Sneha Patra **D15A 41**

Advance Devops Case Study Report

Cloud Deployment with Automation

Concepts Used:

- AWS CodePipeline
- EC2
- S3 Problem

Statement:

Build a simple web application using AWS CodeBuild and deploy it to an S3 bucket. Then, automate the deployment process using AWS CodePipeline, ensuring the application is deployed on an EC2 instance. A sample index.html page will be used for demonstration.

Tasks:

1. Set up AWS CodeBuild for the web app.
2. Create a pipeline that deploys the web app to an S3 bucket.
3. Use AWS CodeDeploy to push updates to an EC2 instance.

1. Introduction Case Study Overview:

This case study focuses on building a simple web application and automating its deployment using a combination of AWS services— AWS CodeBuild, S3, CodePipeline, and CodeDeploy. The task is to create a basic application, package it, and deploy it to an S3 bucket as the initial step. Afterward, AWS CodePipeline is used to automate the deployment, ensuring updates are automatically pushed to an EC2 instance using AWS CodeDeploy. The goal is to demonstrate a seamless, automated deployment pipeline for a web application, which simplifies continuous integration and delivery (CI/CD) processes.

Key Feature and Application:

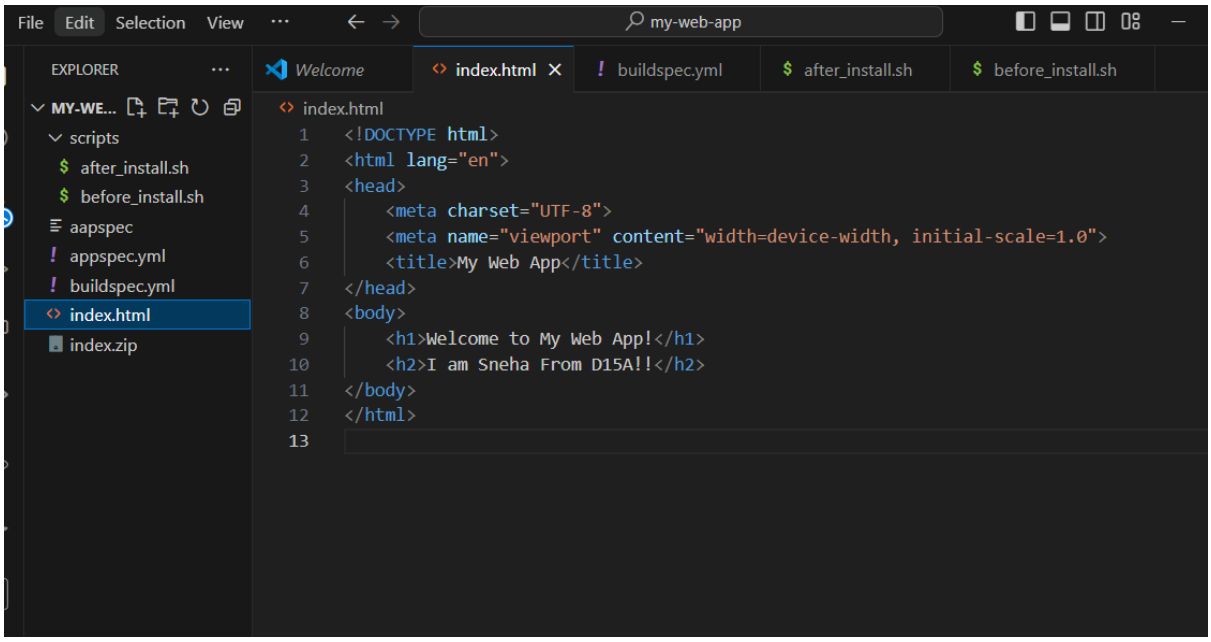
The unique feature of this case study is the integration of multiple AWS services to create a fully automated deployment pipeline. By using AWS

CodeBuild for compiling and packaging the web app, AWS S3 as the storage for the deployed app, and AWS CodePipeline for automating the process, the deployment becomes efficient and scalable. Additionally, CodeDeploy ensures that the web application can be easily pushed and updated on an EC2 instance, facilitating fast iteration and real-time updates to the deployed application. This automation greatly reduces the manual workload involved in deployment and allows for continuous delivery, making it highly practical for modern web applications that need frequent updates.

PROCEDURE & SCREENSHOTS:

Create a Simple Web App

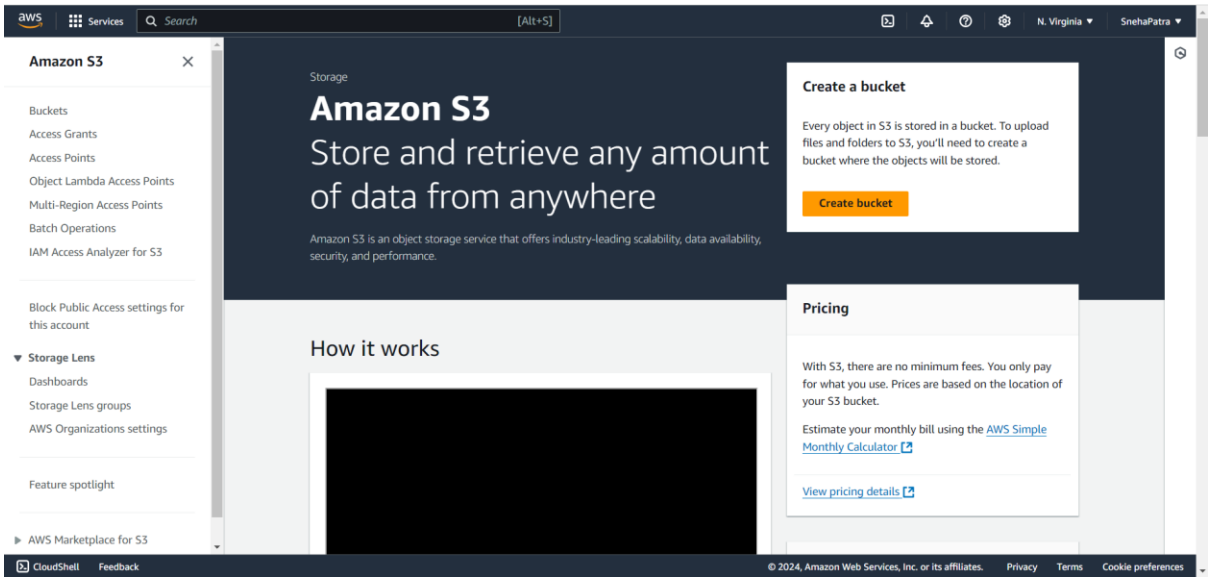
1. First, create a simple web app with an index.html file:



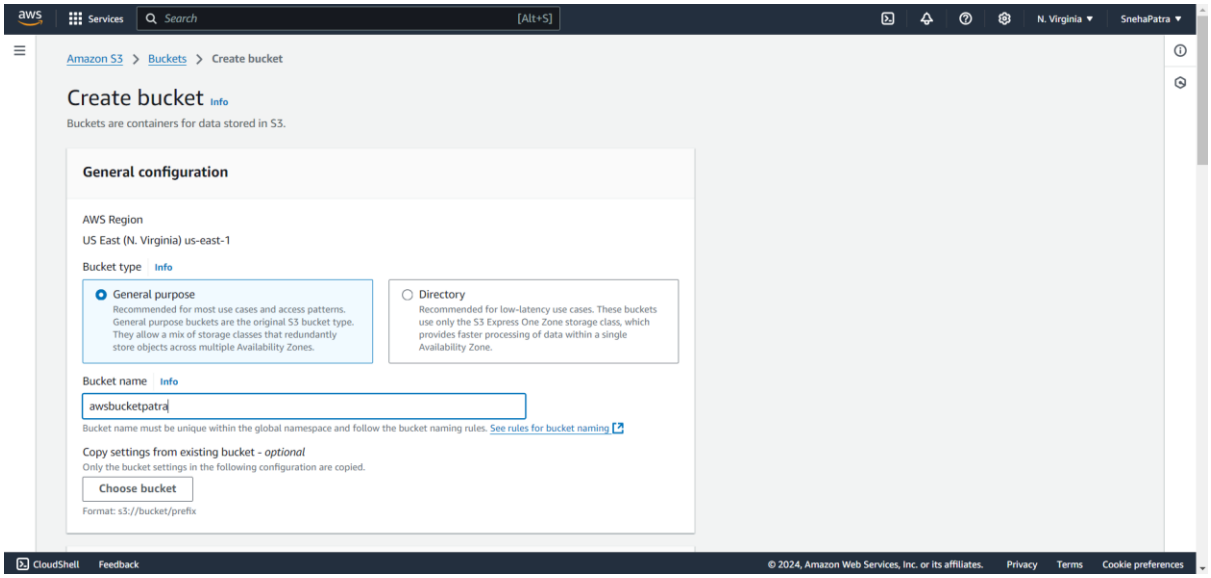
This file will serve as the web page deployed to your S3 bucket and later to the EC2 instance.

2. Set Up S3 Bucket for Web App Hosting

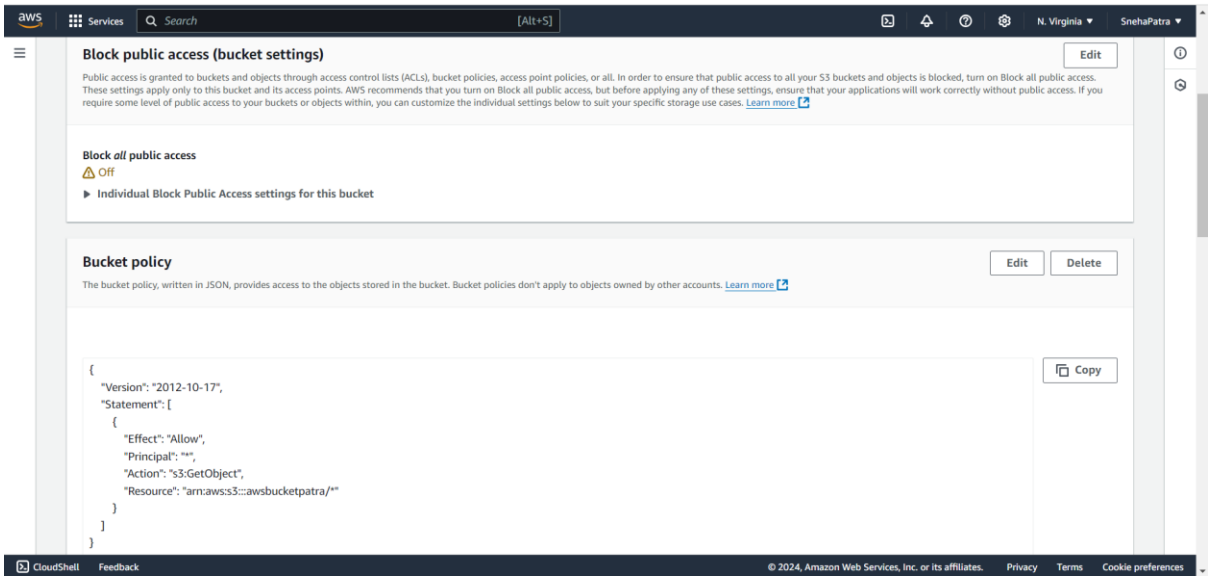
1. **Go to the AWS S3 Console**:
    - Open the S3 console.



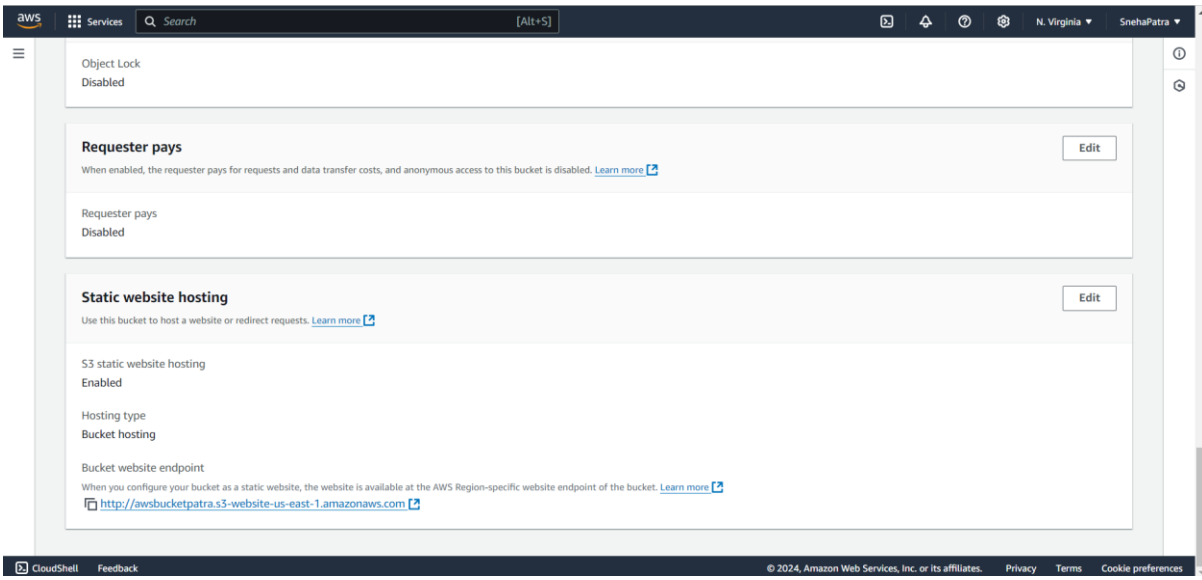
- Create a new S3 bucket, giving it a unique name (e.g., my-s3-web-bucket).



- Under **Permissions**, uncheck the "Block all public access" option, allowing public access for web hosting.

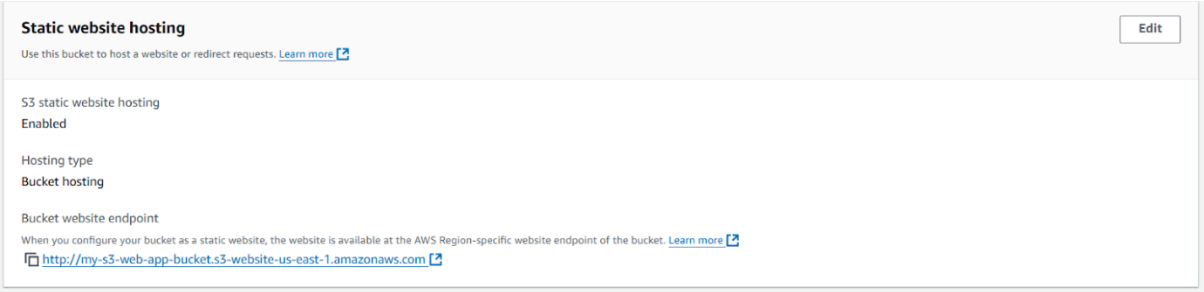


2. **Configure the Bucket for Website Hosting**:
   - Go to the **Properties** tab of your S3 bucket.
   - Scroll down to **Static website hosting**.



   - Enable it, and set the **Index document** as index.html.

- Copy the bucket website URL for testing the web app later.



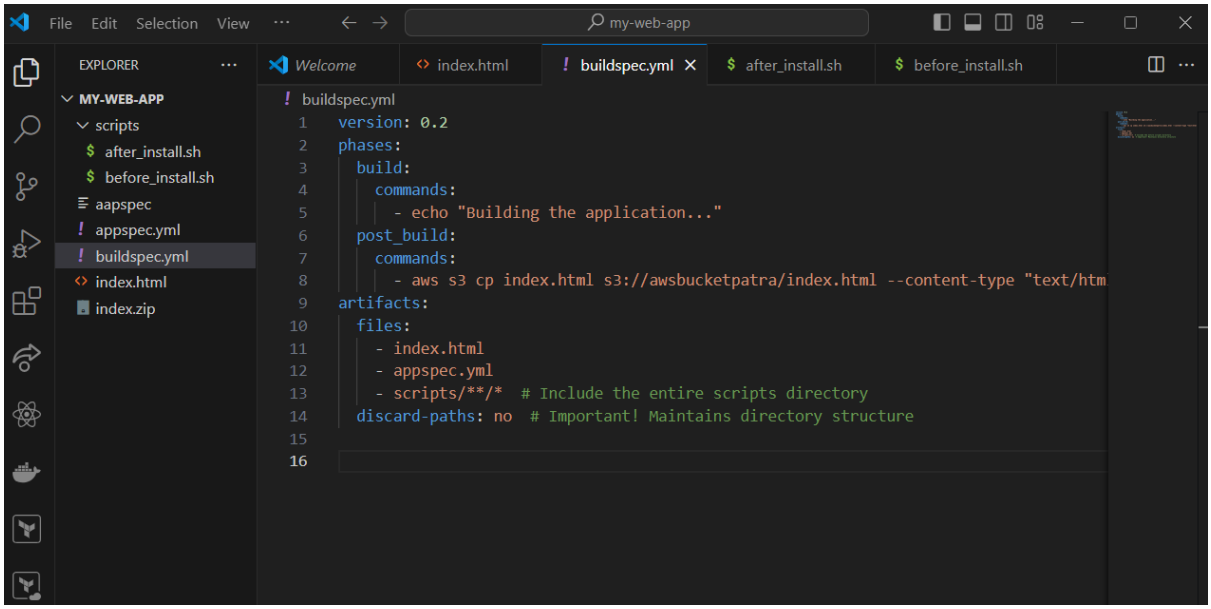
  in my case it is: http://awsbucketpatra.s3-website-us-east-1.amazonaws.com

- if s3 website shows 403 forbidden, its a IAM permission issue.

---

3. Set Up CodeBuild for Your Web App

Create a Buildspec File: In your project directory (where index.html resides), create a buildspec.yml file. This file tells AWS CodeBuild what to do during the build.

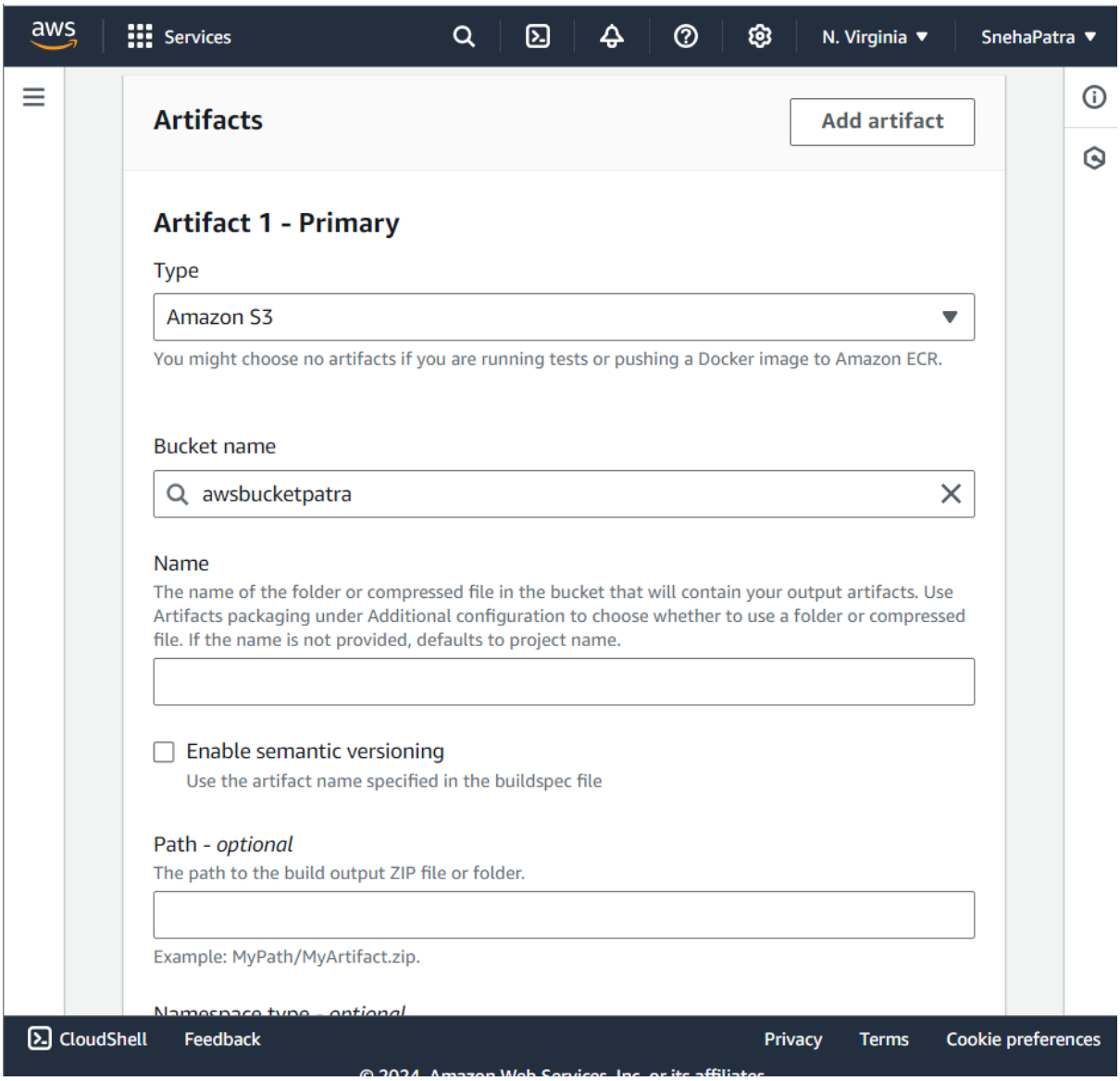


Go to AWS CodeBuild:

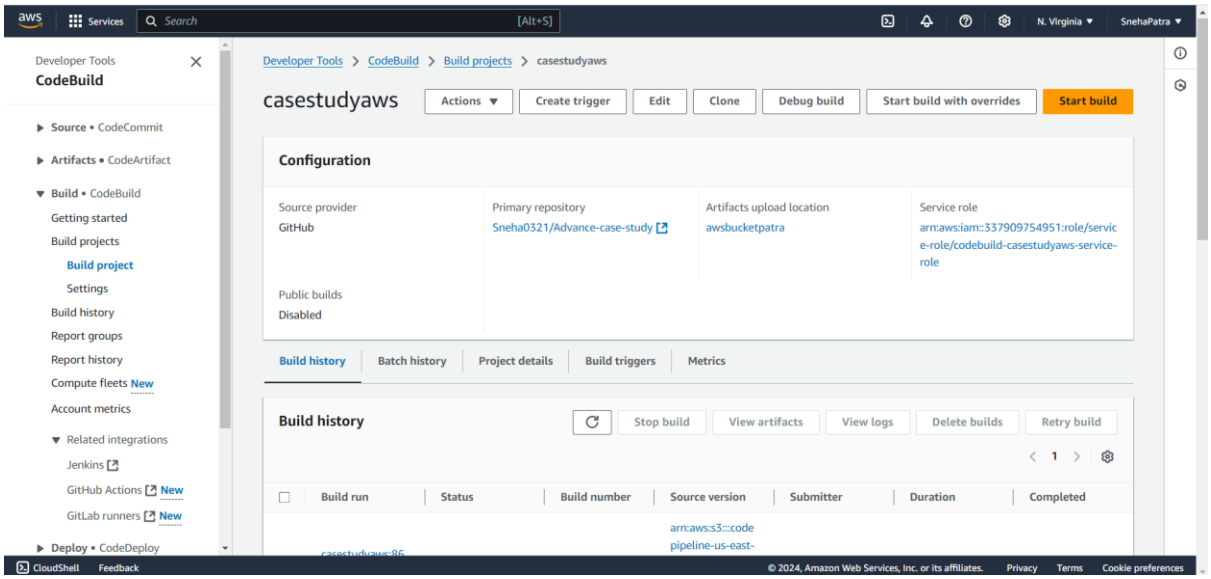
- Open the AWS CodeBuild console.
- Create a new build project.

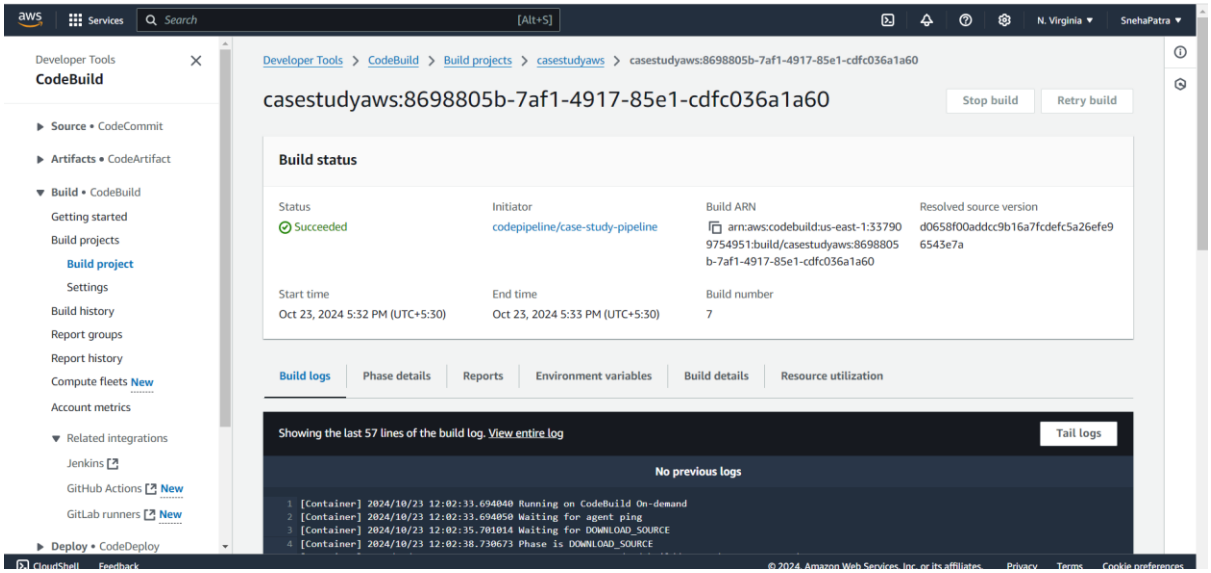
- For **Source**, choose your source repository (e.g., GitHub, Bitbucket, or S3).

-
- specify the buildspec.yml file you created. before this add it to your git repo
- Set **Artifacts** to "S3", and choose the bucket you created earlier.

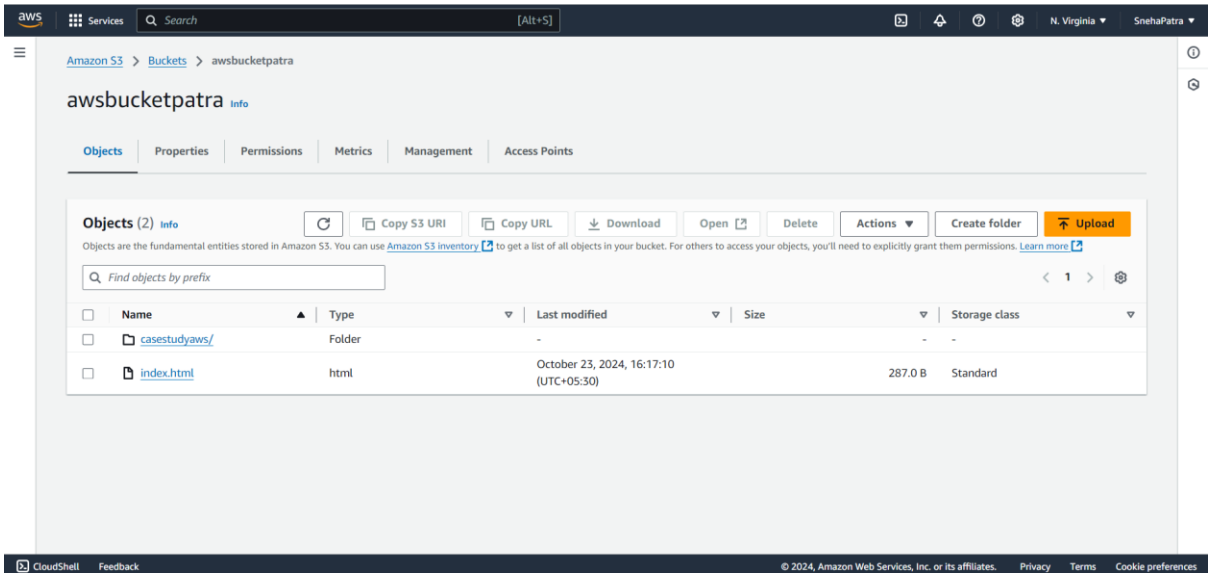


- Create the build project and start the build to ensure it uploads index.html to the S3 bucket.



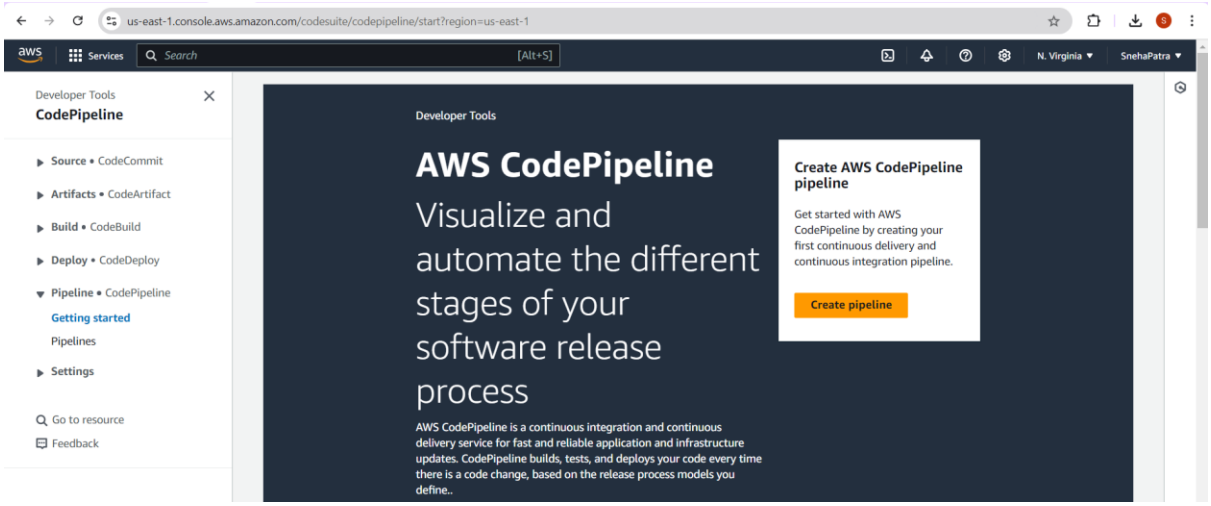


s3 bucket was updated:



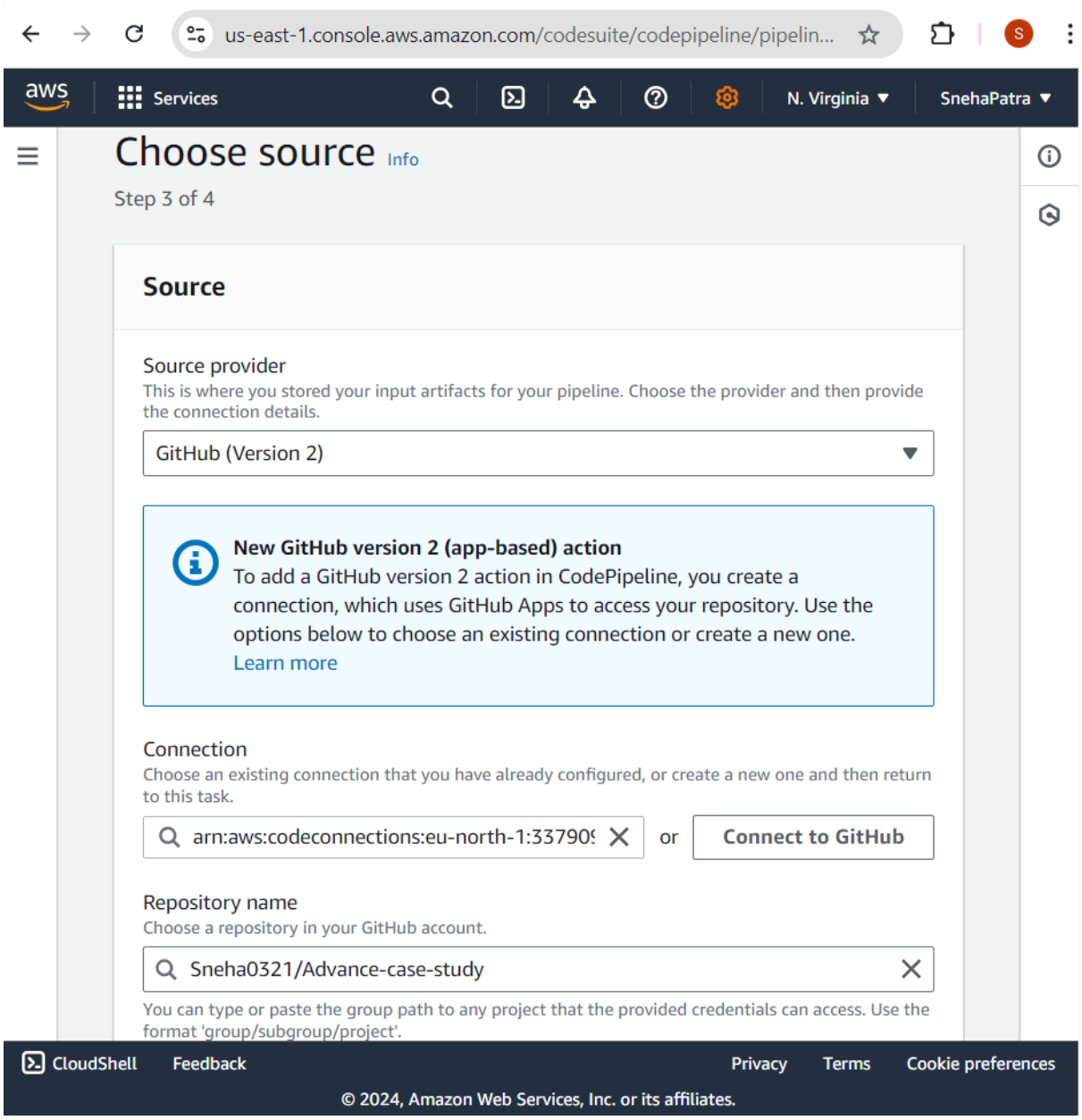
4. Set Up AWS CodePipeline

1. **Go to AWS CodePipeline**:
   - Open the [CodePipeline console](#).



   - Create a new pipeline.

   - For **Source**:

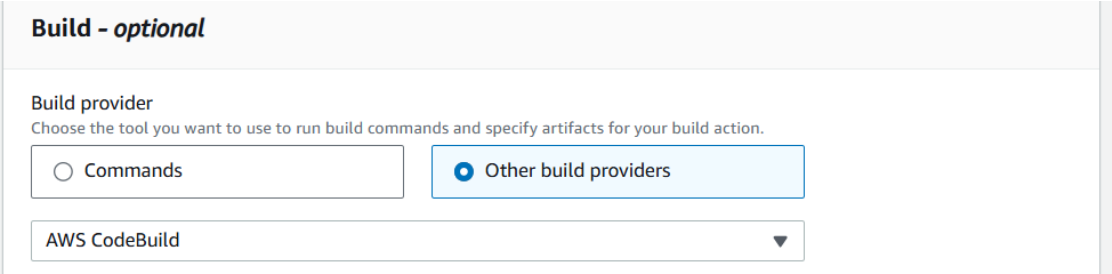


- Select your repository (e.g., GitHub, Bitbucket).

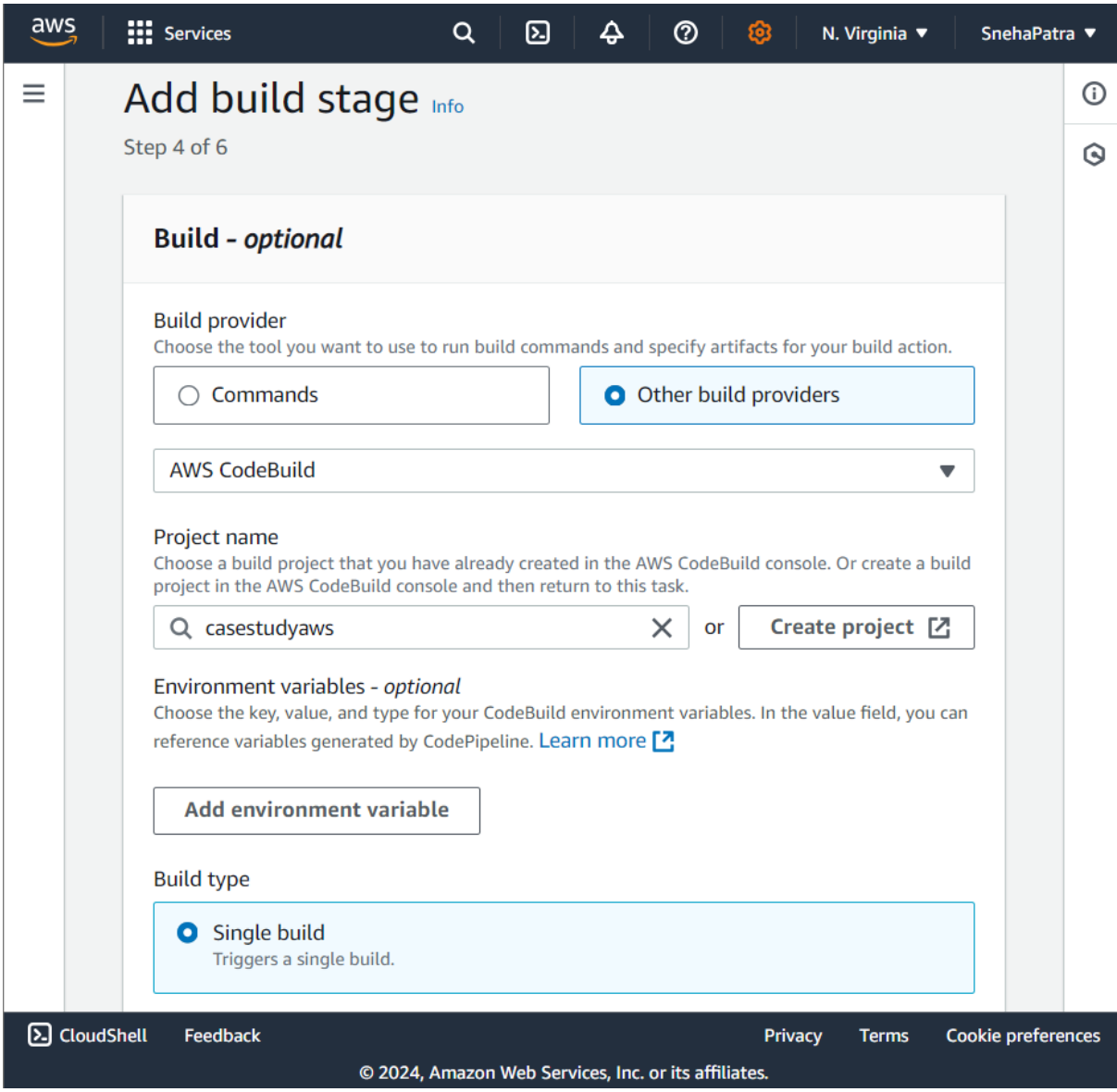
- Connect and choose the appropriate branch where the index.html and buildspec.yml files are.

2. **Add Build Stage**:

- In the **Build stage**, choose **AWS CodeBuild** as the build provider.



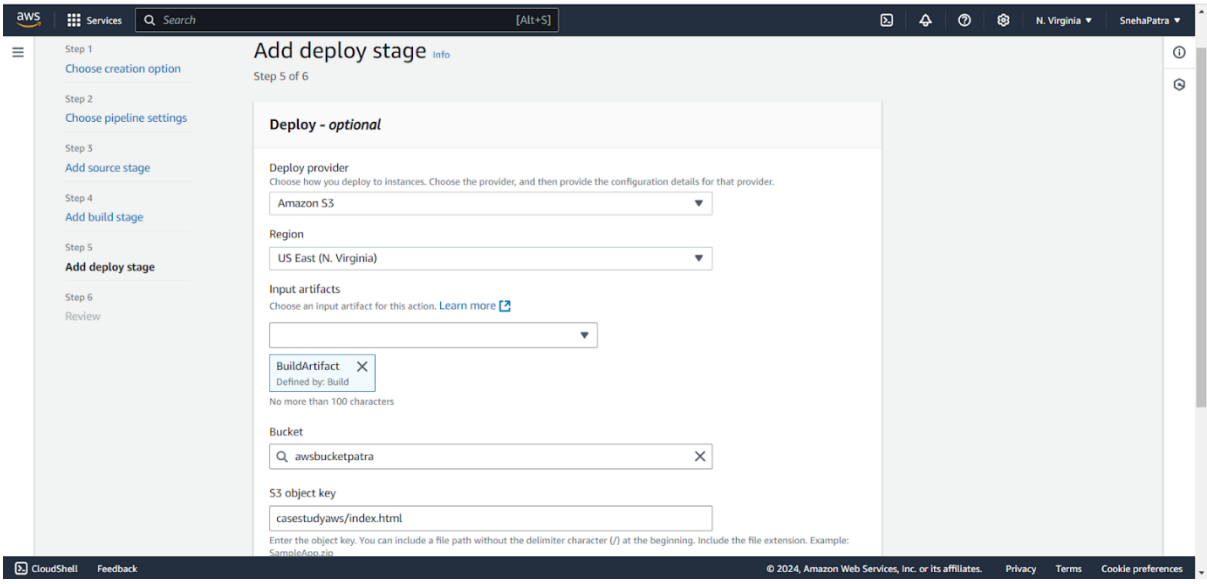
- Select the CodeBuild project you created earlier.



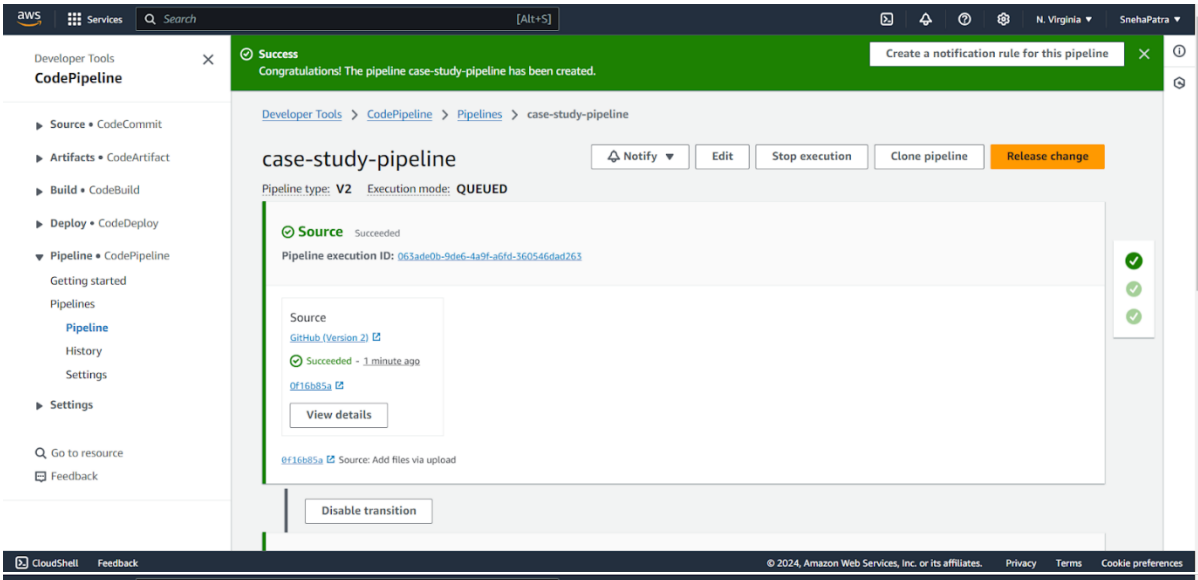
3. **Deploy to S3**:

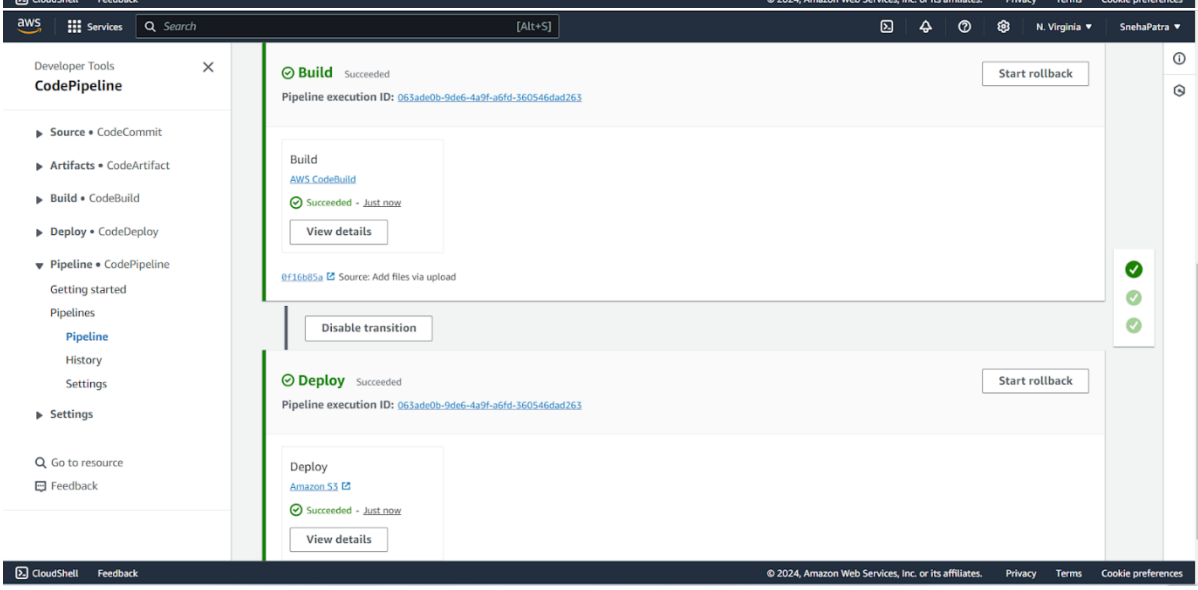
- In the next stage, choose **Deploy**. Select **Amazon S3**. Choose your S3 bucket (my-s3-bucket) where the index.html file will be

deployed.

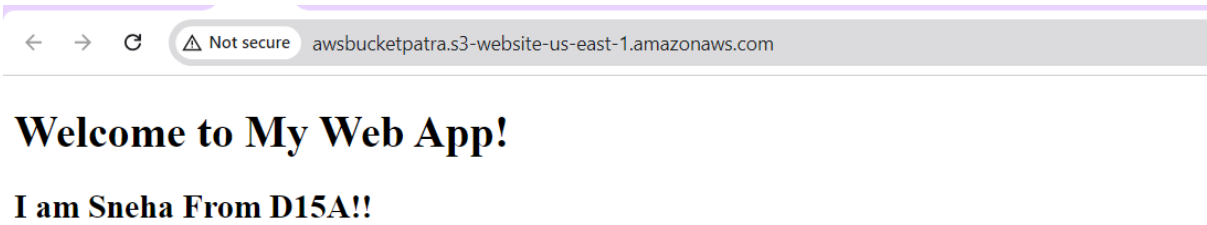


4. **Test the Pipeline**:
    - Once the pipeline is set up, click **Release Change** to start the pipeline. This should fetch the latest code, build it, and upload index.html to the S3 bucket.


aws
Services
Search
[Alt+S]
N. Virginia
SnehaPatra
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Success
Congratulations! The pipeline case-study-pipeline has been created.
Create a notification rule for this pipeline
Developer Tools > CodePipeline > Pipelines > case-study-pipeline
case-study-pipeline
Notify
Edit
Stop execution
Clone pipeline
Release change
Pipeline type: V2
Execution mode: QUEUED
Source
Succeeded
Pipeline execution ID: 063ade0b-9de6-4a9f-a6fd-360546dad263
Source
GitHub (Version 2)
Succeeded - 1 minute ago
0f16b85a
View details
0f16b85a Source: Add files via upload
Disable transition
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
SnehaPatra
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Build
Succeeded
Pipeline execution ID: 063ade0b-9de6-4a9f-a6fd-360546dad263
Start rollback
Build
AWS CodeBuild
Succeeded - Just now
View details
0f16b85a Source: Add files via upload
Disable transition
Deploy
Succeeded
Pipeline execution ID: 063ade0b-9de6-4a9f-a6fd-360546dad263
Start rollback
Deploy
Amazon S3
Succeeded - Just now
View details
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

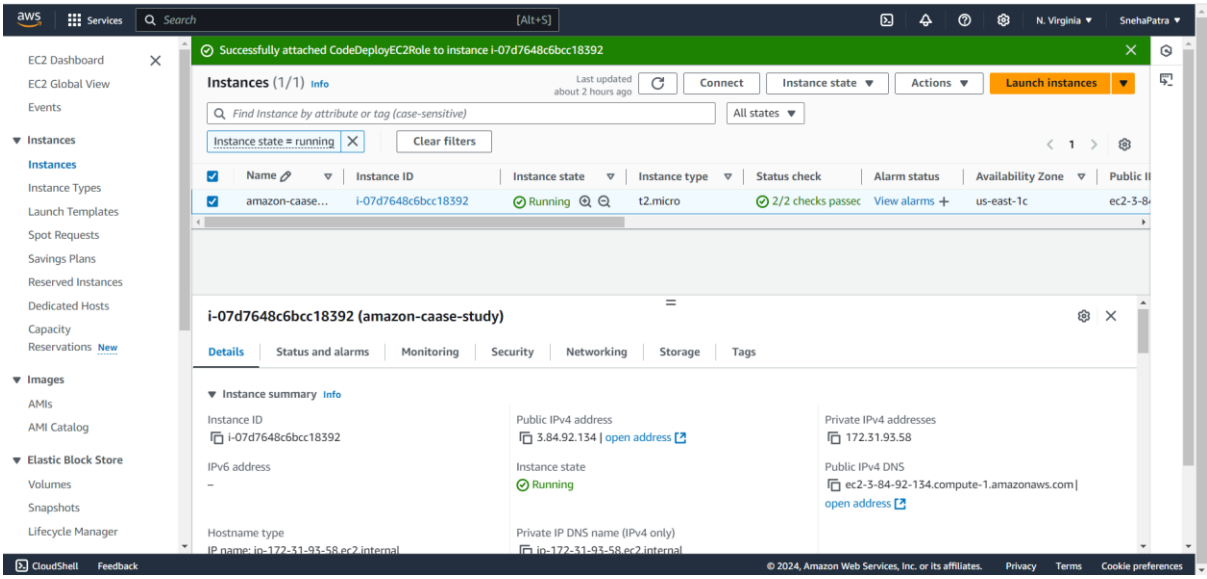
- Visit the S3 bucket's website URL to verify that the index.html page is live.



---

5. Set Up EC2 Instance for Web Hosting

1. **Launch an EC2 Instance**:
   - Open the EC2 console.



   - Launch a new instance, selecting an Amazon Linux 2 AMI.

- Choose the default t2.micro instance type.
- Configure instance settings and storage (use defaults for now).
- In **Configure Security Group**, allow HTTP traffic by adding a rule to open port 80.

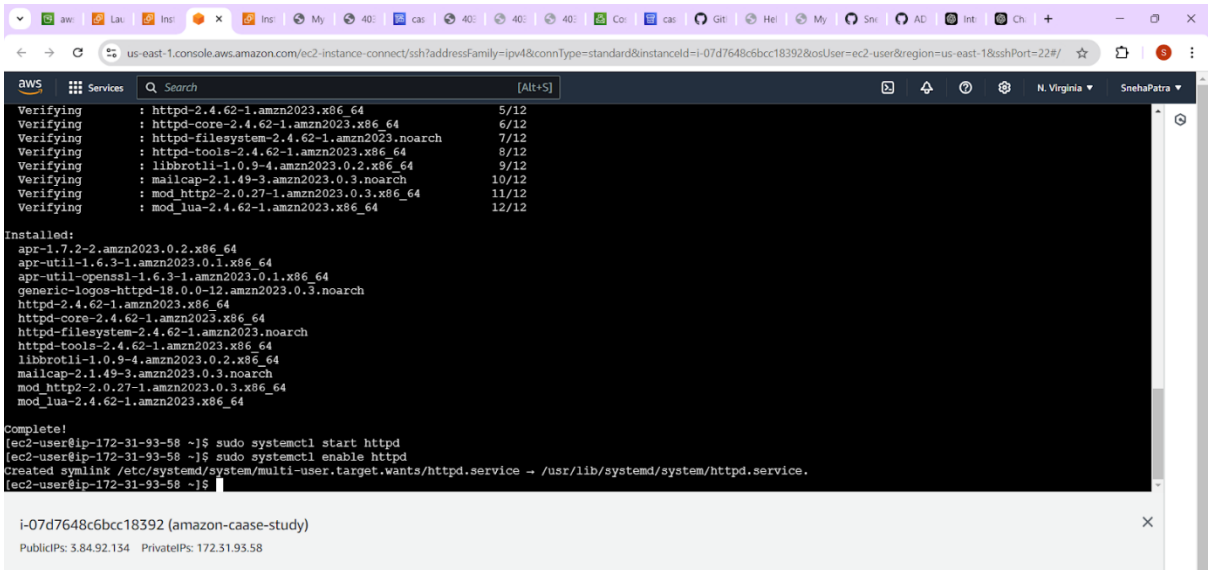
2. **Connect to the EC2 Instance**:
   - Once the instance is running, connect via SSH.

Install the required web server (Apache) on your instance:
sudo yum update -y

sudo yum install httpd -y

sudo systemctl start httpd

sudo systemctl enable httpd

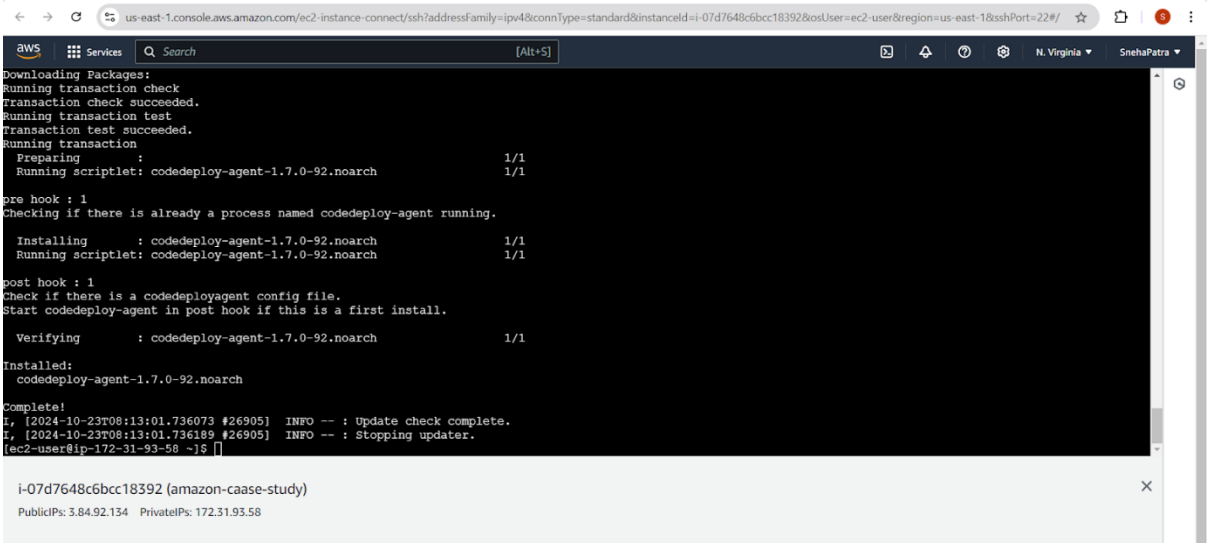


---

6. Set Up AWS CodeDeploy to Push Updates to EC2

1. **Install CodeDeploy Agent on EC2**:
   - Connect to the EC2 instance and Install the CodeDeploy agent:

sudo yum update -y
sudo yum install -y ruby wget
wget https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto



2. **Set Up CodeDeploy Application**:

3.1 Create Application

1. **Using AWS CLI from Local Terminal on pc (not ec2, use your own pc's command line)**:

 # First ensure AWS CLI is installed
aws --version

if not installed, install from https://awscli.amazonaws.com/AWSCLIV2.msi

aws configure

2. Create application:

aws deploy create-application --application-name my-webapp

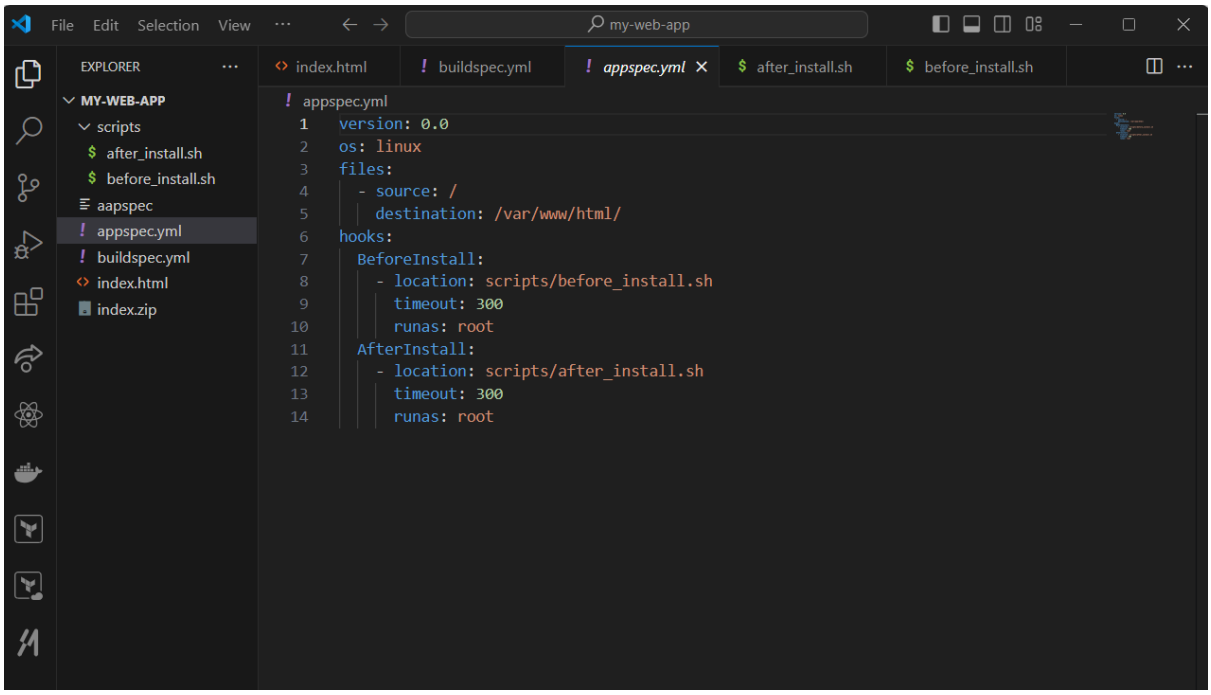
3.2 Create Deployment Group

- Create deployment group:

aws deploy create-deployment-group --application-name my-webapp --deployment-group-name my-webapp-group --service-role-arn arn:aws:iam::ACCOUNT_ID:role/CodeDeployServiceRole

- enter your ACCOUNT_ID above!!

Create appspec.yml for CodeDeploy: In your project folder, create an appspec.yml file to specify how CodeDeploy should handle the deployment:

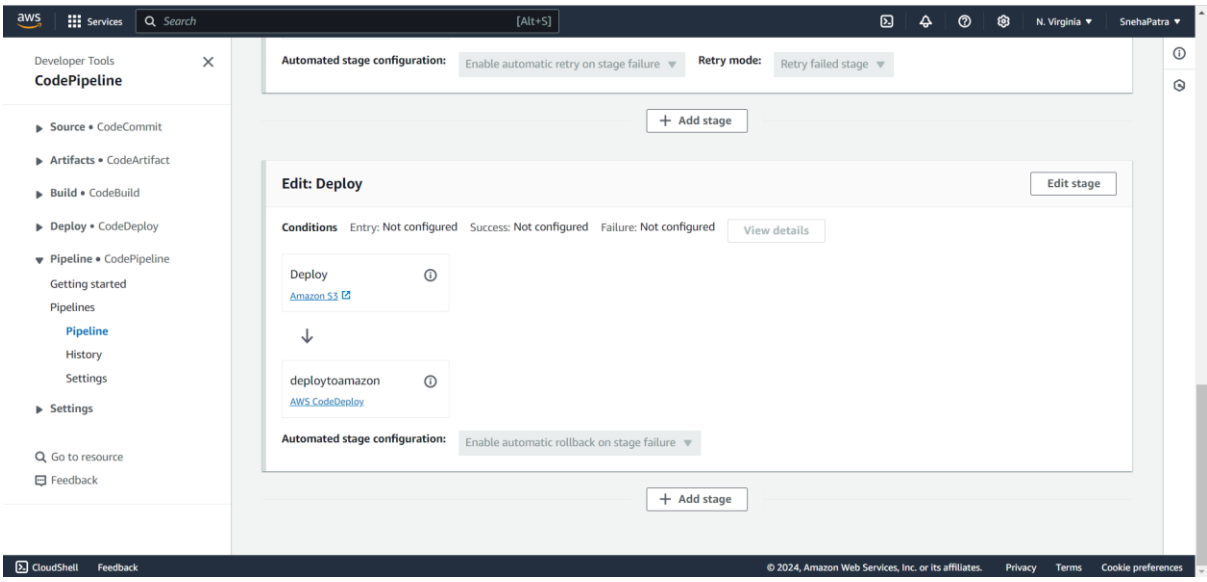


Now create a scripts folder inside your repo which will contain 2 files: before_install.sh and after_install.sh

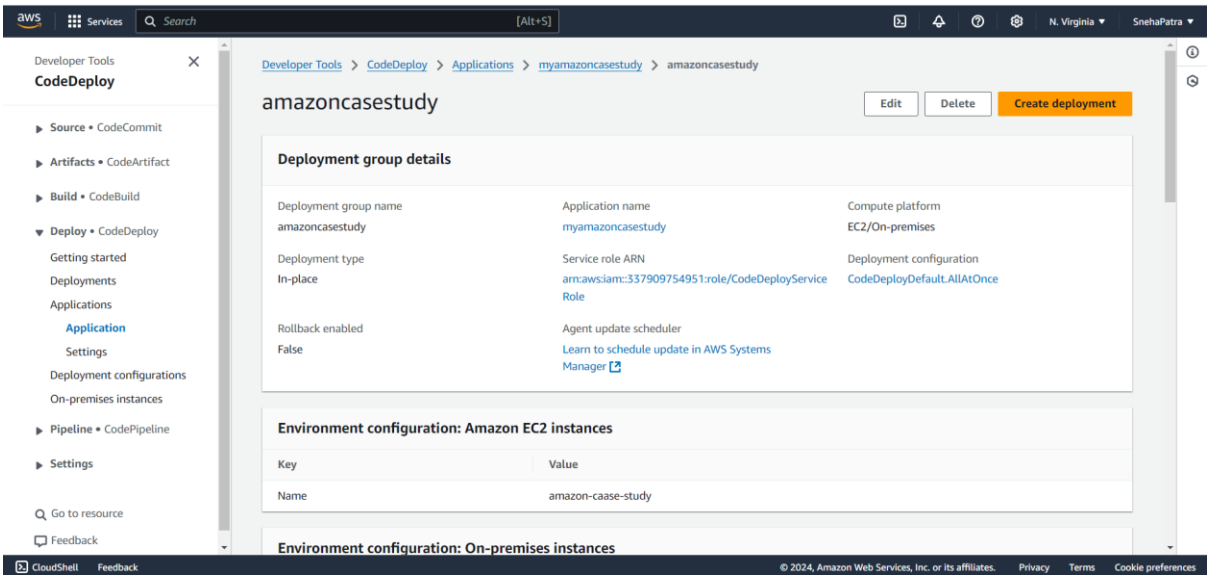
3. **Add Deployment Stage to CodePipeline**:

- Go back to your CodePipeline.

- Add a new stage for deployment.



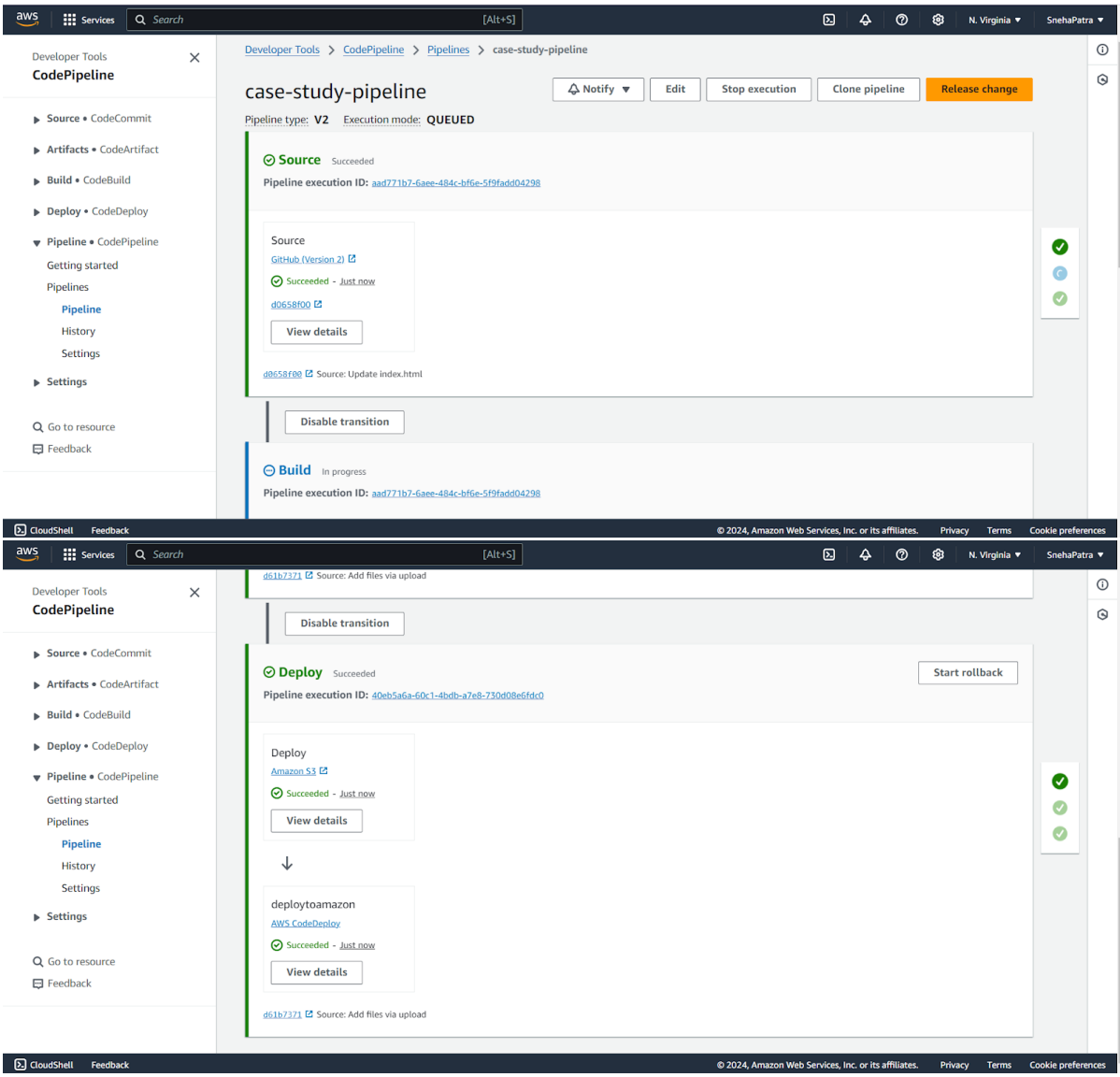
- Select **AWS CodeDeploy** and choose the application and deployment group you created earlier.



4. **Test Deployment**:

- Make a change to index.html in your source repository and push it.

- This should trigger the pipeline, rebuild the app, push it to S3, and deploy it to the EC2 instance.



- You can access your EC2 instance via its public IP to view the updated web app.

---

7. Verify Automation

- Now that your pipeline is set up, any changes to your repository (e.g., modifying the index.html file) should automatically trigger the build, deploy it to S3, and push updates to your EC2 instance.

CHALLENGES FACED:

1. **Permission Issues**: One of the main hurdles was setting the correct permissions in IAM roles and the S3 bucket policy. Errors like 403 Forbidden when accessing S3 were resolved by adjusting the S3 bucket policy.
2. **Missing AppSpec File**: The CodeDeploy process failed initially because the appspec.yml file was not placed correctly in the root directory of the build artifacts.
3. **CodeDeploy Agent Issues**: The EC2 instance had issues with the CodeDeploy agent, such as failing to start or showing Permission Denied errors. These were resolved by restarting the agent and ensuring the instance had appropriate permissions to access S3.

Despite these challenges, the overall deployment was successful, demonstrating the effectiveness of AWS's automation tools in managing continuous deployment workflows.

CONCLUSION:

This experiment successfully demonstrated the process of building and deploying a simple web application using AWS CodePipeline, CodeBuild, CodeDeploy, and EC2. The pipeline automates the workflow from the moment the code is pushed to a repository until the application is deployed on an EC2 instance.