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.

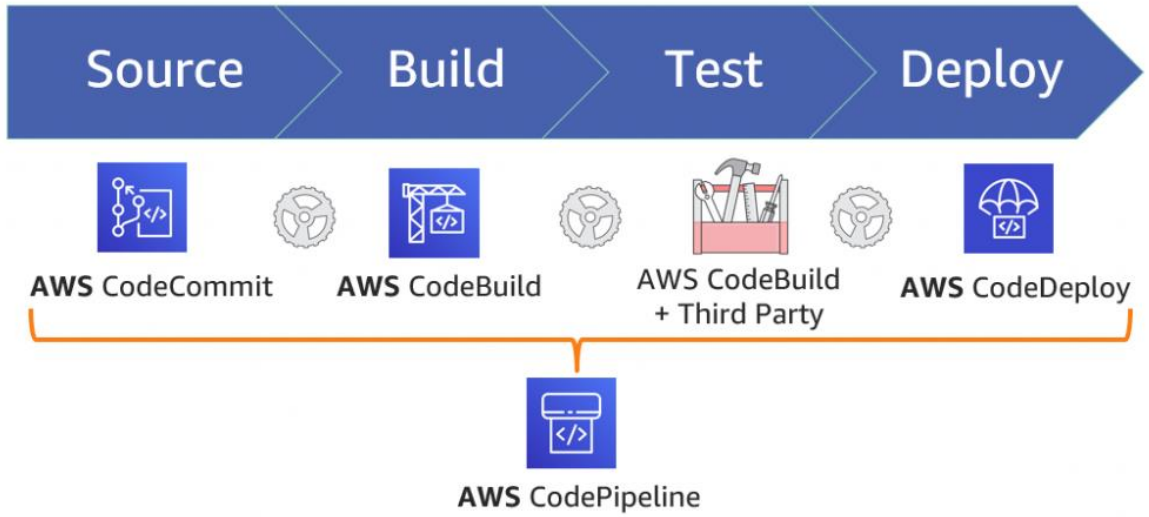
Artifact Handling: Once the code changes are detected, CodePipeline automatically packages the source code into artifacts and stores them in an S3 bucket designated

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

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.

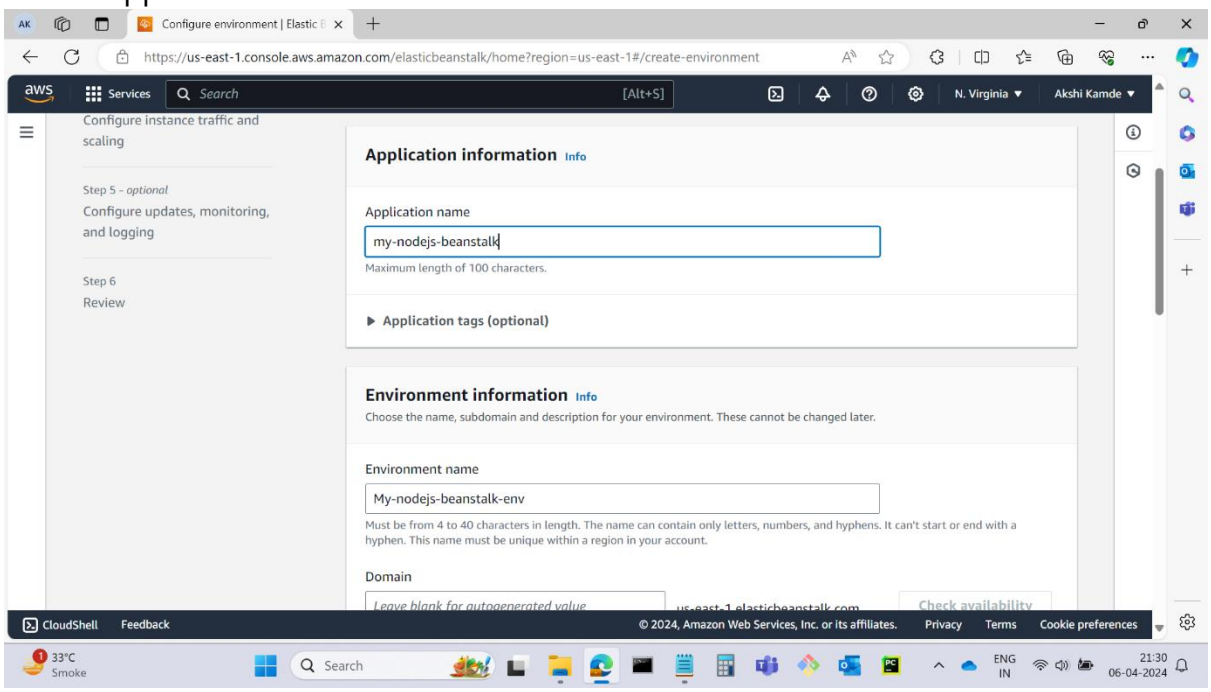
There are 3 major steps in this tutorial.

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.

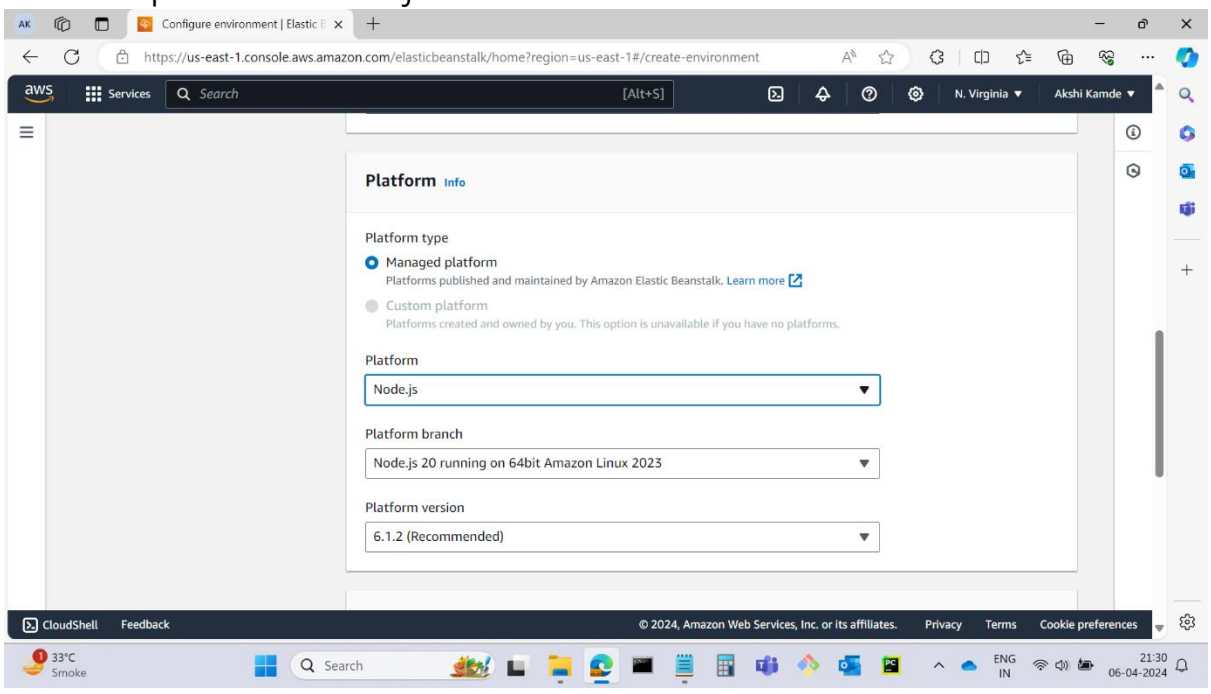
Create a Web Server Environment using Elastic Beanstalk

CodePipeline

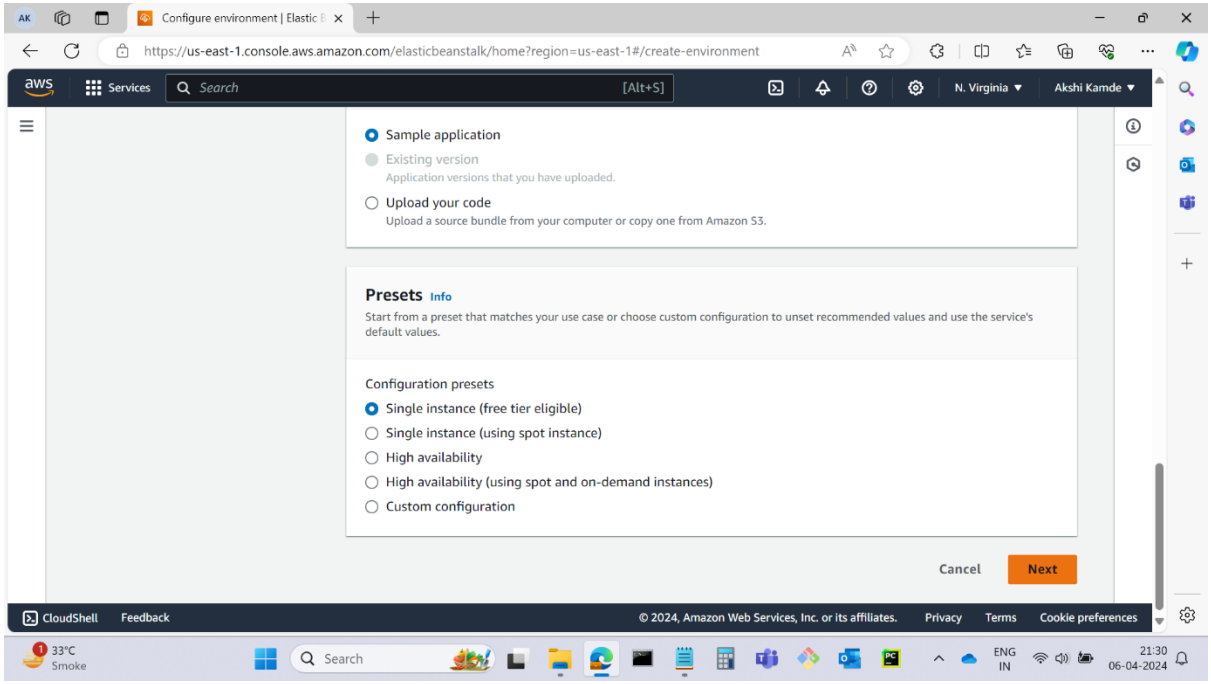
First search for “Elastic Beanstalk” and then click on create application. Add a name for the application.



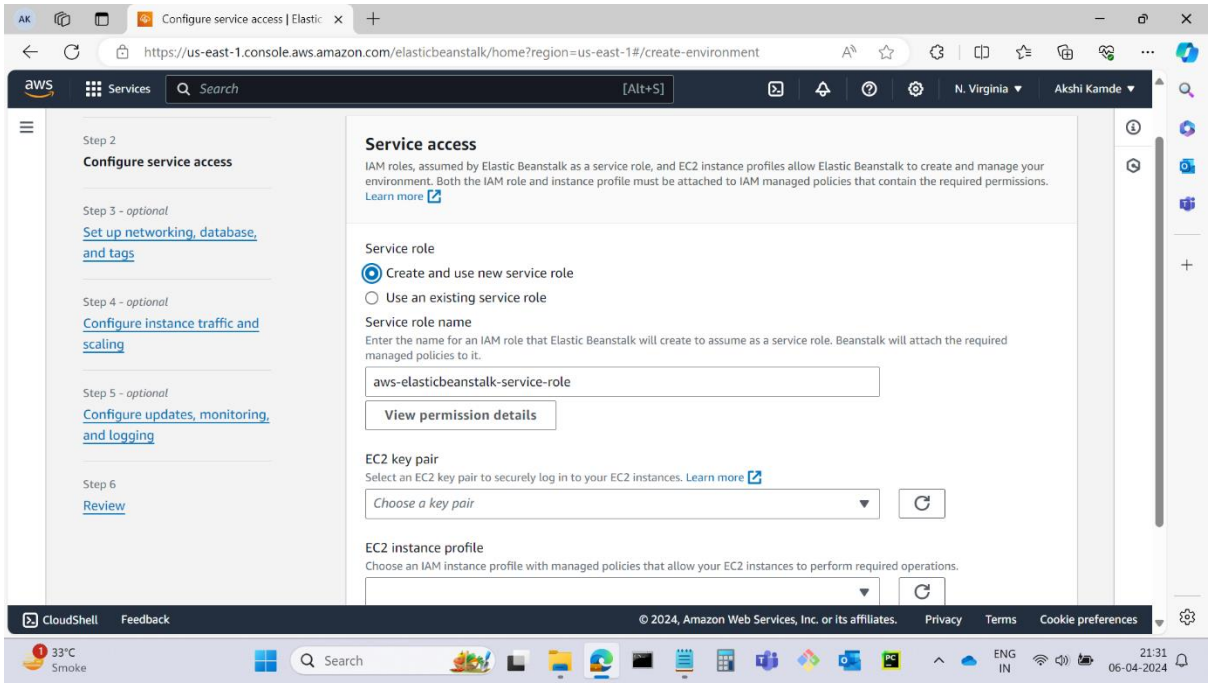
Select the platform as Node.js



Keep the default settings and click on next.



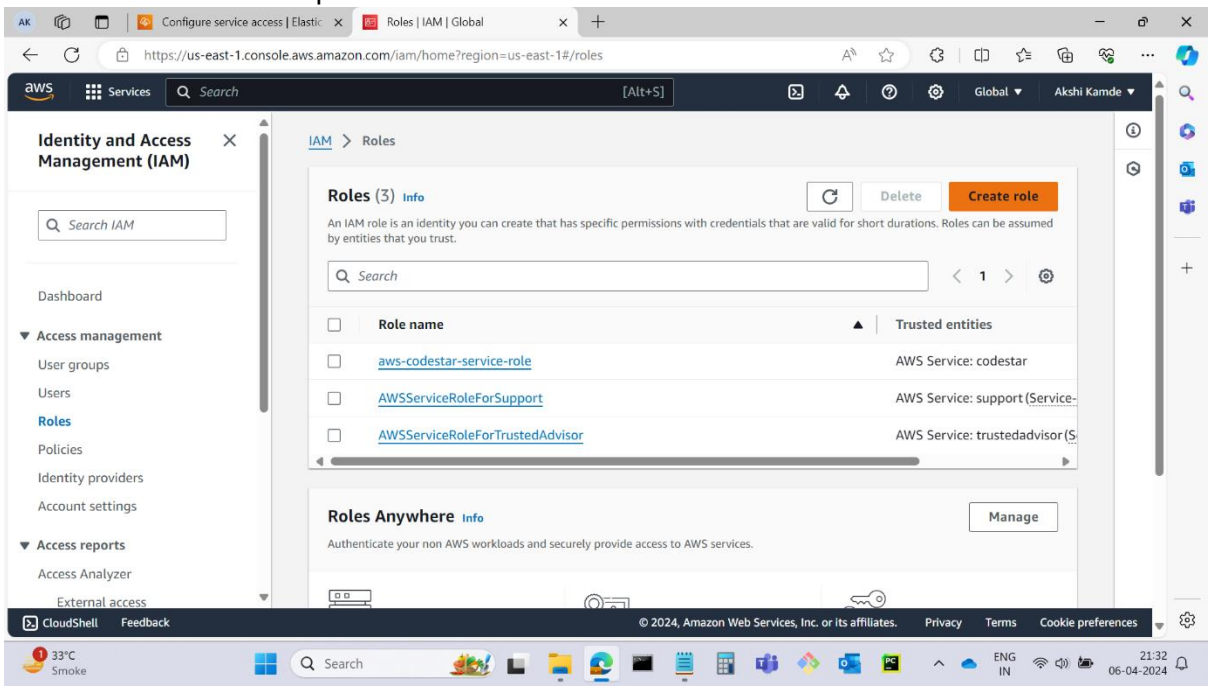
Create and use a new service role.



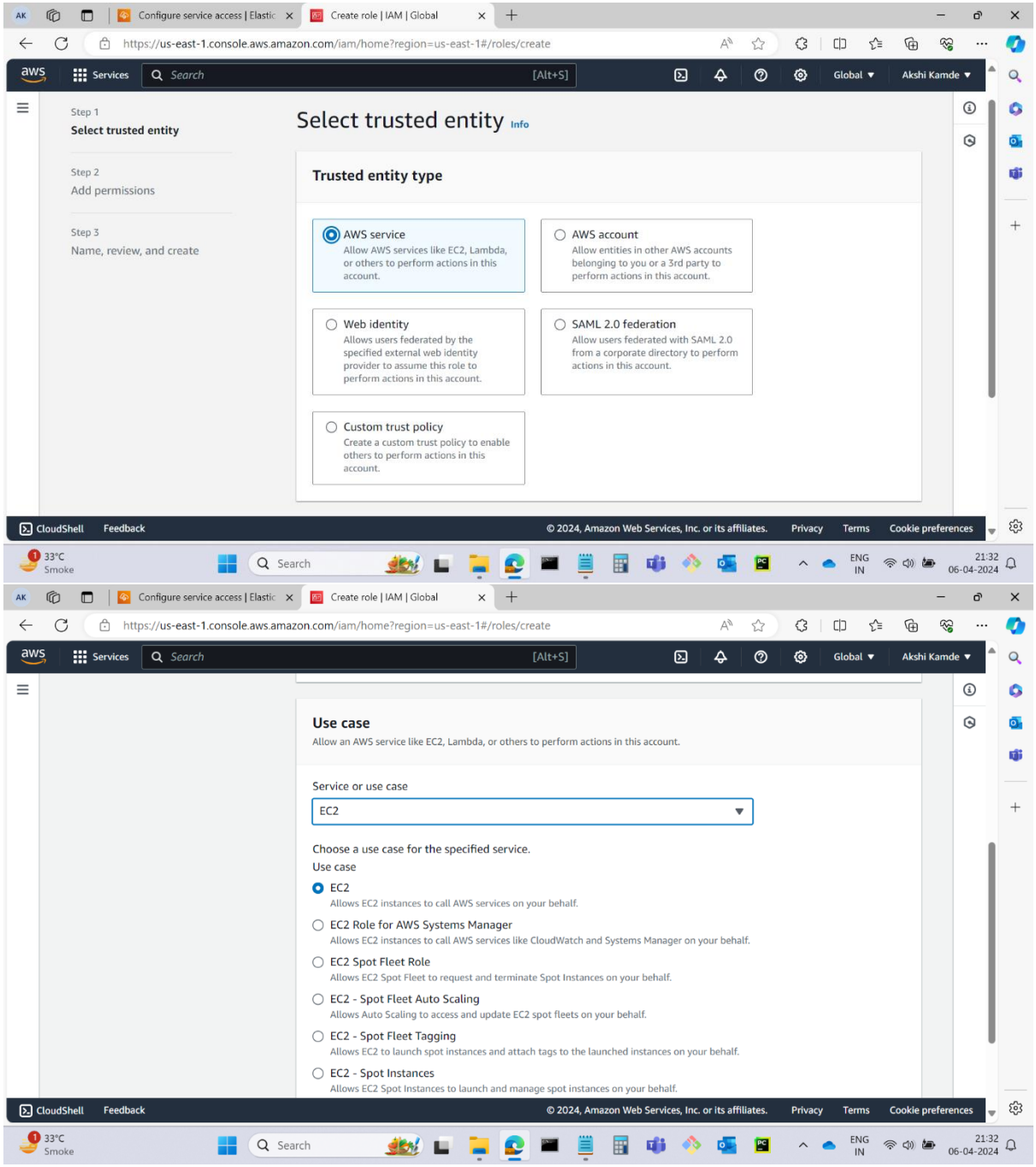
Create a New Service Role and IAM Role with EC2 Permissions:

CodePipeline

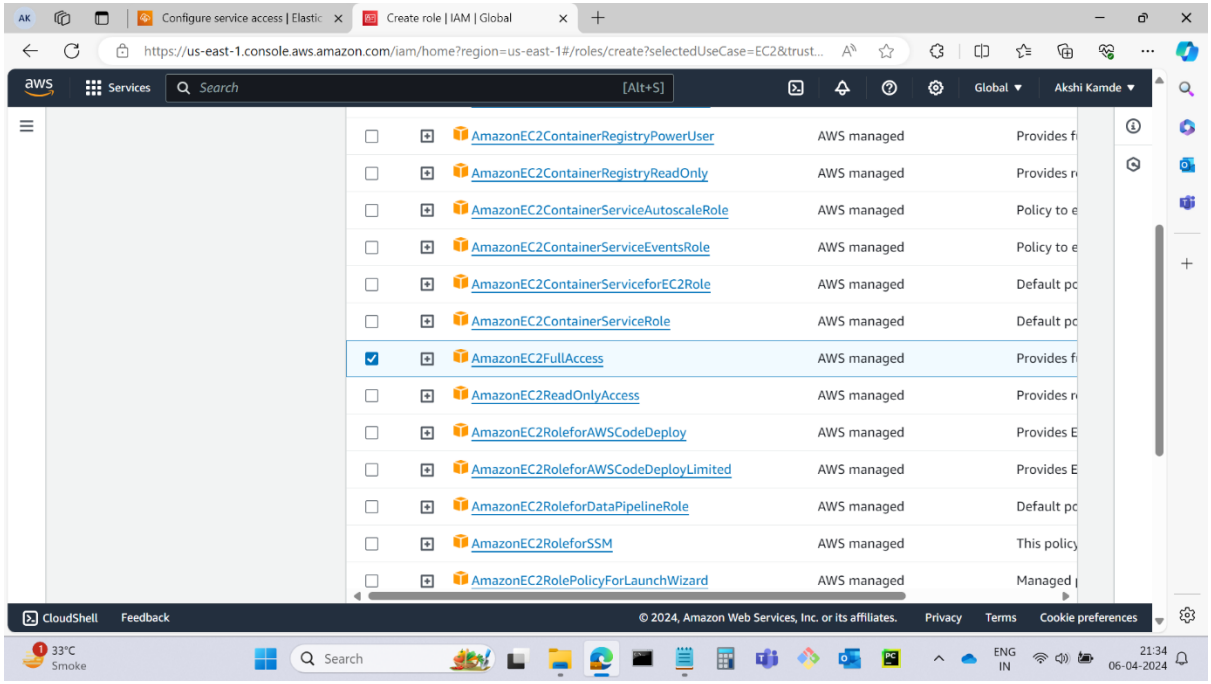
Create an IAM role with EC2 access. For this search for “IAM” in search bar and move to roles from left side panel. Click on “Create role”



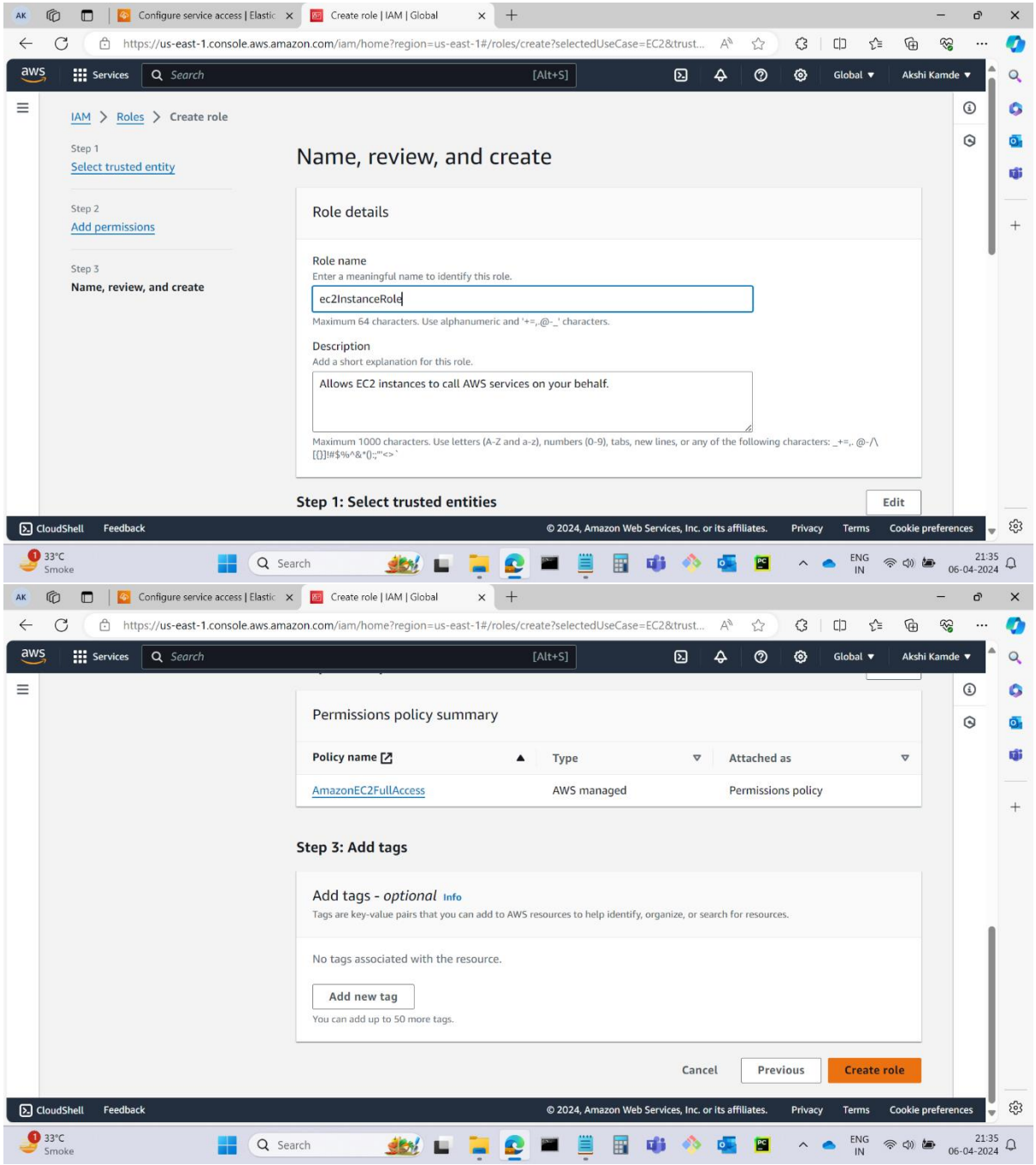
Keep AWS service as trusted entity type and the use case as EC2 and hit next.



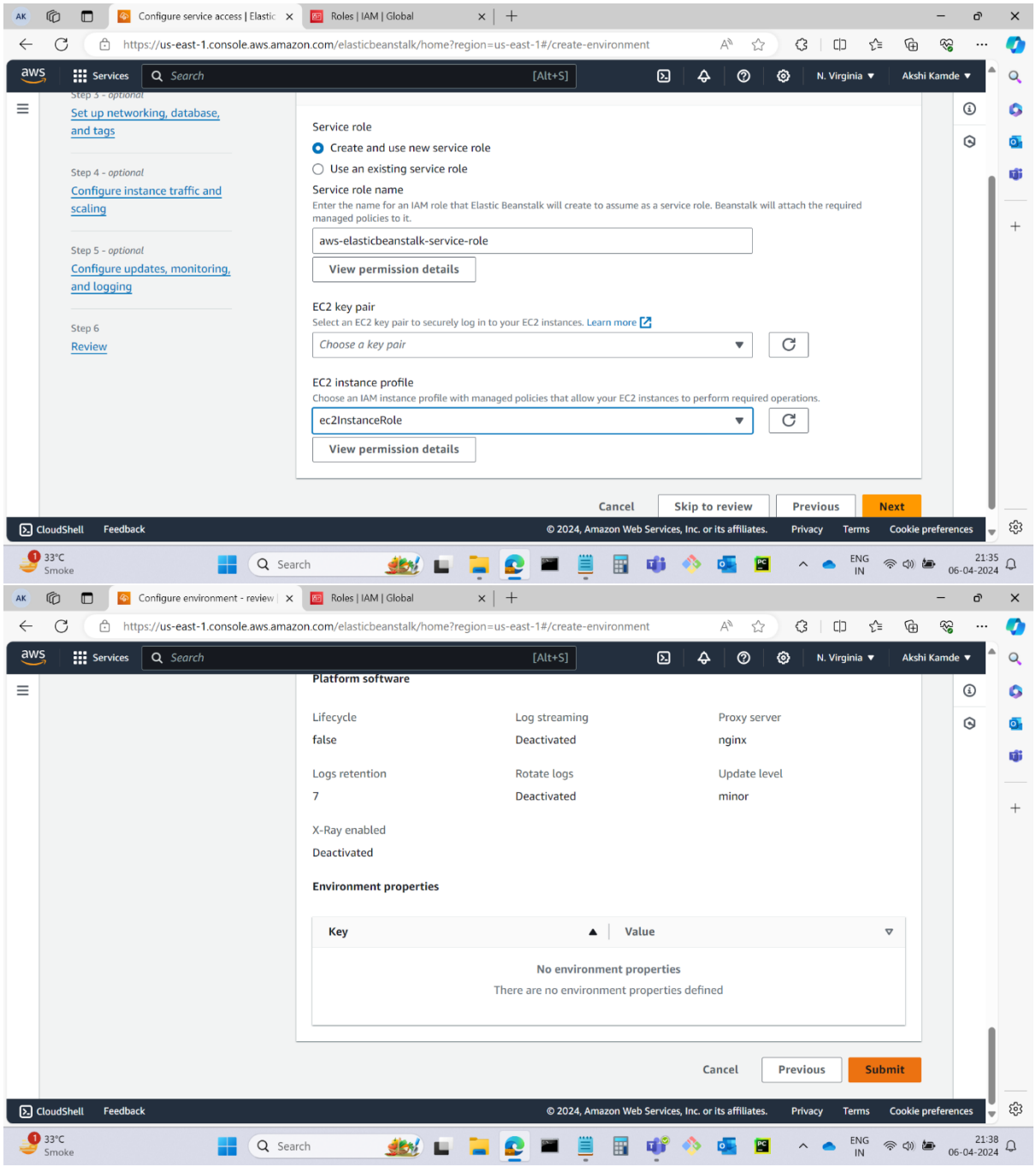
Search for EC2FullAccess and click next.



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

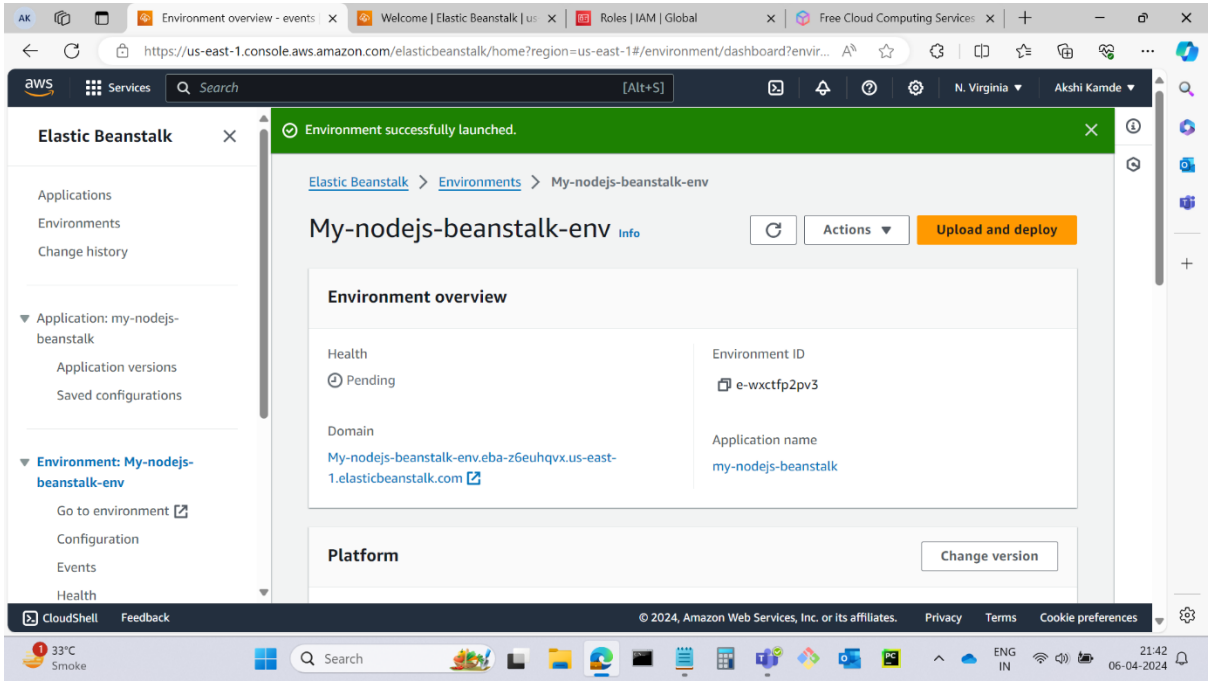


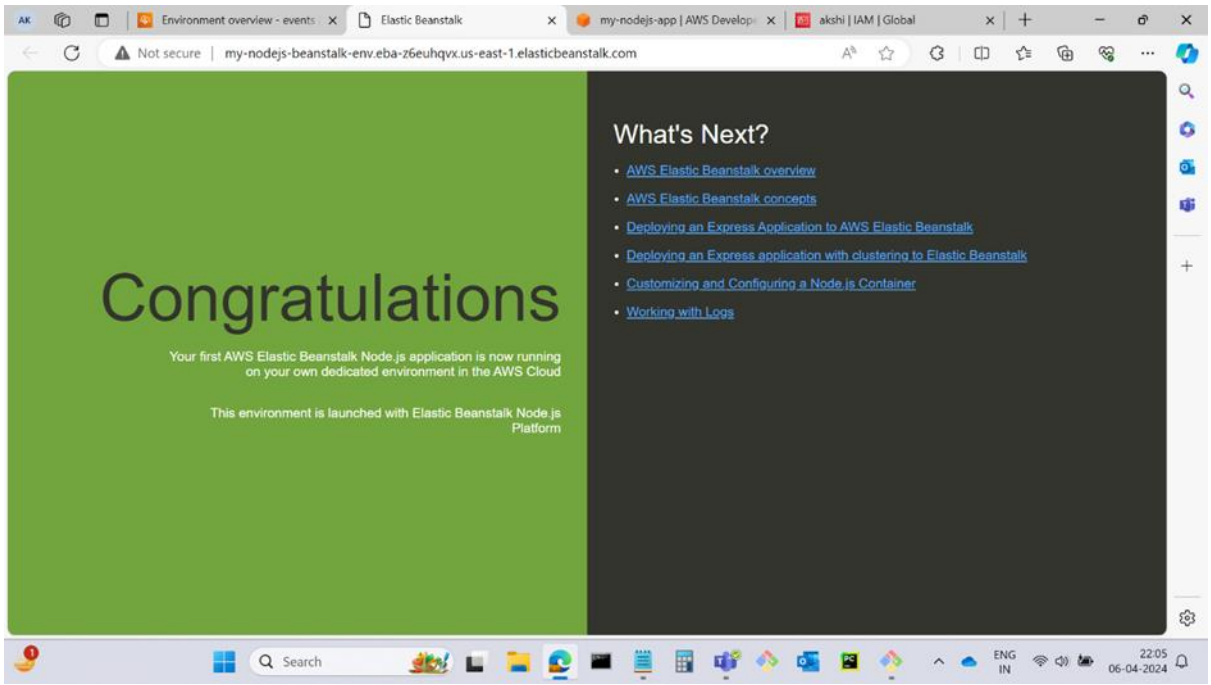
Now you can use this IAM role that was created for creating the elastic beanstalk environment. Click on refresh icon beside EC2 instance profile and select the newly created role from dropdown. Skip the VPC section for now and hit next. A summary will be shown and now you select submit.



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

screen for Elastic Beanstalk.

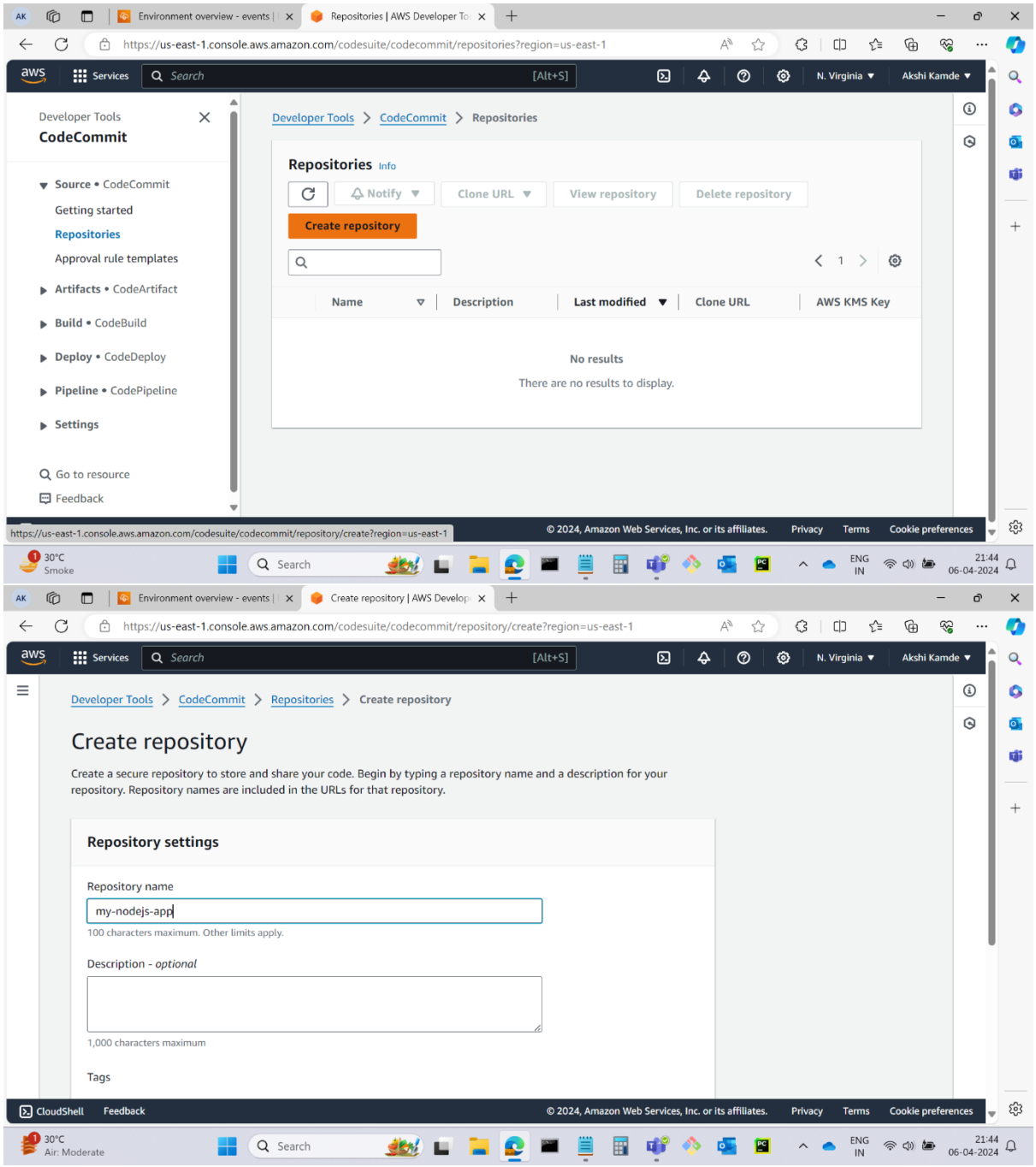




Create a CodeCommit repository and establish a git connection on local

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.

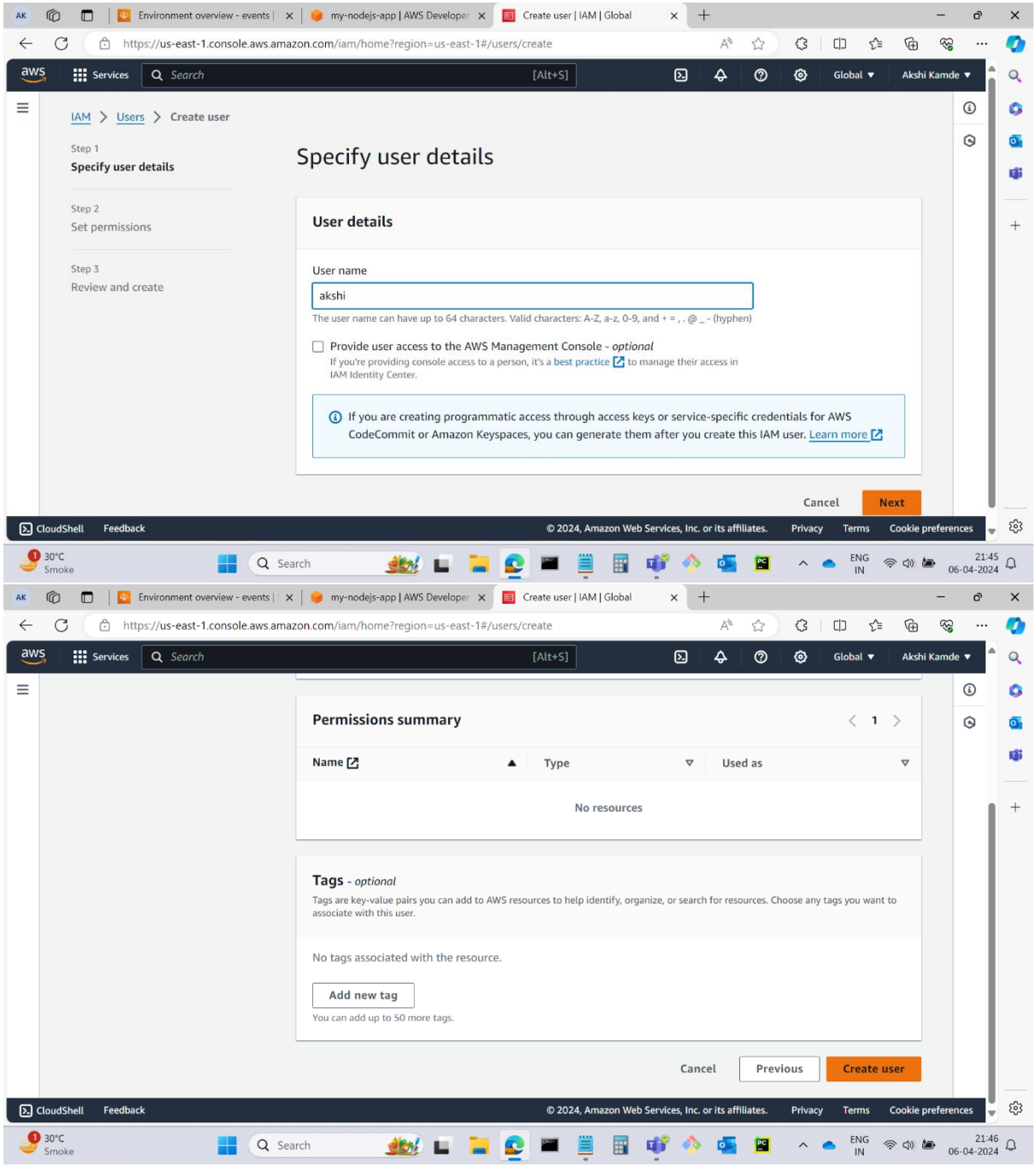
CodePipeline



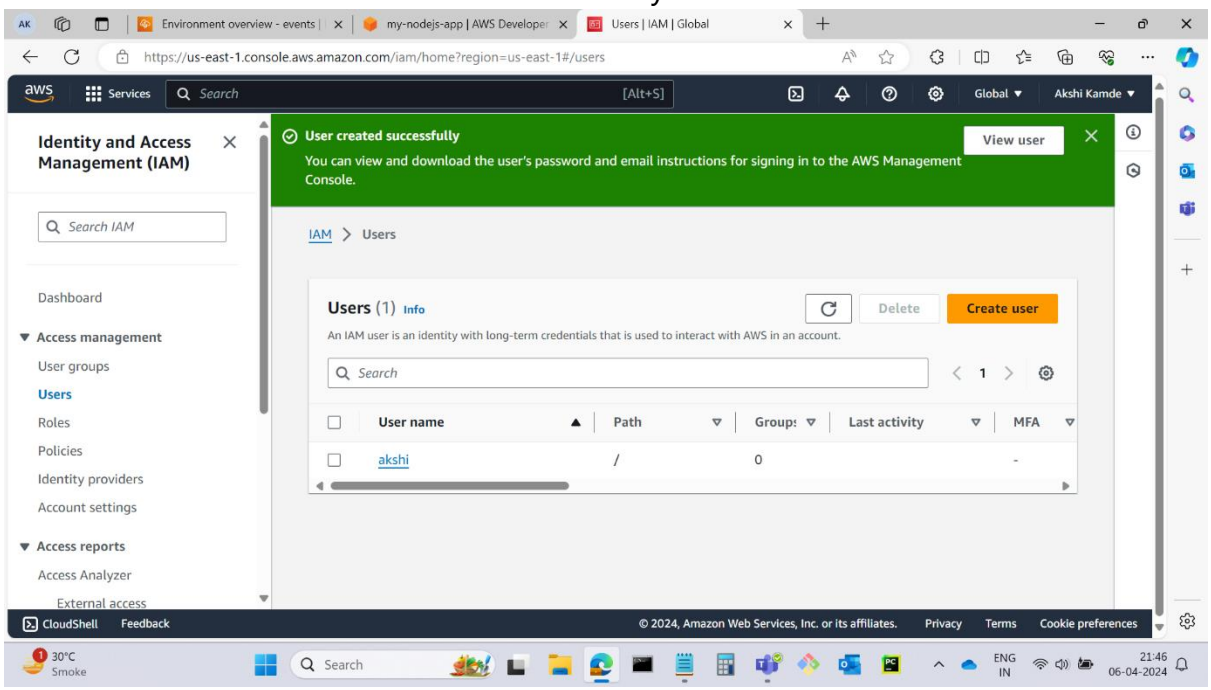
To add files from local, we need to establish a connection between CodeCommit and our local machine. But it is not recommended for root users to establish git connections so for this we shall create an IAM group and then add a new IAM user to this group with necessary permissions.

CodePipeline

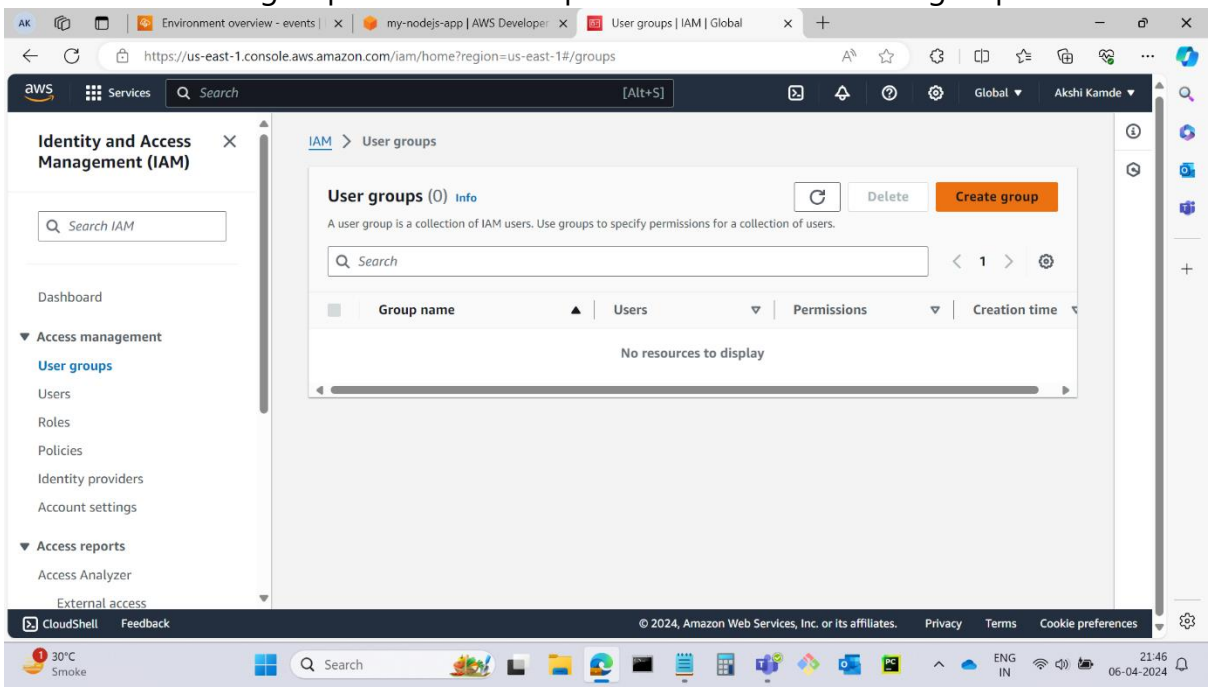
First click on Users from left side panel and fill the details.



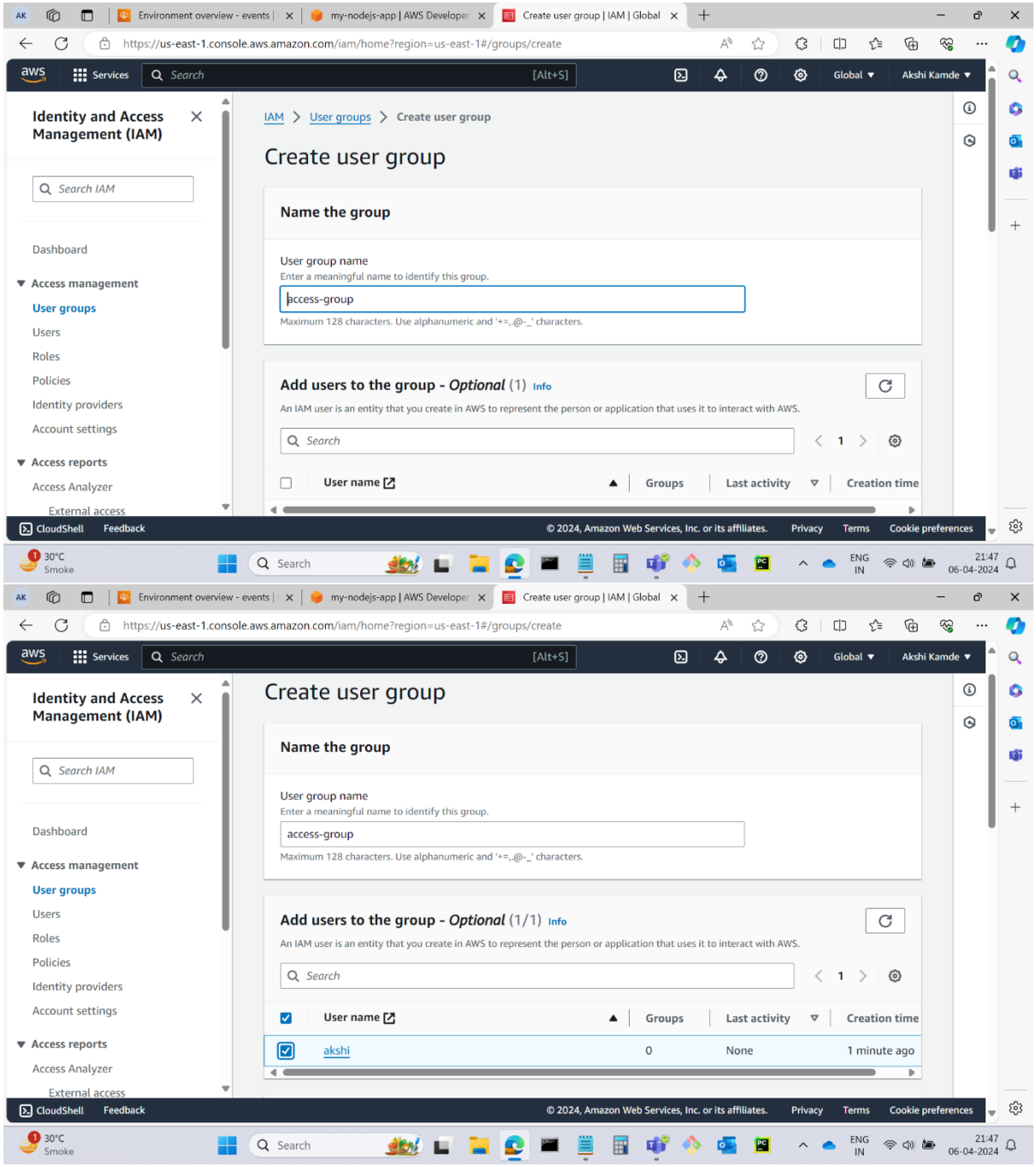
Hit create user and the user will be successfully created.



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

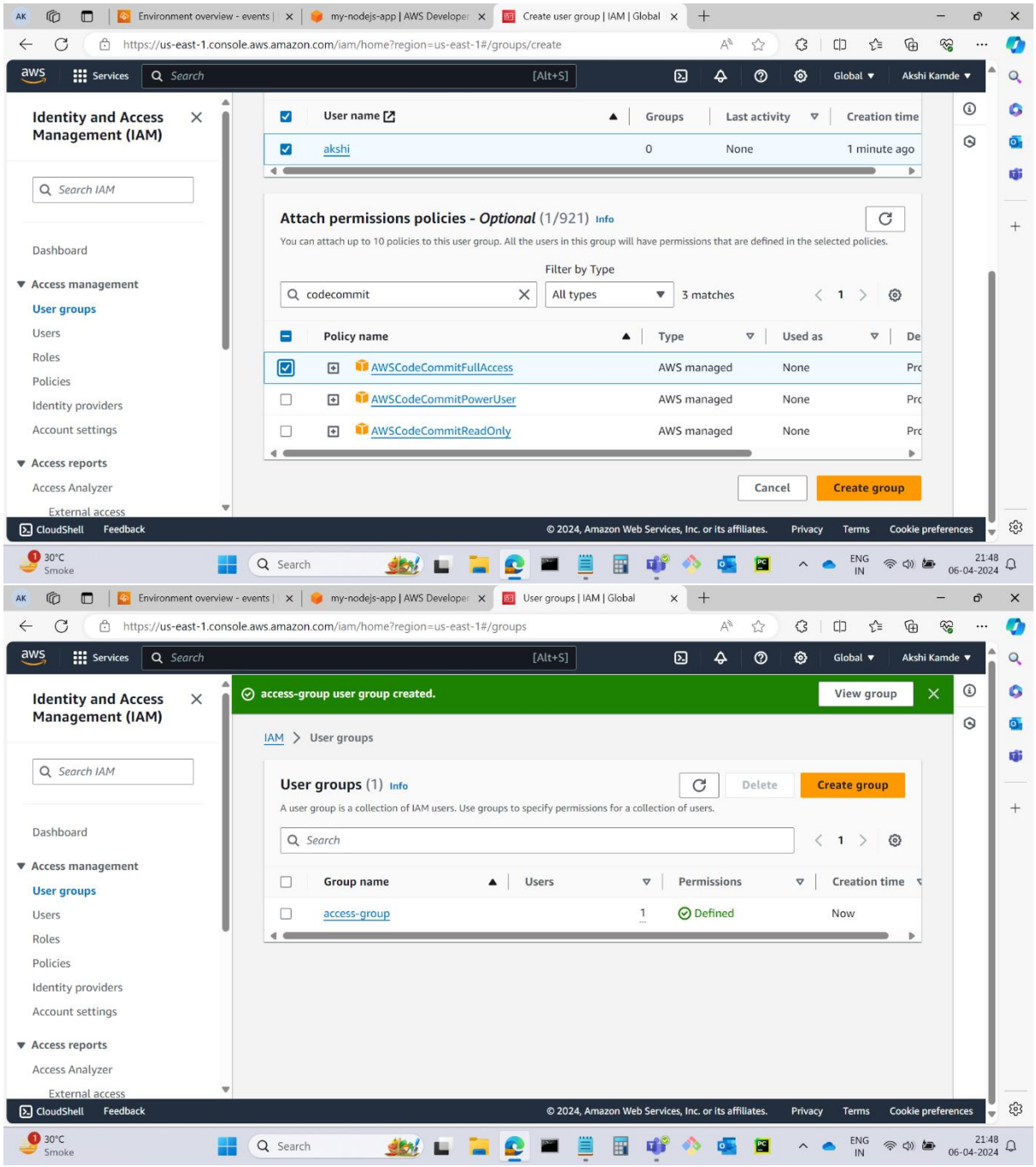


Give a name and add the previously created user to this group.

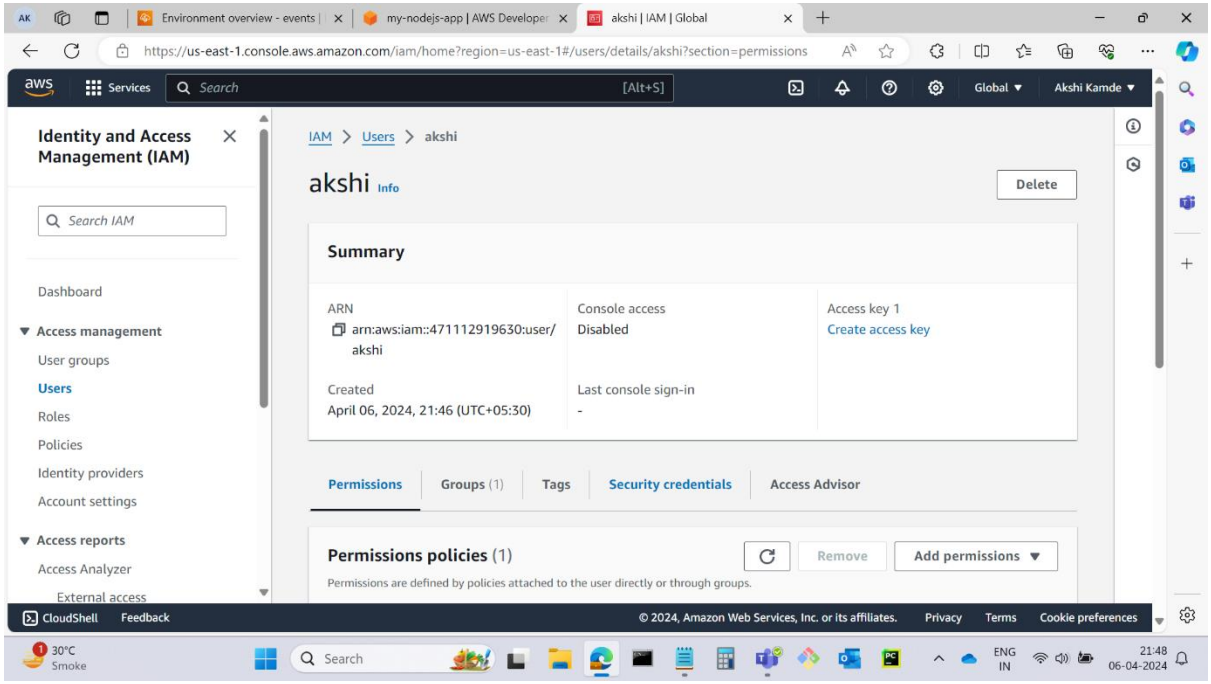


Search for CodeCommit and choose AWSCodeCommitFullAccess as the attached policy for this grup. You can also give Full Administration access. Click on create group.

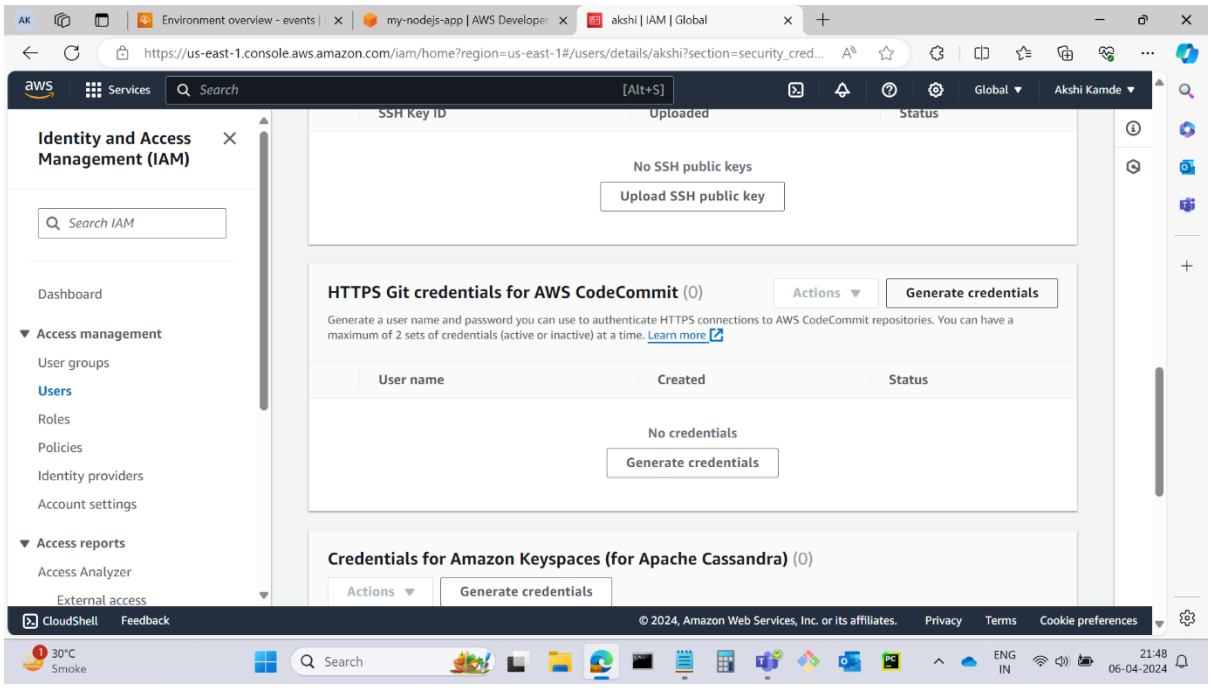
CodePipeline



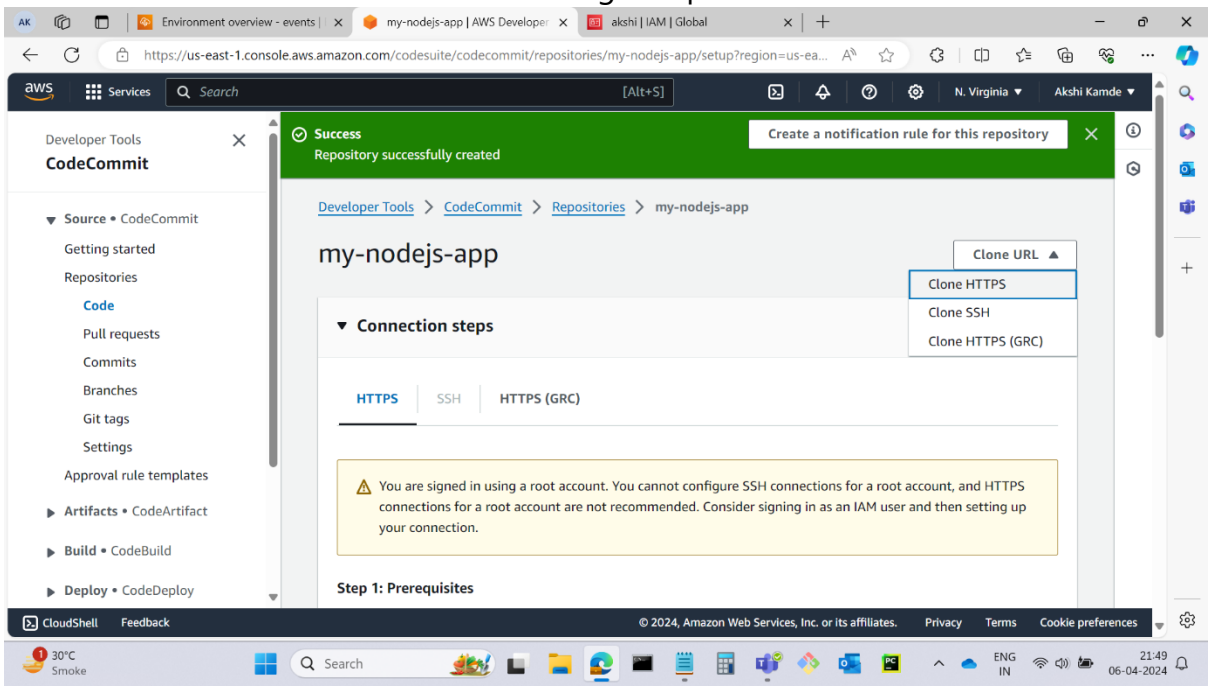
Now go to Users and select the created user. Select “Security Credentials”



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

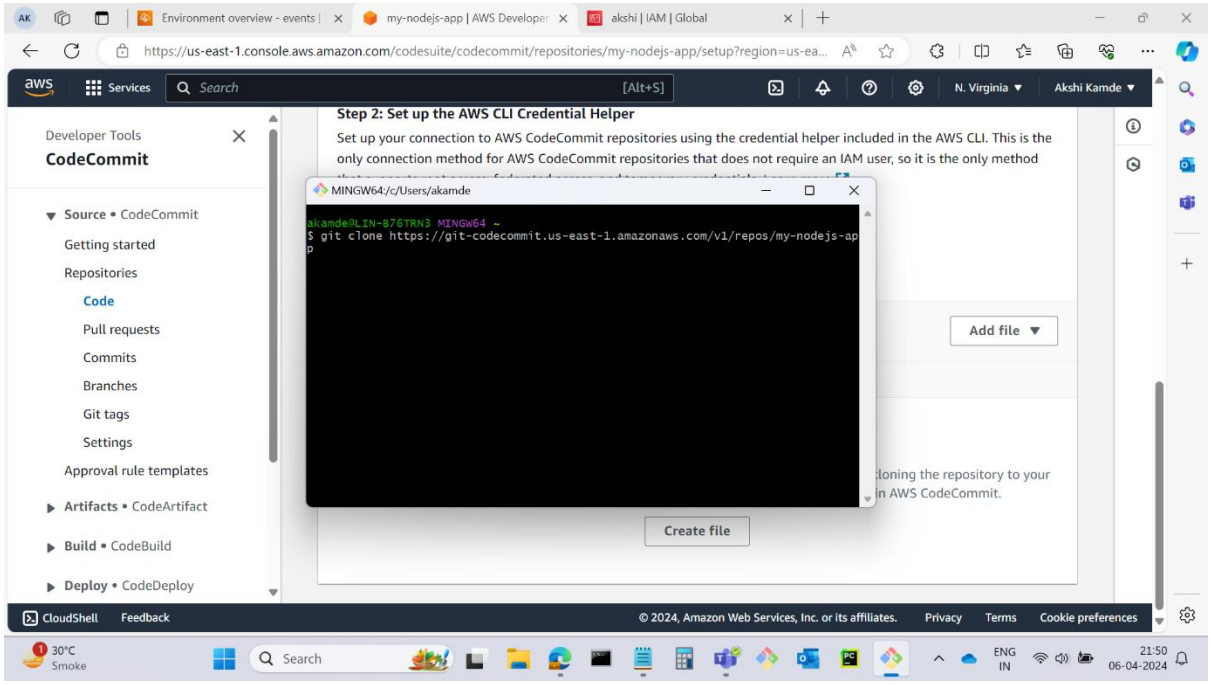


Now switch to the tab where you had create CodeCommit repository and click on Clone URL. Choose Clone HTTPS. URL will get copied.

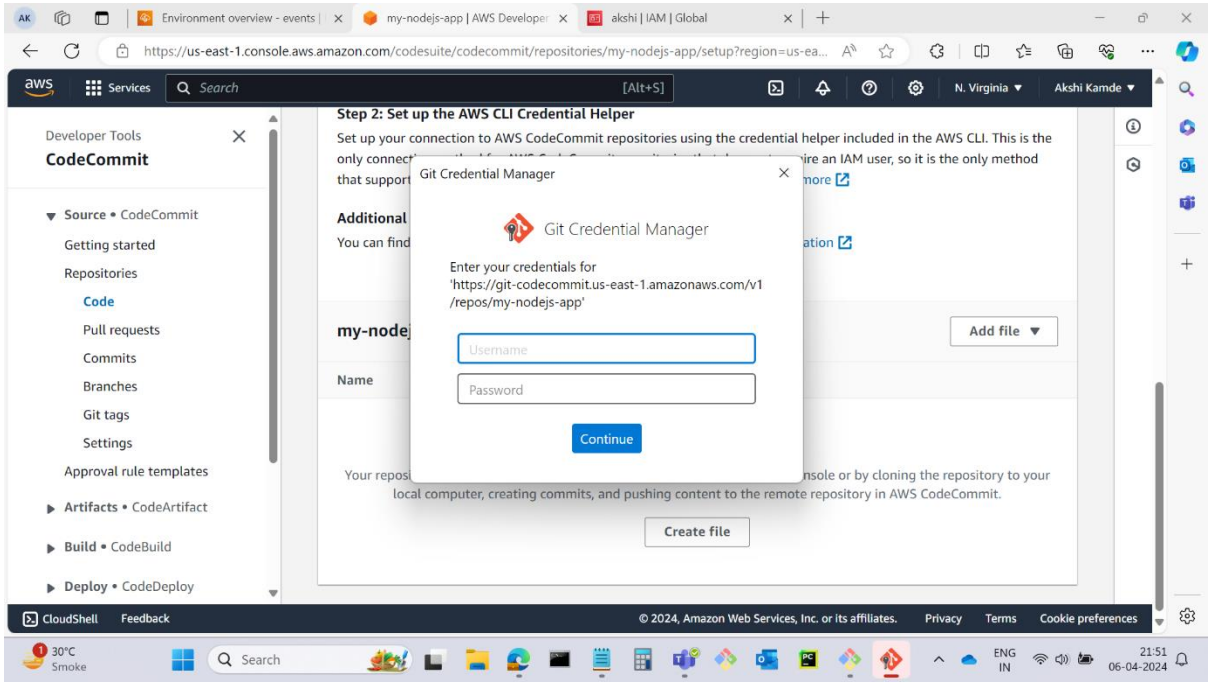


Now open git bash in your local machine and type the following command:

```
git clone <paste the copied URL>
```

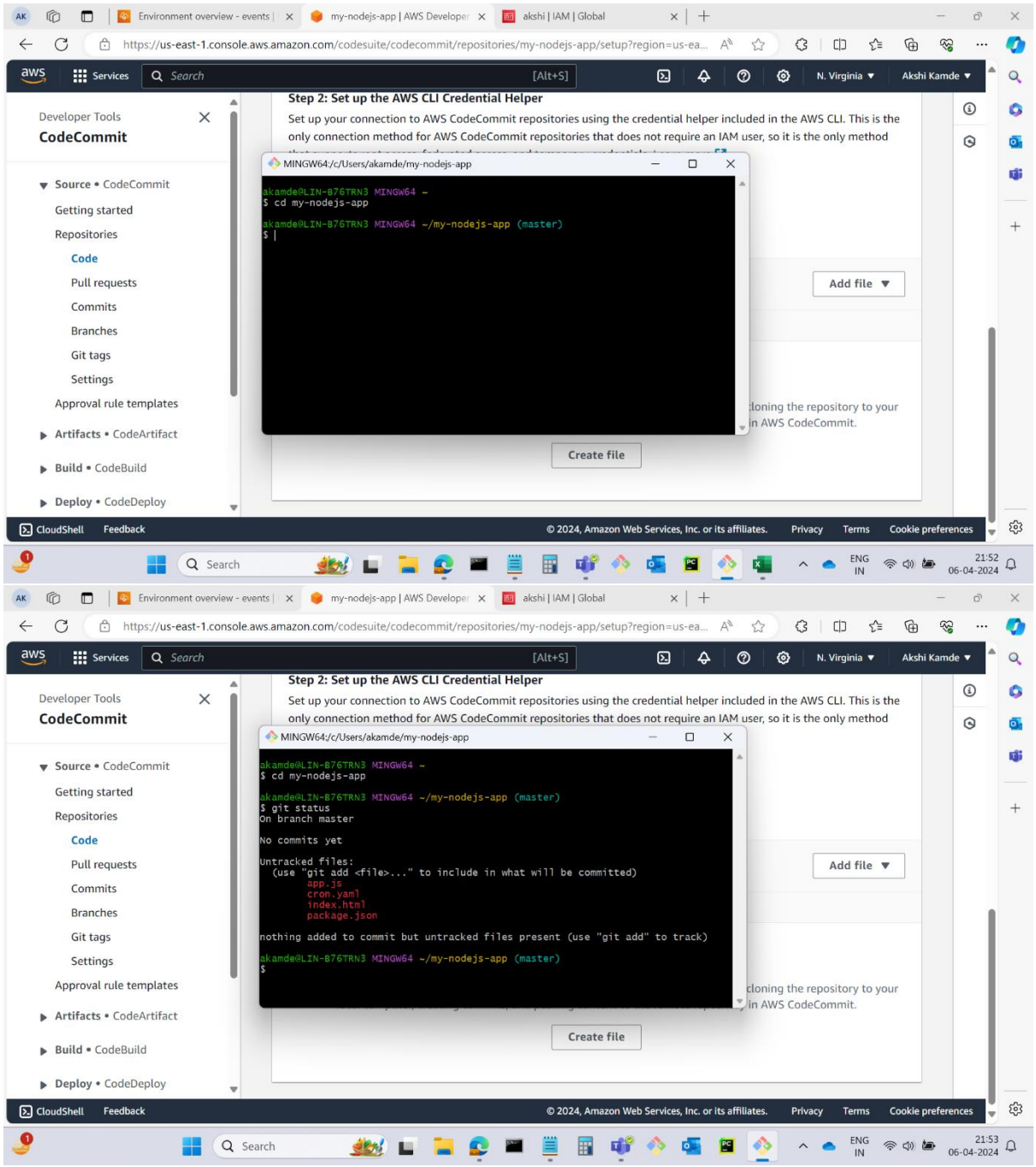


You will be prompted to add the username and password which you got from AWS IAM user. Fill the details and hit continue.

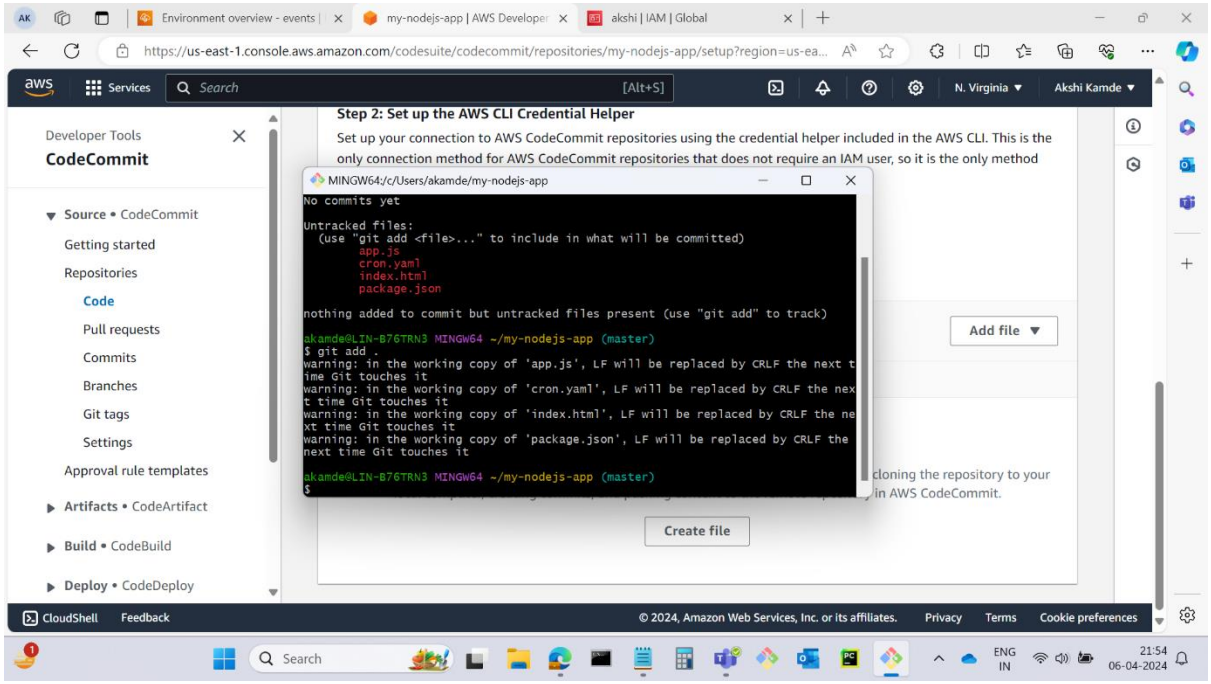


Now cd into the directory/repository name. You will see that you are on the master/main branch. You can either manually copy paste the files or simple use cp command. Then check it using git status

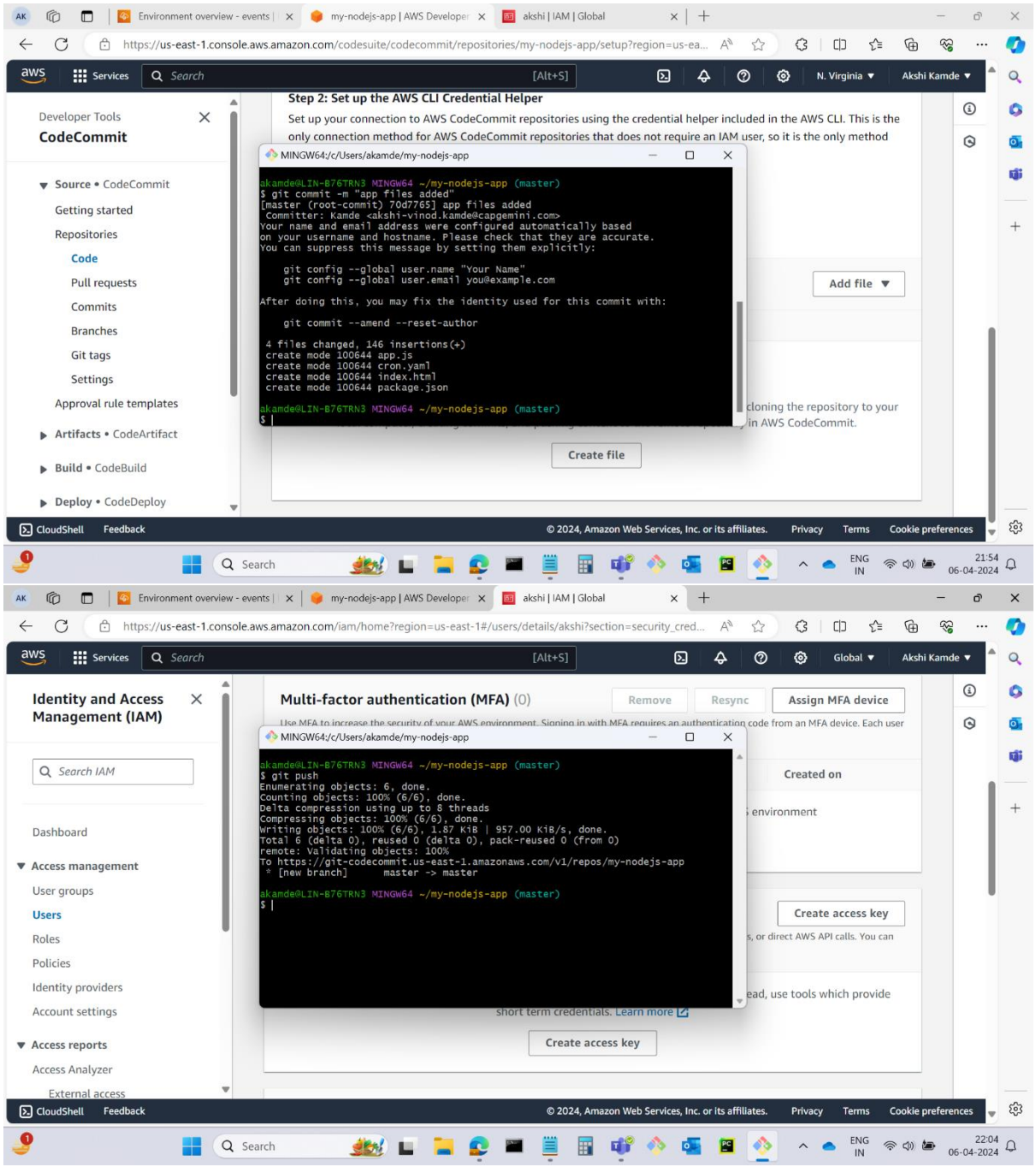
CodePipeline



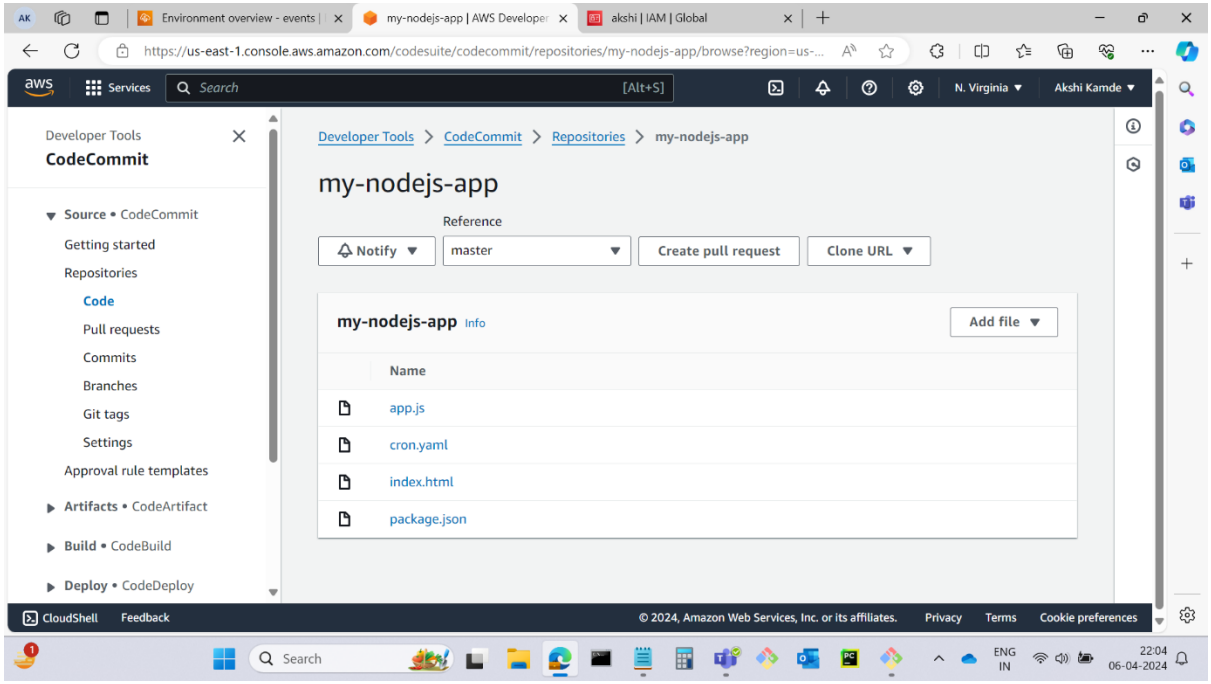
Use commands like git add ., git commit and git push.



CodePipeline

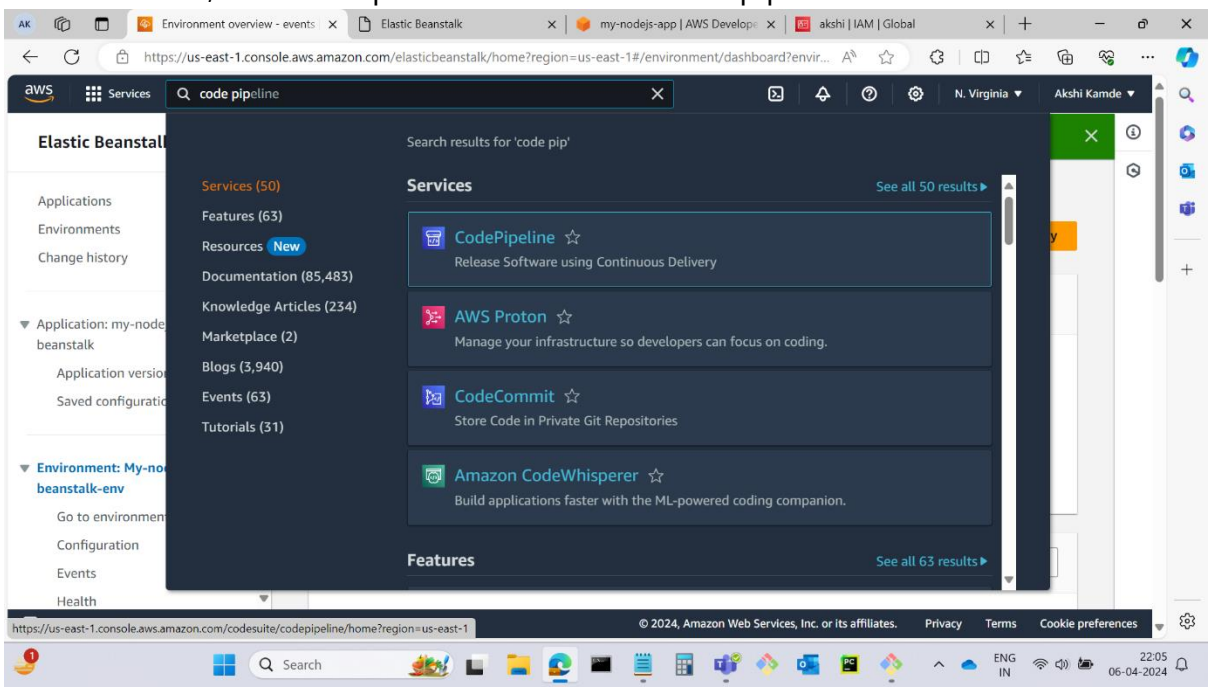


Voila! You can now see all the files on CodeCommit.

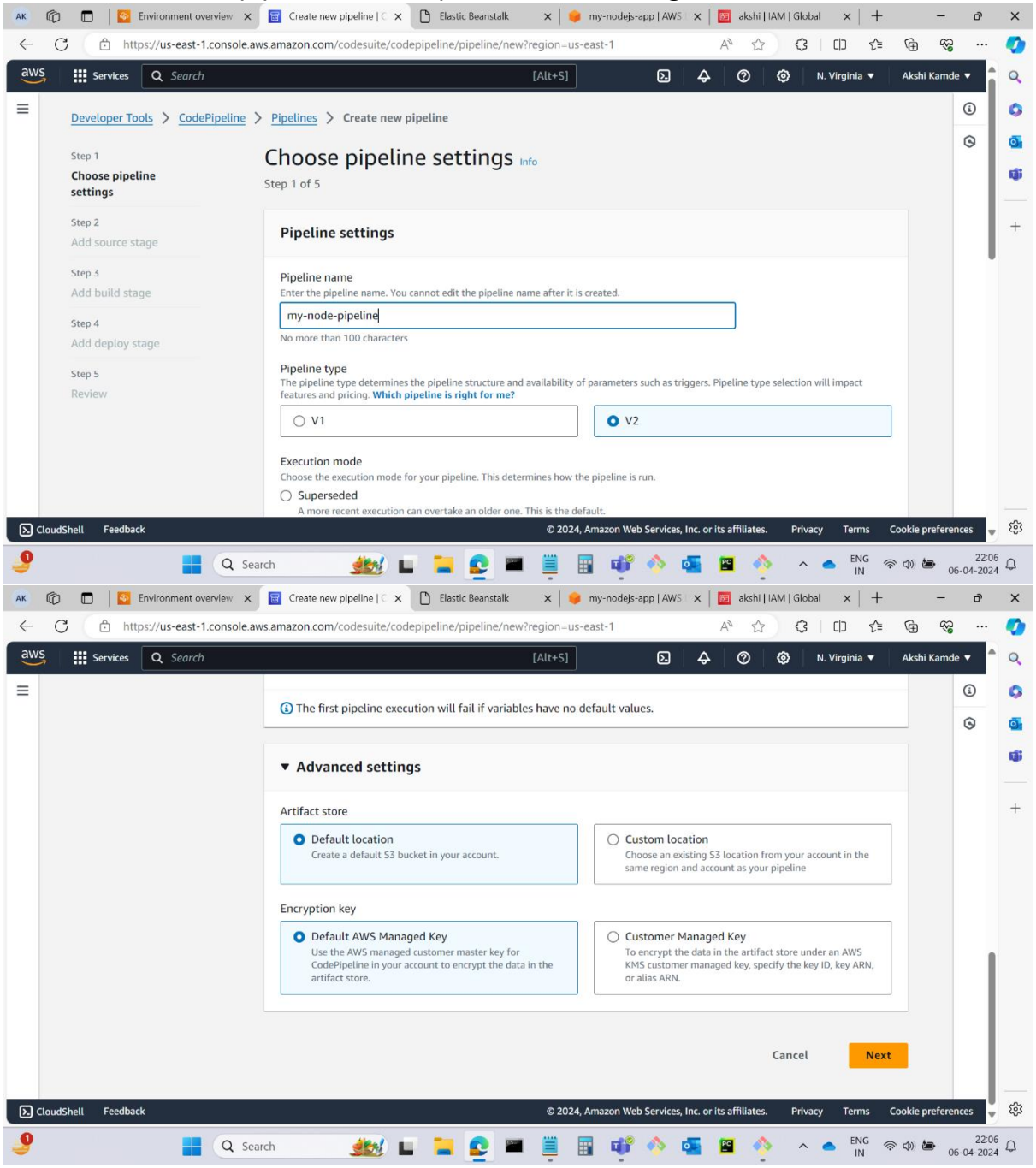


Create a CodePipeline

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

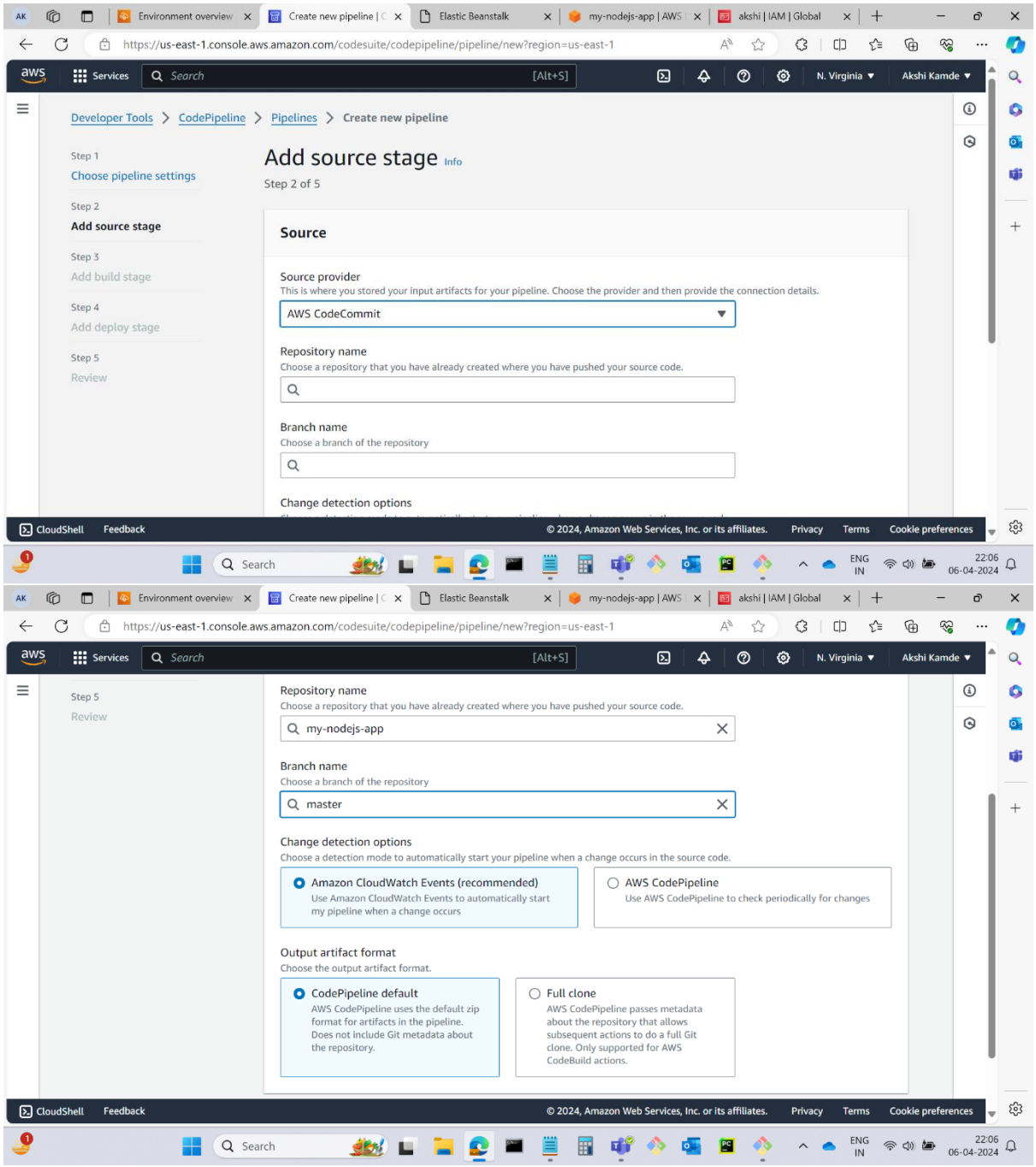


Give a name to the pipeline and keep the default settings.

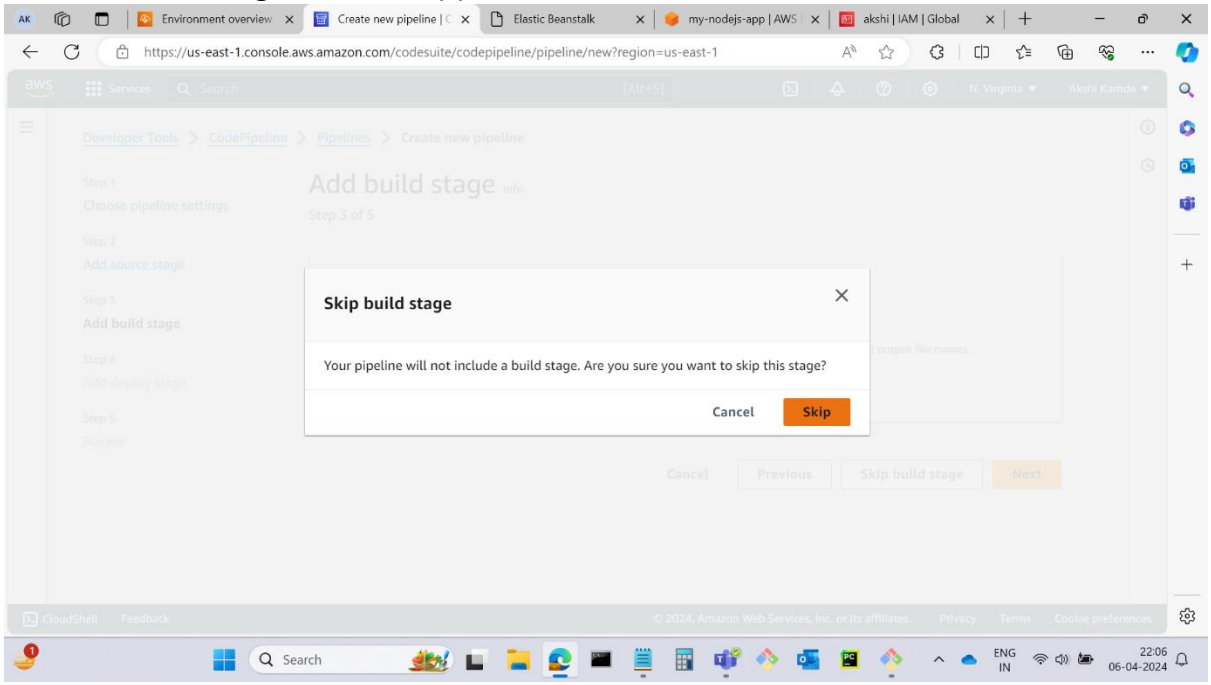


Add source as AWS CodeCommit, select the repository name that was created and branch name. Keep the rest set to default.

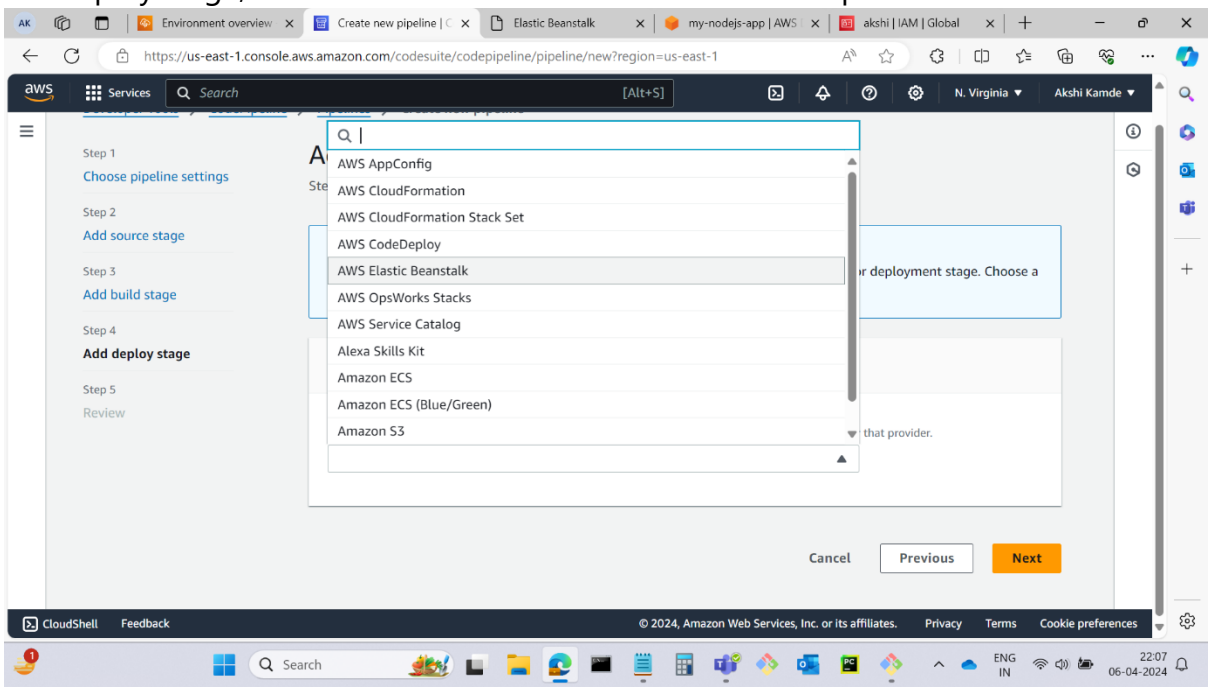
CodePipeline



The next build stage can be skipped.

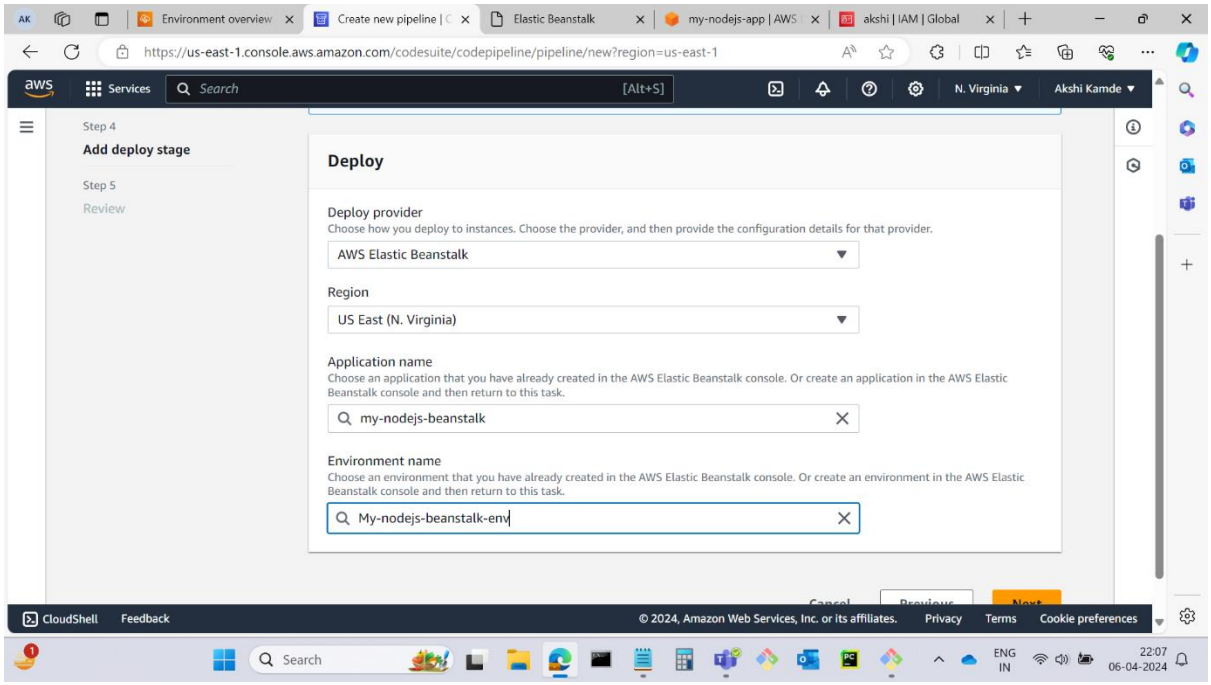


For deploy stage, search for AWS Elastic Beanstalk from dropdown menu.

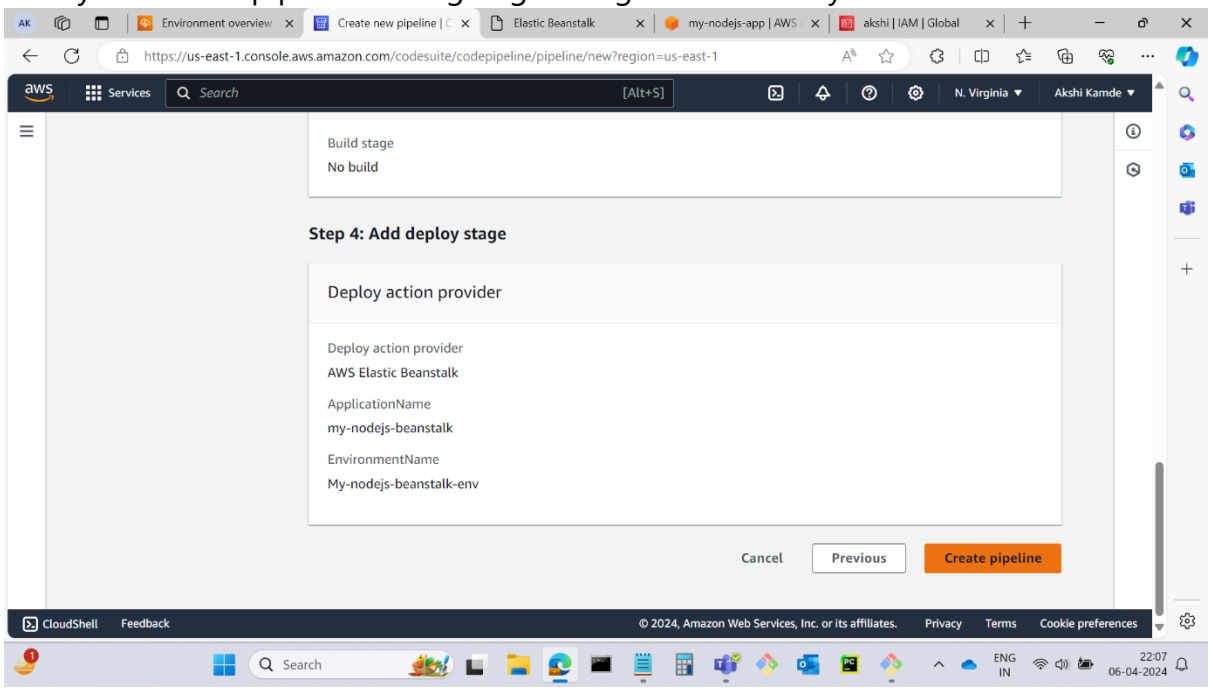


Region will be selected by default, make sure it matches your current region, add the name of the apllication from dropdown. Also select the environment of the application. Currently we have only built one env, if we had built production env then

we could have selected that too.

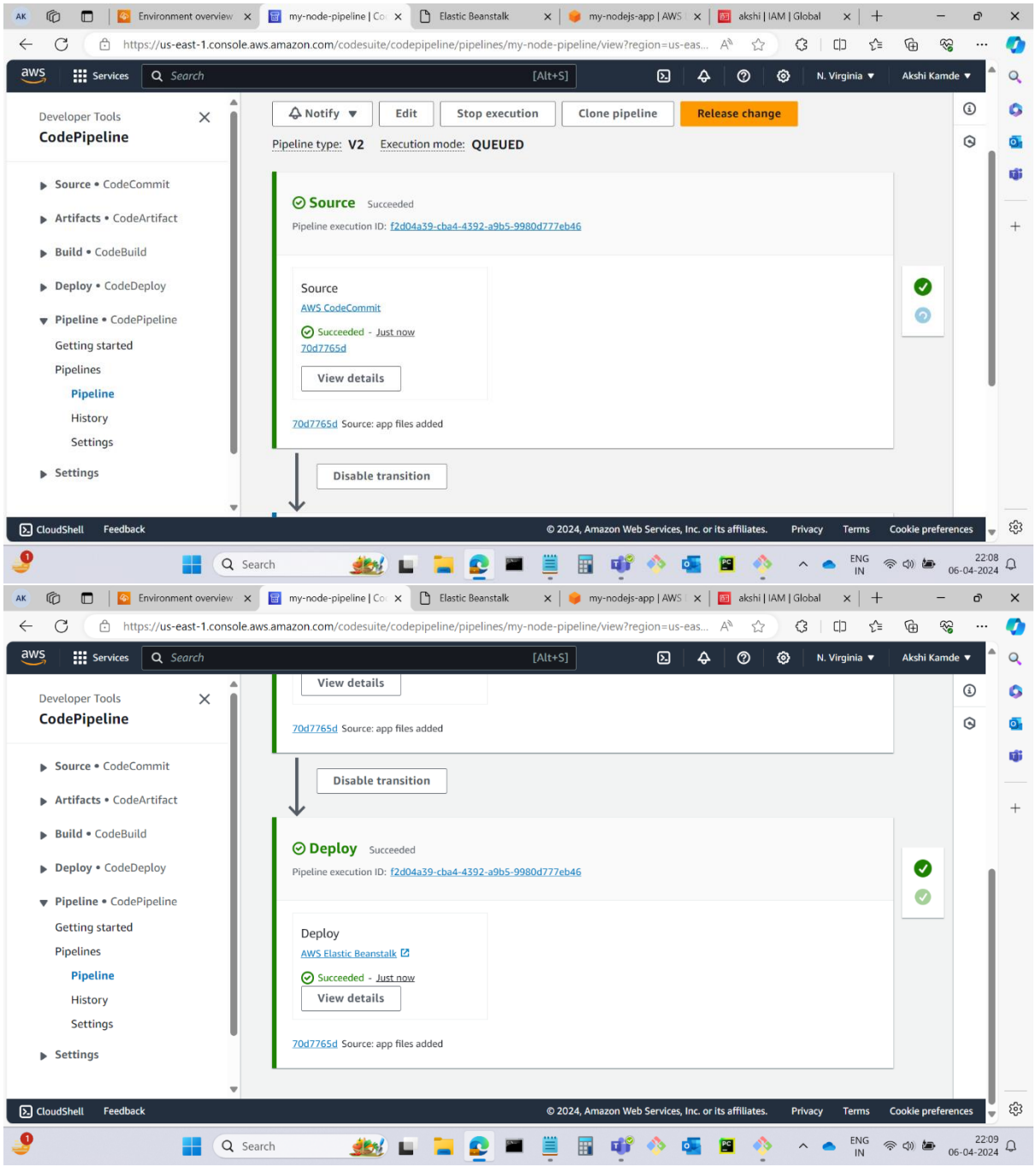


Finally create the pipeline after going through the summary.



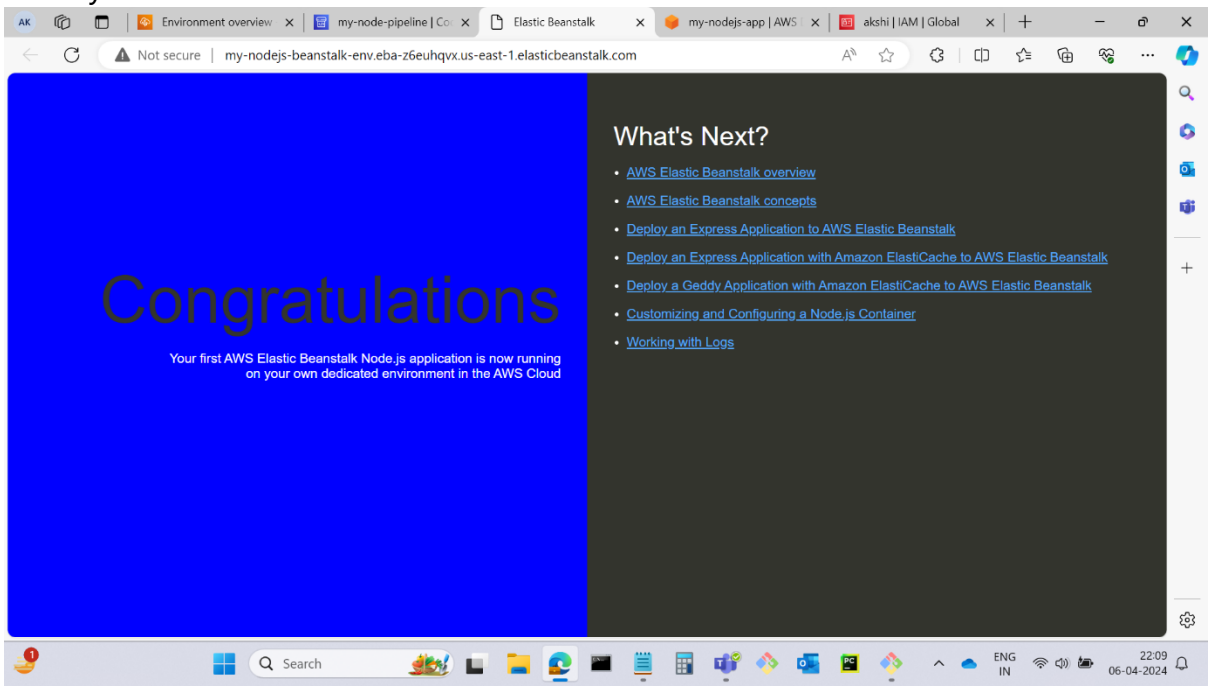
In few minutes you would see that your pipeline is running successfully and application is also deployed. You can click on the link provided for elastic beanstalk from the deploy stage. From here you can check the health of your environment and check for the domain link where your application is running.

CodePipeline

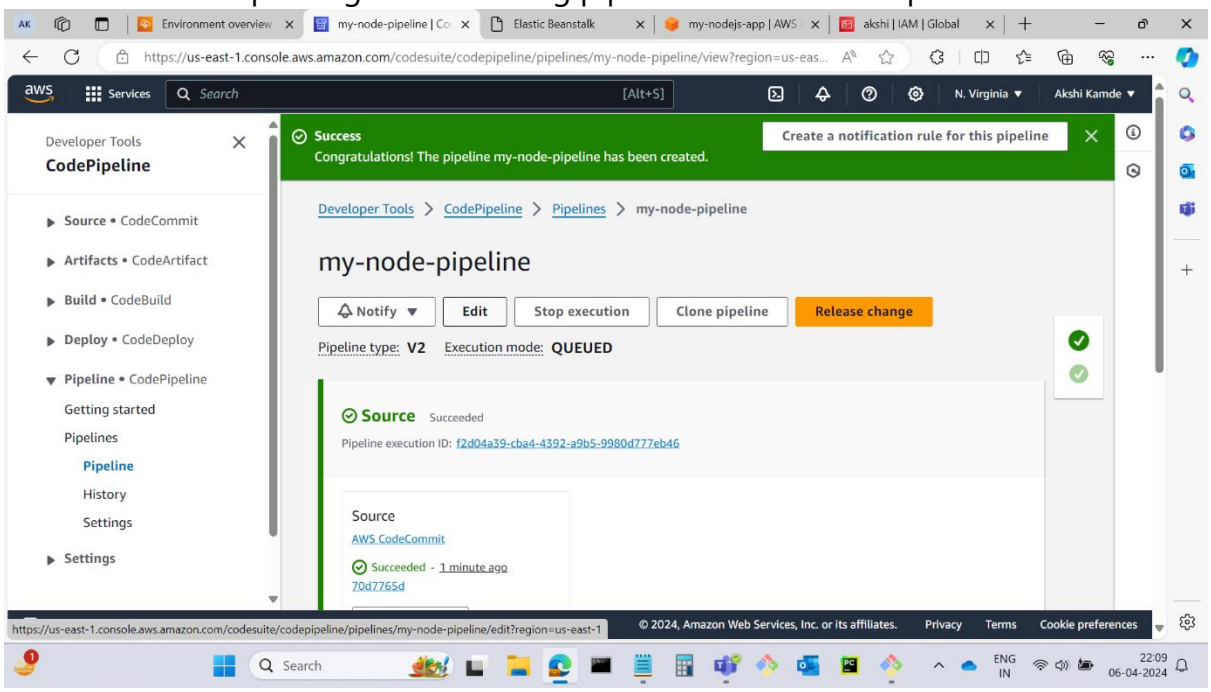


CodePipeline

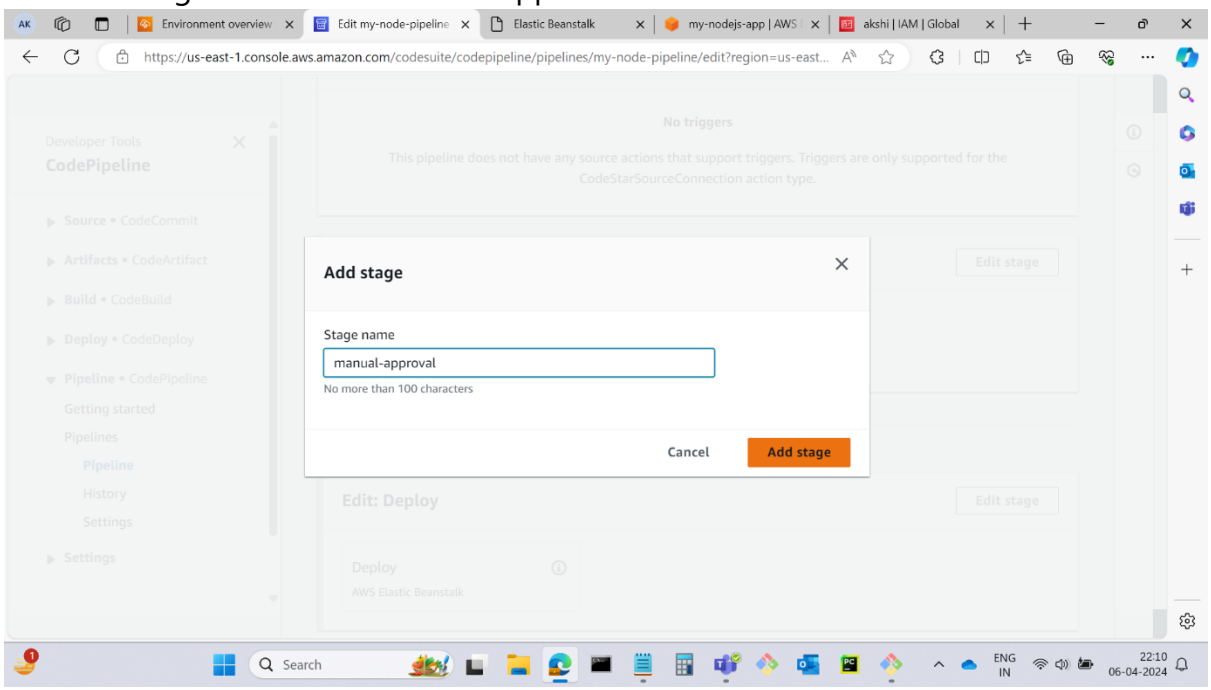
The code changes the color of background from green to blue and that can be seen when you view the site.



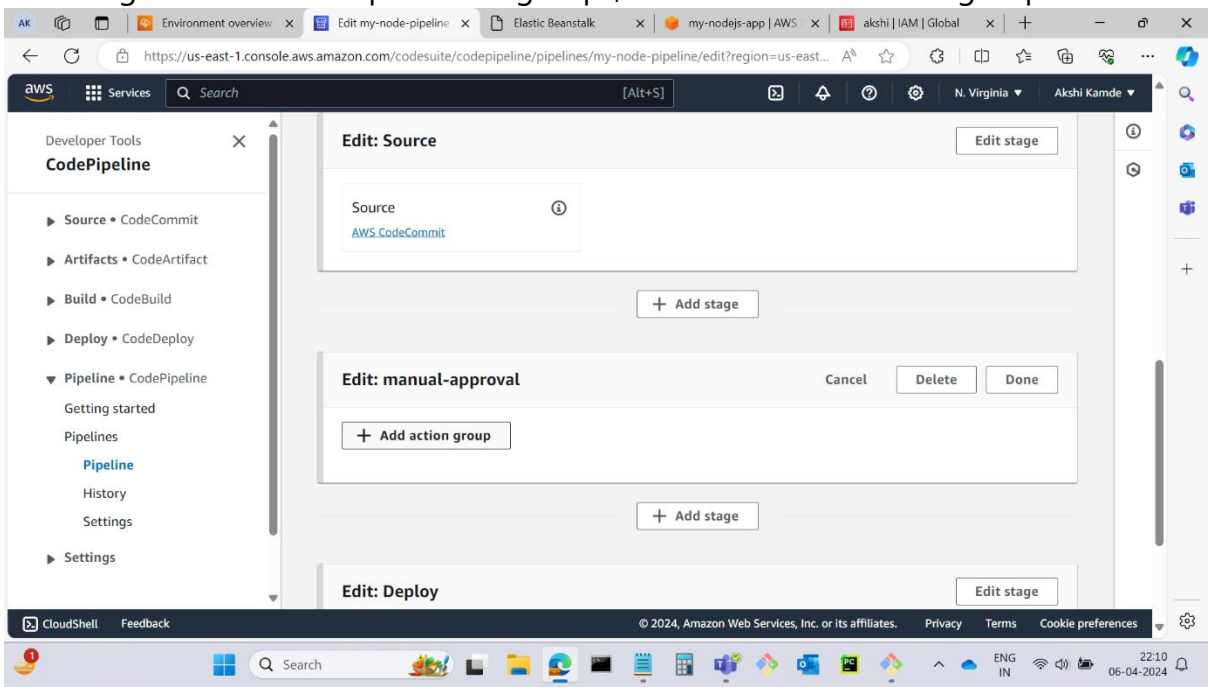
We can add multiple stages in a running pipeline. Select “edit” option.



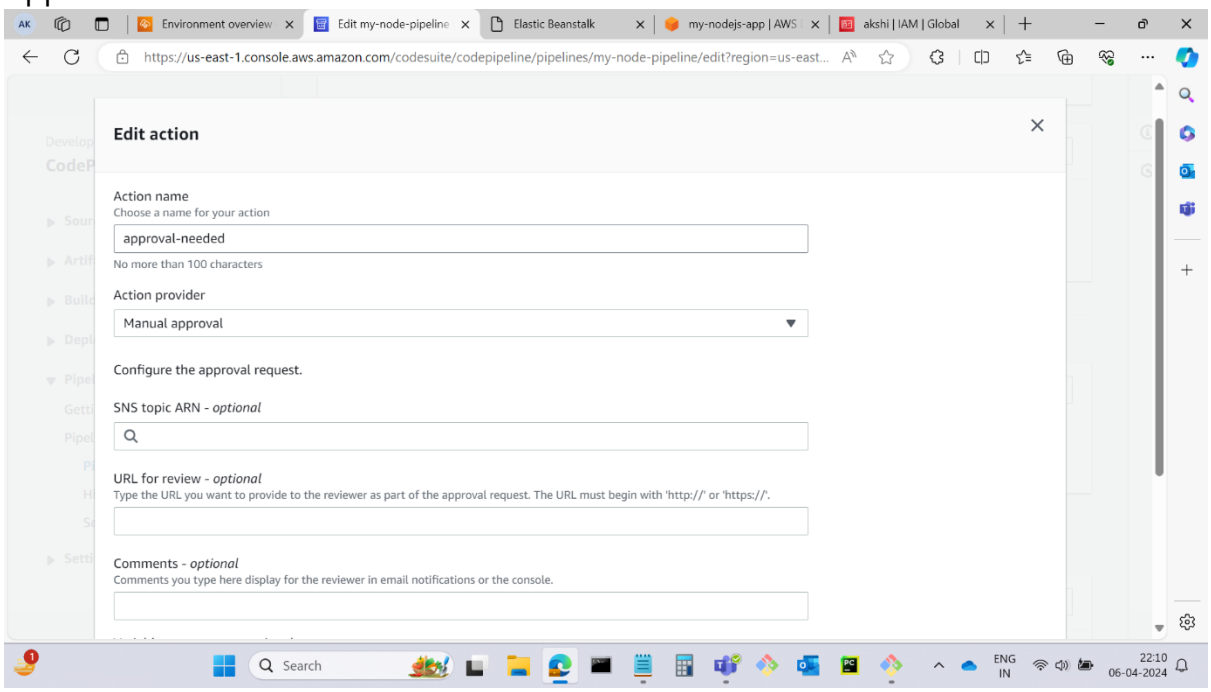
Give this stage a name as “manual-approval”



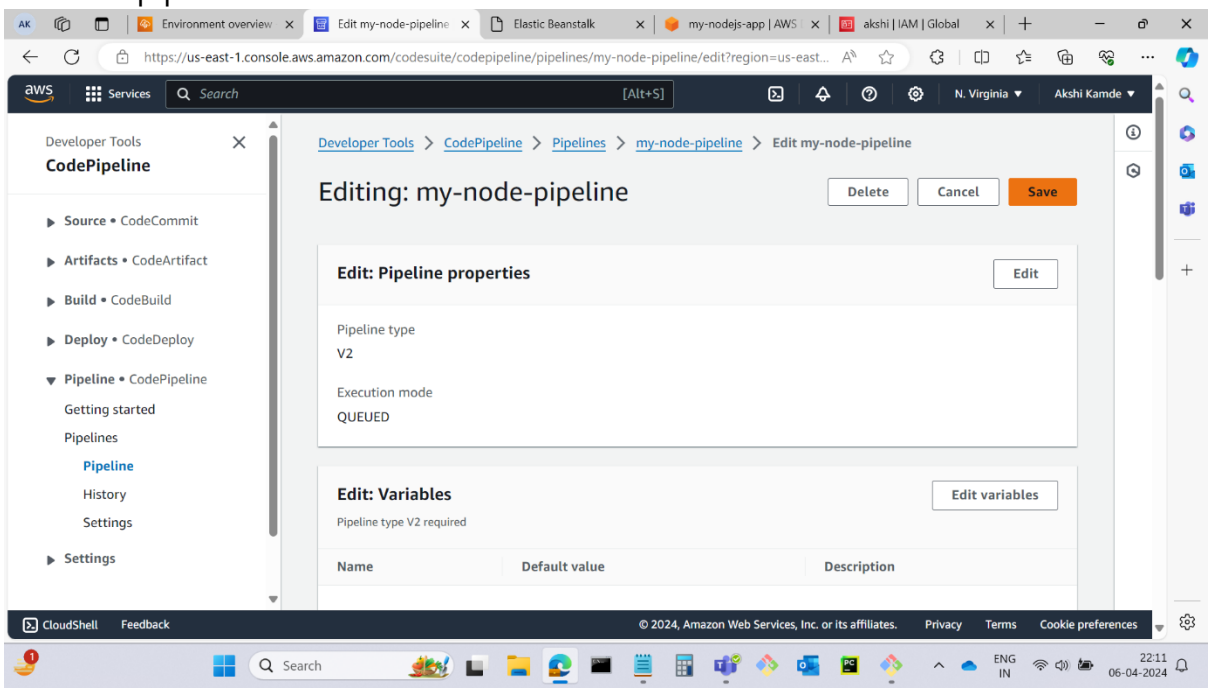
Each stage can have multiple action groups, so click on “add action group”.



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

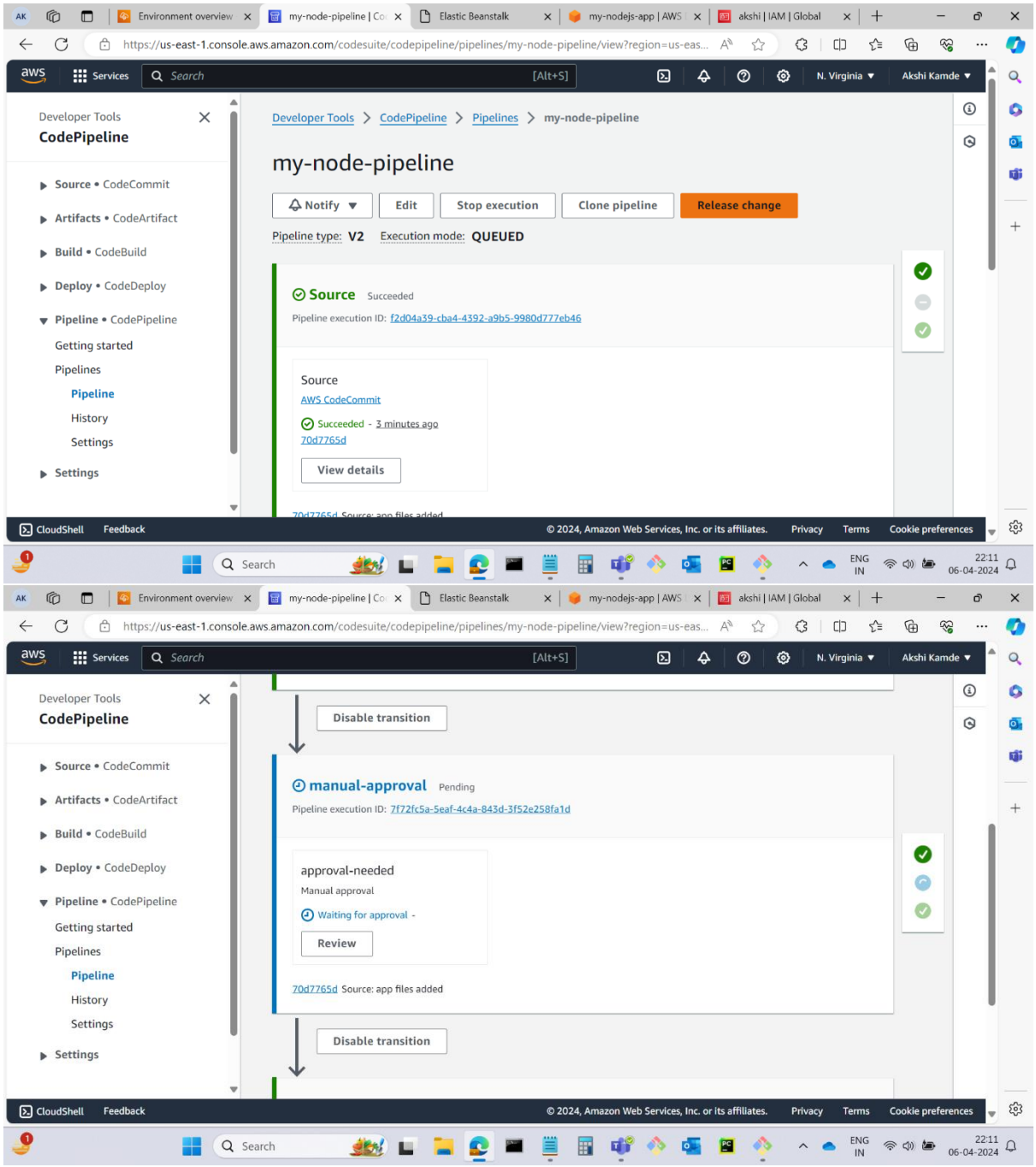


Save this pipeline.

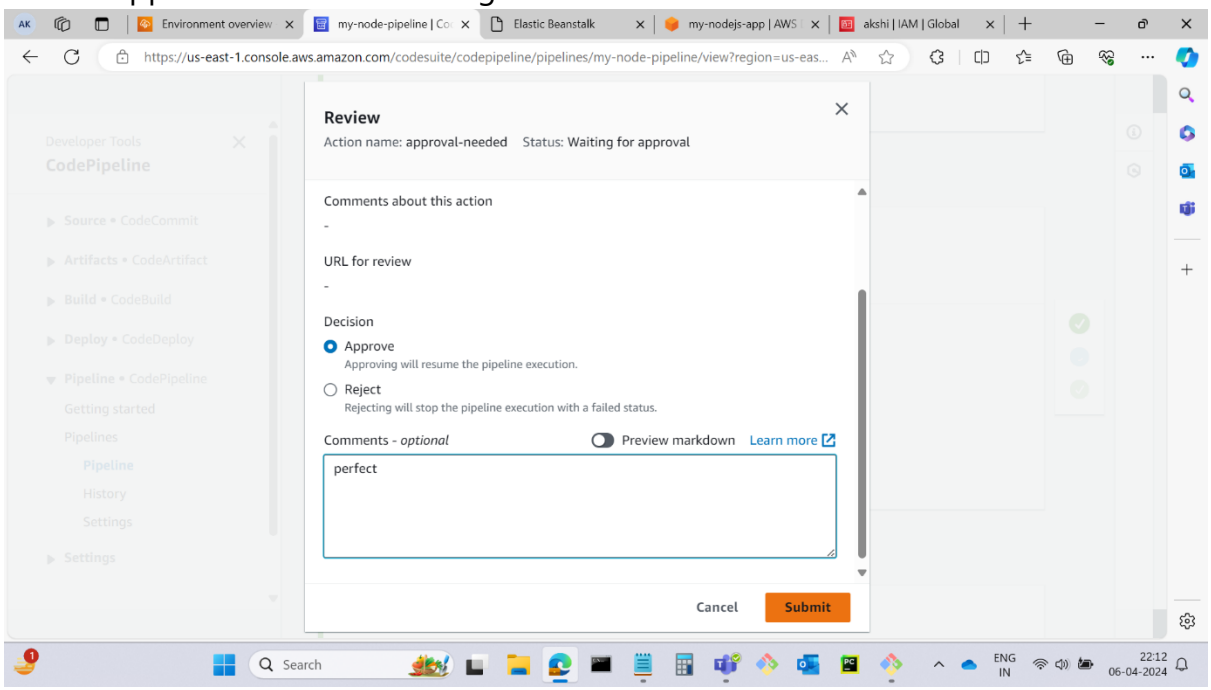


Click on “Release Change”. You will notice that application now pauses at the second stage, waiting for an approval. Click on “Review”

CodePipeline



Select “Approve” and write message. Click on “Submit”



You can also stop a pipeline.