

## Advance Devops Case Study Report

### Cloud Deployment with Automation

#### Concepts Used:

- AWS CodePipeline
- EC2
- S3 Bucket

#### 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 deploying and automating a simple web application using various services from Amazon Web Services (AWS). The goal is to create a cloud-based infrastructure that streamlines the continuous integration and delivery (CI/CD) process.

The project begins with building the web application through AWS CodeBuild, which compiles and packages the code. Once the application is prepared, the deployment is automated using AWS CodePipeline, ensuring smooth transitions from development to production without manual intervention. The final step involves deploying the application to an S3 bucket to host static content and pushing updates to an EC2 instance using AWS CodeDeploy, achieving continuous delivery and scalable performance.

## Key Feature and Application:

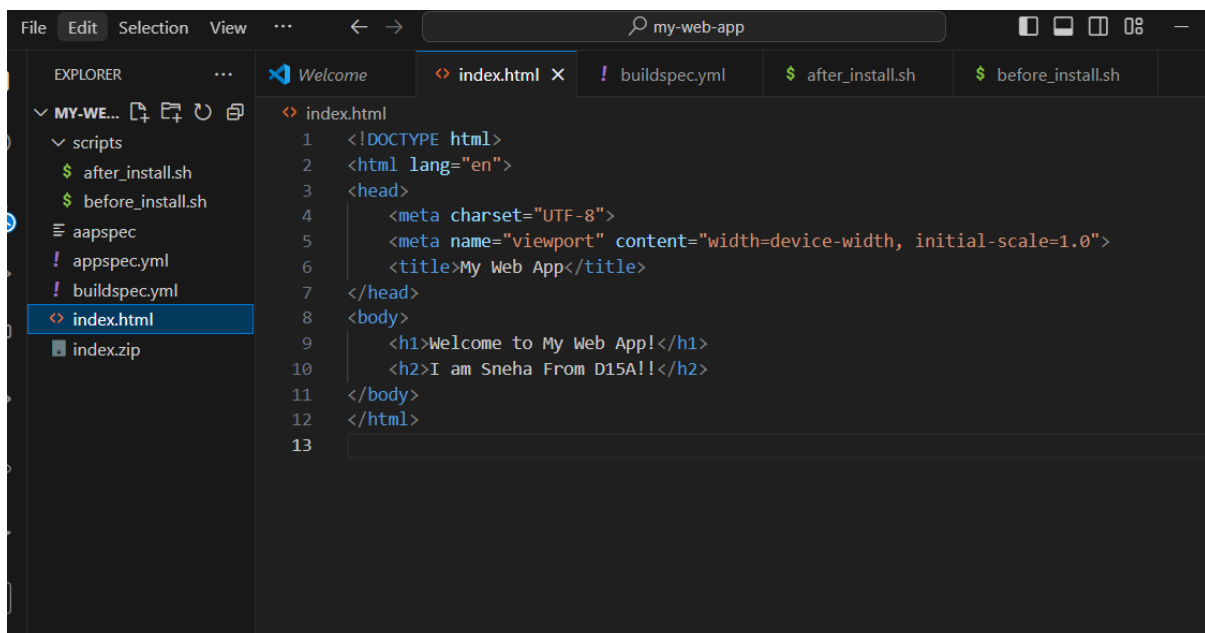
This project emphasizes several core capabilities of cloud infrastructure and automation, including:

- **Automation:** CodePipeline automates the build, deployment, and delivery processes, reducing human effort and minimizing errors.
- **Continuous Delivery:** Every code change triggers an automatic deployment to the S3 bucket or EC2 instance, ensuring that the application is always up-to-date.
- **High Availability and Scalability:** Hosting static content on **S3** and deploying dynamic content on **EC2** ensures that the web application can handle traffic effectively.

## 2. Step-by-Step Explanation:

### Step 1: Create a Simple Web App

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

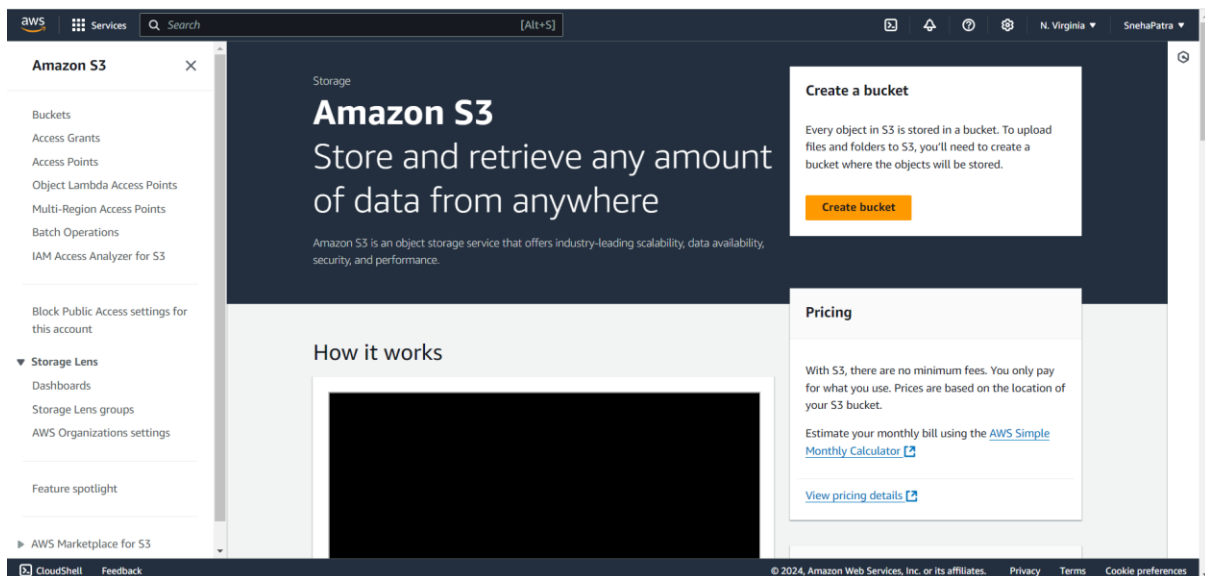
A screenshot of a code editor interface. The Explorer panel on the left shows a project named 'MY-WE...' with a 'scripts' folder containing 'after\_install.sh' and 'before\_install.sh', and a file named 'index.html' which is currently selected. The main editor area displays the content of 'index.html'. The code is a simple HTML document with a doctype, meta tags for charset and viewport, a title 'My Web App', and a body containing two lines of text: 'Welcome to My Web App!' and 'I am Sneha From D15A!!'.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>My Web App</title>
7 </head>
8 <body>
9   <h1>Welcome to My Web App!</h1>
10  <h2>I am Sneha From D15A!!</h2>
11 </body>
12 </html>
13
```

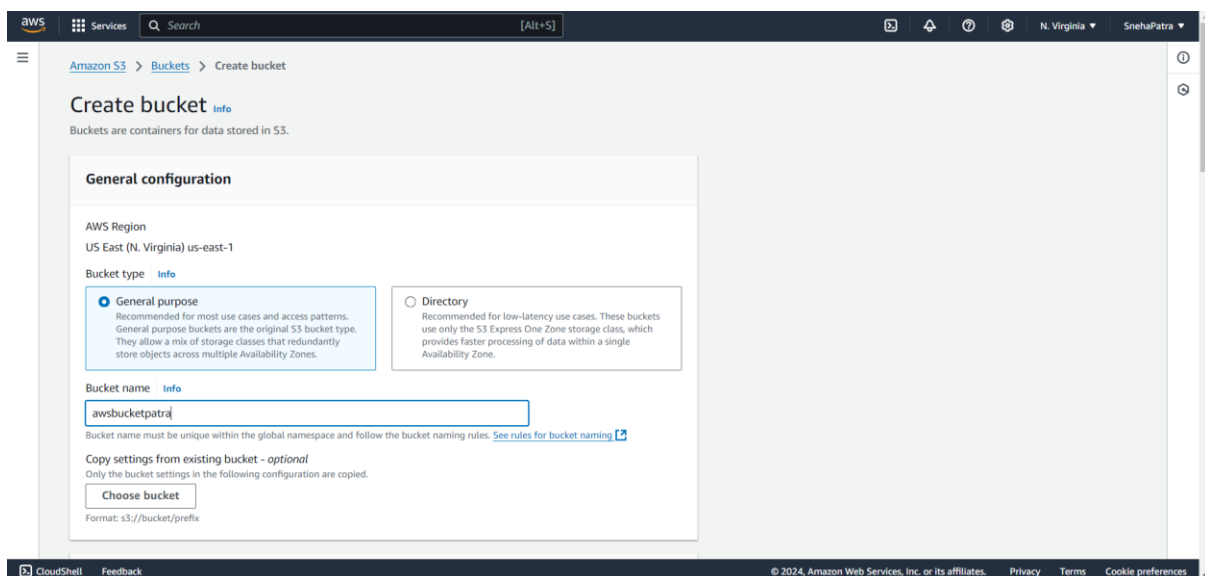
- This file will serve as the web page deployed to your S3 bucket and later to the EC2 instance.

## Step 2: Set Up S3 Bucket for Web App Hosting

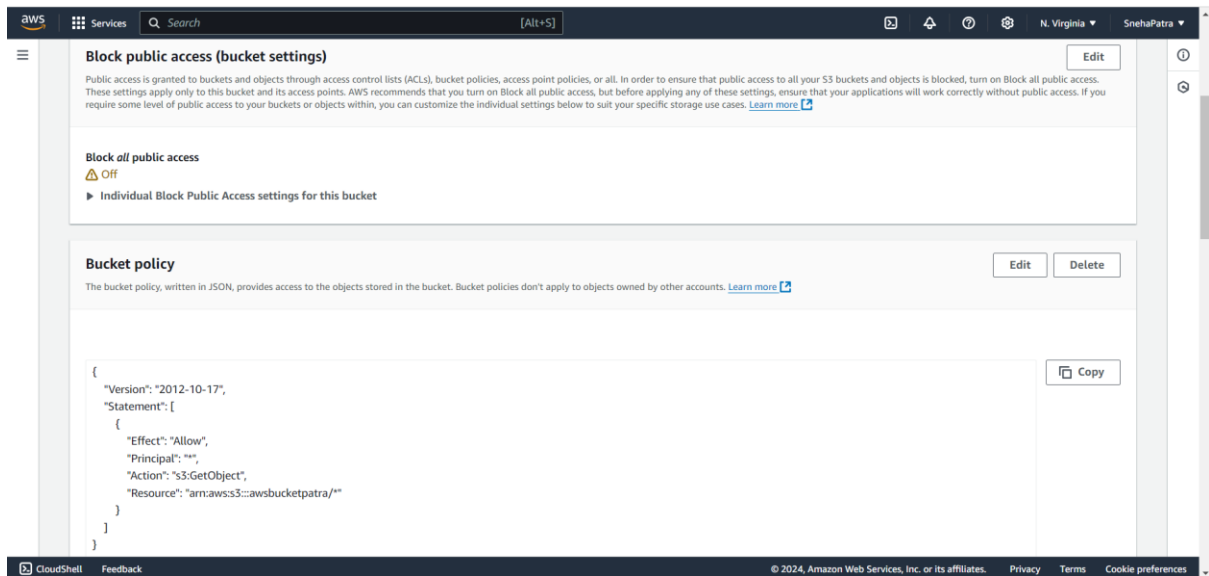
- Go to the AWS S3 Console:



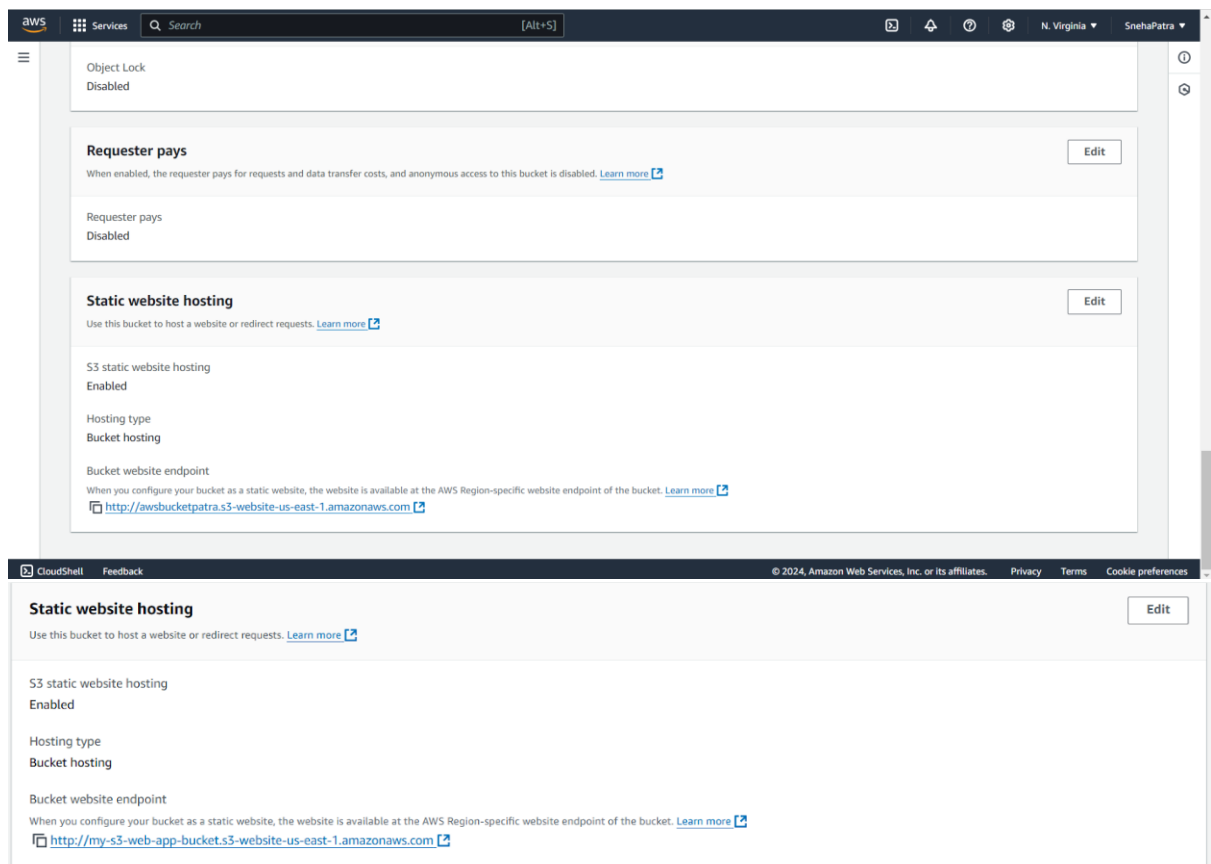
- Create a new S3 bucket, and give it a unique name (e.g. awsbucketpatra).



- Under **Permissions**, uncheck the "Block all public access" option, allowing public access for web hosting.
- Also under permissions add bucket policy So that you don't get configuration error.



- 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.



The bucket website URL: <http://awsbucketpatra.s3-website-us-east-1.amazonaws.com>

### Step 3: Create a CodeBuild Project:

- Create a **buildspec.yml** file to specify build instructions:

version: 0.2

phases:

build:

commands:

- echo "Building the application..."

post\_build:

commands:

- aws s3 cp index.html s3://awsbucketpatra/index.html --content-type "text/html"

artifacts:

files:

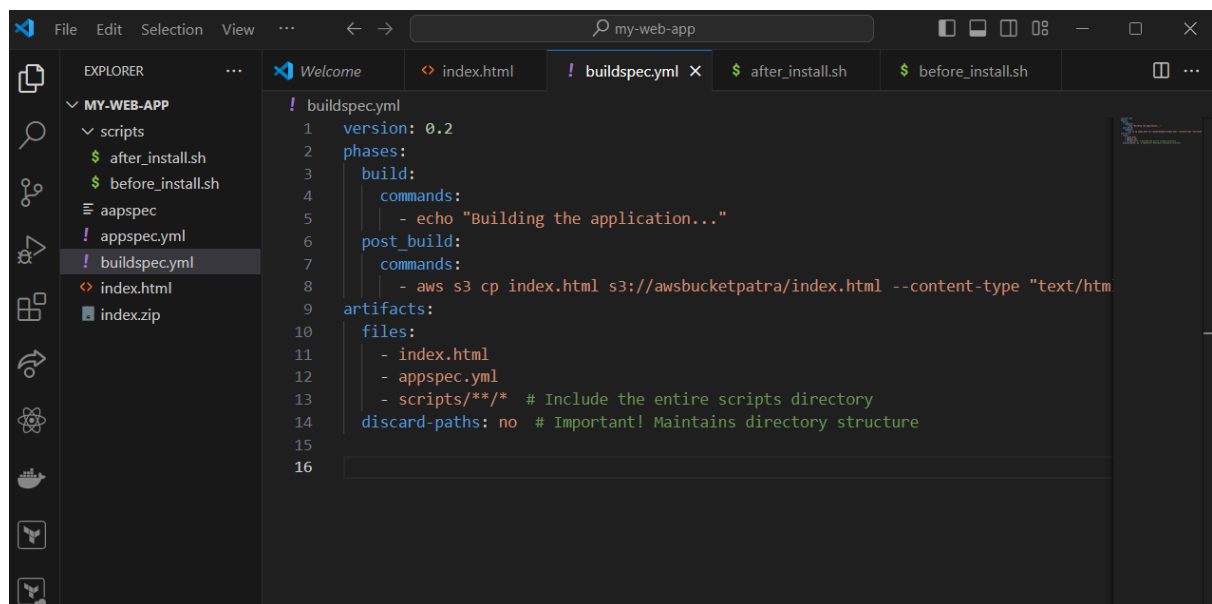
- index.html

- appspec.yml

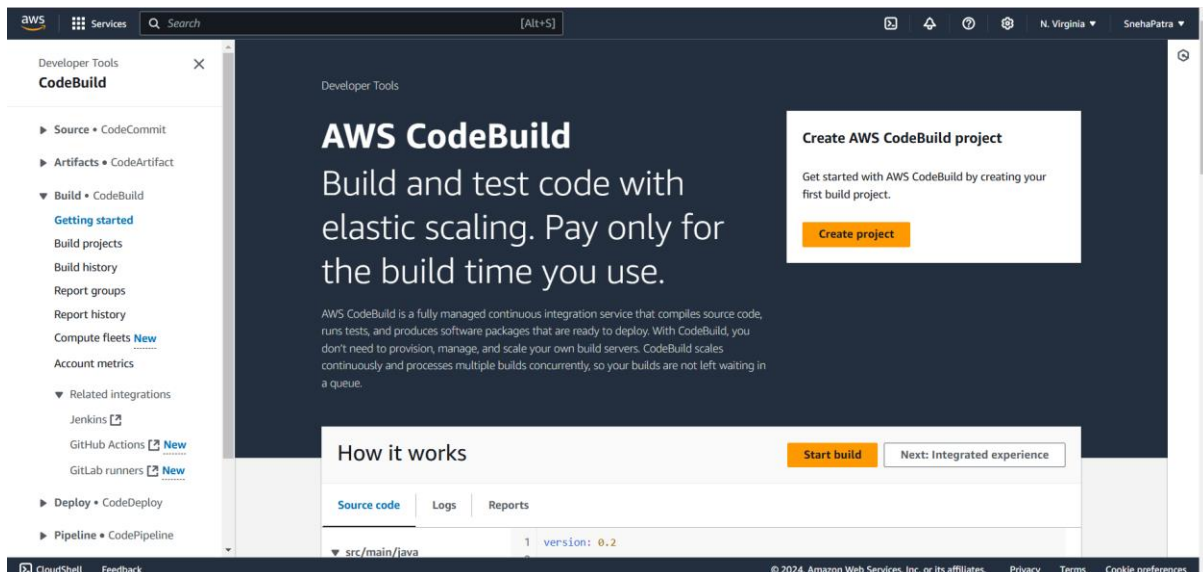
- scripts/\*\*/\* # Include the entire scripts directory

discard-paths: no # Important! Maintains directory structure

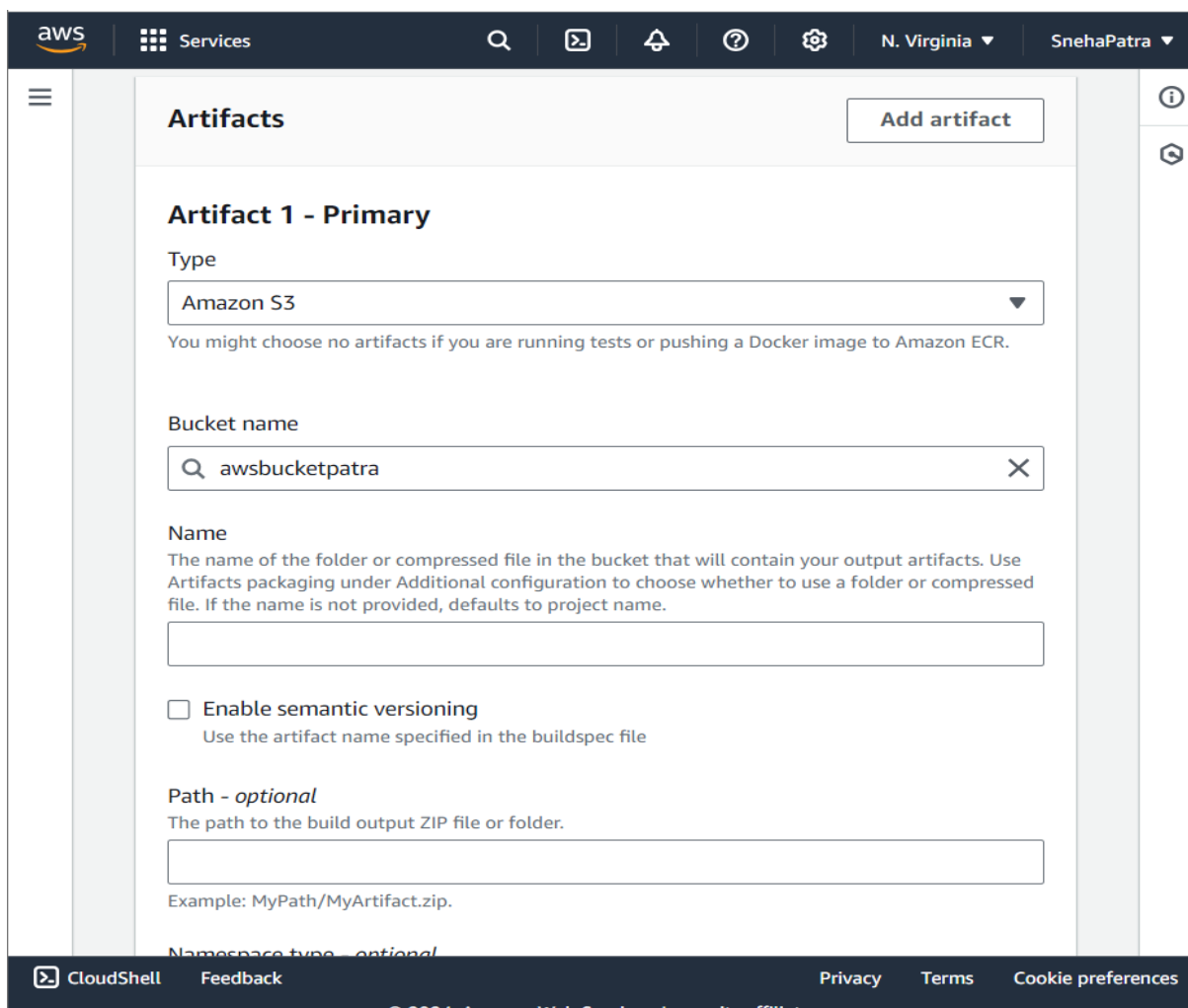
(Note: Add your bucket name.)



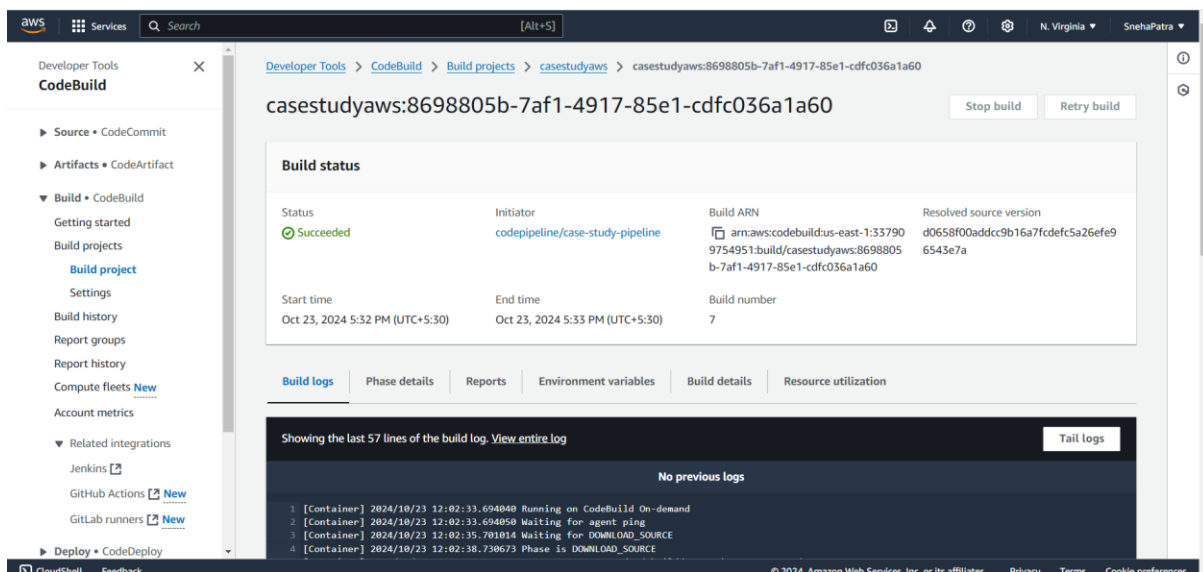
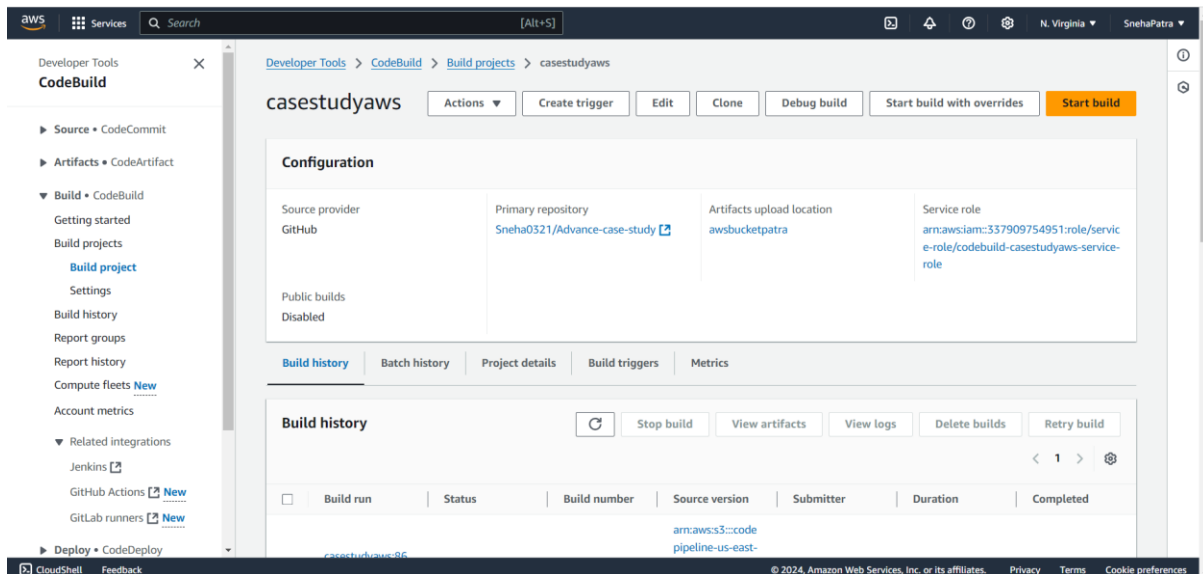
- **Now Go to AWS CodeBuild:**
- Open CodeBuild in the AWS Management Console and click Create project.



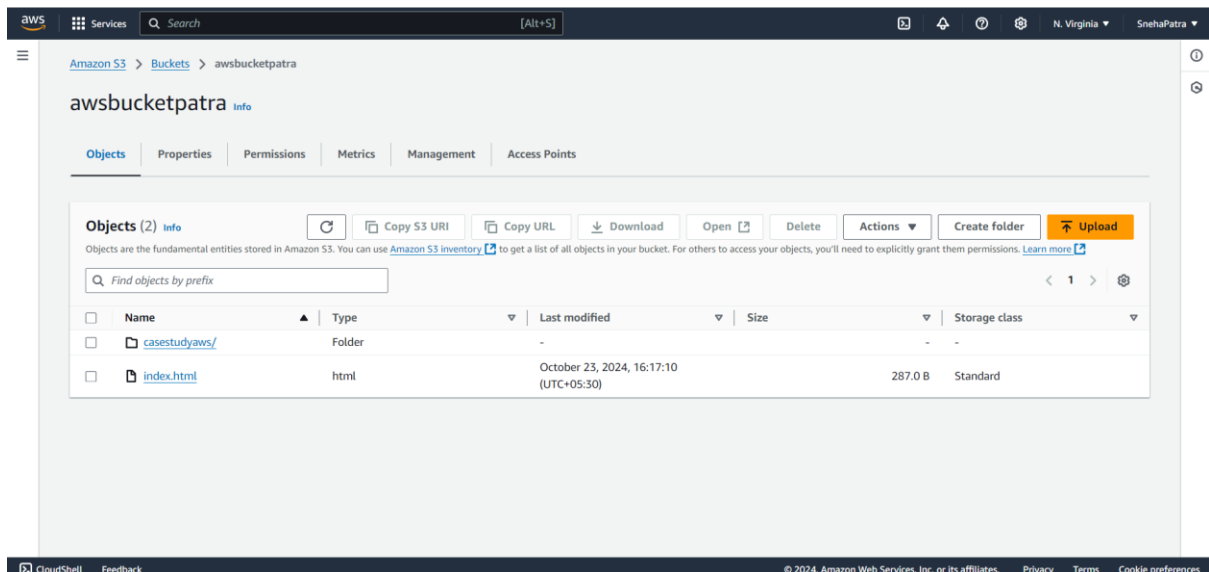
- For **Source**, choose your source repository (e.g., GitHub). Also specify the buildspec.yml file you created. Before this add it to your git repository.
- Set **Artifacts** to Amazon S3, and choose the bucket you created earlier.



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

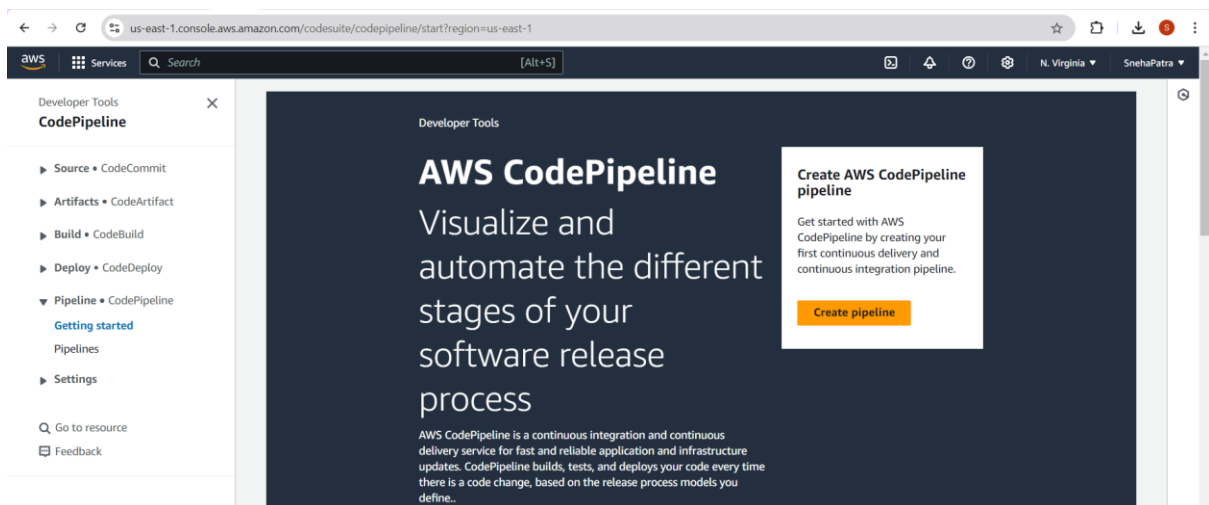


- Once your build is successful your s3 bucket will get updated:



## Step 4: Create a Pipeline to Deploy to S3 Bucket

- Open CodePipeline in the AWS Management Console and click Create pipeline.
- Set the pipeline name (e.g., case-study-pipeline).



- In the **Source Stage**, choose your **GitHub repository**.
- Connect and choose the appropriate branch where the index.html and buildspec.yml files are.



us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelin...

aws Services N. Virginia SnehaPatra


## Choose source Info

Step 3 of 4

### 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) ▼

**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.

arn:aws:codeconnections:eu-north-1:337906... X or [Connect to GitHub](#)

**Repository name**  
Choose a repository in your GitHub account.

Sneha0321/Advance-case-study X

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

CloudShell Feedback Privacy Terms Cookie preferences

© 2024, Amazon Web Services, Inc. or its affiliates.

- Now **Add Build Stage**:
- In the **Build Stage**, choose the CodeBuild project created earlier.
- Make sure that the Input Artifacts are correctly configured.

aws Services

Search

N. Virginia

SnehaPatra

# Add build stage [Info](#)

Step 4 of 6

## Build - optional

**Build provider**  
Choose the tool you want to use to run build commands and specify artifacts for your build action.

☐ Commands ☒ Other build providers

AWS CodeBuild

**Project name**  
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

or

**Environment variables - optional**  
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

**Build type**

☒ Single build  
Triggers a single build.

CloudShell Feedback Privacy Terms Cookie preferences

© 2024, Amazon Web Services, Inc. or its affiliates.

- **Add Deployment to S3:**
- In the **Deploy Stage**, choose **S3** and select the hosting bucket (awsbucketpatra).
- Add your bucket key correctly or else it will throw an error and your deployment will be failed.

**Add deploy stage** info

Step 5 of 6

**Deploy - optional**

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

Amazon S3

**Region**  
US East (N. Virginia)

**Input artifacts**  
Choose an input artifact for this action. [Learn more](#)

BuildArtifact  
Defined by: Build  
No more than 100 characters

**Bucket**  
awsbucketpatra

**S3 object key**  
casesstudyaws/index.html

Enter the object key. You can include a file path without the delimiter character (/) at the beginning. Include the file extension. Example: SampleApp.zip

- **Now Run the Pipeline:**
- Click **Release Change** to test the pipeline.
- Check if the index.html page is accessible from the **S3 static website endpoint URL**.

**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**

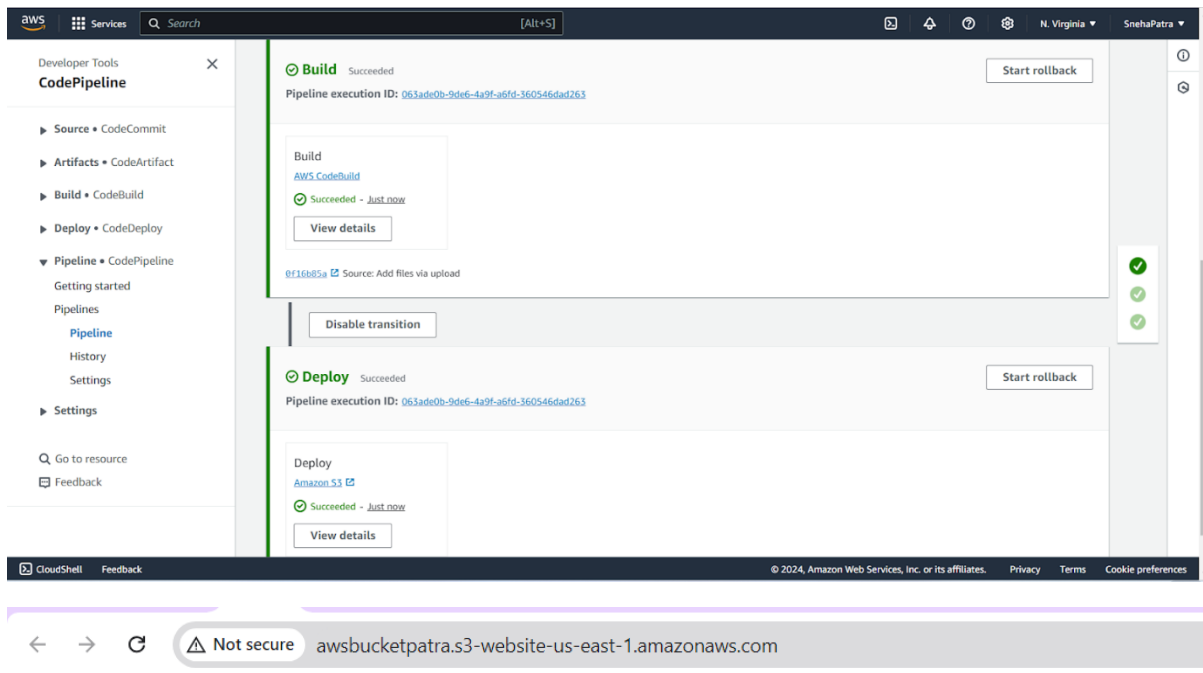
Pipeline type: V2 Execution mode: QUEUED

**Source** Succeeded  
Pipeline execution ID: D63ade0b-9def-4a9f-adfd-360546dad263

Source  
GitHub (Version 2)  
Succeeded - 1 minute ago  
@f16b85a  
View details

@f16b85a Source: Add files via upload

Disable transition

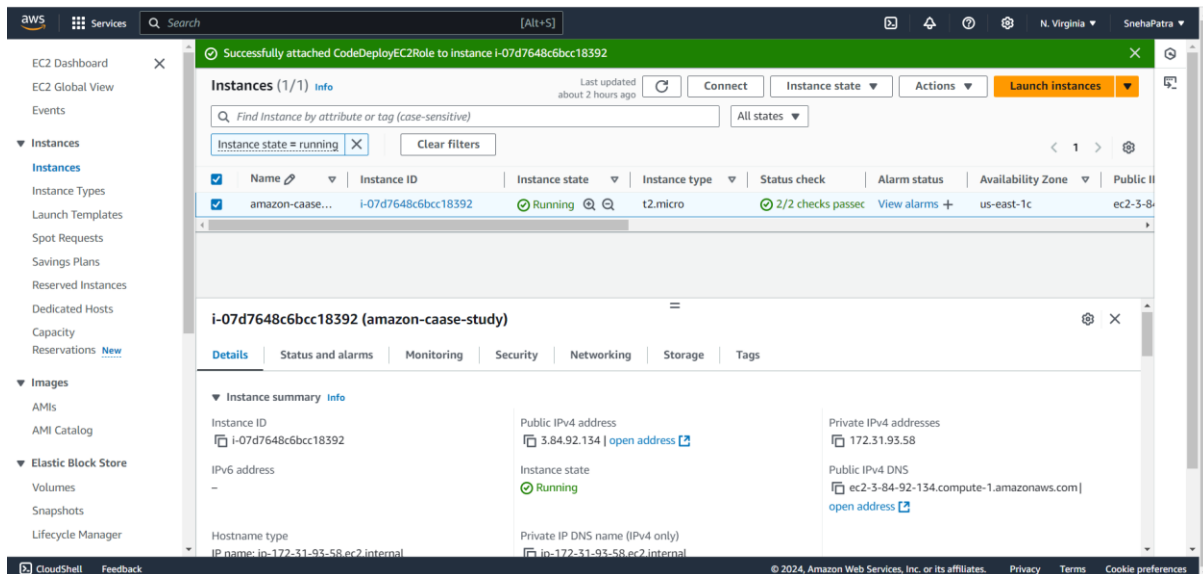


# Welcome to My Web App!

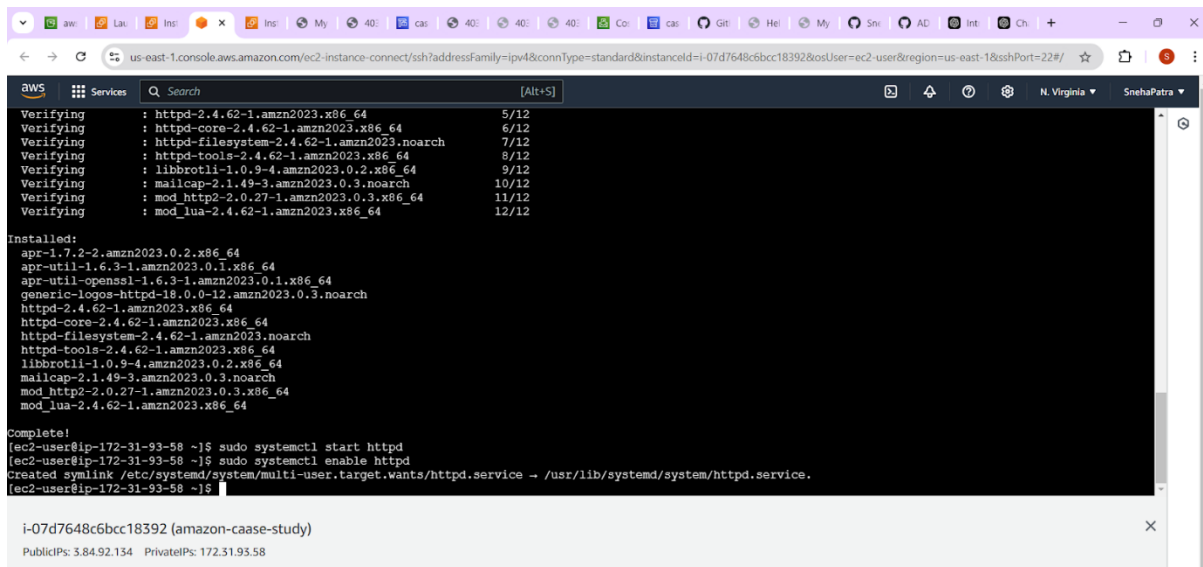
I am Sneha From D15A!!

## Step 5: Set Up EC2 Instance

- **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 default settings).
- In **Configure Security Group**, allow HTTP traffic by adding a rule to open port 80 and also SSH traffic to port 80.
- Then launch your EC2.

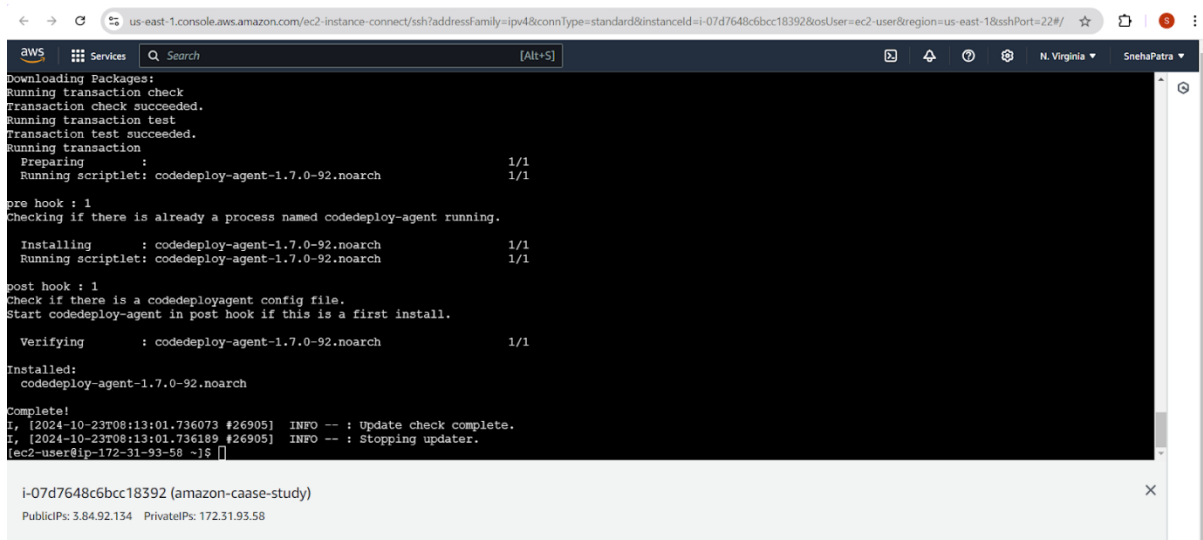


- **Connect to the EC2 Instance:**
- Once the instance is running, connect via SSH key.
- To install the required web server (Apache) on your instance, run the following commands
  - `sudo yum update -y`
  - `sudo yum install httpd -y`
  - `sudo systemctl start httpd`
  - `sudo systemctl enable httpd`



## Step 6: Push Updates to EC2 Instance Using AWS CodeDeploy

- **Install the CodeDeploy Agent on the EC2 Instance:**
- Make sure you have connected to the EC2 instance via **SSH** if not then use the command
  - `ssh -i your-key.pem ec2-user@your-instance-public-ip`
- Then run the following commands to install the CodeDeploy agent:
  - `sudo yum update -y`
  - `sudo yum install ruby -y`
  - `sudo yum install wget -y`
  - `cd /home/ec2-user`
  - `wget https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install`
  - `sudo chmod +x ./install`
  - `sudo ./install auto`
  - `sudo service codedeploy-agent start`



The screenshot shows the AWS Management Console terminal window for an EC2 instance. The terminal output displays the successful installation of the CodeDeploy agent. The process includes downloading packages, running transaction checks, preparing the scriptlet, and installing the agent. The final output shows the agent is installed and the updater is stopped.

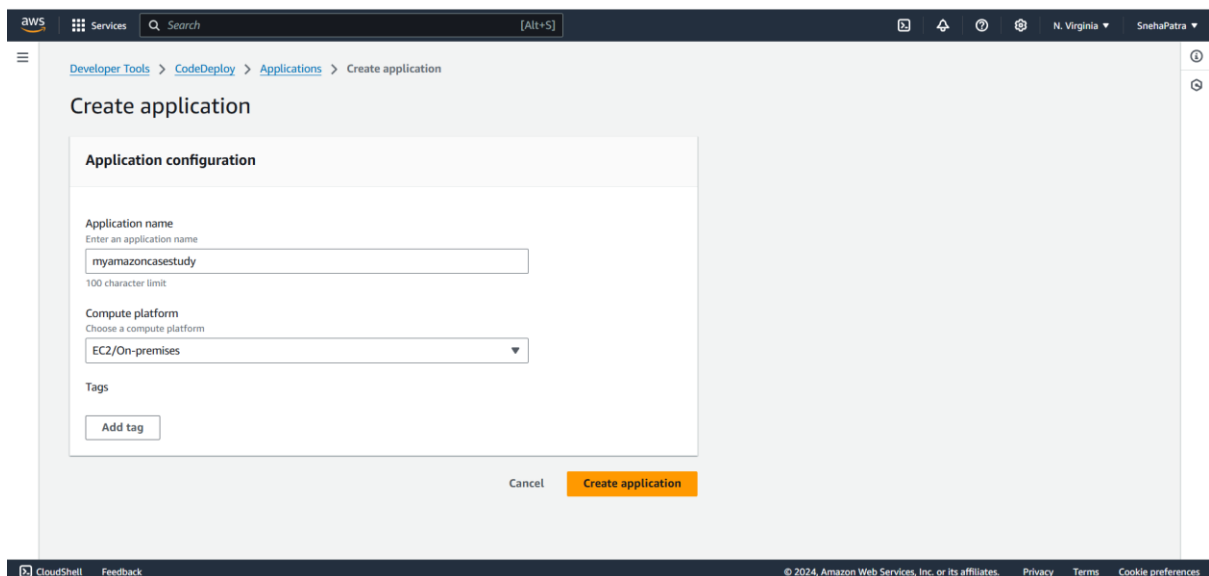
```
Download Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      : codedeploy-agent-1.7.0-92.noarch                1/1
  Running scriptlet: codedeploy-agent-1.7.0-92.noarch                1/1
pre hook : 1
Checking if there is already a process named codedeploy-agent running.
  Installing      : codedeploy-agent-1.7.0-92.noarch                1/1
  Running scriptlet: codedeploy-agent-1.7.0-92.noarch                1/1
post hook : 1
Check if there is a codedeployagent config file.
Start codedeploy-agent in post hook if this is a first install.
  Verifying      : codedeploy-agent-1.7.0-92.noarch                1/1
Installed:
codedeploy-agent-1.7.0-92.noarch
Complete!
1, [2024-10-23T08:13:01.736073 #26905] INFO -- : Update check complete.
1, [2024-10-23T08:13:01.736189 #26905] INFO -- : Stopping updater.
[ec2-user@ip-172-31-93-50 ~]$
```

i-07d7648c6bcc18392 (amazon-caase-study)  
PublicIPs: 3.84.92.134 PrivateIPs: 172.31.93.58

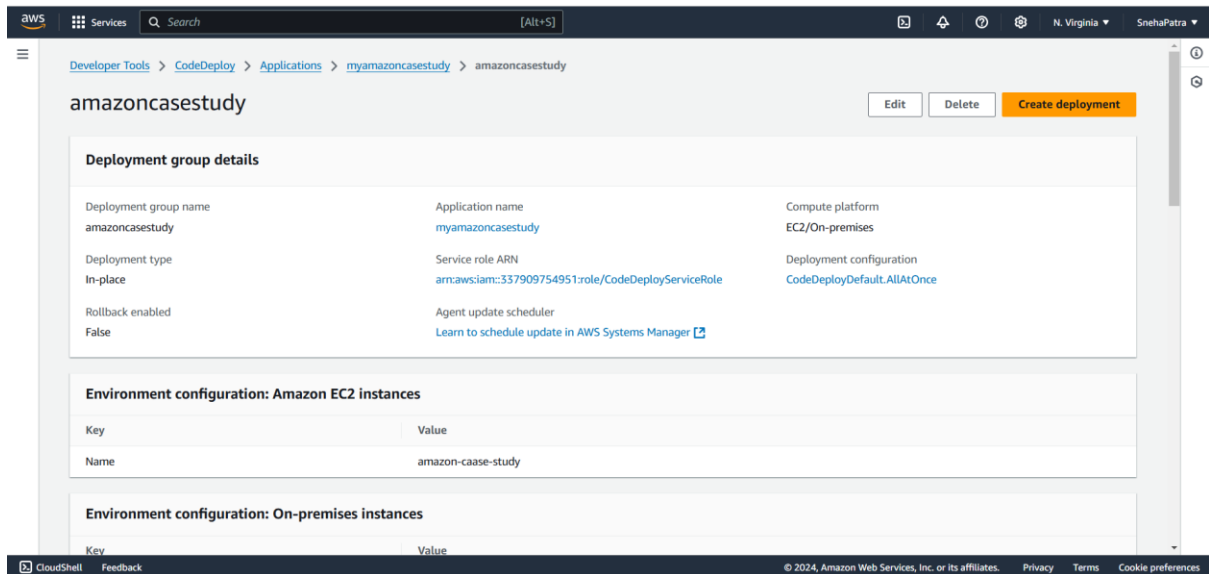
- **Create an Application in CodeDeploy:**



- **Create an Application in CodeDeploy:**
- Enter an application name and choose a compute platform and select create application.



- **Create Deployment Group**
- Name your Deployment group and select your EC2/On-premises.
- Add correct key and value of tag of your EC2 Instance or else it will show no healthy connection and deployment will be failed.



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

version: 0.0

os: linux

files:

- source: /

destination: /var/www/html/

hooks:

BeforeInstall:

- location: scripts/before\_install.sh

timeout: 300

runas: root

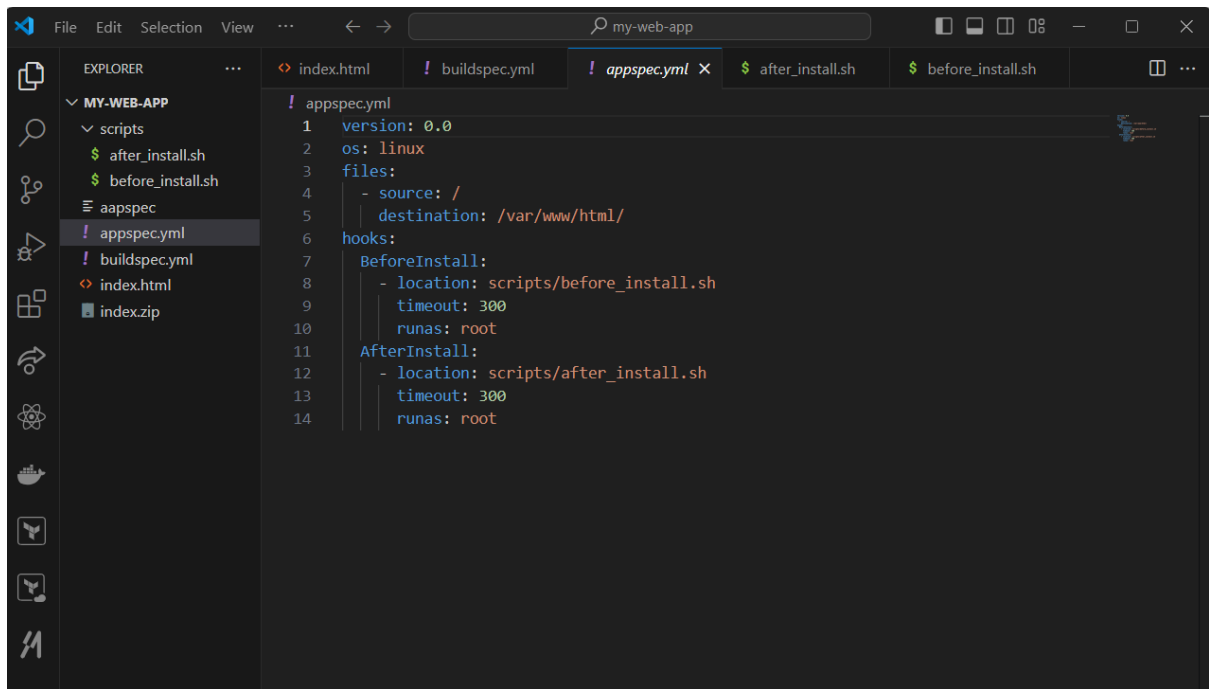
AfterInstall:

- location: scripts/after\_install.sh

timeout: 300

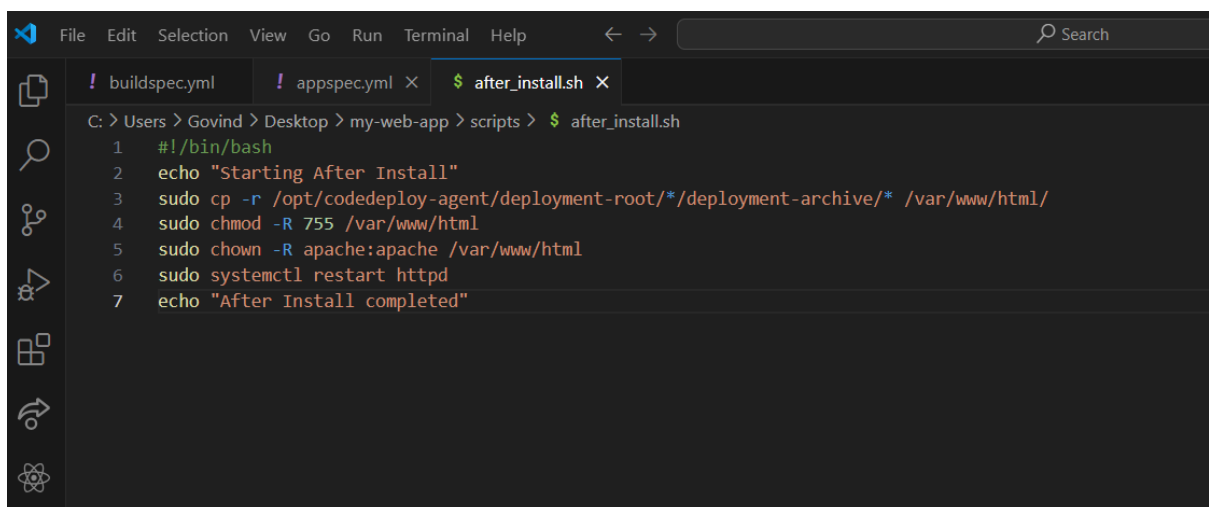
runas: root





- Now create a scripts folder inside your repo which will contain 2 files: before\_install.sh and after\_install.sh.
- **after\_install.sh**  

```
#!/bin/bash
echo "Starting After Install"
sudo cp -r /opt/codedeploy-agent/deployment-root/*/deployment-archive/* /var/www/html/
sudo chmod -R 755 /var/www/html
sudo chown -R apache:apache /var/www/html
sudo systemctl restart httpd
echo "After Install completed"
```

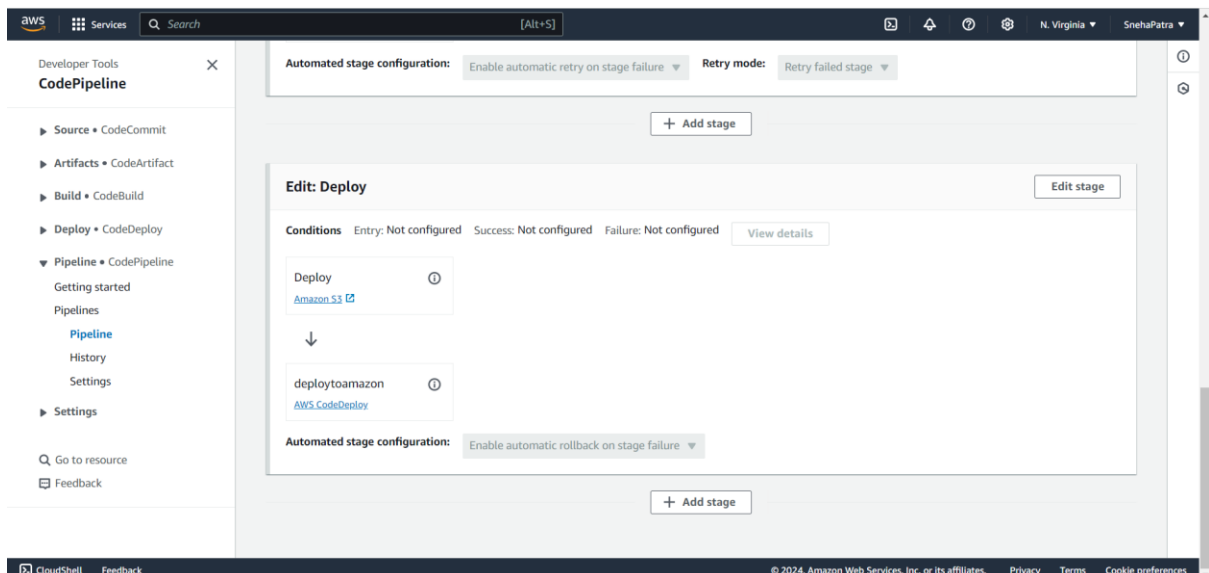


- **before\_install.sh**

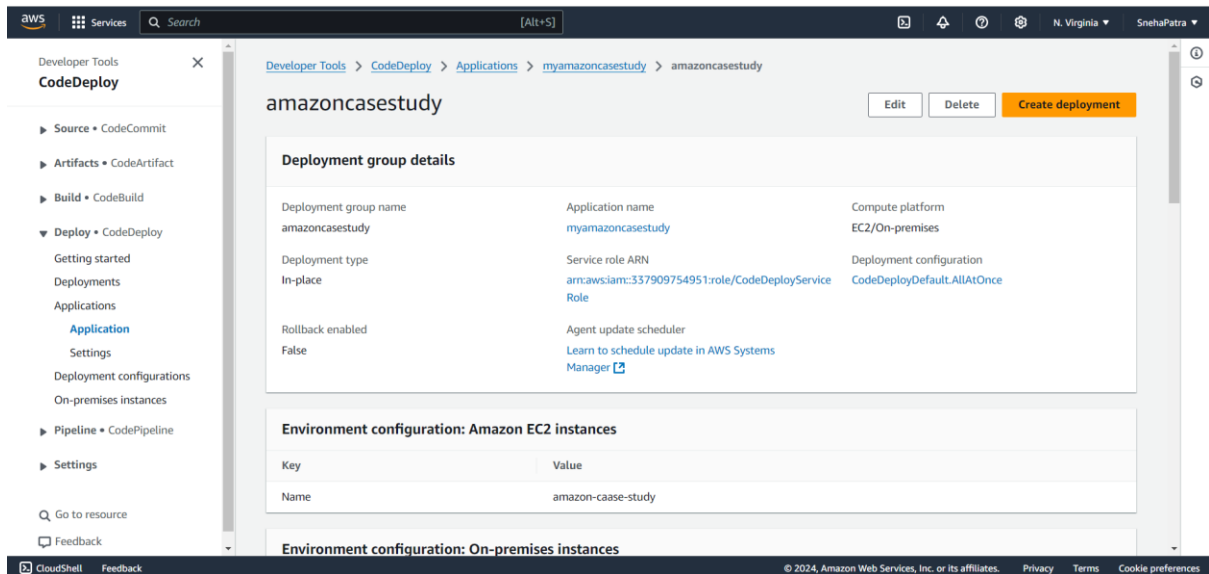
```
#!/bin/bash
echo "Starting Before Install"
sudo yum update -y
sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
echo "Before Install completed"
```

```
C: > Users > Govind > Desktop > my-web-app > scripts > $ before_install.sh
1  #!/bin/bash
2  echo "Starting Before Install"
3  sudo yum update -y
4  sudo yum install -y httpd
5  sudo systemctl start httpd
6  sudo systemctl enable httpd
7  echo "Before Install completed"
```

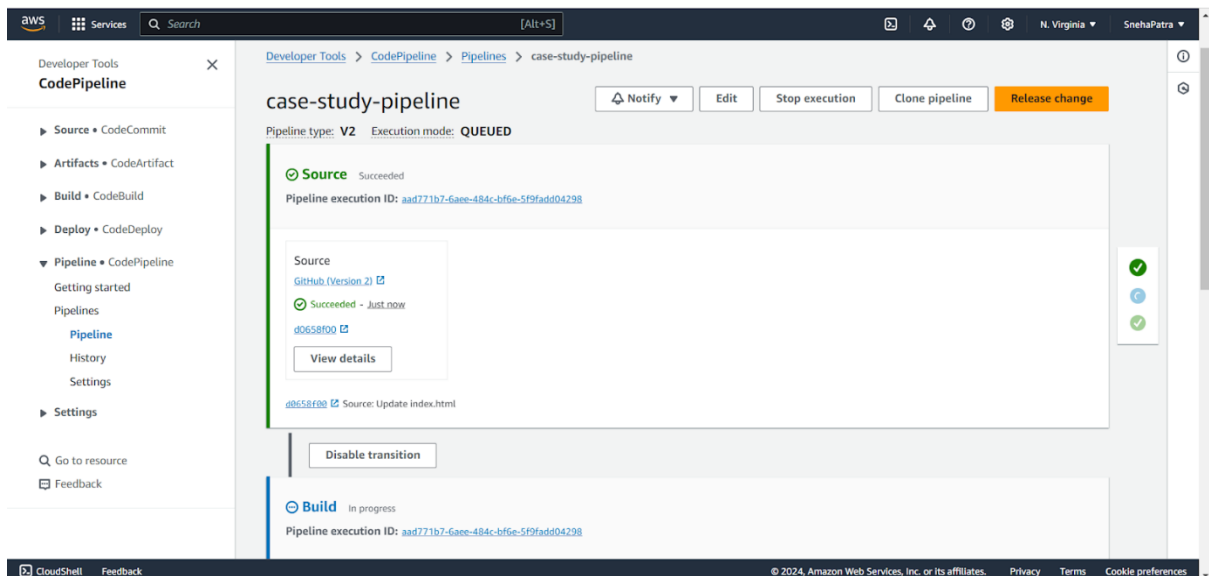
- **Add Deployment Stage to CodePipeline:**
- Go back to your CodePipeline, Add a new stage for deployment.

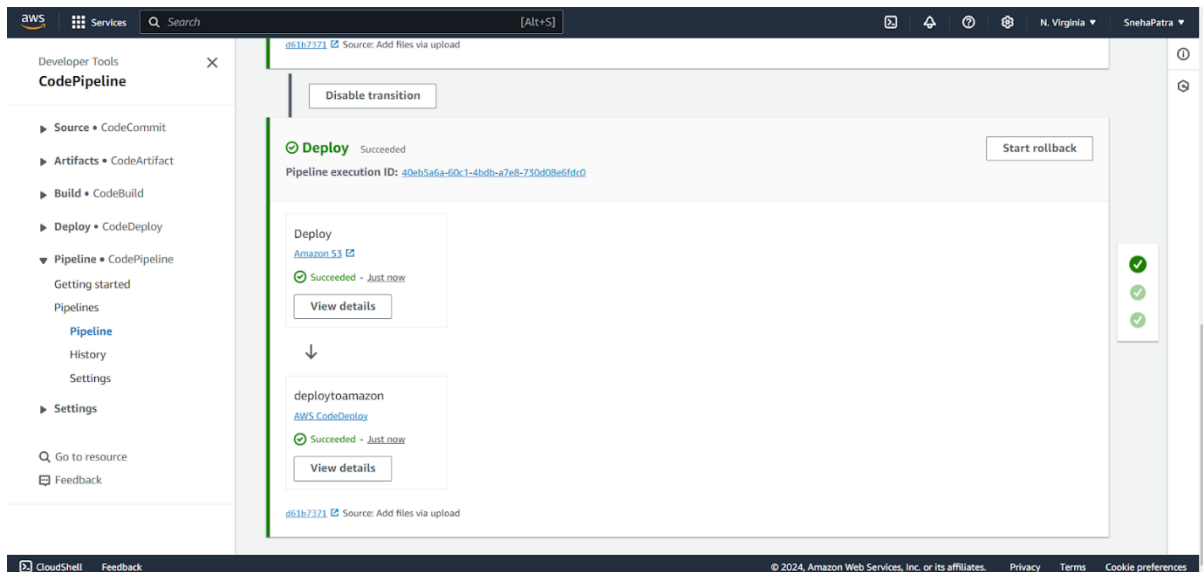


- Select **AWS CodeDeploy** and choose the application and deployment group you created earlier.



- Now you can **Test your Deployment:**
- Make a change to index.html in your source repository and push it.
- This will trigger your pipeline, rebuild the app, push it to S3, and deploy it to the EC2 instance.

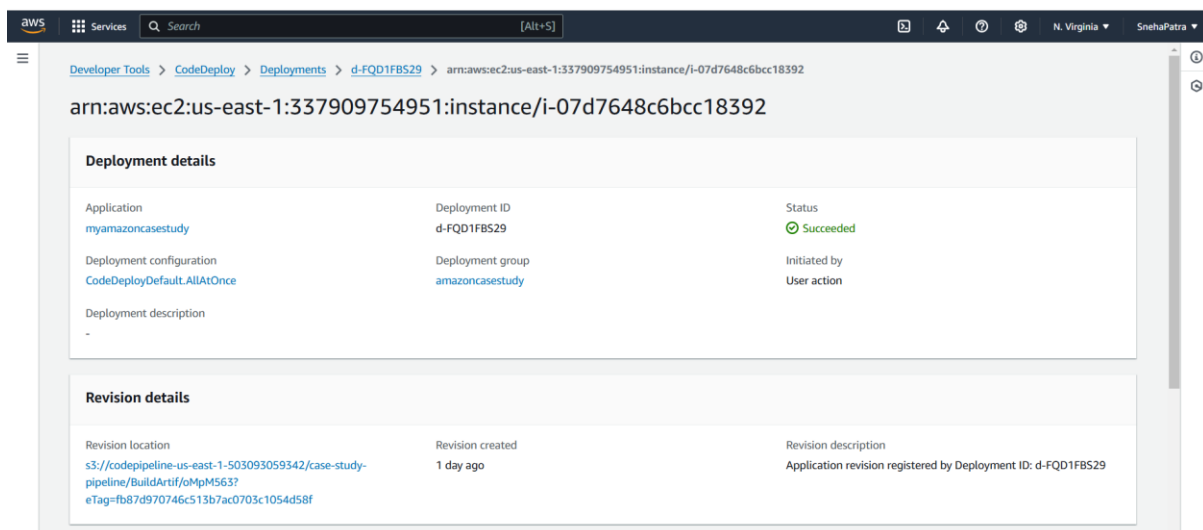




- Now you can access your EC2 instance via its public IP to view the changes you made in your web app.

## Step 7: Verify Your Automation

- Now that your pipeline is set up, any changes to your repository will automatically trigger the build, deploy it to S3, and push updates to your EC2 instance.
- Check the deployment Details and once the deployment is successful.
- Visit the S3 bucket's website URL to verify that the index.html page is live.



Deployment description

-

**Revision details**

Revision location	Revision created	Revision description
s3://codepipeline-us-east-1-503093059342/case-study-pipeline/BuildArtif/oMpM563?eTag=fb87d970746c513b7ac0703c1054d58f	1 day ago	Application revision registered by Deployment ID: d-FQD1FBS29

Event	Duration	Status	Error code	Start time	End time
ApplicationStop	less than one second	✔ Succeeded	-	Oct 23, 2024 5:33 PM (UTC+5:30)	Oct 23, 2024 5:33 PM (UTC+5:30)
DownloadBundle	less than one second	✔ Succeeded	-	Oct 23, 2024 5:33 PM (UTC+5:30)	Oct 23, 2024 5:33 PM (UTC+5:30)
BeforeInstall	1 second	✔ Succeeded	-	Oct 23, 2024 5:33 PM (UTC+5:30)	Oct 23, 2024 5:33 PM (UTC+5:30)
Install	less than one second	✔ Succeeded	-	Oct 23, 2024 5:33 PM (UTC+5:30)	Oct 23, 2024 5:33 PM (UTC+5:30)
AfterInstall	1 second	✔ Succeeded	-	Oct 23, 2024 5:33 PM (UTC+5:30)	Oct 23, 2024 5:33 PM (UTC+5:30)
ApplicationStart	less than one second	✔ Succeeded	-	Oct 23, 2024 5:33 PM (UTC+5:30)	Oct 23, 2024 5:33 PM (UTC+5:30)
ValidateService	less than one second	✔ Succeeded	-	Oct 23, 2024 5:33 PM (UTC+5:30)	Oct 23, 2024 5:33 PM (UTC+5:30)

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Welcome to My Web App!

I am Sneha From D15A!!

This is my amazon case study.

## Error Encountered:

### 1. CodeBuild Errors:

Issue: Build Fails with “Permission Denied” or “Artifact Upload Error”

Solution:

Ensure the CodeBuild service role has s3:PutObject permissions to write to the S3 bucket.

Check if your buildspec.yml file is correctly formatted and in the root directory of the repository.

### 2. S3 Bucket Issues:

Issue: "403 Forbidden" when Accessing the Website from S3

Solution:

Go to S3 > Permissions > Bucket Policy and add the following policy:

```
{
```

```
"Version": "2012-10-17",  
"Statement": [  
  {  
    "Effect": "Allow",  
    "Principal": "*",  
    "Action": "s3:GetObject",  
    "Resource": "arn:aws:s3:::your-bucket-name/*"  
  }  
]  
}
```

### **3. CodePipeline Errors:**

Issue: "Pipeline Failed at Build or Deploy Stage"

Solution:

Confirm that Source Repository credentials are correct (e.g., GitHub integration).

Check the logs in CodePipeline to identify any permission or build errors.

### **4. EC2 and CodeDeploy Errors:**

Issue: Deployment Fails on EC2 Instance

Solution:

Ensure the CodeDeploy agent is running on the EC2 instance.

Verify that the IAM role attached to the EC2 instance has permissions to receive deployments.

Ensure that port 80 is open in the EC2 security group for HTTP access.

### **5. Missing AppSpec File , after\_install.sh and before\_install.sh:**

The CodeDeploy process failed initially because the appspec.yml after\_install.sh and before\_install.sh: file was not placed correctly in the root directory of the build artifacts.

### **Guidelines and Best Practices:**

- Ensure all necessary IAM roles are configured with the right permissions.
- Use monitoring tools like CloudWatch to track deployment status and logs.
- Regularly update the CodeDeploy agent on the EC2 instance.

### **CONCLUSION:**

This experiment successfully demonstrated the process of building and deploying a simple web application using AWS CodePipeline, CodeBuild, CodeDeploy, S3, and EC2. The project showcased how automation can streamline the deployment workflow, starting from pushing code to the repository to hosting the application on an EC2 instance.

By utilizing CodePipeline, the entire CI/CD process was automated, ensuring smooth integration between different AWS services. CodeBuild efficiently compiled the code and prepared artifacts, CodeDeploy ensured seamless updates to the EC2 instance, and S3 was used to host static content.

This case study highlights the importance of cloud-based deployment and automation in modern web development, demonstrating how AWS services can simplify workflows, reduce human intervention, and improve deployment efficiency. The skills and insights gained through this project align with industry-standard DevOps practices, providing valuable experience in managing real-world cloud infrastructure.