

Task Description:

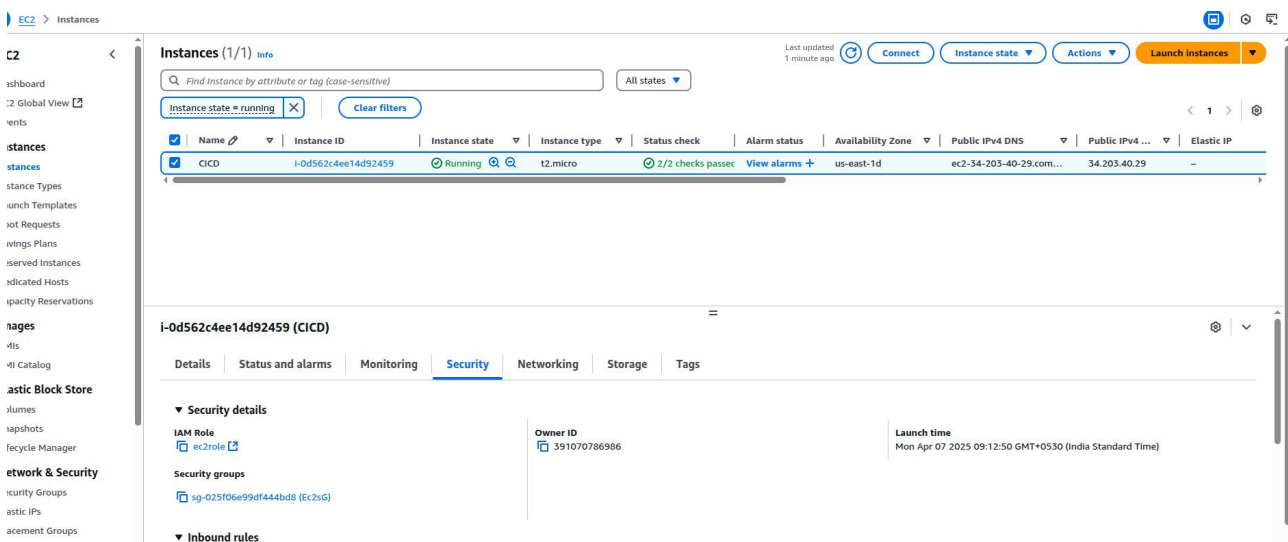
Deploy a simple web application using AWS code commit, code build and deploy & access via browser and automate via codepipeline.

Solution:

STEP 1: Prepare EC2 Instance (CodeDeploy Target)

1. Launch EC2 Instance

- Amazon Linux 2 or Ubuntu
- Assign a key pair
- Open ports: HTTP (80), SSH (22)
- Attach IAM Role with this policy: `AmazonEC2RoleforAWSCodeDeploy` (or create a custom one with CodeDeploy + S3 access)



The screenshot displays the AWS Management Console for EC2 Instances. The instance 'CICD' (ID: i-0d562c4ee14d92459) is shown in a 'Running' state. The console provides details for the instance, including its IAM role 'ec2role', security group 'sg-025f06e99df444bd8', and launch time 'Mon Apr 07 2025 09:12:50 GMT+0530 (India Standard Time)'. The instance is running on the 't2.micro' instance type in the 'us-east-1d' availability zone.

2. Install CodeDeploy Agent

We'll update your canvas with the proper script using the **correct region**.

Update your script in canvas to:

```
sudo yum update -y
sudo yum install ruby wget -y
cd /home/ec2-user
wget https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/install
chmod +x ./install
```

```
sudo ./install auto
sudo systemctl start codedeploy-agent
sudo systemctl enable codedeploy-agent
```

Replace `us-east-1` with your actual region.

STEP 2: Prepare GitHub Repo

Your GitHub repo should have:

```
my-web-app/
├── index.html (or app files)
├── buildspec.yml
├── appspec.yml
├── scripts/
│   └── restart_server.sh
```

Example `buildspec.yml`:

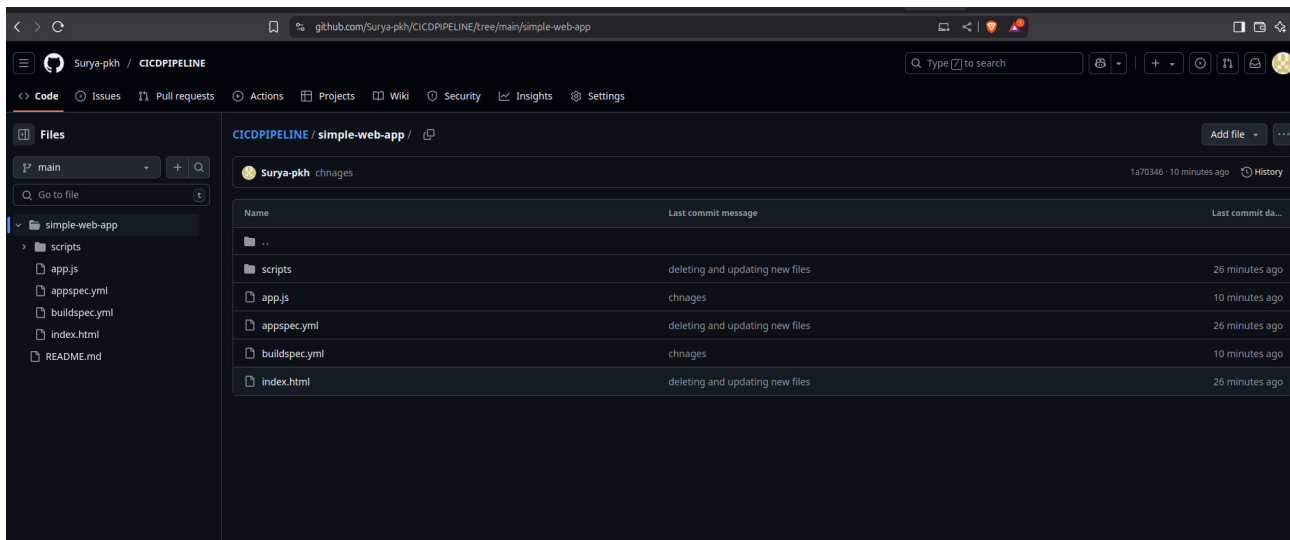
```
version: 0.2
phases:
  build:
    commands:
      - echo Build started on `date`
artifacts:
  files:
    - '**/*'
```

Example `appspec.yml`:

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
hooks:
  AfterInstall:
    - location: scripts/restart_server.sh
      timeout: 300
      runas: root
```

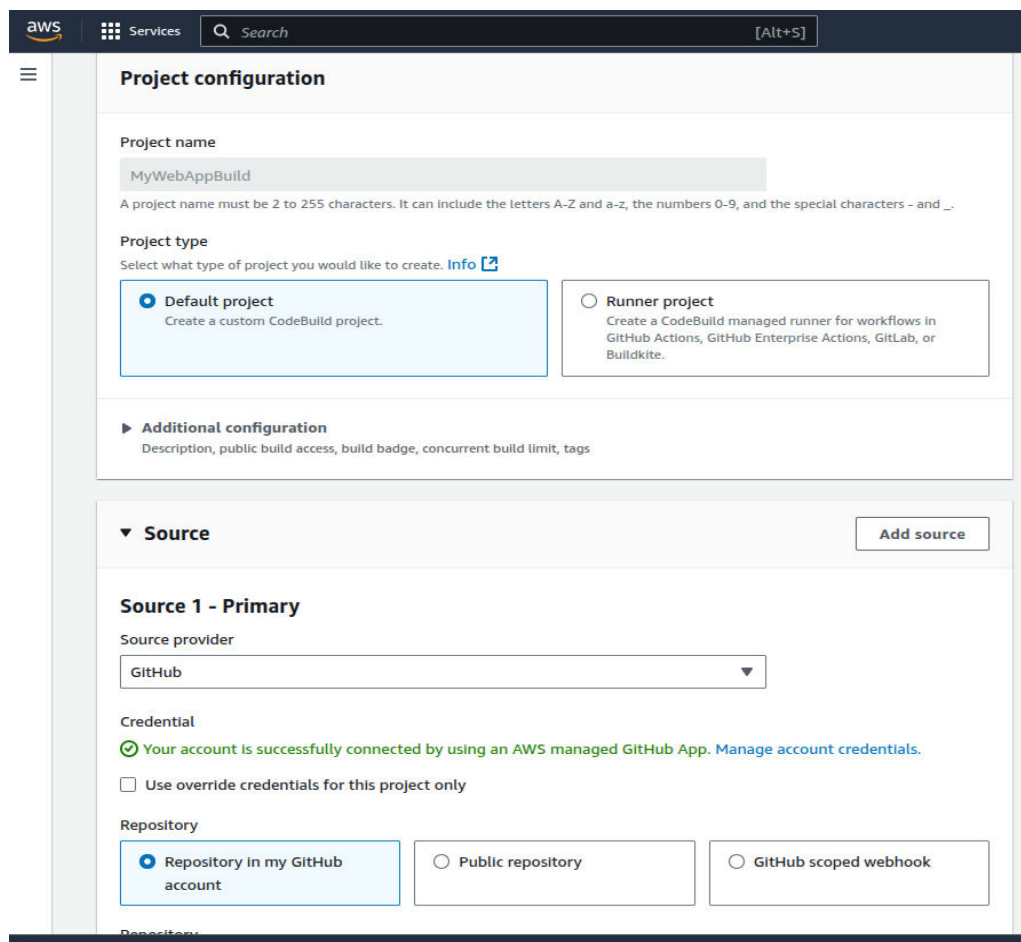
Example `restart_server.sh`:

```
#!/bin/bash
sudo systemctl restart httpd || sudo systemctl restart nginx
```



🔧 STEP 3: Set Up CodeBuild

1. Go to **CodeBuild > Create Build Project**
2. Set Name: MyWebAppBuild



3. Source Provider: **GitHub**

The screenshot shows the AWS CodePipeline console interface. At the top, there's a navigation bar with the AWS logo, a 'Services' menu, a search bar, and a '[Alt+S]' shortcut. Below the navigation bar, a description field contains the text: 'Description, public build access, build badge, concurrent build limit, tags'. The main content area is titled 'Source' and includes an 'Add source' button. Under the 'Source' section, 'Source 1 - Primary' is selected. The 'Source provider' dropdown is set to 'GitHub'. The 'Credential' section shows a green checkmark and the message: 'Your account is successfully connected by using an AWS managed GitHub App. Manage account credentials.' There is an unchecked checkbox for 'Use override credentials for this project only'. The 'Repository' section has three radio buttons: 'Repository in my GitHub account' (selected), 'Public repository', and 'GitHub scoped webhook'. Below this, the 'Repository' text input field contains 'https://github.com/Surya-pkh/CICDPIPELINE', with a search icon on the left, a clear icon (X) on the right, and a refresh icon (circular arrow) to the right of the field. The 'Source version - optional Info' section has a text input field with the placeholder 'Enter a pull request, branch, commit ID, tag, or reference and a commit ID.' Below this is an 'Additional configuration' section with a right-pointing triangle icon and the text 'Git clone depth, Git submodules, Build status config'. The bottom section is titled 'Primary source webhook events Info' and includes a 'Webhook - optional Info' link. A checked checkbox is labeled 'Rebuild every time a code change is pushed to this repository'.

aws Services Search [Alt+S]

Description, public build access, build badge, concurrent build limit, tags

▼ Source Add source

Source 1 - Primary

Source provider

GitHub

Credential

✓ Your account is successfully connected by using an AWS managed GitHub App. [Manage account credentials.](#)

☐ Use override credentials for this project only

Repository

☒ Repository in my GitHub account ☐ Public repository ☐ GitHub scoped webhook

Repository

https://github.com/Surya-pkh/CICDPIPELINE X ↻

Source version - optional [Info](#)

Enter a pull request, branch, commit ID, tag, or reference and a commit ID.

Additional configuration

Git clone depth, Git submodules, Build status config

▼ Primary source webhook events [Info](#)

Webhook - optional [Info](#)

☒ Rebuild every time a code change is pushed to this repository

4. Environment:


- Managed image: Amazon Linux
- Runtime: Node.js or standard
- Allow AWS to create a new role


The screenshot displays the AWS CodeBuild 'Environment' configuration interface. The top navigation bar includes the AWS logo, 'Services', a search bar, and a '[Alt+S]' shortcut. The left sidebar shows a menu icon. The main content area is titled 'Environment' and contains several sections:

- Provisioning model**: Includes an 'Info' link and two options:
 - On-demand** (selected): Automatically provision build infrastructure in response to new builds.
 - Reserved capacity**: Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.
- Environment image**: Includes two options:
 - Managed image** (selected): Use an image managed by AWS CodeBuild.
 - Custom image**: Specify a Docker image.
- Compute**: Includes two options:
 - EC2** (selected): Optimized for flexibility during action runs.
 - Lambda**: Optimized for speed and minimizes the start up time of workflow actions.
- Running mode**: Includes two options:
 - Container** (selected): Running on Docker container.
 - Instance**: Running on EC2 Instance directly.
- Operating system**: A dropdown menu showing 'Amazon Linux'.
- Runtime(s)**: A dropdown menu showing 'Standard'.
- Image**: A dropdown menu showing 'aws/codebuild/amazonlinux-x86_64-standard:5.0'.
- Image version**: A dropdown menu showing 'Always use the latest image for this runtime version'.

5. Artifacts:

- Type: Amazon S3 (create a new bucket)

 Services [Alt+S]



▼ Artifacts Add artifact

Artifact 1 - Primary

Type

Amazon S3 ▼

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

Bucket name

Name

The name of the folder or compressed file in the bucket that will contain your output artifacts. Use Artifacts packaging under Additional configuration to choose whether to use a folder or compressed file. If the name is not provided, defaults to project name.

☐ **Enable semantic versioning**
Use the artifact name specified in the buildspec file

Path - optional
The path to the build output ZIP file or folder.

Example: MyPath/MyArtifact.zip.

Namespace type - optional

None ▼

Choose Build ID to insert the build ID into the path to the build output ZIP file or folder, e.g. MyPath/MyBuildID/MyArtifact.zip. Otherwise, choose None.

Artifacts packaging

☒ **None**
The artifact files will be uploaded to the bucket.

☐ **Zip**
AWS CodeBuild will upload artifacts into a compressed file that is put into the specified bucket.

6. Buildspec: From source (buildspec.yml)

aws Services Search [Alt+S]

that is put into the specified bucket.

☐ **Disable artifact encryption**
Disable encryption if using the artifact to publish a static website or sharing content with others

► **Additional configuration**
Cache, encryption key

▼ **Logs**

CloudWatch

☐ **CloudWatch logs - optional**
Checking this option will upload build output logs to CloudWatch.

S3

☐ **S3 logs - optional**
Checking this option will upload build output logs to S3.

▼ **Service role permissions**

Service role
Choose an existing service role from your account

arn:aws:iam::391070786986:role/service-role/codebuild-MyWebAppBuild-serv X

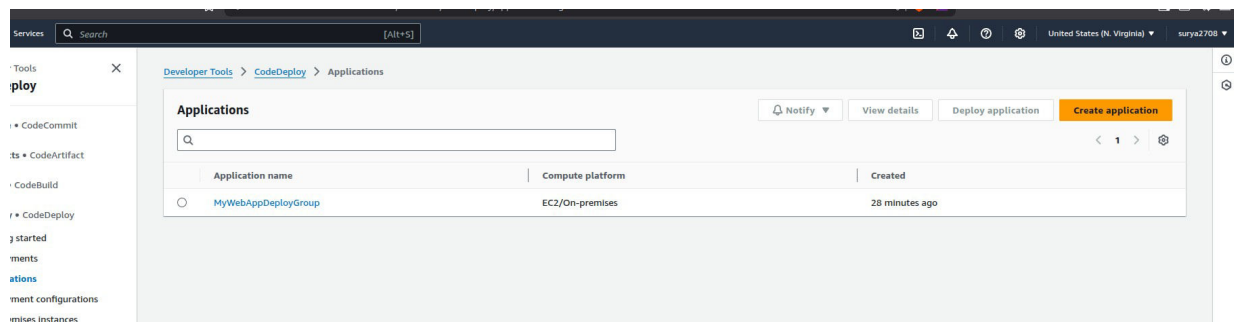
☒ **Allow AWS CodeBuild to modify this service role so it can be used with this build project**

Cancel Update project



STEP 4: Set Up CodeDeploy

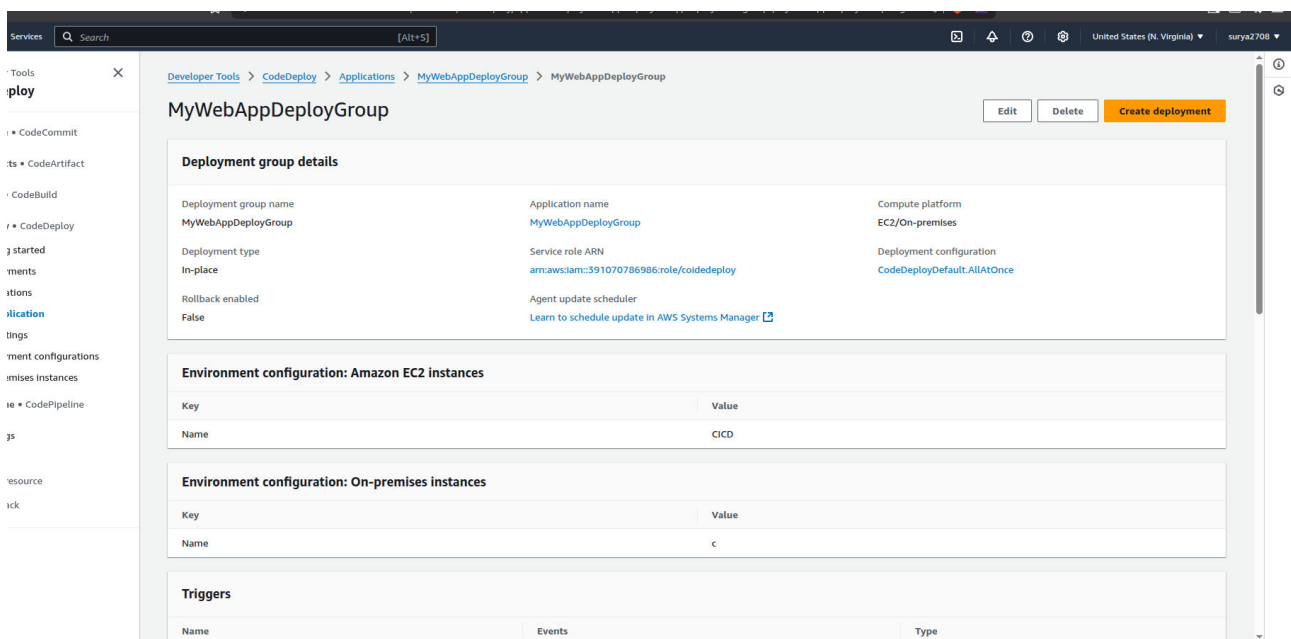
1. Go to **CodeDeploy** > **Applications** > **Create application**



2. Compute Platform: EC2/On-premise

3. Create Deployment Group:

- Name: MyWebAppDeployGroup
- Service role: Create a role with `AWSCodeDeployRole`
- Target: EC2 instance (tag-based or manually attach)



STEP 5: Create CodePipeline

1. Go to **CodePipeline** > **Create pipeline**
2. Pipeline Name: MyWebAppPipeline
3. Source:

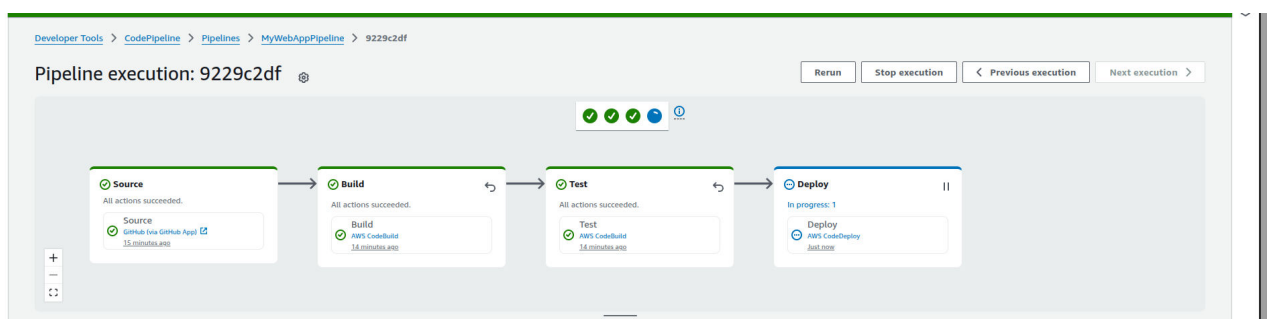
- Provider: **GitHub**
- Connect and select your repo/branch

4. Build:

- Provider: **CodeBuild**
- Choose the project MyWebAppBuild

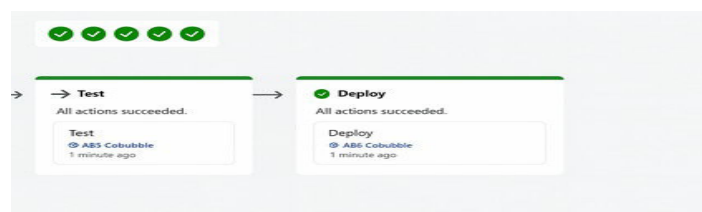
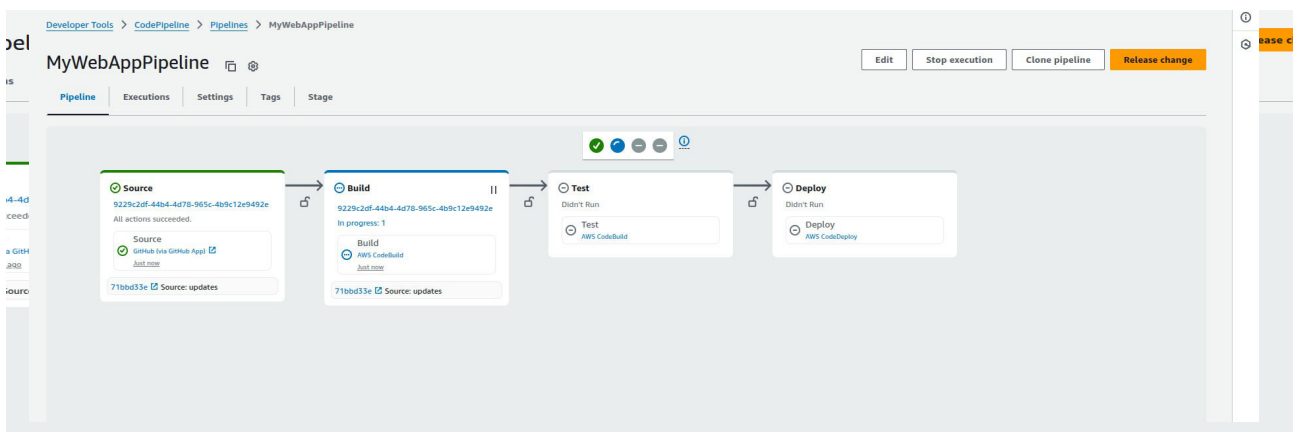
5. Deploy:

- Provider: **CodeDeploy**
- Choose the app and deployment group

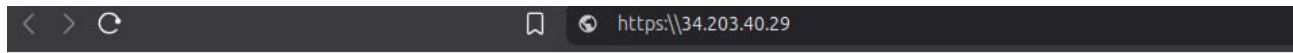


✓ STEP 6: Trigger Deployment

- Push any code change to GitHub.
- CodePipeline will automatically build and deploy to EC2.



- Visit: `http://<your-ec2-public-ip>` to see your web app.



Hello, World!
