

CodePipeline

AWS CodePipeline is a continuous integration and continuous delivery (CI/CD) service provided by Amazon Web Services (AWS). It is a visual workflow to orchestrate CICD.

It helps in automating the process of building, testing, and deploying code changes.

Source>build > test> deploy > load testing

Action Types	Action Providers
Source	codecommit, ecr,s3,bitbucket/github
Build	codebuild, jenkins
Test	codebuild, jenkins, device farm, BlazeMeter
Deploy	codedeploy, elastic beanstalk, cloudformation, ecs, OPsWorks, s3
Invoke	lambda, step functions, manual approval

In AWS CodePipeline, you can have a minimum of 2 stages and a maximum of up to 50 stages in a single pipeline.

Also, max 50 actions and min 1 action is required inside each stage.

Pipeline Configuration Options

AWS CodePipeline provides a flexible configuration setup for defining your pipeline's stages, actions, and integration points. These configuration options include

1. Specifying the pipeline name as a unique identifier
2. Defining a service role with appropriate permissions to access AWS services involved in the pipeline
3. Setting up an S3 bucket to serve as the artifact store, and optionally configuring encryption using either AWS Key Management Service (KMS) or other encryption methods.

Pipeline Workflow Overview:

Source Stage: In this scenario, development (dev) pushes code changes to AWS CodeCommit, a fully managed source control service by AWS. The code changes trigger the pipeline.

Artifact Handling: Once the code changes are detected, CodePipeline automatically packages the source code into artifacts and stores them in an S3 bucket designated

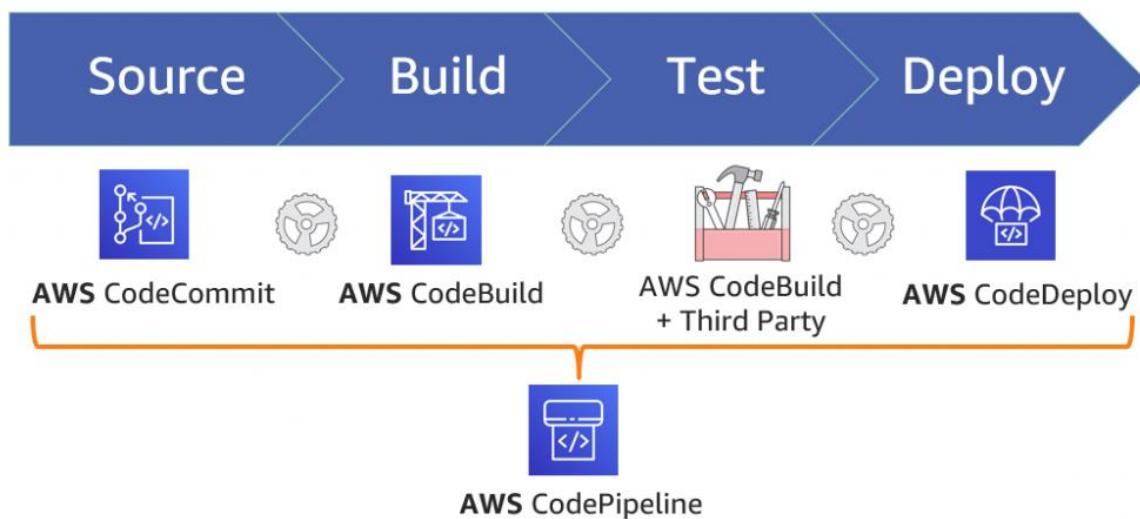
CodePipeline

as the artifact store. These artifacts represent the input for subsequent stages in the pipeline.

Build Stage: The pipeline then invokes AWS CodeBuild, a fully managed build service, to execute the build process. CodeBuild pulls the source artifacts from the S3 bucket, compiles the code, runs tests, and produces built output artifacts.

Artifact Storage: The built output artifacts generated by CodeBuild are stored in the same S3 bucket used for storing input artifacts. These artifacts serve as the input for the deployment stage.

Deployment Stage: The pipeline triggers AWS CodeDeploy, an automated deployment service, to deploy the built artifacts to the target environment (e.g., staging, production). CodeDeploy retrieves the built artifacts from the S3 bucket and orchestrates the deployment process to the specified deployment targets (e.g., EC2 instances, Lambda functions).



Why use CloudFormation for CodePipeline?

Infrastructure Tracking: CloudFormation allows you to define your CodePipeline infrastructure as code, which can be tracked and version-controlled alongside your application code. This ensures that changes to your pipeline's infrastructure are documented and reversible, providing visibility and accountability for all modifications.

Reproducibility Across Environments: With CloudFormation templates, you can easily replicate your CodePipeline and associated infrastructure in different AWS regions or accounts. This simplifies the process of setting up consistent pipelines across

CodePipeline

development, testing, staging, and production environments, promoting reliability and consistency in deployments.

Efficient Resource Management: CloudFormation enables you to quickly provision, update, or delete pipeline resources using automation. This streamlines the management of your pipeline infrastructure, making it easy to scale resources up or down as needed and reducing manual intervention in resource management tasks.

Quick Removal of Resources: When it's time to decommission a pipeline or its associated resources, CloudFormation allows for the swift and complete removal of those resources. This ensures that unused resources do not incur unnecessary costs and simplifies cleanup efforts, maintaining a tidy and efficient AWS environment.

Tutorial 1

This tutorial demonstrates the creation of a simple two-stage CodePipeline with source as CodeCommit and deploy as Elastic Beanstalk. It also includes setting up an Elastic Beanstalk environment, creating a CodeCommit repository, establishing a git connection, and adding manual approval to the pipeline.

There are 3 major steps in this tutorial.

1. **Elastic Beanstalk Setup:**

- Create an Elastic Beanstalk application with Node.js platform.
- Use a newly created IAM role with EC2 access for environment setup.

2. **CodeCommit Setup:**

- Create a CodeCommit repository and IAM user/group with access.
- Clone the repository locally and push changes.

3. **CodePipeline Creation:**

- Set up a CodePipeline with CodeCommit as source and Elastic Beanstalk as deploy stage.
- Add manual approval stage for deployment changes.
- Approve changes for deployment.

Create a Web Server Environment using Elastic Beanstalk

CodePipeline

First search for "Elastic Beanstalk" and then click on create application. Add a name for the application.

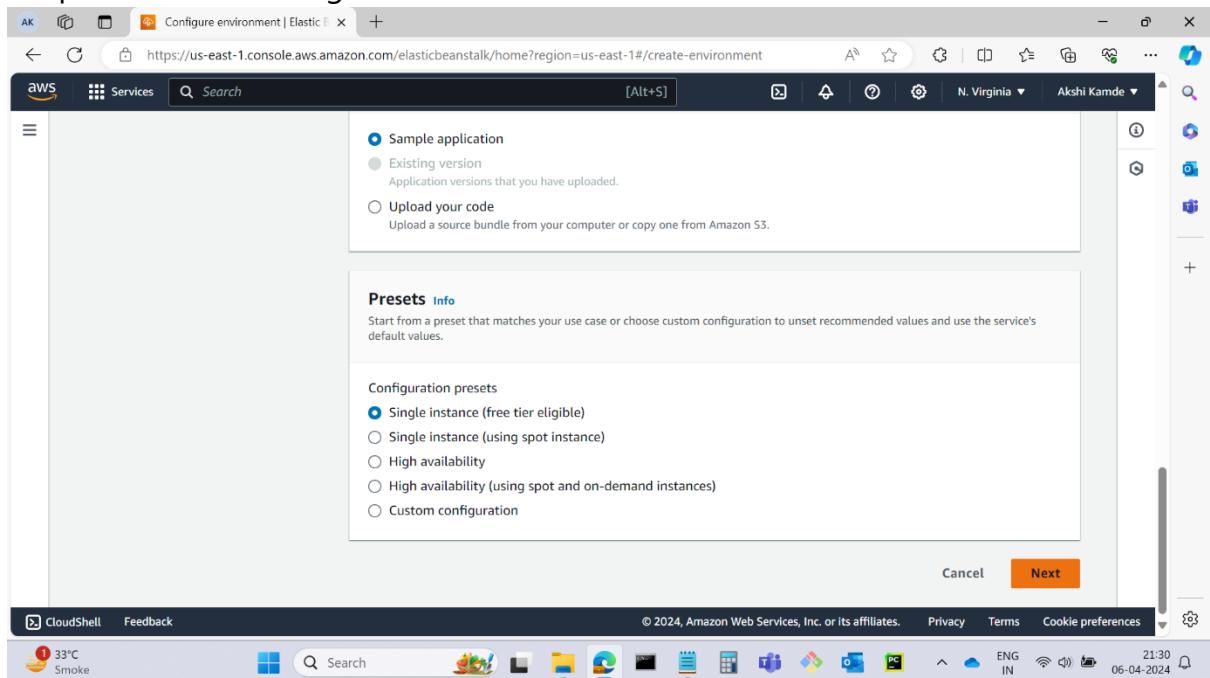
The screenshot shows the 'Configure environment' step in the AWS Elastic Beanstalk console. The 'Application information' section has 'Application name' set to 'my-nodejs-beanstalk'. The 'Environment information' section has 'Environment name' set to 'My-nodejs-beanstalk-env'. A 'Domain' field is present with the placeholder 'Leave blank for autogenerated value'. A 'Check availability' button is visible next to the domain field. The browser status bar at the bottom shows the URL as https://us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/create-environment.

Select the platform as Node.js

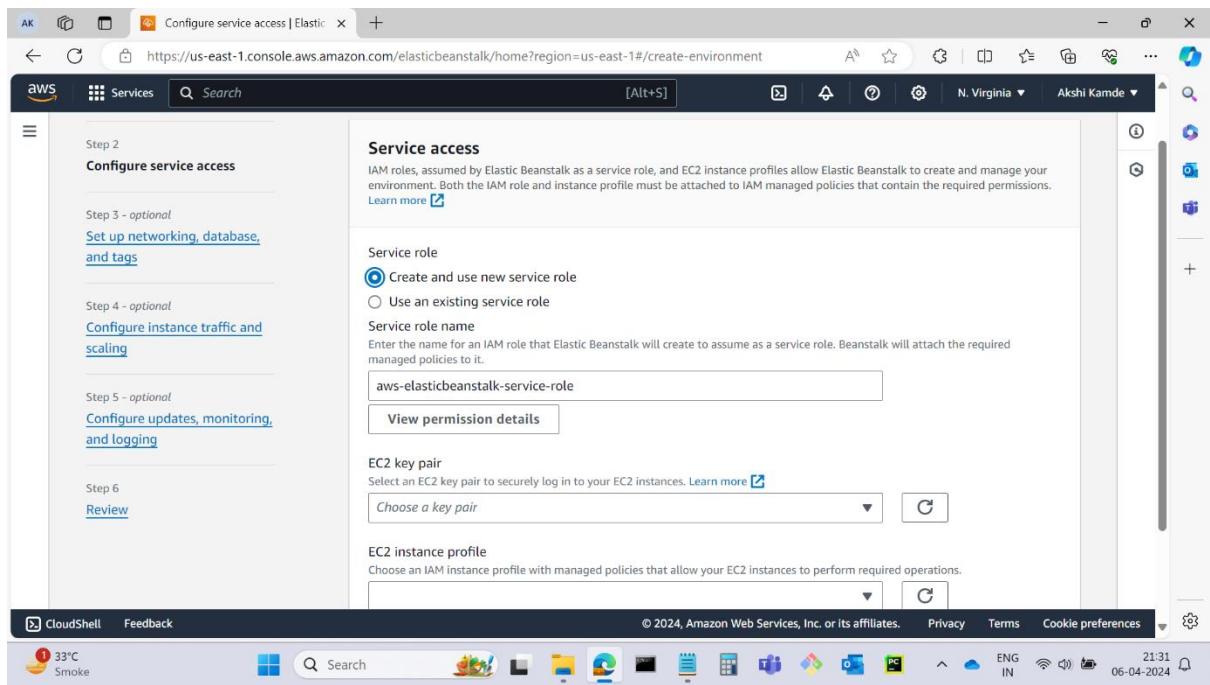
The screenshot shows the 'Platform' configuration step in the AWS Elastic Beanstalk console. Under 'Platform type', 'Managed platform' is selected. The 'Platform' dropdown is set to 'Node.js'. The 'Platform branch' dropdown is set to 'Node.js 20 running on 64bit Amazon Linux 2023'. The 'Platform version' dropdown is set to '6.1.2 (Recommended)'. The browser status bar at the bottom shows the URL as https://us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/create-environment.

CodePipeline

Keep the default settings and click on next.



Create and use a new service role.



Create a New Service Role and IAM Role with EC2 Permissions:

CodePipeline

Create an IAM role with EC2 access. For this search for "IAM" in search bar and move to roles from left side panel. Click on "Create role"

The screenshot shows the AWS IAM Roles page. The left sidebar is collapsed, showing options like Dashboard, Access management, Roles, Access reports, and External access. The main content area is titled 'Roles (3) Info' and contains a table with three rows:

Role name	Trusted entities
aws-codestar-service-role	AWS Service: codestar
AWSServiceRoleForSupport	AWS Service: support (Service...
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (S...

Below the table, there is a section titled 'Roles Anywhere' with a 'Manage' button.

CodePipeline

Keep AWS service as trusted entity type and the use case as EC2 and hit next.

The image consists of three vertically stacked screenshots from the AWS IAM console, showing the process of creating a new role:

- Screenshot 1: Select trusted entity**

This step shows the "Trusted entity type" options:
 - AWS service** (selected): Allows AWS services like EC2, Lambda, or others to perform actions in this account.
 - AWS account**: Allows entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
 - Web identity**: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
 - SAML 2.0 federation**: Allows users federated with SAML 2.0 from a corporate directory to perform actions in this account.
 - Custom trust policy**: Create a custom trust policy to enable others to perform actions in this account.
- Screenshot 2: Use case**

This step shows the "Service or use case" dropdown set to "EC2". It lists the "Use case" options for EC2:
 - EC2** (selected): Allows EC2 instances to call AWS services on your behalf.
 - EC2 Role for AWS Systems Manager**: Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
 - EC2 Spot Fleet Role**: Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
 - EC2 - Spot Fleet Auto Scaling**: Allows Auto Scaling to access and update EC2 spot fleets on your behalf.
 - EC2 - Spot Fleet Tagging**: Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.
 - EC2 - Spot Instances**: Allows EC2 Spot Instances to launch and manage spot instances on your behalf.
- Screenshot 3: Next Step**

This screenshot shows the "Next Step" button at the bottom of the wizard, indicating the user has completed the previous steps.

CodePipeline

Search for EC2FullAccess and click next.

The screenshot shows the AWS IAM 'Create role' wizard. The current step is 'Configure service access'. Under the 'AWS Lambda' section, the 'EC2' checkbox is checked. Other roles listed include:

Role Name	Type	Description
AmazonEC2ContainerRegistryPowerUser	AWS managed	Provides full access to the Amazon EC2 Container Registry
AmazonEC2ContainerRegistryReadOnly	AWS managed	Provides read-only access to the Amazon EC2 Container Registry
AmazonEC2ContainerServiceAutoscaleRole	AWS managed	Policy to enable the Amazon EC2 Container Service to automatically scale your application
AmazonEC2ContainerEventsRole	AWS managed	Policy to enable the Amazon EC2 Container Service to receive events from AWS Lambda
AmazonEC2ContainerServiceforEC2Role	AWS managed	Default policy for the Amazon EC2 Container Service to interact with the Amazon EC2 service
AmazonEC2ContainerServiceRole	AWS managed	Default policy for the Amazon EC2 Container Service to interact with other services
AmazonEC2FullAccess	AWS managed	Provides full access to the Amazon EC2 service
AmazonEC2ReadOnlyAccess	AWS managed	Provides read-only access to the Amazon EC2 service
AmazonEC2RoleforAWSCodeDeploy	AWS managed	Provides EC2 instances to AWS CodeDeploy
AmazonEC2RoleforAWSCodeDeployLimited	AWS managed	Provides EC2 instances to AWS CodeDeploy with limited permissions
AmazonEC2RoleforDataPipelineRole	AWS managed	Default policy for the Amazon EC2 Data Pipeline to interact with the Amazon EC2 service
AmazonEC2RoleforSSM	AWS managed	This policy is used by the AWS Systems Manager (SSM) agent to interact with the Amazon EC2 service
AmazonEC2RolePolicyForLaunchWizard	AWS managed	Managed policy for the Launch Wizard to interact with the Amazon EC2 service

CodePipeline

Type in the name of the role as given below click on create role.

The screenshots show the AWS IAM 'Create role' wizard:

- Step 1: Select trusted entity**: Shows the 'Role details' section where 'Role name' is set to 'ec2InstanceRole'. The 'Description' field contains 'Allows EC2 instances to call AWS services on your behalf.'
- Step 2: Add permissions**: Shows the 'Permissions policy summary' table with one item: 'AmazonEC2FullAccess' (AWS managed, Attached as Permissions policy).
- Step 3: Add tags**: Shows the 'Add tags - optional' section with a note about key-value pairs for identifying resources. It shows 'No tags associated with the resource.' and a 'Add new tag' button.

Now you can use this IAM role that was created for creating the elastic beanstalk environment. Click on refresh icon beside EC2 instance profile and select the newly created role from dropdown. Skip the VPC section for now and hit next. A summary will be shown and now you select submit.

CodePipeline

The image consists of three vertically stacked screenshots of the AWS Elastic Beanstalk console, showing the configuration of a new environment.

Screenshot 1: Step 3 - optional
This screenshot shows the "Set up networking, database, and tags" step. It includes fields for a Service role (set to "Create and use new service role") and a Service role name ("aws-elasticbeanstalk-service-role"). There is also a "View permission details" button. Below this, there are sections for "EC2 key pair" and "EC2 instance profile".

Screenshot 2: Step 6
This screenshot shows the "Review" step. It displays the configuration summary, including the service role and instance profile selected in the previous steps.

Screenshot 3: Configure environment - review
This screenshot shows the "Configure environment - review" step. It displays the platform software configuration, which includes:

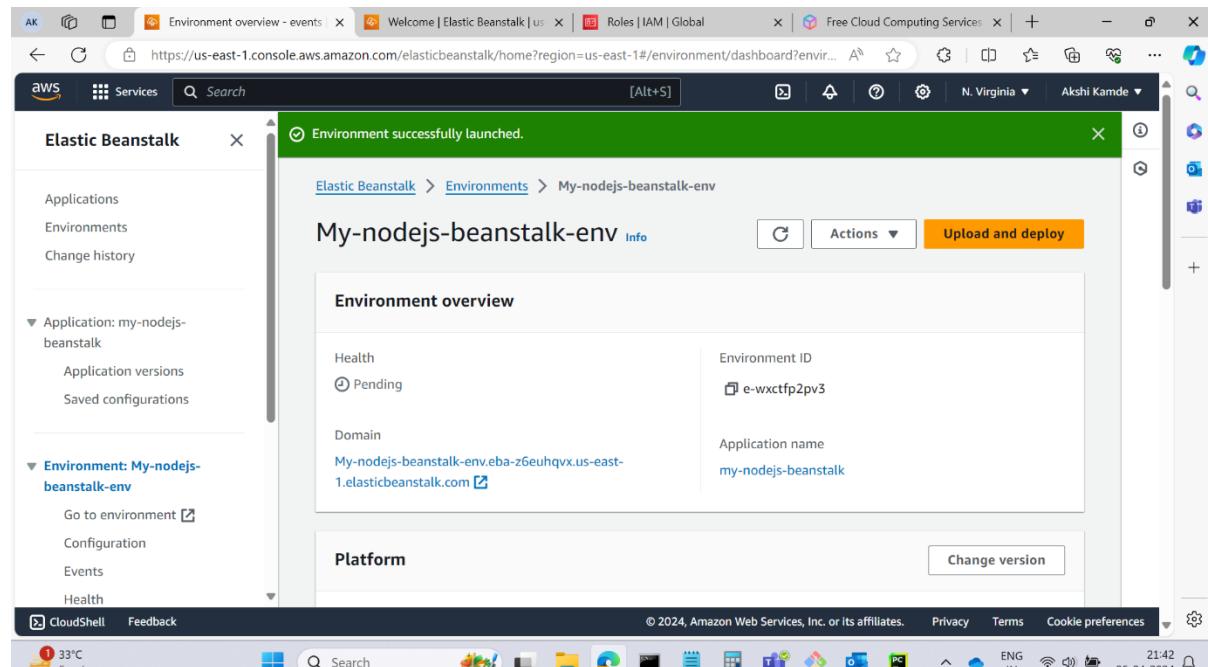
Key	Value
Lifecycle	false
Log streaming	Deactivated
Proxy server	nginx
Logs retention	7
Rotate logs	Deactivated
Update level	minor
X-Ray enabled	Deactivated

Below this, it shows the "Environment properties" section, which states "No environment properties" and "There are no environment properties defined".

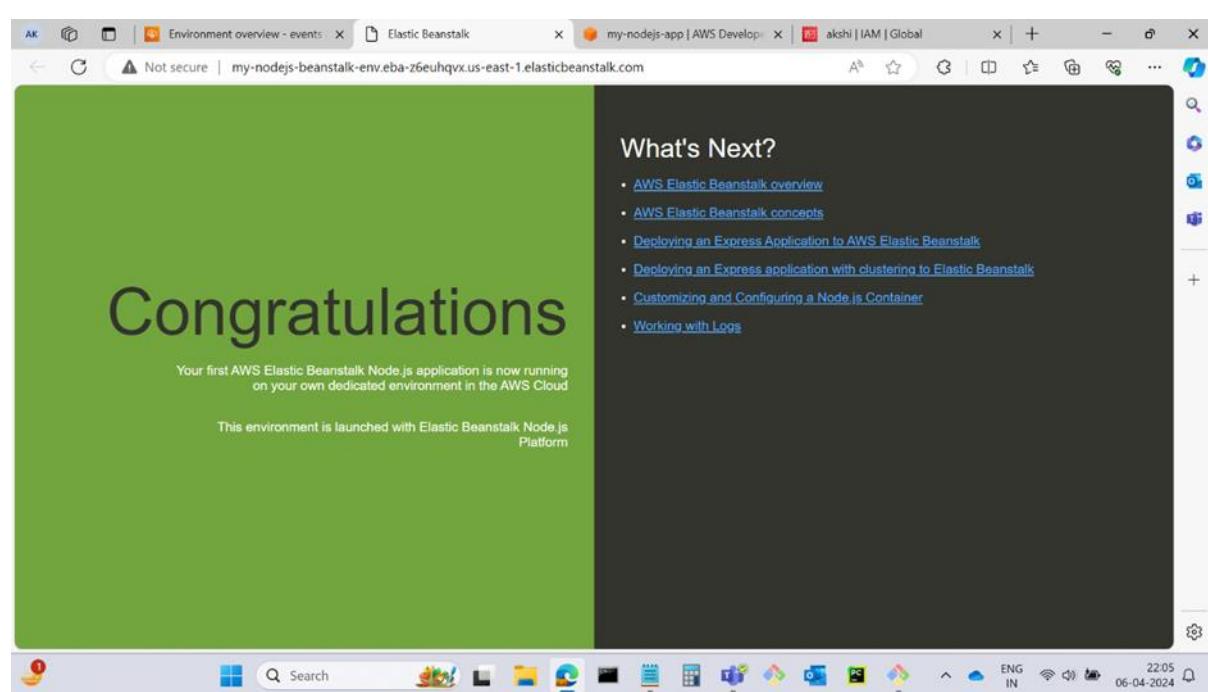
After waiting for about 5-10 minutes, your beanstalk environment will be ready with a Domain address link. Click on this link. You will be able to see the below default

CodePipeline

screen for Elastic Beanstalk.



The screenshot shows the AWS Elastic Beanstalk environment overview for 'My-nodejs-beanstalk-env'. A green banner at the top says 'Environment successfully launched.' The environment ID is e-wxctfp2pv3. The application name is 'my-nodejs-beanstalk'. The platform section shows 'Change version'.



The screenshot shows the 'What's Next?' section of the AWS Elastic Beanstalk environment launch confirmation page. It lists several links for further reading, such as 'AWS Elastic Beanstalk overview', 'AWS Elastic Beanstalk concepts', and 'Deploying an Express Application to AWS Elastic Beanstalk'.

Create a CodeCommit repository and establish a git connection on local

For this, search for CodeCommit in search bar by opening a new tab. Click on create a repository. Type the name of the repository. For example, my-nodejs-app. Give a description if required and then click on create. Your empty repository will be created.

CodePipeline

The screenshot shows the AWS CodeCommit service interface. On the left, a navigation sidebar lists various services under 'Developer Tools' and 'CodeCommit'. The main area displays a table titled 'Repositories' with one row, 'No results', indicating there are no repositories to display. A prominent orange button labeled 'Create repository' is centered above the table. Below the table, a message states 'There are no results to display.'

In the second part of the screenshot, the user has navigated to the 'Create repository' page. The title bar reads 'Create repository | AWS Developer Tools'. The page contains a form for 'Repository settings'. The 'Repository name' field is filled with 'my-nodejs-app'. The 'Description - optional' field is empty. There is also a 'Tags' section which is currently empty.

To add files from local, we need to establish a connection between CodeCommit and our local machine. But it is not recommended for root users to establish git connections so for this we shall create an IAM group and then add a new IAM user to this group with necessary permissions.

CodePipeline

First click on Users from left side panel and fill the details.

The image consists of three vertically stacked screenshots of the AWS IAM 'Create user' wizard, showing the process of creating a new IAM user named 'akshi'.

Step 1: Specify user details

- User name: akshi
- Description: The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen).
- Checkboxes:
 - Provide user access to the AWS Management Console - *optional*
 - If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.
- Information box: If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keypairs, you can generate them after you create this IAM user. [Learn more](#)

Step 2: Set permissions

Step 3: Review and create

Permissions summary

Name	Type	Used as
No resources		

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Buttons: Cancel, Previous, Create user

CodePipeline

Hit create user and the user will be successfully created.

The screenshot shows the AWS IAM Users page. A green success message at the top states "User created successfully" and "You can view and download the user's password and email instructions for signing in to the AWS Management Console." Below this, the "Users (1) Info" section is displayed, showing a table with one row for "akshi". The table columns are "User name", "Path", "Group", "Last activity", and "MFA". The "User name" column shows "akshi". The "Path" column shows "/". The "Group" column shows "0". The "Last activity" and "MFA" columns show "-". There are "Create user" and "Delete" buttons at the top right of the table.

Now choose User groups from left side panel and click on Create group.

The screenshot shows the AWS IAM User groups page. A green success message at the top states "User groups (0) Info" and "A user group is a collection of IAM users. Use groups to specify permissions for a collection of users." Below this, the "User groups (0) Info" section is displayed, showing a table with no resources. The table columns are "Group name", "Users", "Permissions", and "Creation time". The "Group name" column is empty. The "Users", "Permissions", and "Creation time" columns also show "-". There is a "Create group" button at the top right of the table.

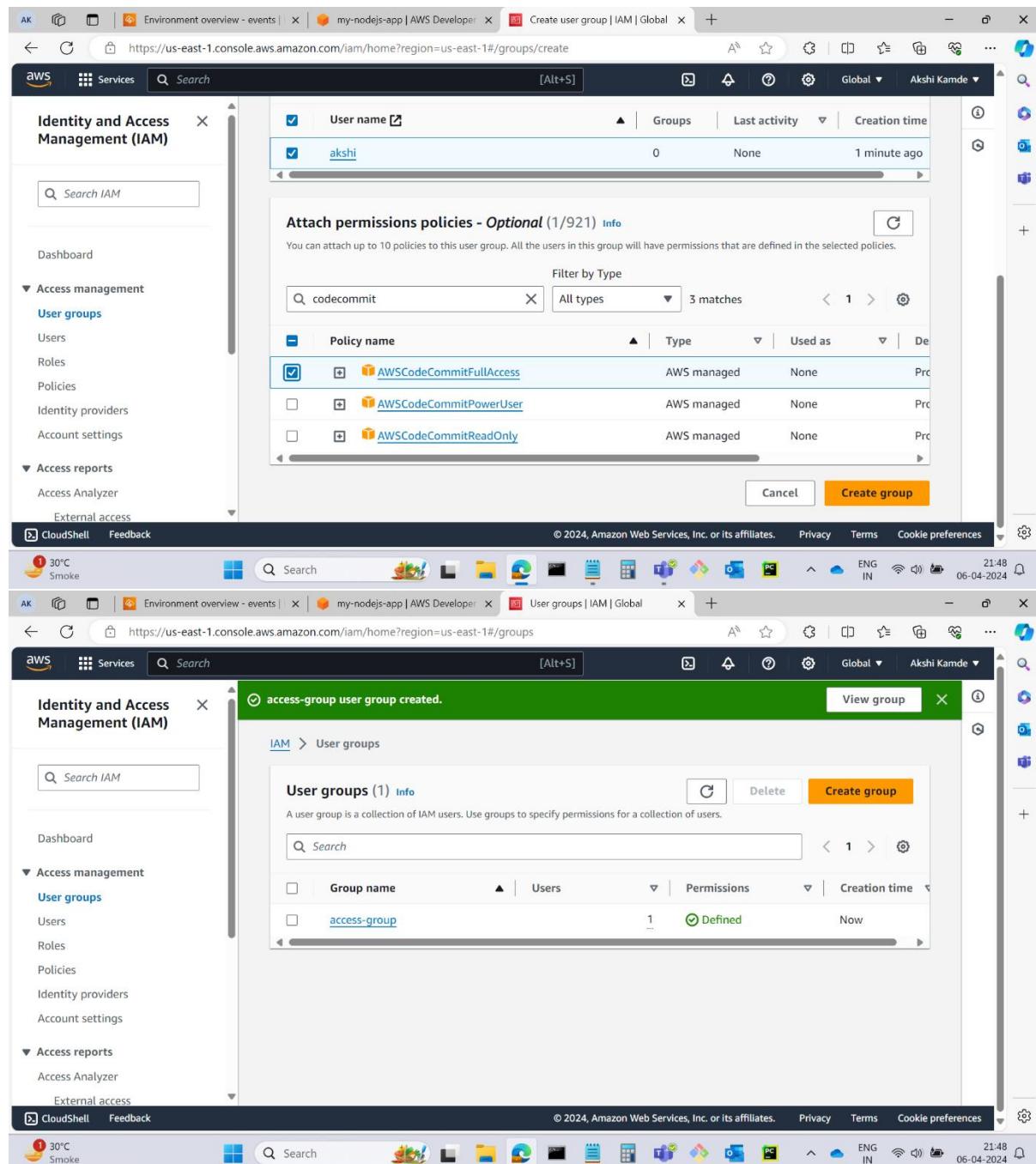
CodePipeline

Give a name and add the previously created user to this group.

The screenshots show the 'Create user group' wizard in the AWS IAM console. In the first screenshot, the 'User group name' field contains 'access-group'. In the second screenshot, the 'User name' dropdown shows 'akshi' selected. In the third screenshot, the 'User name' dropdown shows 'akshi' selected again, indicating it has been added to the group.

Search for CodeCommit and choose AWSCodeCommitFullAccess as the attached policy for this group. You can also give Full Administration access. Click on create group.

CodePipeline



The screenshot shows two side-by-side browser windows displaying the AWS IAM User Groups creation process.

Top Window: A modal dialog titled "Attach permissions policies - Optional (1/921)" is open over the IAM User Groups list. It shows a search bar for "codecommit" and a table of policies. One policy, "AWSCodeCommitFullAccess", is selected and highlighted in blue. Other policies listed are "AWSCodeCommitPowerUser" and "AWSCodeCommitReadOnly".

Policy name	Type	Used as	Description
AWSCodeCommitFullAccess	AWS managed	None	Full access to AWS CodeCommit
AWSCodeCommitPowerUser	AWS managed	None	Power user access to AWS CodeCommit
AWSCodeCommitReadOnly	AWS managed	None	Read-only access to AWS CodeCommit

Bottom Window: The main IAM User Groups page shows a success message: "access-group user group created." The "User groups (1)" section lists the newly created "access-group".

Group name	Users	Permissions	Creation time
access-group	1	Defined	Now

CodePipeline

Now go to Users and select the created user. Select "Security Credentials"

The screenshot shows the AWS IAM User details page for a user named 'akshi'. The 'Summary' section displays the ARN (arn:aws:iam::471112919630:user/akshi), console access status (Disabled), and creation date (April 06, 2024, 21:46 (UTC+05:30)). Below the summary, there are tabs for 'Permissions', 'Groups (1)', 'Tags', 'Security credentials' (which is selected), and 'Access Advisor'. Under the 'Security credentials' tab, it shows 'Permissions policies (1)' attached to the user. At the bottom of the page, there are links for 'CloudShell', 'Feedback', and system status indicators.

Scroll down and select HTTPS git credential. Click on Generate credentials. Download the credentials.

The screenshot shows the 'Security credentials' section for the 'akshi' user. It includes a table for SSH Key ID, a section for HTTPS Git credentials for AWS CodeCommit (0), and a section for Credentials for Amazon Keyspaces (for Apache Cassandra) (0). The 'HTTPS Git credentials for AWS CodeCommit' section has a 'Generate credentials' button. The bottom of the page shows standard navigation links like CloudShell, Feedback, and system status.

CodePipeline

Now switch to the tab where you had create CodeCommit repository and click on Clone URL. Choose Clone HTTPS. URL will get copied.

The screenshot shows the AWS CodeCommit interface. A green success banner at the top states "Success" and "Repository successfully created". Below it, the repository name "my-nodejs-app" is displayed. On the right, a dropdown menu shows "Clone URL" selected. Under "Connection steps", the "HTTPS" tab is active. A warning message in a yellow box states: "⚠ You are signed in using a root account. You cannot configure SSH connections for a root account, and HTTPS connections for a root account are not recommended. Consider signing in as an IAM user and then setting up your connection." At the bottom, there's a section titled "Step 1: Prerequisites". The browser status bar at the bottom indicates the URL is https://us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/my-nodejs-app/setup?region=us-east-1.

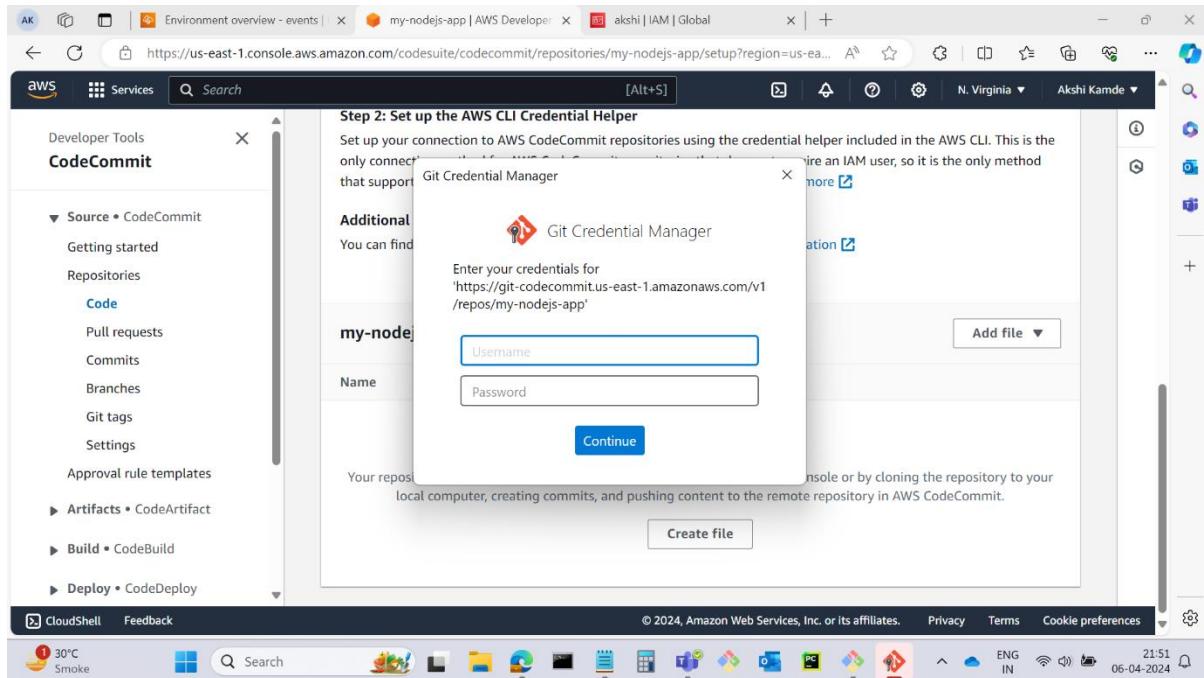
Now open git bash in your local machine and type the following command:

git clone <paste the copied URL>

The screenshot shows the AWS CodeCommit interface again. A modal window titled "Step 2: Set up the AWS CLI Credential Helper" is open. It instructs users to set up their connection to AWS CodeCommit repositories using the credential helper included in the AWS CLI. A terminal window within the modal shows the command "git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/my-nodejs-app" being run. The terminal output indicates the repository is being cloned to the local machine. The browser status bar at the bottom indicates the URL is https://us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/my-nodejs-app/setup?region=us-east-1.

CodePipeline

You will be prompted to add the username and password which you got from AWS IAM user. Fill the details and hit continue.



Now cd into the directory/repository name. You will see that you are on the master/main branch. You can either manually copy paste the files or simple use cp command. Then check it using git status

CodePipeline

The screenshot shows a Windows desktop environment with two browser windows open in a Microsoft Edge browser. Both windows display the same AWS CloudShell interface for setting up the AWS CLI Credential Helper.

Top Window (Screenshot 1):

```
MINGW64:/c/Users/akamde/my-nodejs-app
$ cd my-nodejs-app
$ ls
$ git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    app.js
    cron.yaml
    index.html
    package.json
nothing added to commit but untracked files present (use "git add" to track)
$
```

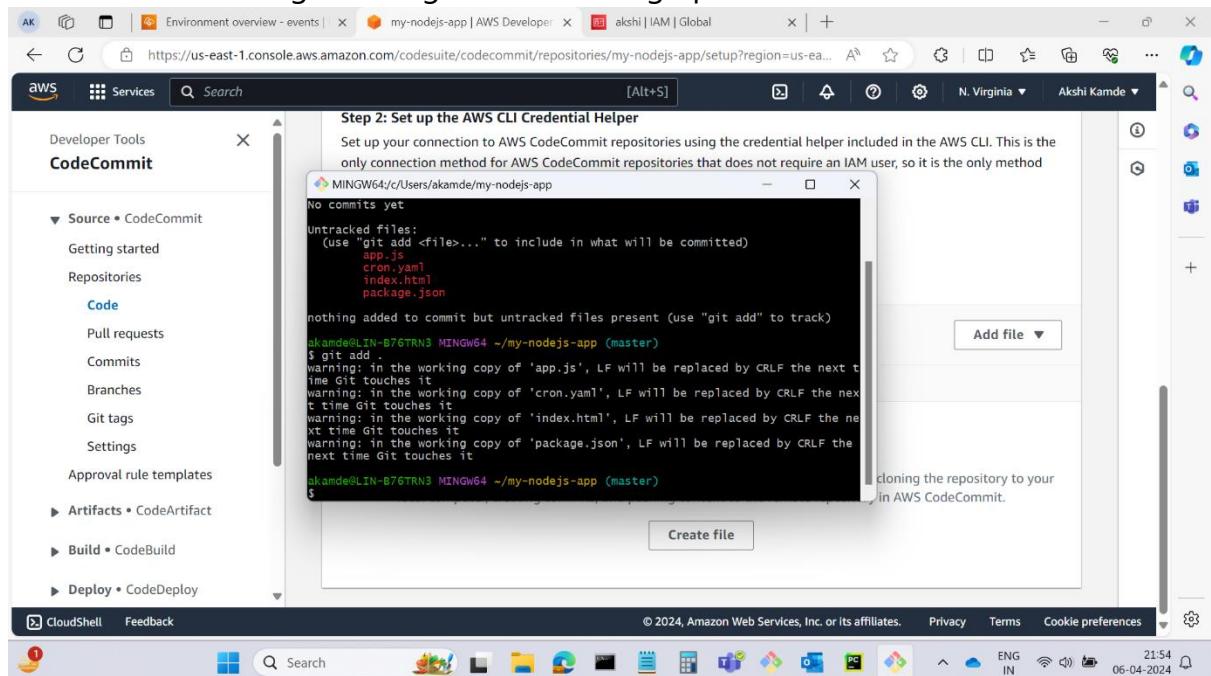
Bottom Window (Screenshot 2):

```
MINGW64:/c/Users/akamde/my-nodejs-app
$ cd my-nodejs-app
$ git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    app.js
    cron.yaml
    index.html
    package.json
nothing added to commit but untracked files present (use "git add" to track)
$
```

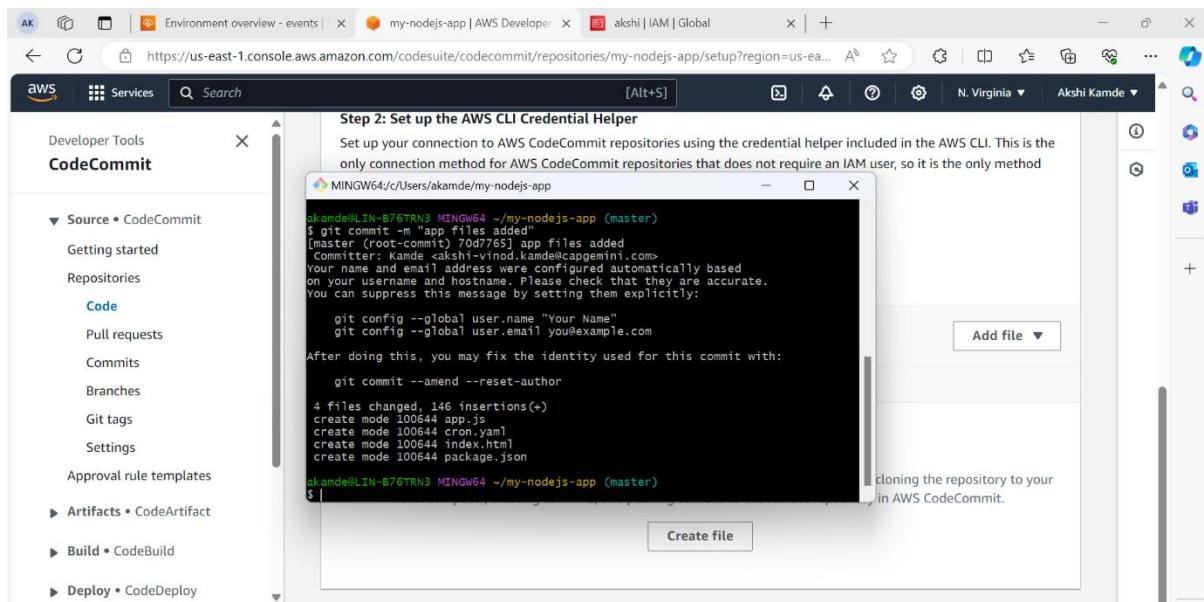
In both windows, the AWS CloudShell interface is visible, showing terminal commands being run. The terminal output includes instructions for cloning the repository to AWS CodeCommit. The AWS navigation bar at the top of each window shows the URL as <https://us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/my-nodejs-app/setup?region=us-east-1>.

CodePipeline

Use commands like git add ., git commit and git push.



CodePipeline



The screenshot shows the AWS CodeCommit setup process in a CloudShell window. The title bar says "Step 2: Set up the AWS CLI Credential Helper". The main content area displays a terminal session with the following text:

```
okamde@LIN-B76TRN3 MINGW64 ~/my-nodejs-app (master)
$ git commit -m "app files added"
[master (root-commit) 70d7765] app files added
Committer: Kamde <akshi-vinod.kamde@gemini.com>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.

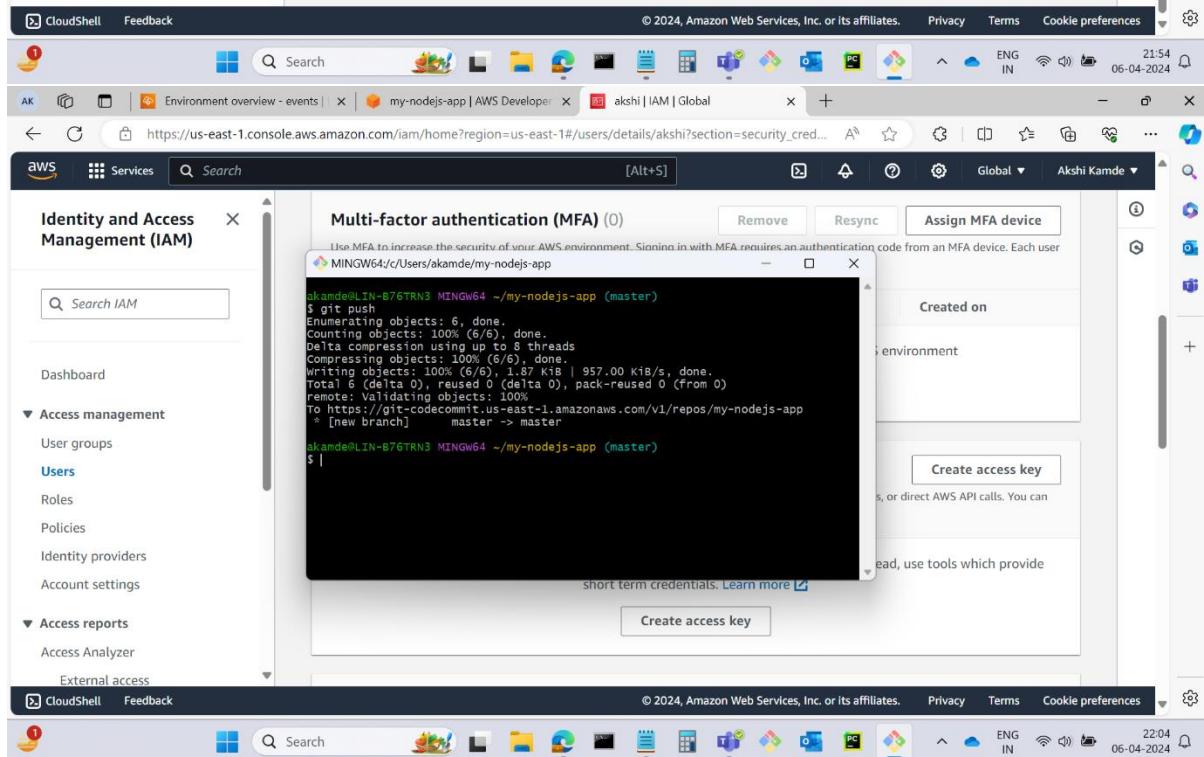
You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"
git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:
git commit --amend --reset-author

4 files changed, 146 insertions(+)
create mode 100644 app.js
create mode 100644 cron.yaml
create mode 100644 index.html
create mode 100644 package.json
okamde@LIN-B76TRN3 MINGW64 ~/my-nodejs-app (master)
```

A tooltip on the right side of the terminal window reads: "cloning the repository to your AWS CodeCommit".



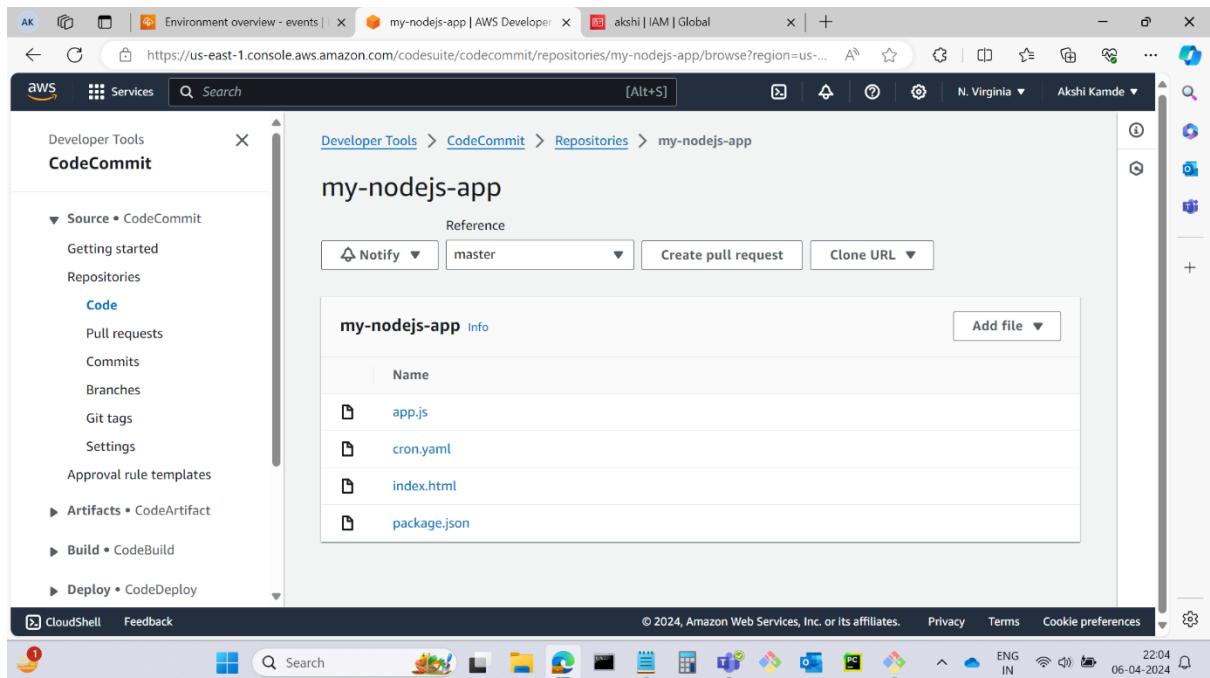
The screenshot shows the IAM Multi-factor authentication (MFA) setup process in a CloudShell window. The title bar says "Multi-factor authentication (MFA) (0)". The main content area displays a terminal session with the following text:

```
okamde@LIN-B76TRN3 MINGW64 ~/my-nodejs-app (master)
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.87 KiB | 957.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/my-nodejs-app
 * [new branch] master -> master
okamde@LIN-B76TRN3 MINGW64 ~/my-nodejs-app (master)
$ |
```

A tooltip on the right side of the terminal window reads: "Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user environment".

CodePipeline

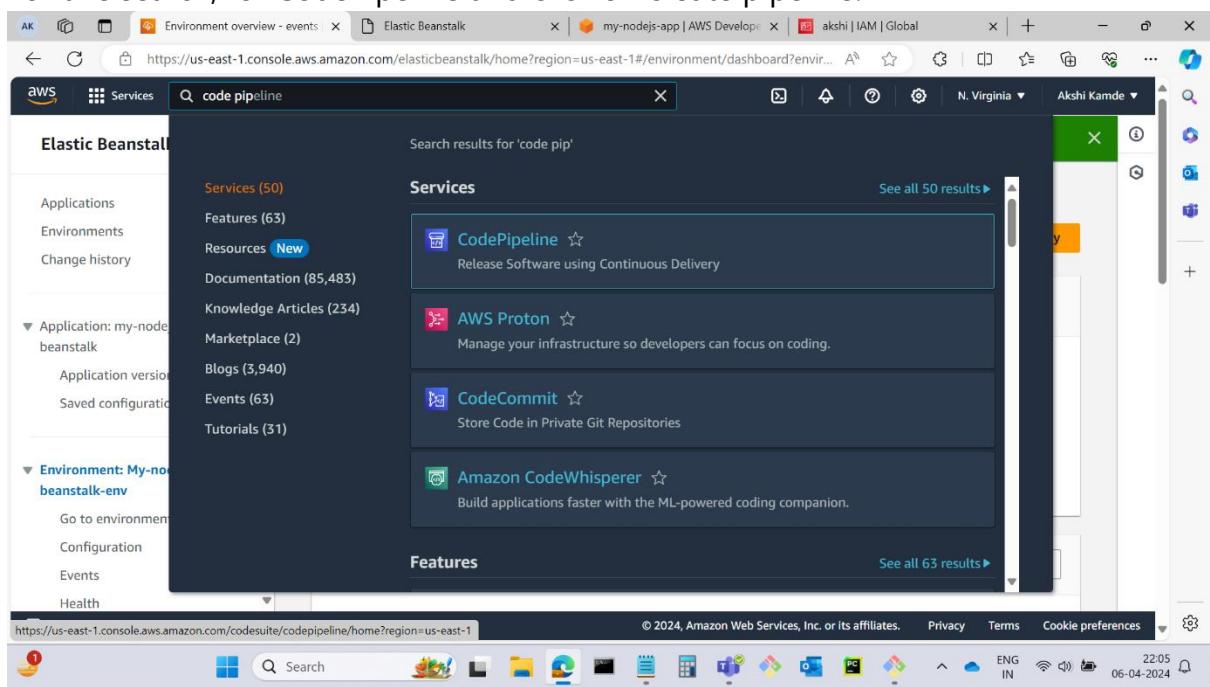
Voila! You can now see all the files on CodeCommit.



The screenshot shows the AWS CodeCommit interface. The left sidebar has sections for Source (CodeCommit), Code (Pull requests, Commits, Branches, Git tags, Settings), Artifacts (CodeArtifact), Build (CodeBuild), and Deploy (CodeDeploy). The main area displays the repository 'my-nodejs-app' with a list of files: app.js, cron.yaml, index.html, and package.json. There are buttons for Notify, master branch, Create pull request, and Clone URL.

Create a CodePipeline

For this search, for CodePipeline and click on create pipeline.



The screenshot shows the AWS search results for 'code pipeline'. The search bar contains 'code pipeline'. The results are categorized into Services and Features. Under Services, 'CodePipeline' is listed with the description 'Release Software using Continuous Delivery'. Other services like AWS Proton, CodeCommit, and Amazon CodeWhisperer are also shown. Under Features, there are 63 results, but none are explicitly named in the visible part of the list.

CodePipeline

Give a name to the pipeline and keep the default settings.

The image consists of three vertically stacked screenshots of the AWS CodePipeline 'Create new pipeline' wizard, specifically Step 1: Choose pipeline settings. The top screenshot shows the initial setup with a pipeline name 'my-node-pipeline' and the V2 pipeline type selected. The middle screenshot shows the 'Advanced settings' section where artifact store and encryption key options are configured. The bottom screenshot shows the final step of the wizard, where the pipeline is being created.

Step 1
Choose pipeline settings

Pipeline settings

Pipeline name: my-node-pipeline

Pipeline type: V2

Execution mode: Superseded

Advanced settings

Artifact store: Default location

Encryption key: Default AWS Managed Key

Next

Add source as AWS CodeCommit, select the repository name that was created and branch name. Keep the rest set to default.

CodePipeline

The image consists of three vertically stacked screenshots of the AWS CodePipeline 'Create new pipeline' wizard, specifically Step 2: Add source stage.

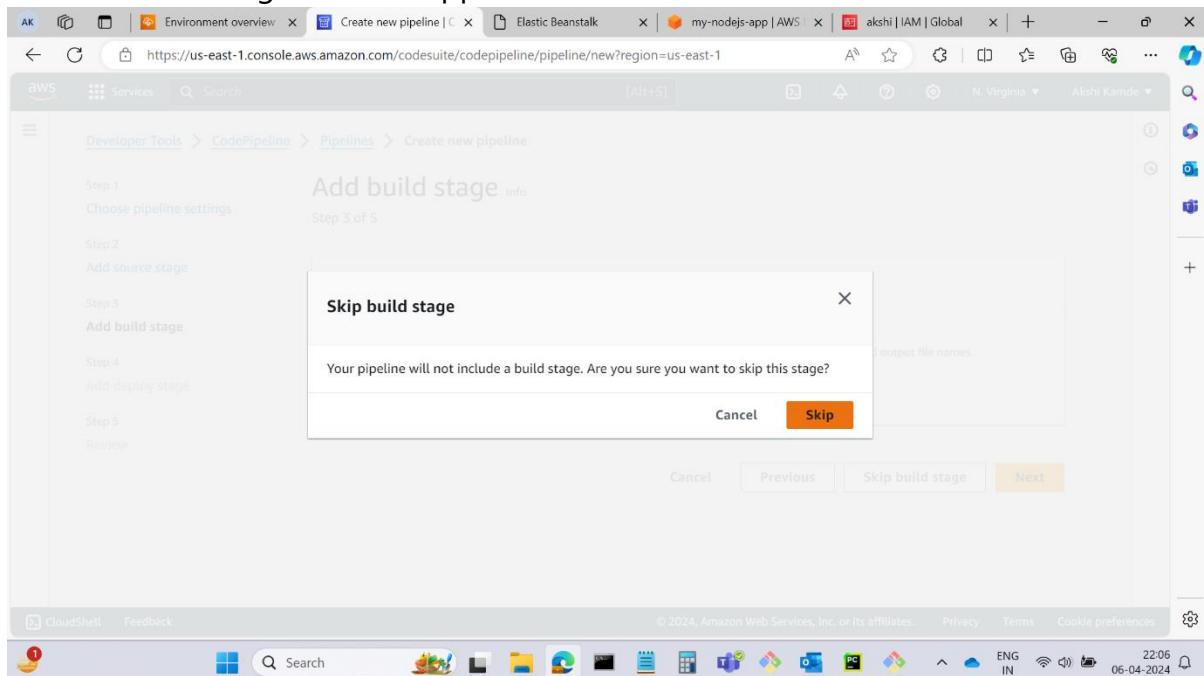
Screenshot 1: Shows the 'Source provider' dropdown set to 'AWS CodeCommit'. The 'Repository name' field contains 'my-nodejs-app' and the 'Branch name' field contains 'master'. The 'Change detection options' section shows two options: 'Amazon CloudWatch Events (recommended)' (selected) and 'AWS CodePipeline'.

Screenshot 2: Similar to Screenshot 1, but the 'Output artifact format' section is visible, showing 'CodePipeline default' (selected) and 'Full clone'.

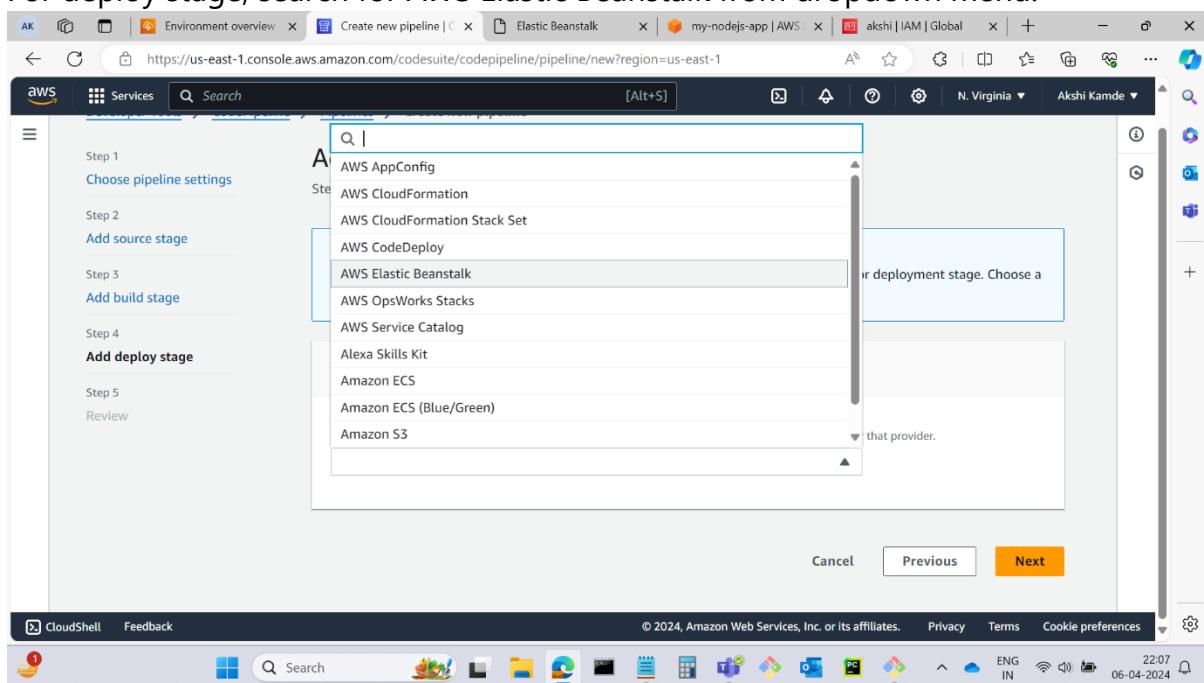
Screenshot 3: A blank screenshot of the same interface, likely a transition between steps.

CodePipeline

The next build stage can be skipped.



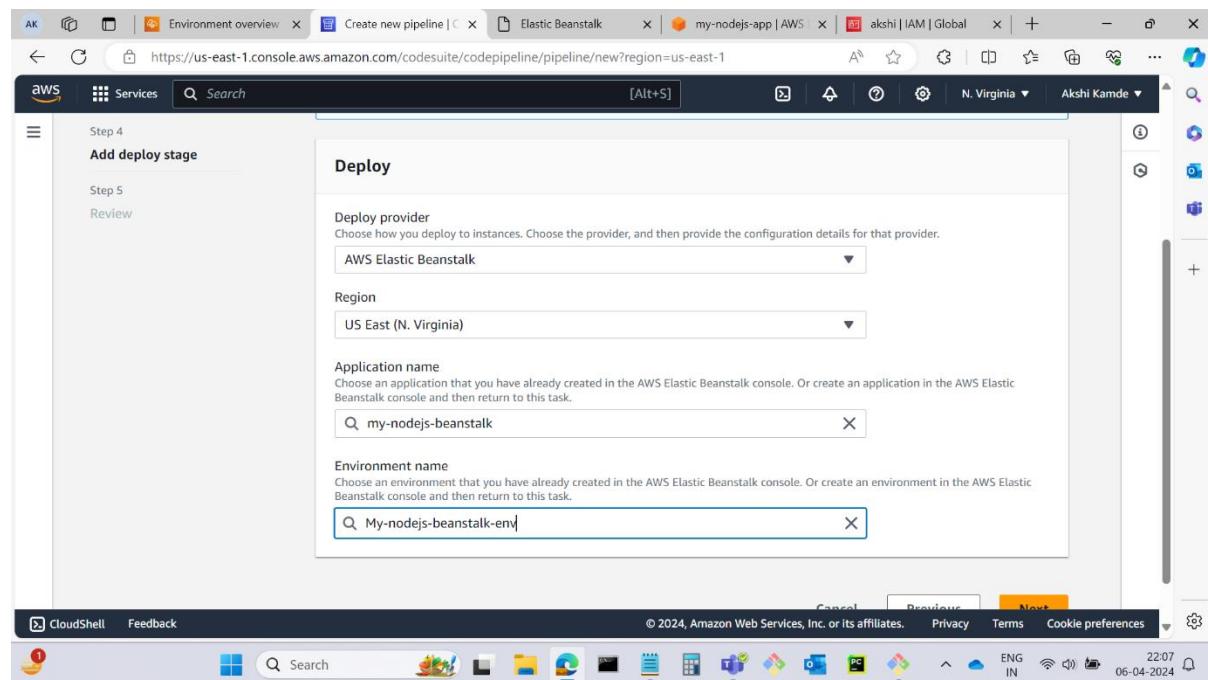
For deploy stage, search for AWS Elastic Beanstalk from dropdown menu.



Region will be selected by default, make sure it matches your current region, add the name of the application from dropdown. Also select the environment of the application. Currently we have only built one env, if we had built production env then

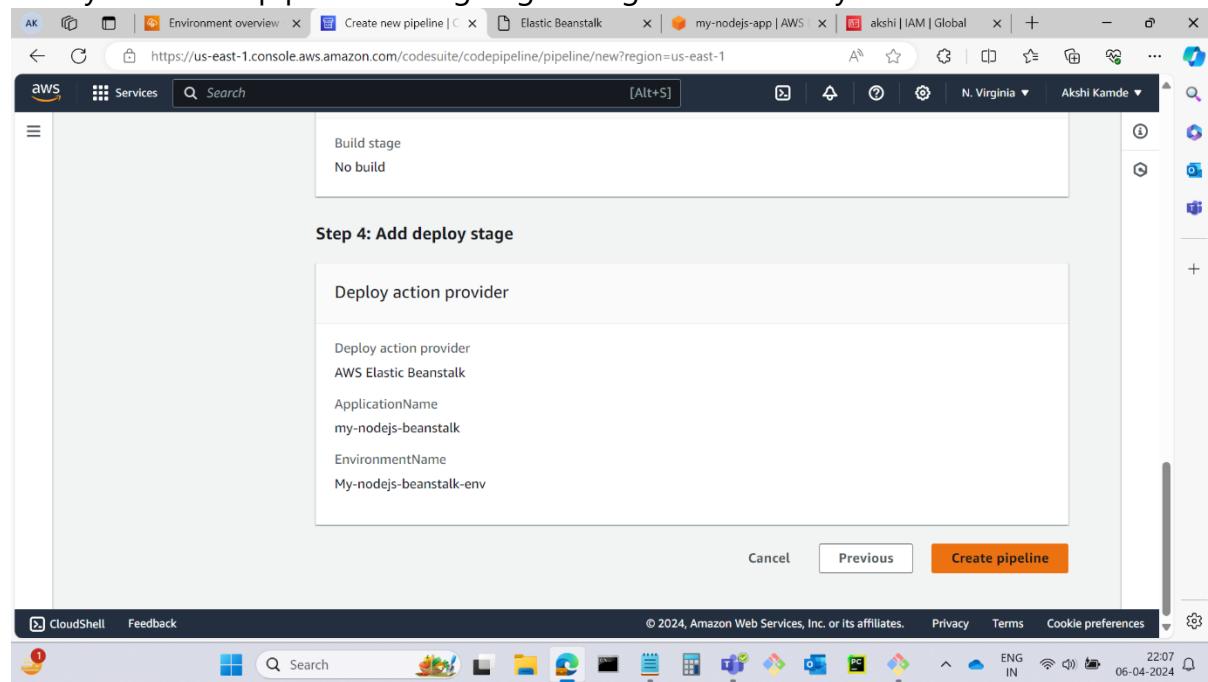
CodePipeline

we could have selected that too.



The screenshot shows the 'Deploy' configuration step in the AWS CodePipeline console. The 'Deploy provider' is set to 'AWS Elastic Beanstalk'. The 'Region' is 'US East (N. Virginia)'. The 'Application name' is 'my-nodejs-beanstalk' and the 'Environment name' is 'My-nodejs-beanstalk-env'. The interface includes a search bar and a 'Next' button at the bottom right.

Finally create the pipeline after going through the summary.



The screenshot shows the 'Create pipeline' summary step. It displays the 'Build stage' (No build) and the 'Step 4: Add deploy stage' configuration. Under 'Deploy action provider', it lists 'AWS Elastic Beanstalk', 'ApplicationName: my-nodejs-beanstalk', and 'EnvironmentName: My-nodejs-beanstalk-env'. The 'Create pipeline' button is highlighted in orange at the bottom right.

In few minutes you would see that your pipeline is running successfully and application is also deployed. You can click on the link provided for elastic beanstalk from the deploy stage. From here you can check the health of your environment and check for the domain link where your application is running.

CodePipeline

The screenshot shows the AWS CodePipeline console interface across three separate browser tabs. Each tab displays a pipeline execution with two stages: Source and Deploy.

- Top Tab:** Pipeline type: V2, Execution mode: QUEUED. The Source stage is Succeeded (green checkmark). The Deploy stage is also Succeeded (green checkmark). A "Disable transition" button is visible between the stages.
- Middle Tab:** Pipeline type: V2, Execution mode: QUEUED. The Source stage is Succeeded (green checkmark). The Deploy stage is Succeeded (green checkmark). A "Disable transition" button is visible between the stages.
- Bottom Tab:** Pipeline type: V2, Execution mode: QUEUED. The Source stage is Succeeded (green checkmark). The Deploy stage is Succeeded (green checkmark). A "Disable transition" button is visible between the stages.

The pipeline structure is as follows:

```
graph TD; Source[Source] --> Deploy1[Deploy]; Deploy1 --> Deploy2[Deploy]
```

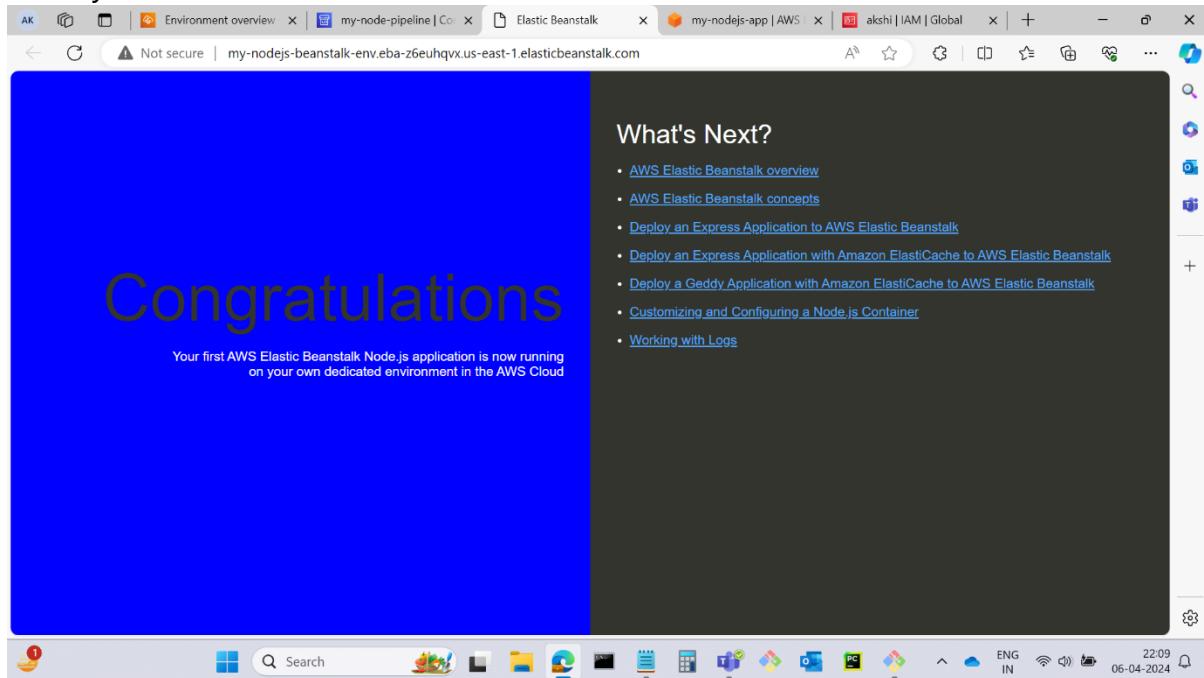
Each stage is associated with an AWS service:

- Source: AWS CodeCommit
- Deploy: AWS Elastic Beanstalk

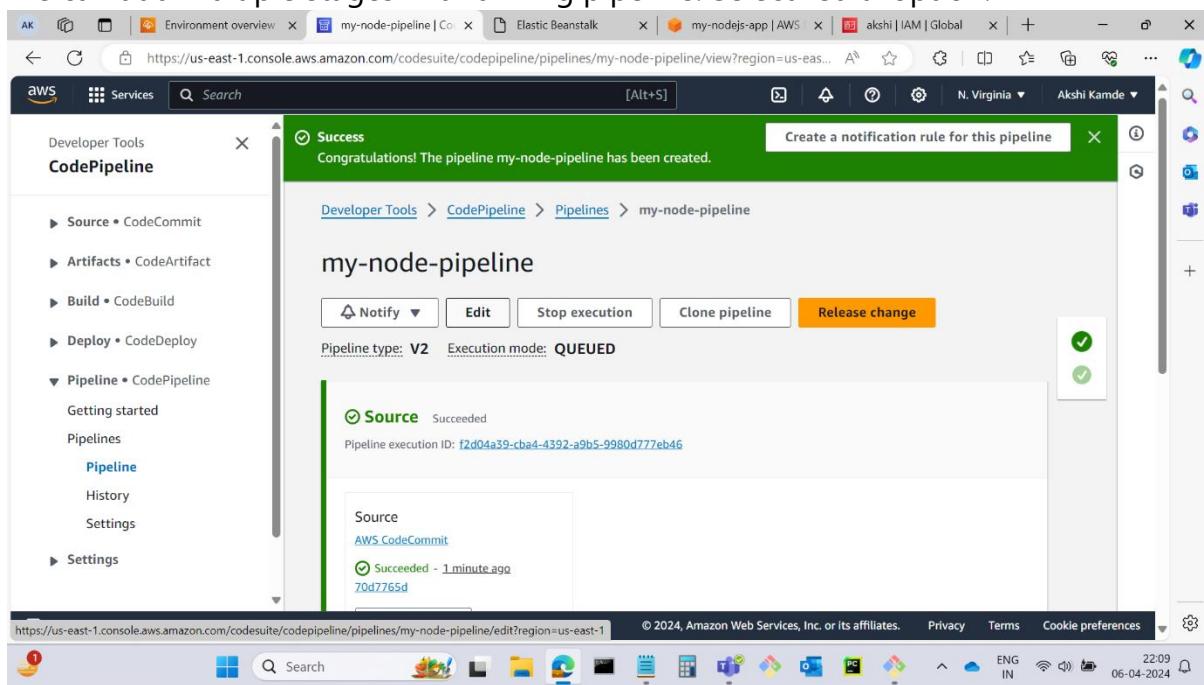
The pipeline ID is f2d04a39-cba4-4392-a9b5-9980d777eb46.

CodePipeline

The code changes the color of background from green to blue and that can be seen when you view the site.

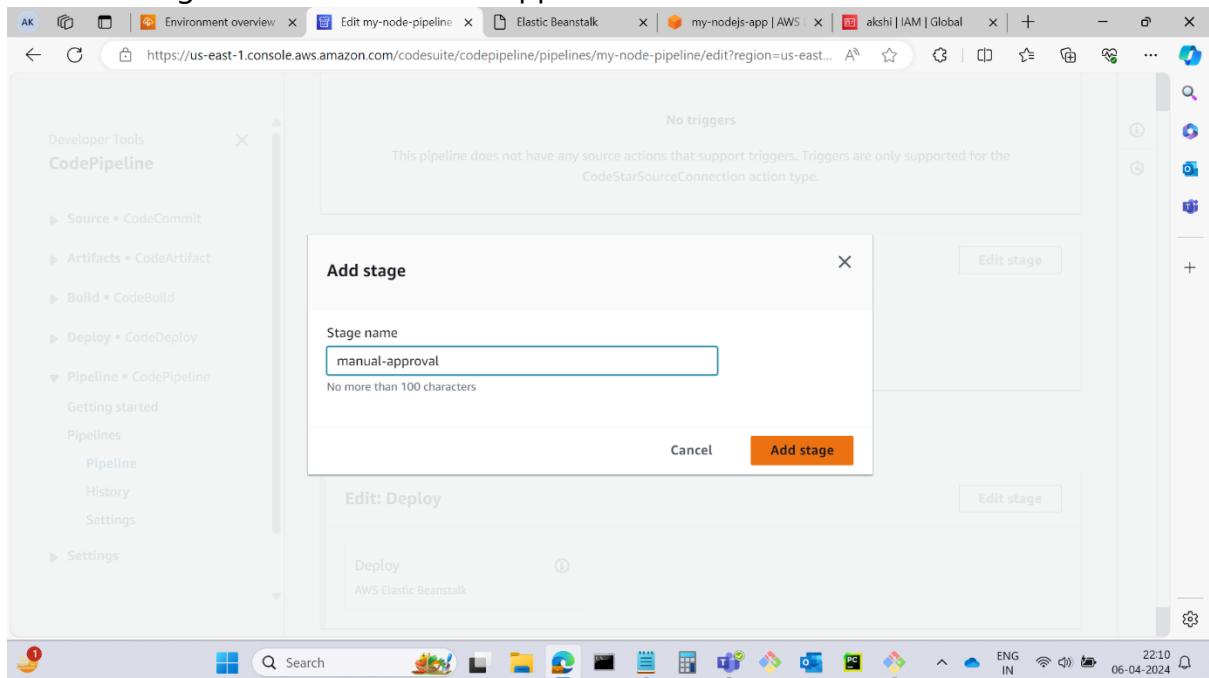


We can add multiple stages in a running pipeline. Select "edit" option.



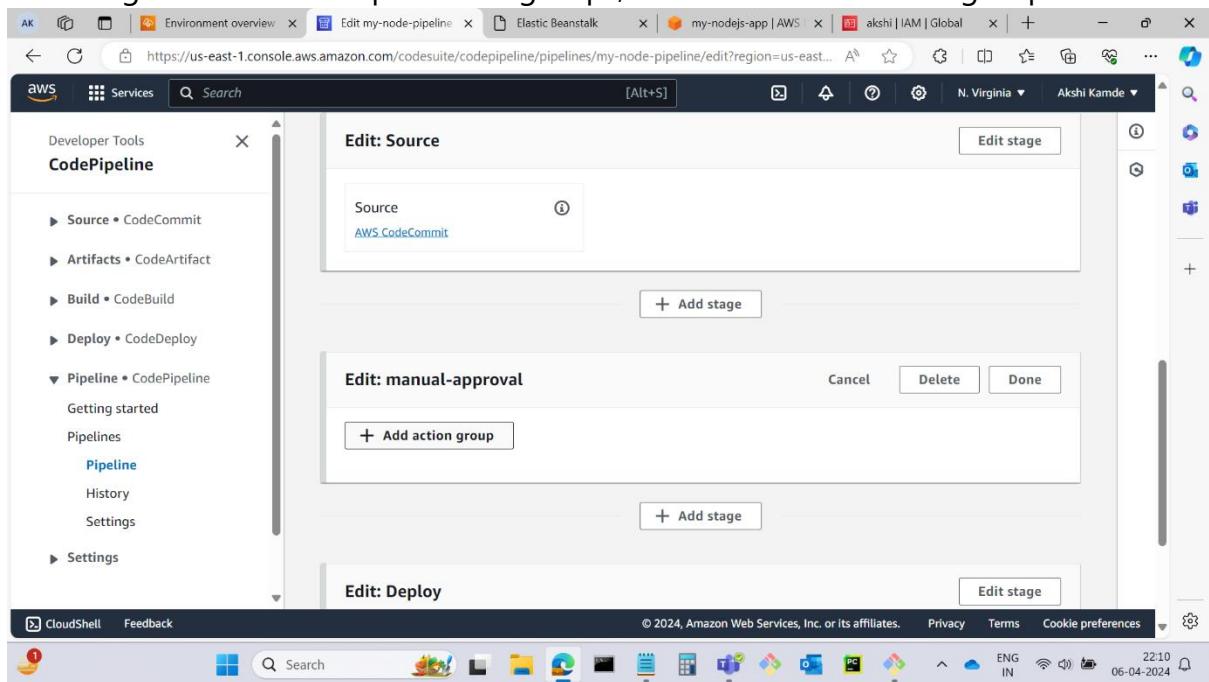
CodePipeline

Give this stage a name as "manual-approval"



The screenshot shows the AWS CodePipeline console with a pipeline named "my-node-pipeline". A modal window titled "Add stage" is open, prompting for a "Stage name". The name "manual-approval" is typed into the input field. Below the input field, there is a note: "No more than 100 characters". At the bottom right of the modal, the "Add stage" button is highlighted with a red border.

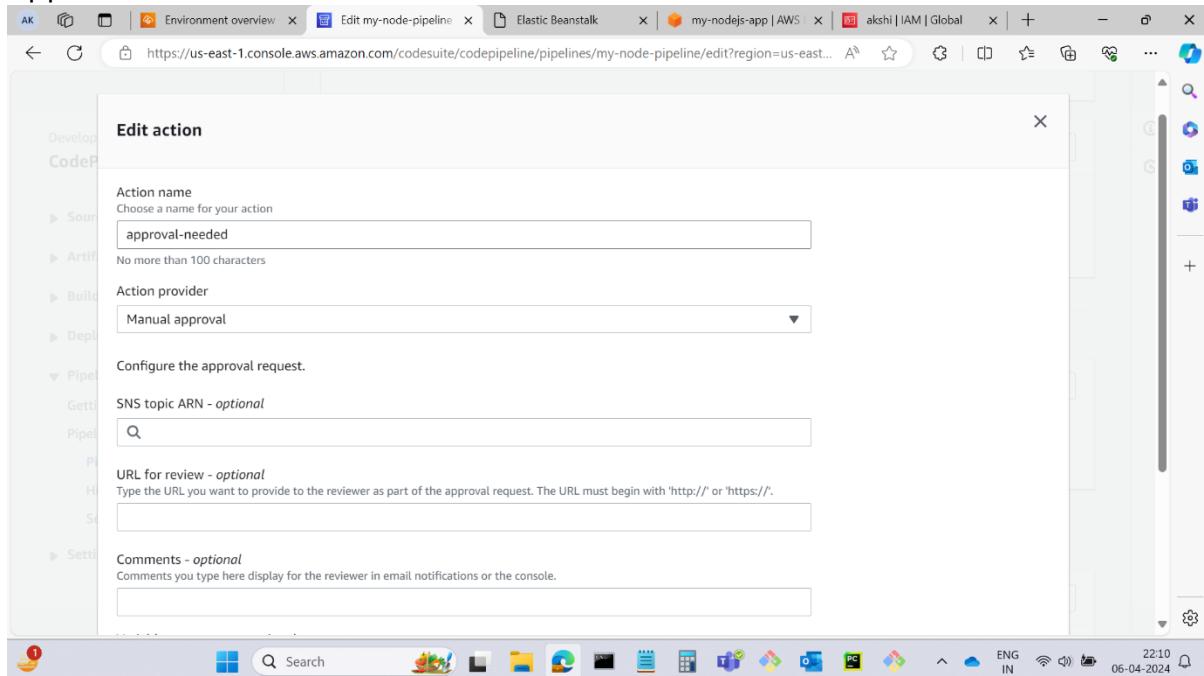
Each stage can have multiple action groups, so click on "add action group".



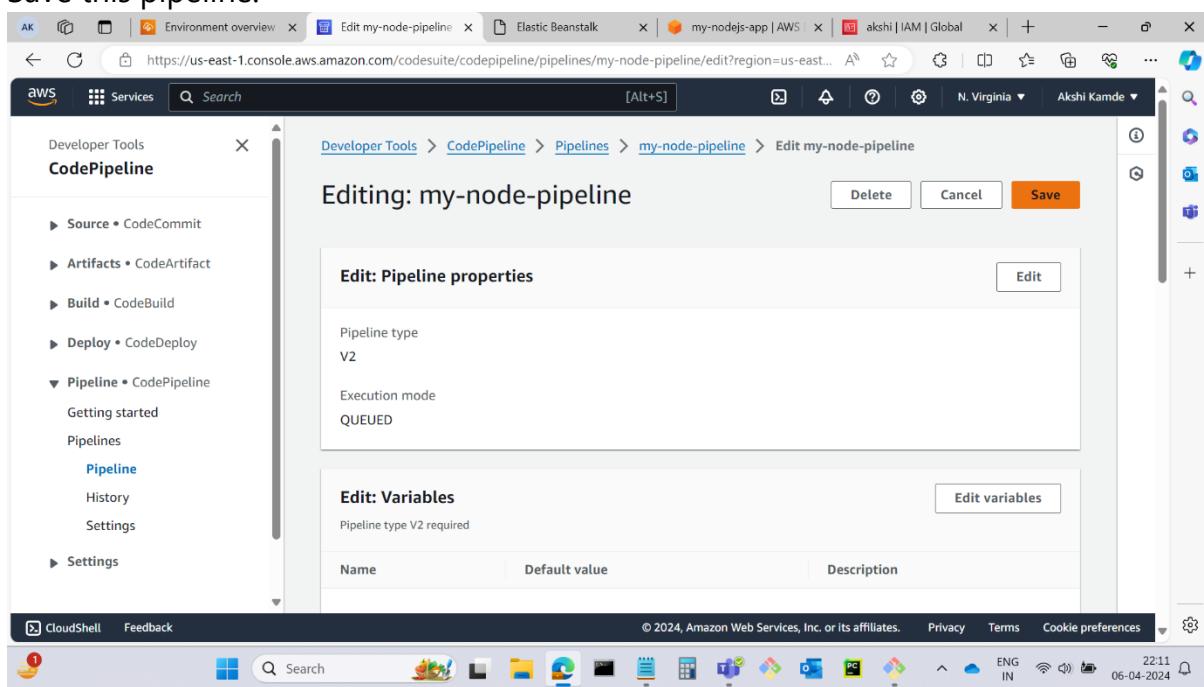
The screenshot shows the AWS CodePipeline console with the "my-node-pipeline" pipeline selected. The "Pipeline" tab is active. The "Edit: manual-approval" stage is currently being edited. A modal window titled "Edit: manual-approval" is open, containing a single button: "+ Add action group". This button is highlighted with a red border.

CodePipeline

In here, type the name as "approval-needed", the action provider would be "Manual Approval".



Save this pipeline.



Click on "Release Change". You will notice that application now pauses at the second stage, waiting for an approval. Click on "Review"

CodePipeline

The screenshot shows the AWS CodePipeline console interface. The pipeline, named "my-node-pipeline", is currently in the "QUEUED" state. The first stage, "Source", has completed successfully. The second stage, "manual-approval", is pending approval. A callout box highlights the "Review" button within this stage's details panel. Two "Disable transition" buttons are also visible, one above and one below the approval stage.

Developer Tools > CodePipeline > Pipelines > my-node-pipeline

my-node-pipeline

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded

Succeeded - 3 minutes ago

View details

manual-approval Pending

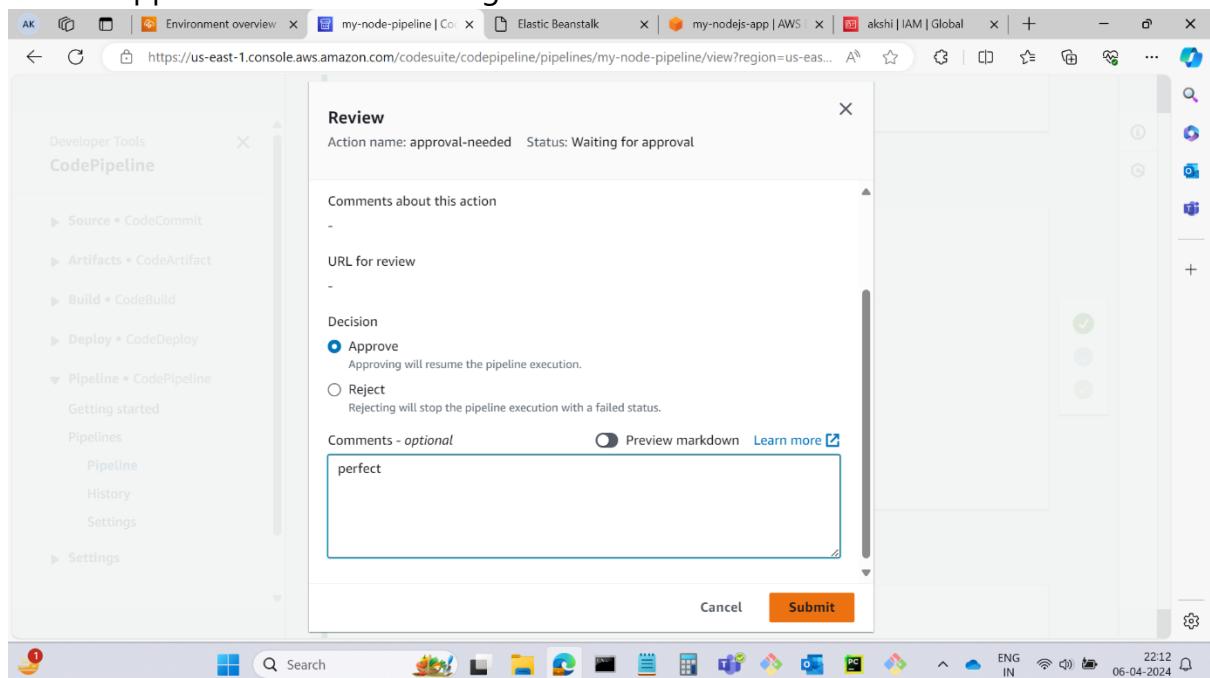
Review

Disable transition

Disable transition

CodePipeline

Select "Approve" and write message. Click on "Submit"



You can also stop a pipeline.

