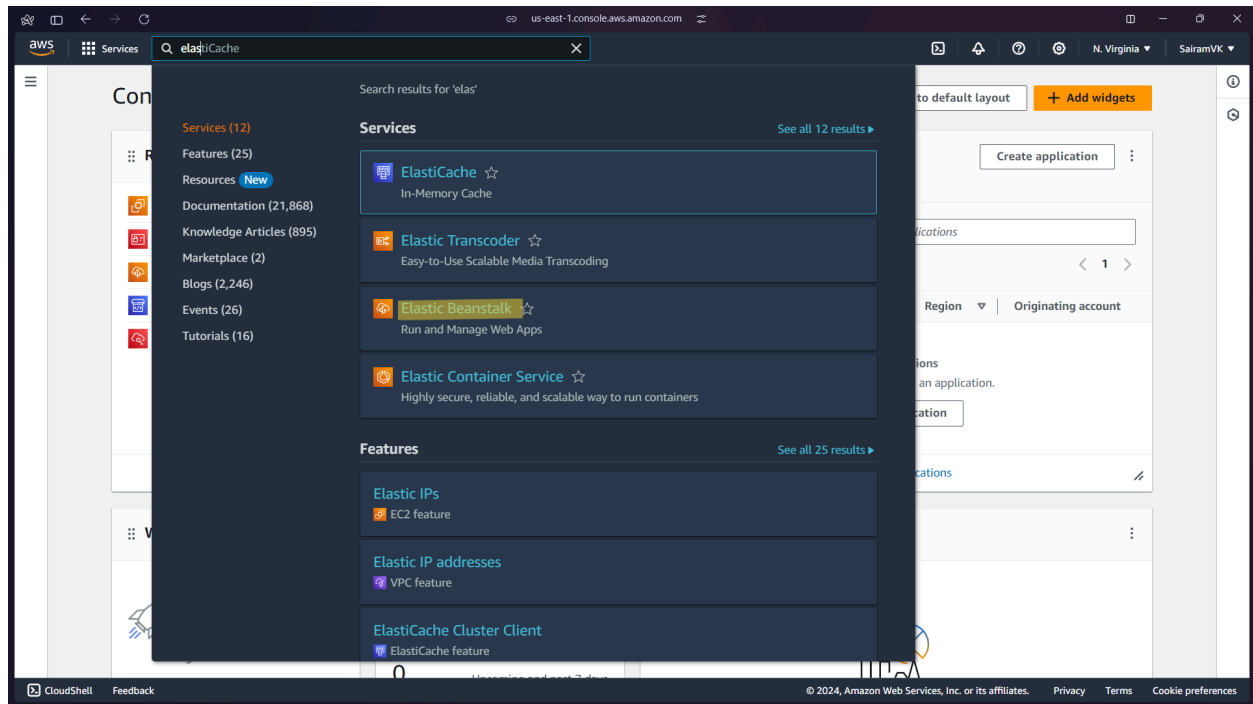
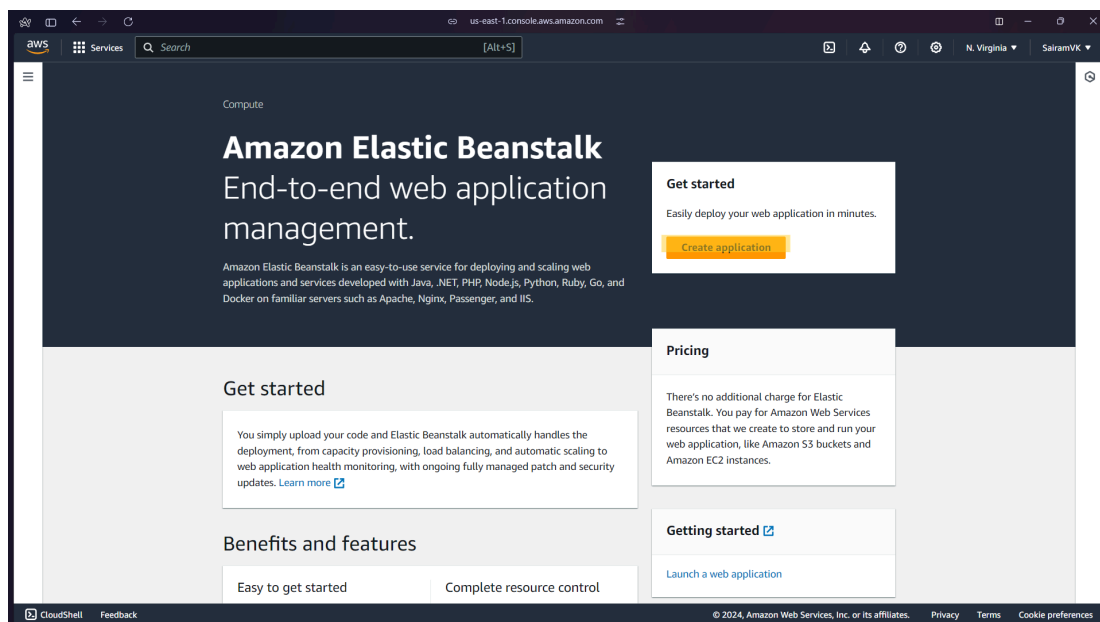


Aim: To Build Your Application using AWS CodeBuild and Deploy on S3/ SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Step 1: Login to your AWS console. Search for Elastic Beanstalk in the searchbar near services.



Step 2: Go to Elastic Beanstalk and click on Create Application



Step 3: Enter the name of your application. Scroll down and in the platform, select platform as PHP. Keep the application code as Sample Application. Set the instance to single instance. Click on NEXT.

The screenshot shows the 'Configure environment' page in the AWS console, specifically Step 1. The left sidebar lists the steps: 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), and Step 6 (Review). The main content area is titled 'Configure environment' and includes the following sections:

- Environment tier:** Two options are shown: 'Web server environment' (selected) and 'Worker environment'. The 'Web server environment' description is 'Run a website, web application, or web API that serves HTTP requests.' The 'Worker environment' description is 'Run a worker application that processes long-running workloads on demand or performs tasks on a schedule.'
- Application information:** The 'Application name' field is filled with 'MyWebApp27'. Below it, it says 'Maximum length of 100 characters.' There is also an 'Application tags (optional)' section.
- Environment information:** The 'Environment name' field is filled with 'MyWebApp27-env'. Below it, it says 'Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.'

The bottom of the page shows the AWS logo, 'CloudShell', 'Feedback', and copyright information: '© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

The screenshot shows the 'Configure environment' page in the AWS console, specifically Step 2. The left sidebar is the same as in Step 1. The main content area is titled 'Configure environment' and includes the following sections:

- Environment description:** A text area for describing the environment.
- Platform:** The 'Platform type' section has 'Managed platform' selected. Below it, the 'Platform' dropdown is set to 'PHP'. The 'Platform branch' dropdown is set to 'PHP 8.3 running on 64bit Amazon Linux 2023'. The 'Platform version' dropdown is set to '4.3.1 (Recommended)'.
- Application code:** This section is partially visible at the bottom of the screenshot.

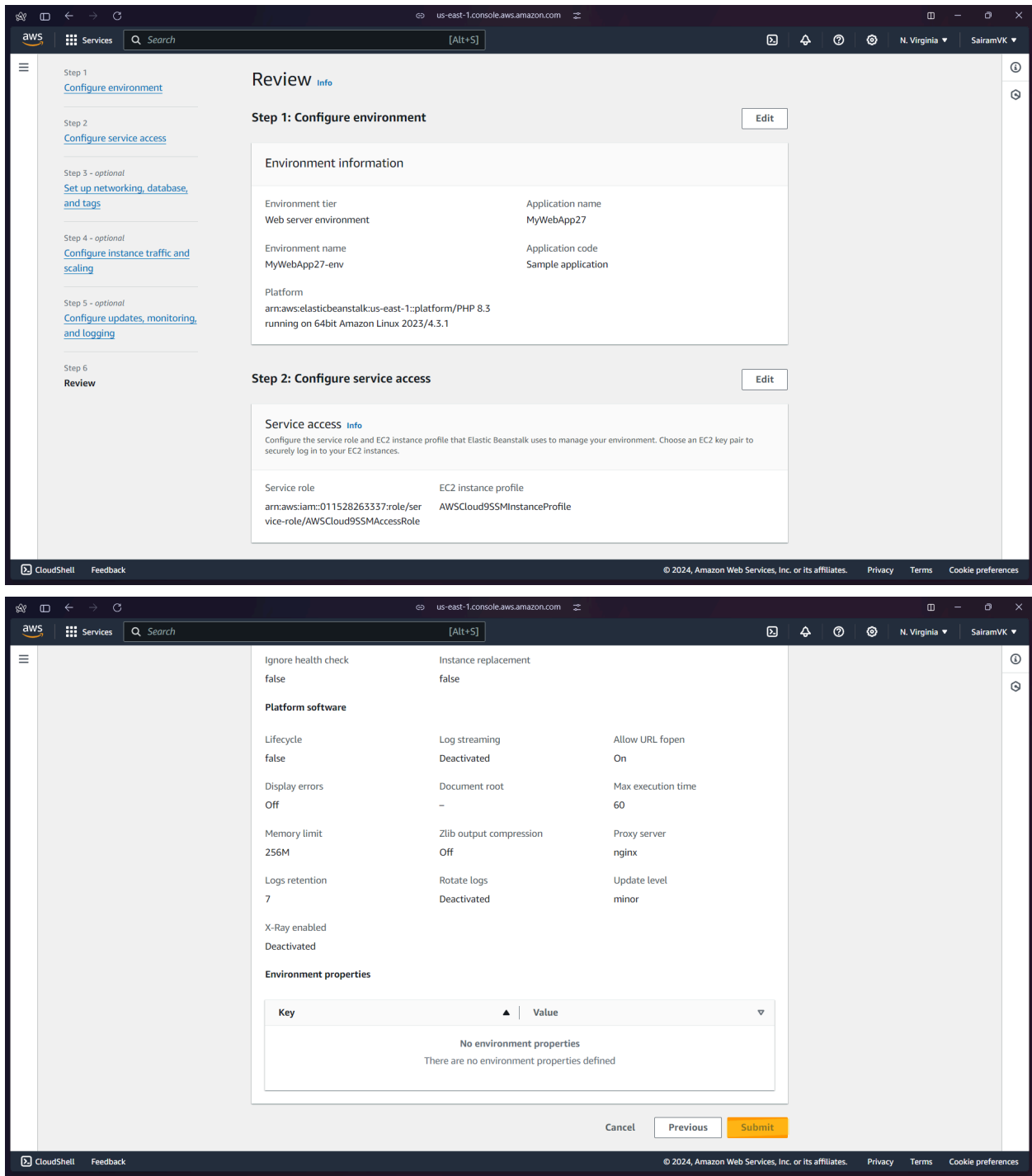
The bottom of the page shows the AWS logo, 'CloudShell', 'Feedback', and copyright information: '© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

The screenshot shows the AWS Elastic Beanstalk console in the 'Configure environment' step. The 'Platform branch' is set to 'PHP 8.3 running on 64bit Amazon Linux 2023' and the 'Platform version' is '4.3.1 (Recommended)'. Under 'Application code', the 'Sample application' option is selected. Under 'Presets', the 'Single instance (free tier eligible)' option is selected. The 'Next' button is visible at the bottom right.

Step 4: Use an existing service role and choose whatever service role is available on your account.

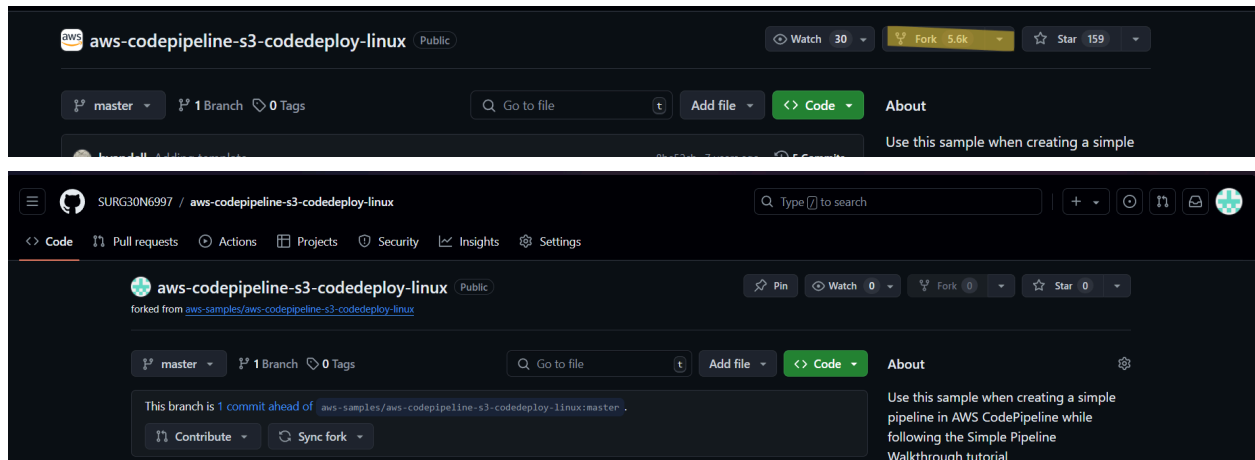
The screenshot shows the AWS Elastic Beanstalk console in the 'Configure service access' step. The 'Service role' section has 'Use an existing service role' selected, and 'AWSCloud9SSMAccessRole' is chosen from the dropdown. The 'EC2 key pair' section has 'Choose a key pair' selected. The 'EC2 instance profile' section has 'AWSCloud9SSMInstanceProfile' selected. The 'Next' button is visible at the bottom right.

Step 5: Review the settings that you have set up for your application and submit your application.

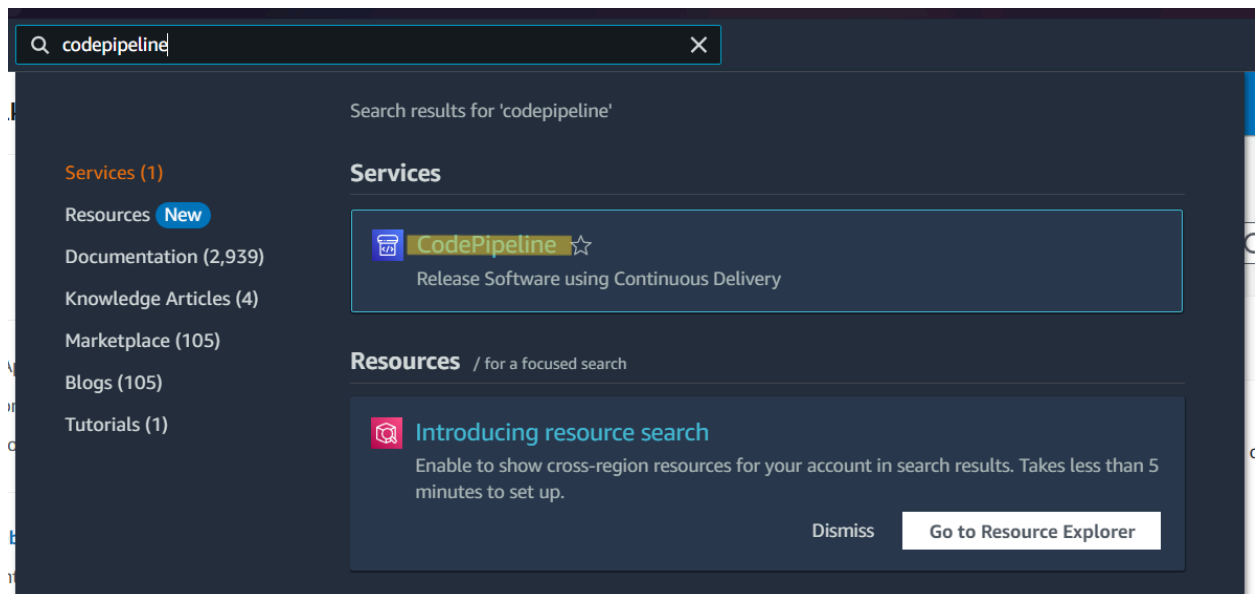


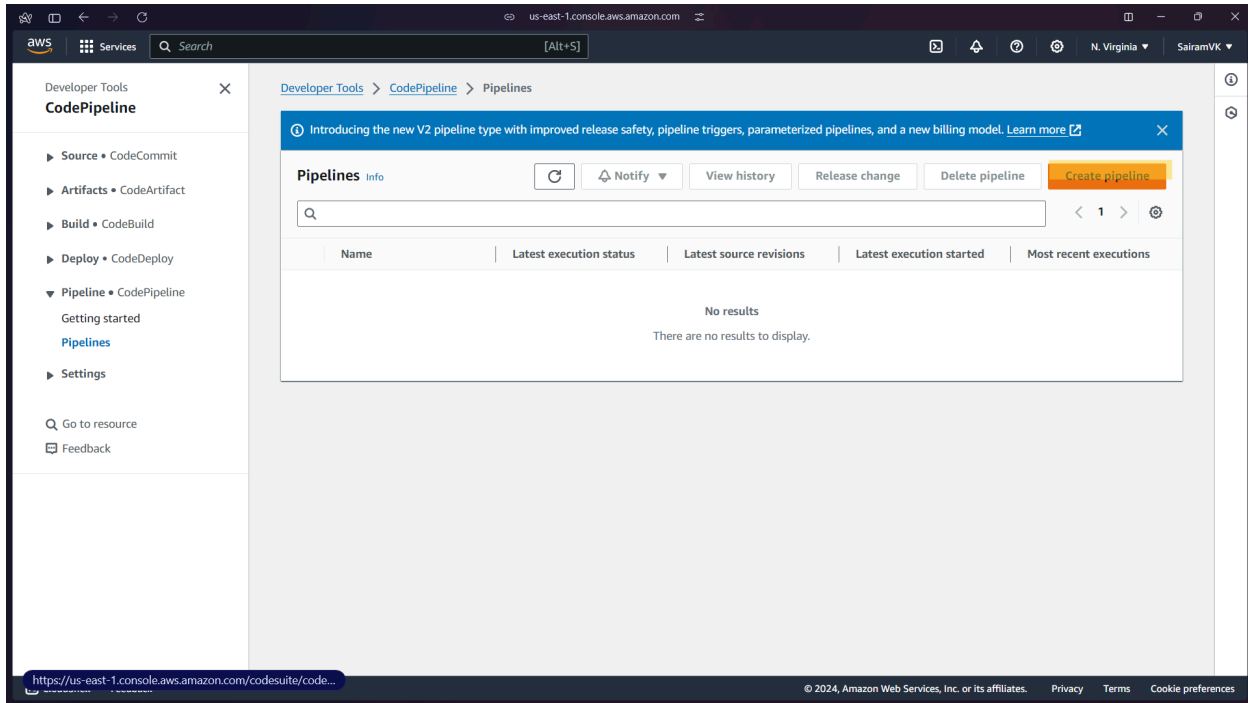
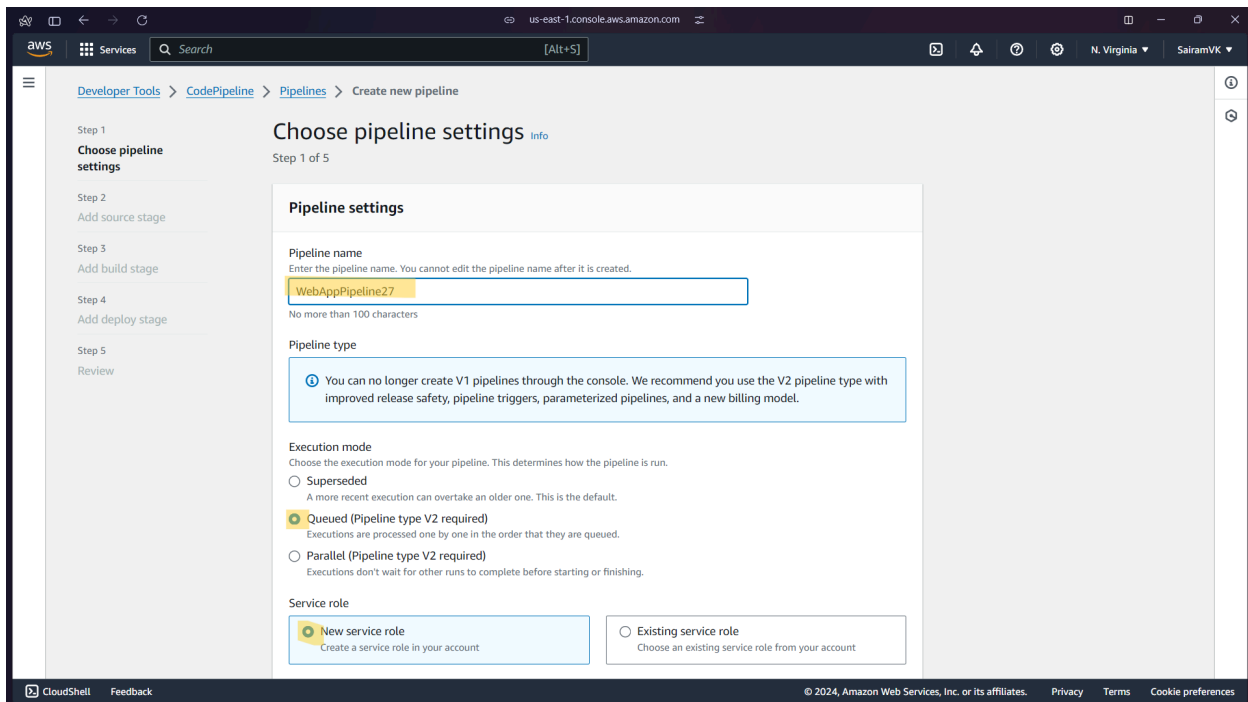
Step 6: Go to the github link below. This is a github with a sample code for deploying a file on AWS CodePipeline. Fork this repository into your personal github.

<https://github.com/aws-samples/aws-codepipeline-s3-codedeploy-linux>



Step 7: Search CodePipeline in the services tab and click on it.



Step 8: Click on Create Pipeline.**Step 9: Give a name to your Pipeline. A new service role would be created with the name of the Pipeline.**

The screenshot shows the 'Create new service role' page in the AWS IAM console. The 'Role name' field is populated with 'AWSCodePipelineServiceRole-us-east-1-WebAppPipeline27'. The checkbox 'Allow AWS CodePipeline to create a service role so it can be used with this new pipeline' is checked. The 'Variables' section indicates no variables are defined at the pipeline level, with an 'Add variable' button and a note that the first pipeline execution will fail if variables have no default values. The 'Advanced settings' section is collapsed. At the bottom, there are 'Cancel' and 'Next' buttons.

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-WebAppPipeline27

Type your service role name

☒ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

Variables

You can add variables at the pipeline level. You can choose to assign the value when you start the pipeline. Choosing this option requires pipeline type V2. [Learn more](#)

No variables defined at the pipeline level in this pipeline.

Add variable

You can add up to 50 variables.

[i](#) The first pipeline execution will fail if variables have no default values.

[▶](#) Advanced settings

Cancel Next

Step 10: Select a source provider (as Github (Version 2)). Click on Connect to Github to connect your github.

The screenshot shows the 'Add source stage' page in the AWS CodePipeline console. The 'Source provider' dropdown is set to 'GitHub (Version 2)'. A blue box highlights the 'New GitHub version 2 (app-based) action' section. The 'Connection' section has a search bar and a 'Connect to GitHub' button. The 'Repository name' and 'Default branch' fields are also present.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1
[Choose pipeline settings](#)

Step 2
Add source stage

Step 3
Add build stage

Step 4
Add deploy stage

Step 5
[Review](#)

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 2)

[i](#) **New GitHub version 2 (app-based) action**
To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. [Learn more](#)

Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.

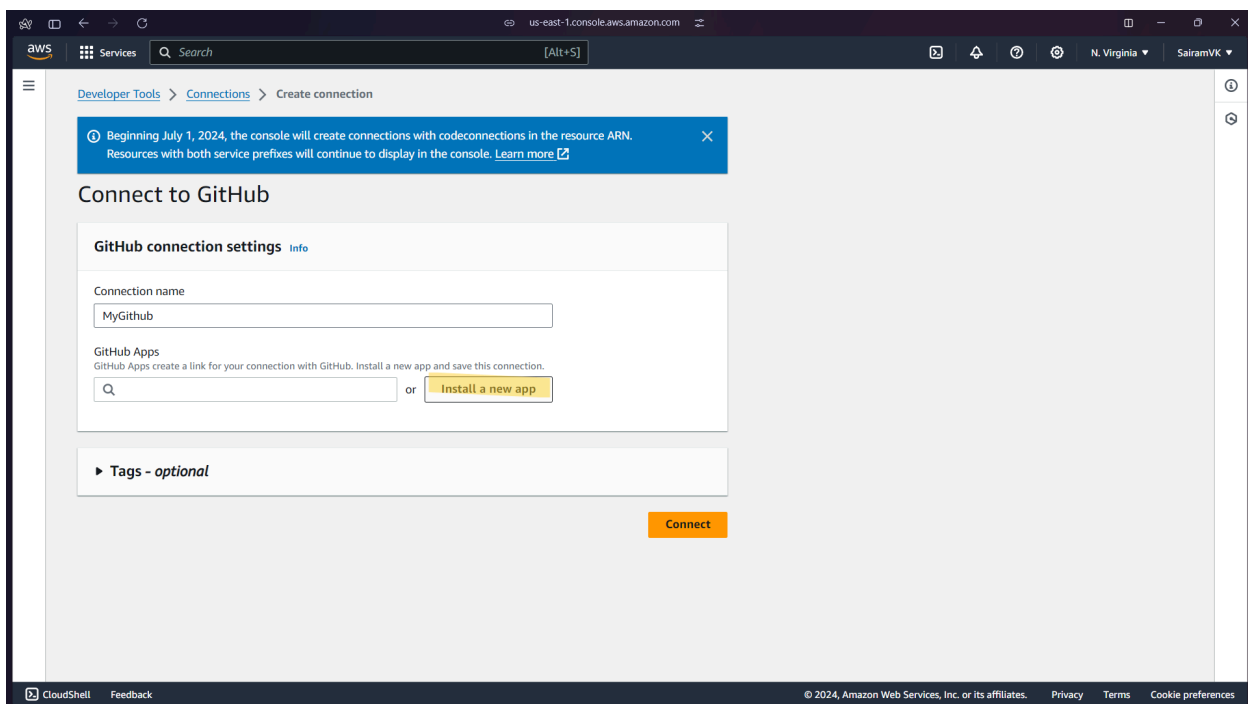
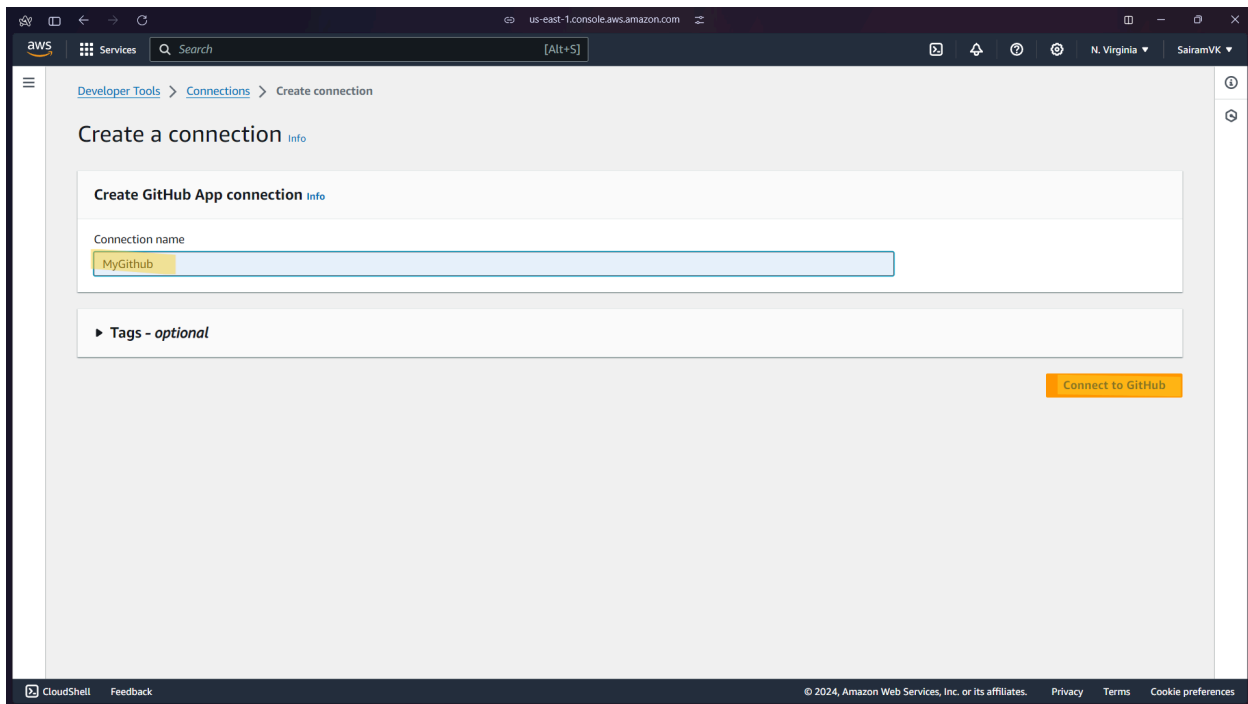
or Connect to GitHub

Repository name
Choose a repository in your GitHub account.

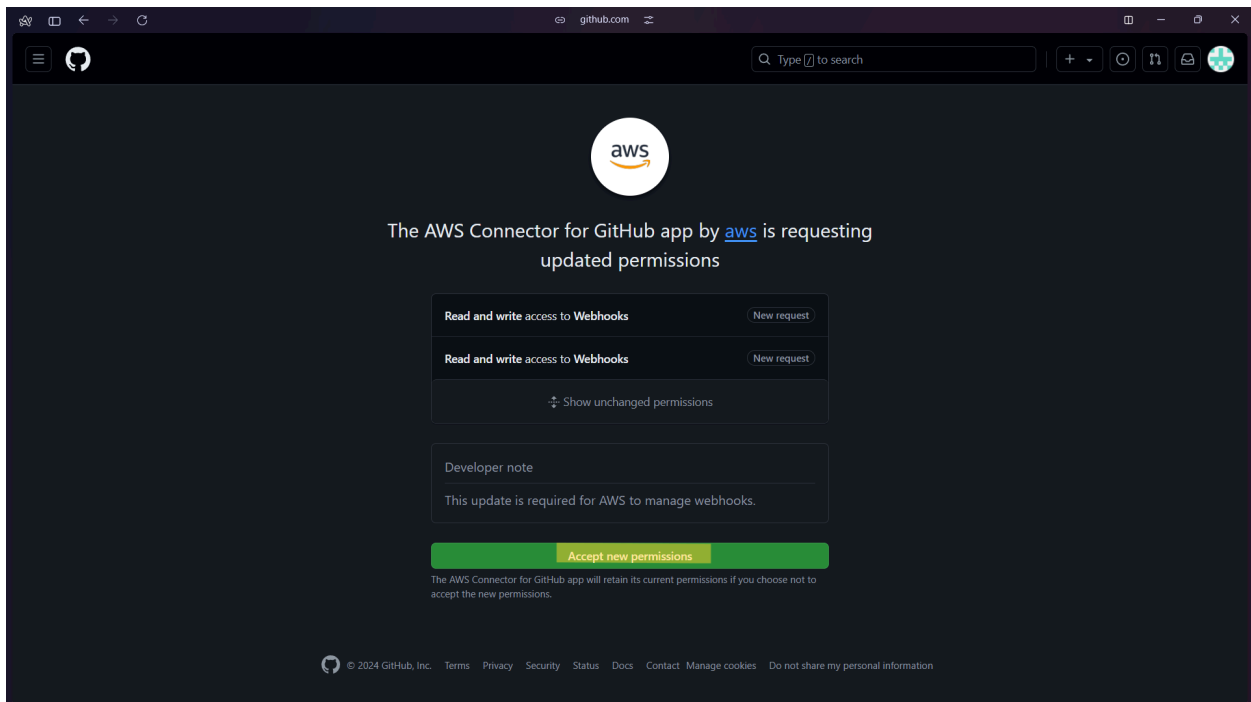
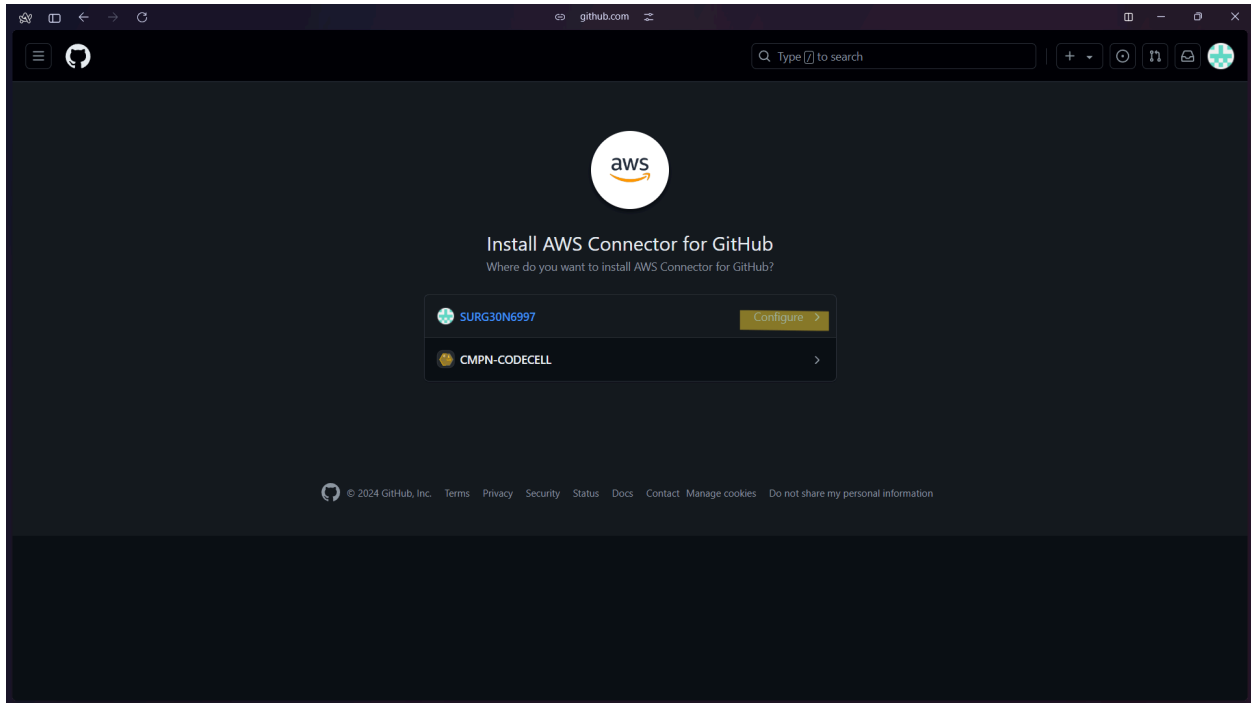
You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

Default branch
Default branch will be used only when pipeline execution starts from a different source or manually started.

Step 11: Give a name to your GitHub app Connection and click on Connect. This will give you a prompt to either to select a GitHub app or install a new app. If this is your first time, click on Install a new app.

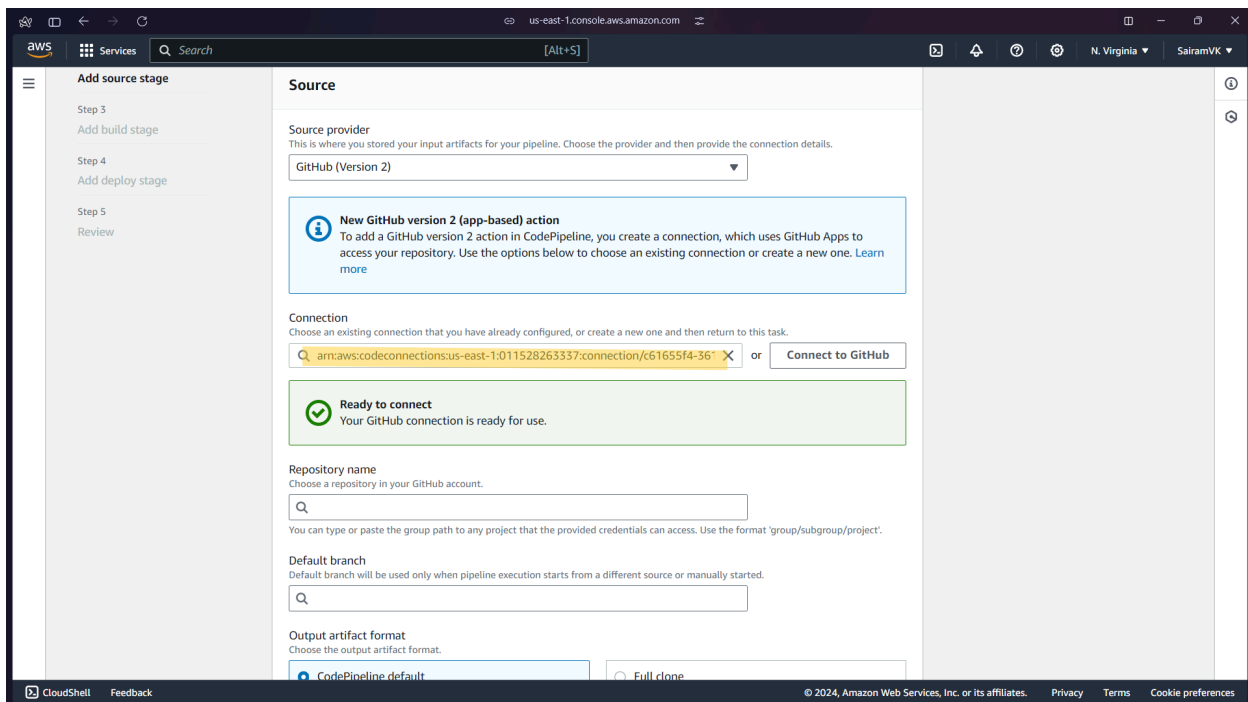
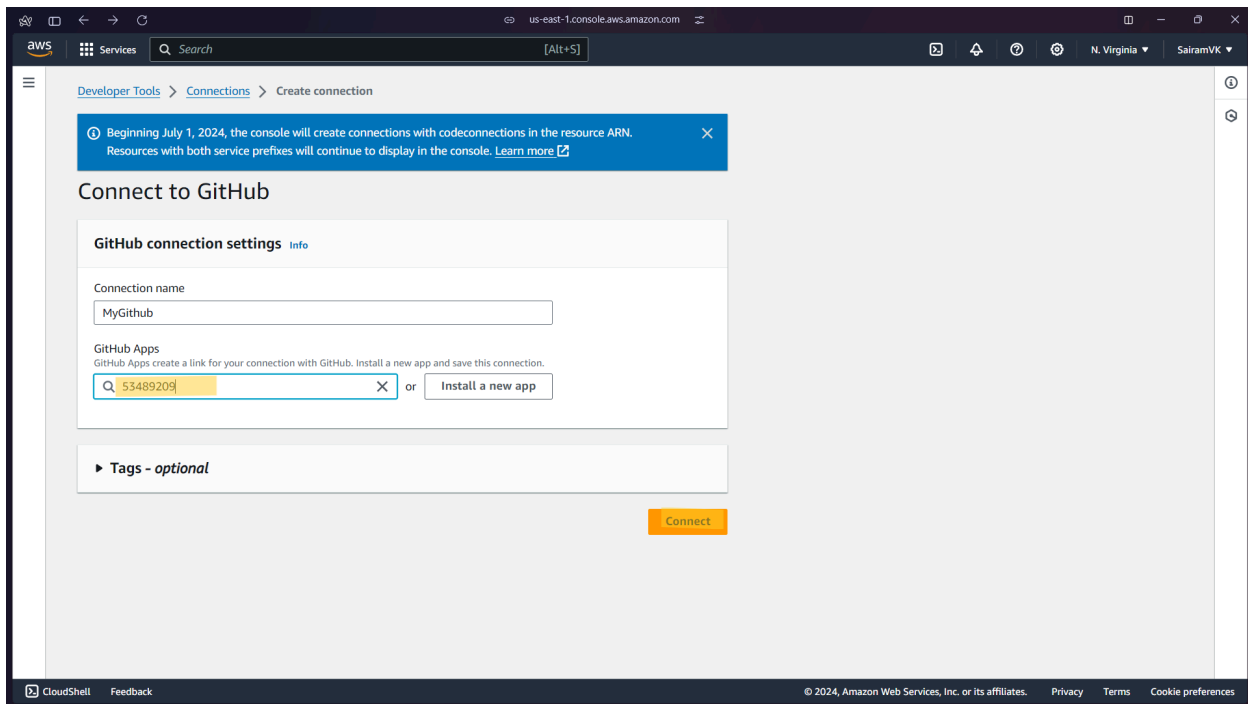


Step 12: This will direct you to install AWS Connector on your GitHub. Install it to your account and give it its permissions.



Step 13: After the app is set up, it gives the number in the text field. Click on Connect.

After clicking on connect, the link is shown in the connection field and AWS shows that GitHub connection is ready to use.



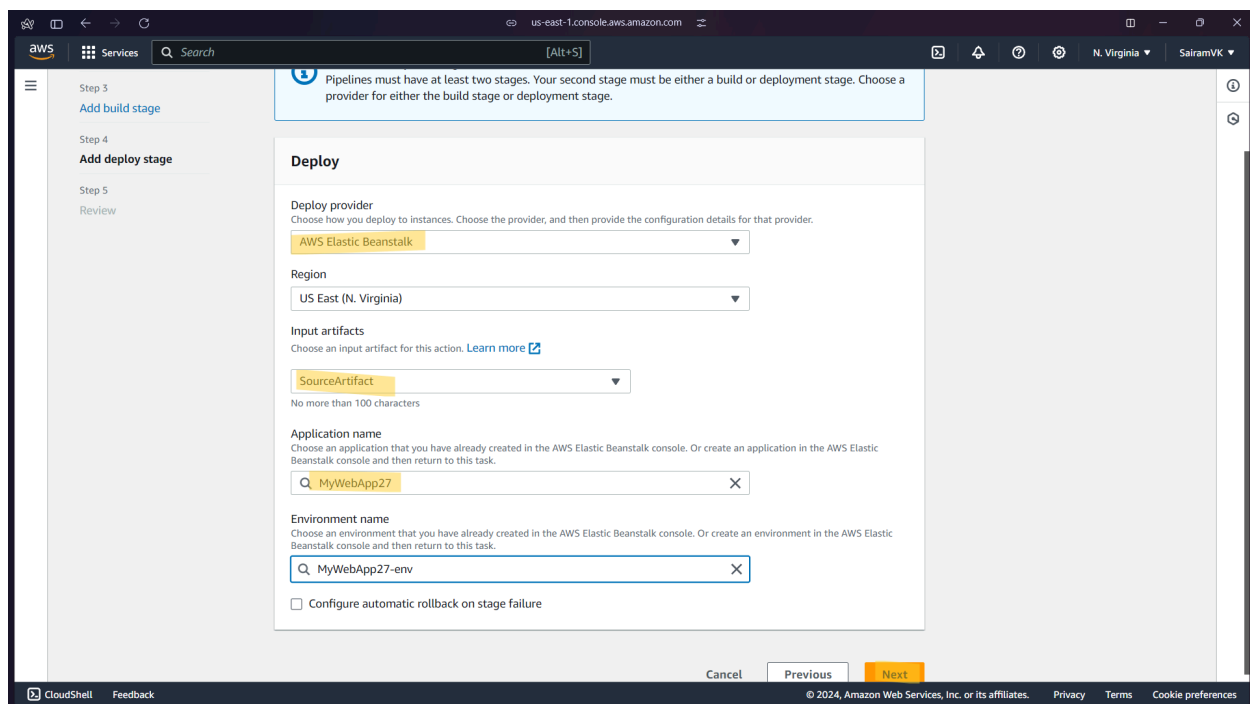
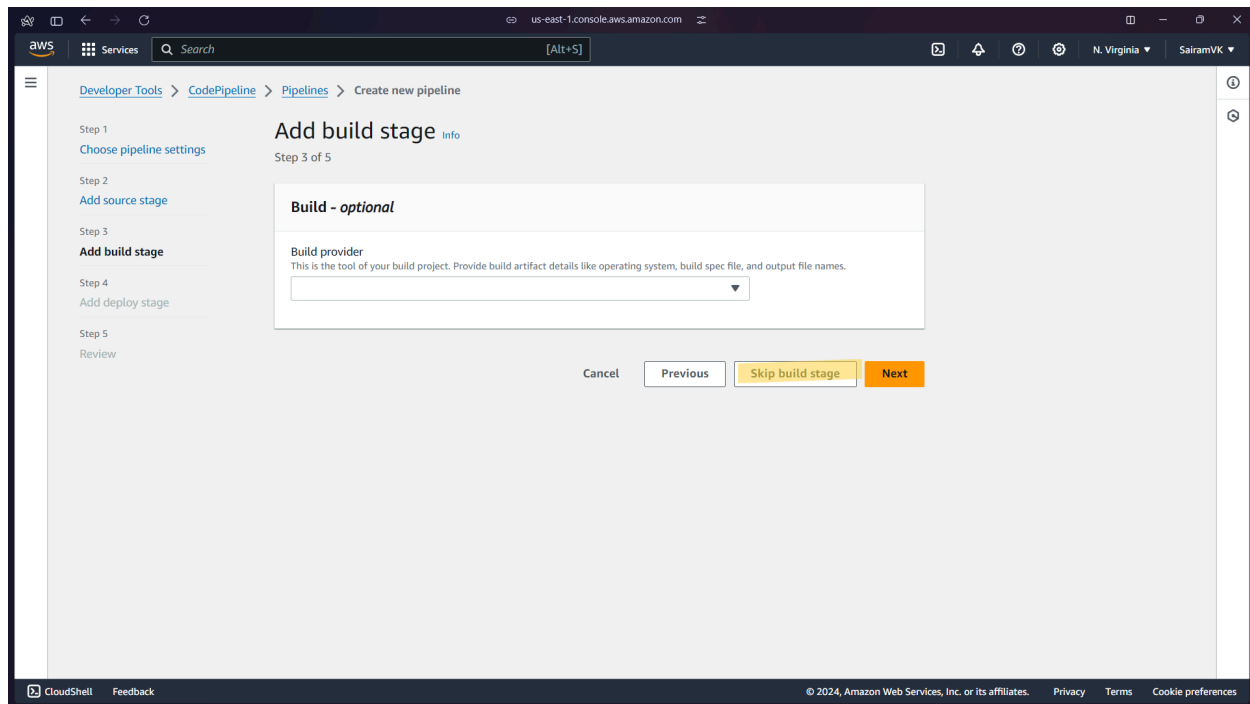
Step 14: Select the repository that you had forked to your GitHub. After that select the branch on which the files are present (default is Master).

The screenshot shows the AWS CodePipeline console in the 'Step 5: Review' stage. The configuration for a new GitHub version 2 (app-based) action is displayed. The 'Connection' section shows a selected connection 'arn:aws:codeconnections:us-east-1:011528263337:connection/c61655f4-36' and a 'Connect to GitHub' button. The 'Ready to connect' status is confirmed. The 'Repository name' is set to 'SURG30N6997/aws-codepipeline-s3-codedeploy-linux'. The 'Default branch' is set to 'master'. The 'Output artifact format' is set to 'CodePipeline default'. The 'Trigger' section is partially visible at the bottom.

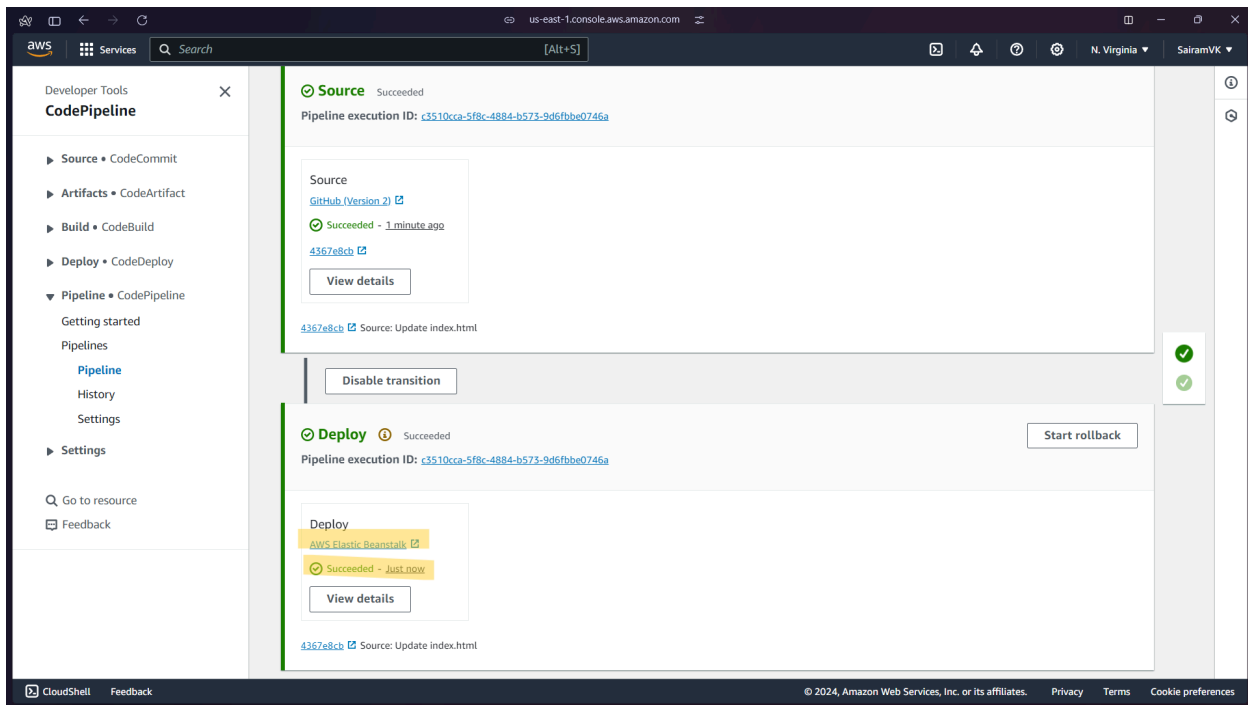
Step 15: Set the Trigger type as no filter. This would allow it to the website to update as soon as some change is made in the github.

The screenshot shows the AWS CodePipeline console in the 'Step 5: Review' stage, focusing on the 'Trigger' configuration. The 'Default branch' is set to 'master'. The 'Output artifact format' is set to 'CodePipeline default'. The 'Trigger type' is set to 'No filter', which starts the pipeline on any push and clones the HEAD. The 'Specify filter' and 'Do not detect changes' options are also visible. The 'Next' button is highlighted in orange.

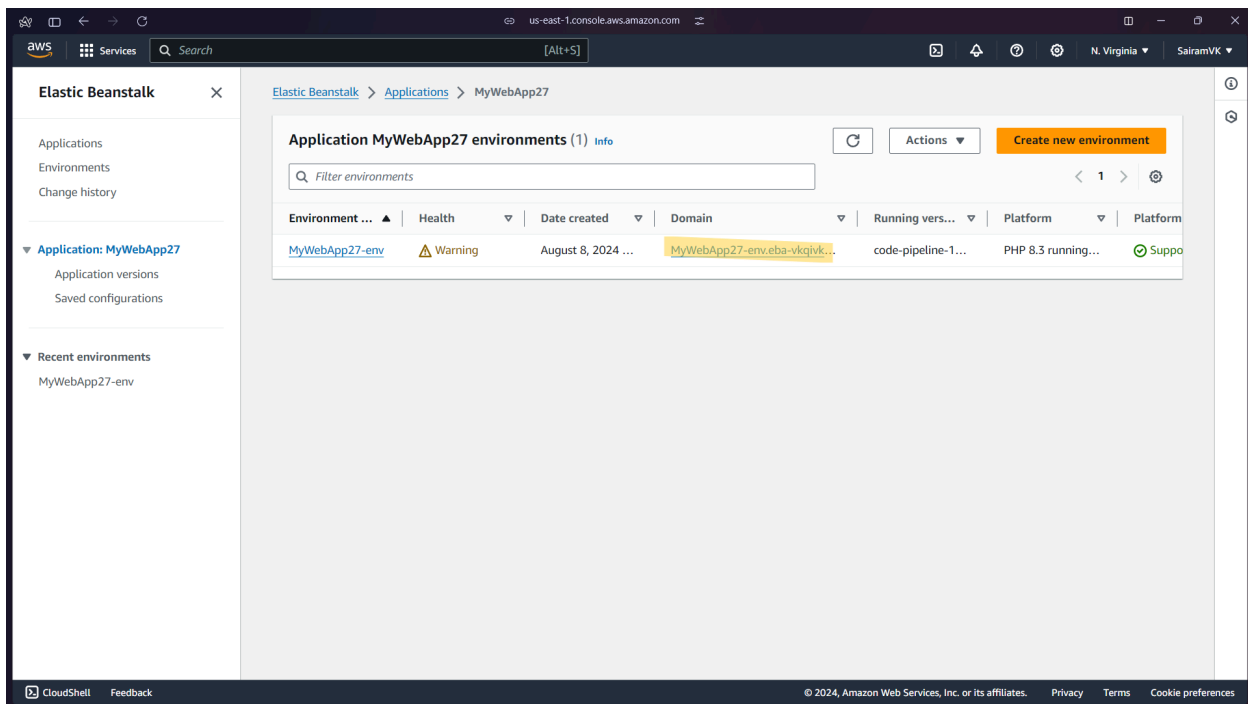
Step 16: Skip the build stage and go to Deploy. Select the deploy provider as AWS Elastic Beanstalk and Input Artifact as SourceArtifact. The application name would be the name of your Elastic Beanstalk. Then click on next.



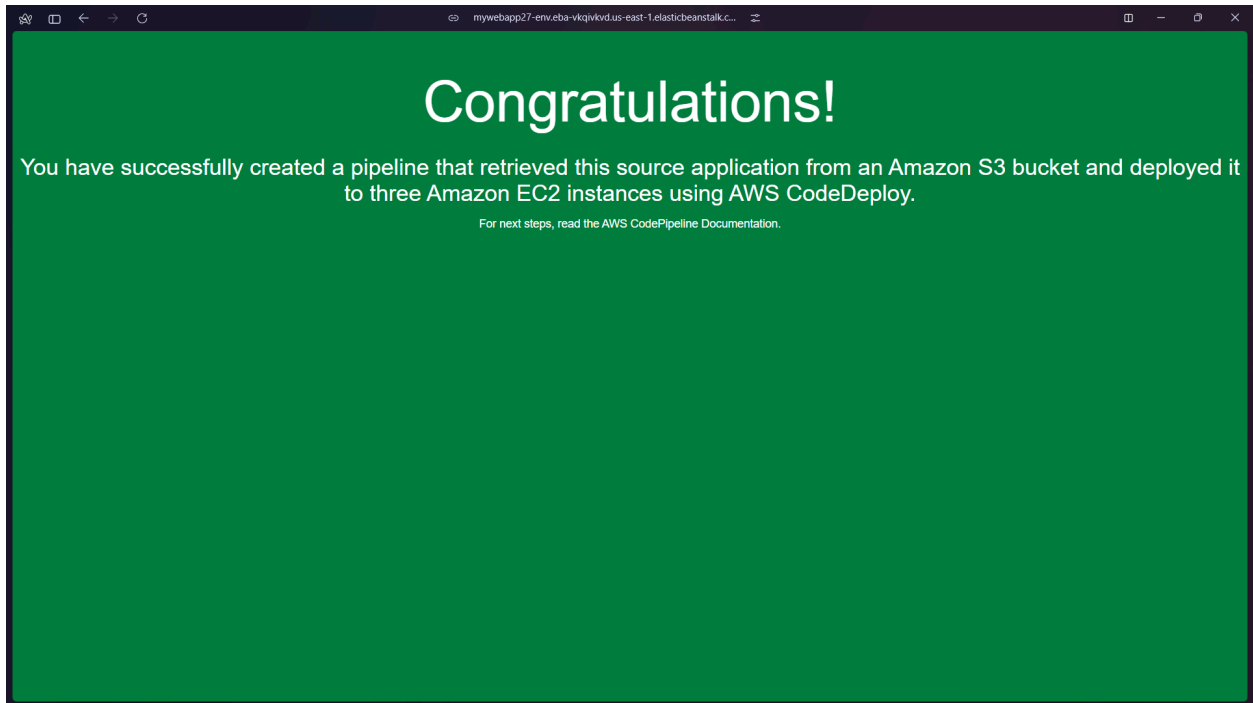
Step 19: Once the deployment is complete, click on the AWS Elastic Beanstalk under Deploy.



Step 20: This will redirect you to the application screen of Elastic Beanstalk. Click on the link shown under Domain.



Step 21: This will successfully show the sample website hosted.



Step 22: Now, we make some changes to the index.html file in the github.

For eg: If you make some changes to the <h2> tag.

Once the changes are committed, when the website is refreshed, the changes can be seen.

