**Code Deploy / CICD Pipeline**

#### **1) Create a CodeDeploy Application**

1. Navigate to **CodeDeploy** and click **Create Application**.
   * **Application Name**: demo-app-application
   * **Compute Platform**: EC2
2. Select **Create Application** to save the setup.
3. Next, on the same page, select **Create Deployment Group**.
   * **Deployment Group Name**: demo-app-application-group
   * **Service Role**: Attach a role with the following policies:
     + AmazonEC2FullAccess
     + AmazonEC2RoleforAWSCodeDeploy
     + AmazonS3FullAccess
     + AWSCodeDeployFullAccess
     + AWSCodeDeployRole
     + Attach the role’s ARN to **Service Role** in the deployment group configuration.
   * **Deployment Type**: In-Place
   * **Environment Configuration**: Amazon EC2
4. Create a new EC2 instance named demo-app-ec2.
   * **Key Name**: Add tags with key Name and value demo-app-ec2.

#### **2) Install the CodeDeploy Agent**

The CodeDeploy Agent manages deployments by listening to deployment instructions and executing tasks like file copying and service restarts.

Connect to your EC2 instance and run the following bash script to install the CodeDeploy agent:  
bash  
Copy code  
#!/bin/bash

sudo apt --fix-broken install

sudo apt-get update

sudo apt-get install -y ruby3.0

cd /home/ubuntu

wget https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/install

chmod +x ./install

sudo ./install auto

sudo service codedeploy-agent start

sudo systemctl status codedeploy-agent

1. Verify the us-east-1 region in the URL matches your deployment region.

#### **3) Project Directory Setup**

1. Within your project directory, create a folder named scripts, and add the following files:

**start\_nginx.sh**:  
bash  
Copy code  
#!/bin/bash

sudo service nginx start

**install\_nginx.sh**:  
bash  
Copy code  
#!/bin/bash

sudo apt-get update

sudo apt-get install -y nginx

Create the **appspec.yml** file in your project root:  
yaml  
Copy code  
version: 0.0

os: linux

files:

- source: /

destination: /var/www/html/

hooks:

AfterInstall:

- location: scripts/install\_nginx.sh

timeout: 300

runas: root

ApplicationStart:

- location: scripts/start\_nginx.sh

timeout: 300

runas: root

1. Commit and push these files to your **CodeCommit** repository.
2. Run **CodeBuild** to build your code and push artifacts to an S3 bucket. Copy the S3 bucket URL for CodeDeploy.

#### **4) Create a Deployment in CodeDeploy**

1. In **CodeDeploy**, select your application and start a deployment.
   * **Deployment Group**: This will auto-populate with the deployment group name.
   * **Revision Type**: S3
   * **Revision Location**: Paste the S3 URL (e.g., s3://code-build-artifact-1209333/build-specs-folder/buildspec-output.zip).
2. Ensure your EC2 instance has the necessary permissions by attaching a role with the following policies:
   * AmazonEC2FullAccess
   * AmazonS3FullAccess
   * AWSCodeDeployFullAccess
3. Test the deployment by checking the public IP of your EC2 instance in a browser.

Restart the CodeDeploy agent if necessary:  
bash  
Copy code  
sudo service codedeploy-agent restart

#### **5) CodePipeline Setup**

1. Go to **CodePipeline** and select **Create Pipeline**.
   * **Pipeline Name**: Choose a unique name.
   * **Service Role**: New Service Role
2. Select **Next**.
3. Configure Source:
   * **Source Provider**: AWS CodeCommit
   * **Repository**: Select the demoapp repository
   * **Branch**: master
   * **Change Detection Options**: AWS CodePipeline
4. Configure Build:
   * **Build Provider**: AWS CodeBuild
   * **Project Name**: demo-app-build
   * **Build Type**: Single Build
5. Configure Deploy:
   * **Deploy Provider**: AWS CodeDeploy
   * **Application Name**: demo-app-application
   * **Deployment Group**: demo-app-deploy-grp
6. Click **Next** to complete your pipeline configuration.

Your CodePipeline will now trigger automatically whenever changes are pushed to CodeCommit, streamlining your CI/CD process.