**Create a CI-CD Using CodeCommit, CodeBuild, CodeDeploy and CodePipeline on AWS:**

In this section, you use CodePipeline to deploy code maintained in a CodeCommit repository to a single Amazon EC2 instance. Your pipeline is triggered when you push a change to the CodeCommit repository. The pipeline deploys your changes to an Amazon EC2 instance using CodeDeploy as the deployment service.

The pipeline has two stages:

A source stage (**Source**) for your CodeCommit source action.

A deployment stage (**Deploy**) for your CodeDeploy deployment action.

Step 1: Create a CodeCommit Repository

First, you create a repository in CodeCommit. Your pipeline gets source code from this repository when it runs. You also create a local repository where you maintain and update code before you push it to the CodeCommit repository.

**To create a CodeCommit repository**

1. Open the CodeCommit console at <https://console.aws.amazon.com/codecommit/>.
2. In the Region selector, choose the AWS Region where you want to create the repository and pipeline. For more information, see [AWS Regions and Endpoints](https://docs.aws.amazon.com/general/latest/gr/rande.html).
3. On the **Repositories** page, choose **Create repository**.
4. On the **Create repository** page, in **Repository name**, enter a name for your repository (for example, **MyDemoRepo**).
5. Choose **Create**.

**To set up a local repository**

In this step, you set up a local repository to connect to your remote CodeCommit repository.

1. With your new repository open in the console, choose **Clone URL** on the top right of the page, and then choose **Clone SSH**. The address to clone your Git repository is copied to your clipboard.
2. In your terminal or command line, navigate to a local directory where you'd like your local repository to be stored. In this tutorial, we use /tmp.
3. Run the following command to clone the repository, replacing the SSH address with the one you copied in the previous step. This command creates a directory called MyDemoRepo. You copy a sample application to this directory.

git clone ssh://git-codecommit.us-west-2.amazonaws.com/v1/repos/MyDemoRepo

## Step 2: Add Sample Code to Your CodeCommit Repository

In this step, you download code for a sample application that was created for a CodeDeploy sample walkthrough, and add it to your CodeCommit repository.

1. Download the following file: [SampleApp\_Linux.zip](https://docs.aws.amazon.com/codepipeline/latest/userguide/samples/SampleApp_Linux.zip)
2. Unzip the files from [SampleApp\_Linux.zip](https://docs.aws.amazon.com/codepipeline/latest/userguide/samples/SampleApp_Linux.zip) into the local directory you created earlier (for example, /tmp/MyDemoRepo or c:\temp\MyDemoRepo).

Change directories to your local repo:

(For Linux, macOS, or Unix) cd /tmp/MyDemoRepo

(For Windows) cd c:\temp\MyDemoRepo

Run the following command to stage all of your files at once:

git add -A

Run the following command to commit the files with a commit message:

git commit -m "Add sample application files"

Run the following command to push the files from your local repo to your CodeCommit repository:

git push

The files you downloaded and added to your local repo have now been added to the master branch in your CodeCommit MyDemoRepo repository and are ready to be included in a pipeline.

## Step 3: Create an Amazon EC2 Linux Instance and Install the CodeDeploy Agent

In this step, you create the Amazon EC2 instance where you deploy a sample application. As part of this process, you install the CodeDeploy agent on the Amazon EC2 instance. The CodeDeploy agent is a software package that enables an instance to be used in CodeDeploy deployments. You also attach an IAM role to the instance (known as an instance role) to allow it to fetch files that the CodeDeploy agent uses to deploy your application.

**To create an instance role**

1. Open the IAM console at <https://console.aws.amazon.com/iam/>).
2. From the console dashboard, choose **Roles**.
3. Choose **Create role**.
4. Under **Choose the service that will use this role**, select **EC2**, and then choose **Next: Permissions**.
5. Search for and select the policy named **AmazonEC2RoleforAWSCodeDeploy**, and then choose **Next: Tags**.
6. Choose **Next: Review**. Enter a name for the role (for example, **EC2InstanceRole**), and then choose **Create role**.

**To launch an instance**

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. From the console dashboard, choose **Launch Instance**.
3. On **Step 1: Choose an Amazon Machine Image (AMI)**, locate **Amazon Linux 2 AMI (HVM)**, and then choose **Select**. (This AMI is labeled "Free tier eligible" and can be found at the top of the list.)
4. On the **Step 2: Choose an Instance Type** page, choose the free tier eligible t2.micro type as the hardware configuration for your instance, and then choose **Next: Configure Instance Details**.
5. On the **Step 3: Configure Instance Details** page, do the following:
   * In **Number of instances**, enter 1.
   * In **Auto-assign Public IP**, choose **Enable**.
   * In **IAM role**, choose the IAM role you created in the previous procedure (for example, **EC2InstanceRole**).
6. Expand **Advanced Details**, and in the **User data** field, enter the following:

#!/bin/bash

yum -y update

yum install -y ruby

yum install -y aws-cli

cd /home/ec2-user

aws s3 cp s3://aws-codedeploy-us-east-2/latest/install . --region us-east-2

chmod +x ./install

./install auto

This code installs the CodeDeploy agent on your instance as it is created.

1. Leave the rest of the items on the **Step 3: Configure Instance Details** page unchanged. Choose **Next: Add Storage**, leave the **Step 4: Add Storage** page unchanged, and then choose **Next: Add Tags**.
2. Choose **Add Tag**. In **Key**, enter **MyCodePipelineDemo**, and then choose **Next: Configure Security Group**. Later, you create a CodeDeploy application that deploys the sample application to this instance. CodeDeploy selects instances to deploy based on the tags that are attached to instances.
3. On the **Step 6: Configure Security Group** page, do the following:
   * Next to **Assign a security group**, choose **Create a new security group**.
   * In the row for **SSH**, under **Source**, choose **My IP**.
   * Choose **Add Rule**, choose **HTTP**, and then under **Source**, choose **My IP**.
4. Choose **Review and Launch**.
5. On the **Review Instance Launch** page, choose **Launch**, and then do one of the following when prompted for a key pair:
   * If you already have a key pair to use with Amazon EC2 instances, select **Choose an existing key pair**, and then select your key pair.
   * If you have not created a key pair yet, select **Create a new key pair**, enter a name for the key pair, and then choose **Download Key Pair**. This is your only chance to save the private key file. Be sure to download it. Save the private key file in a safe place. You must provide the name of your key pair when you launch an instance. You must provide the corresponding private key each time you connect to the instance.

When you are ready, select the acknowledgement check box, and then choose **Launch Instances**.

1. Choose **View Instances** to close the confirmation page and return to the console.
2. You can view the status of the launch on the **Instances** page. When you launch an instance, its initial state is pending. After the instance starts, its state changes to running, and it receives a public DNS name. (If the **Public DNS** column is not displayed, choose the **Show/Hide** icon, and then select **Public DNS**.)
3. It can take a few minutes for the instance to be ready for you to connect to it. View the information in the **Status Checks** column to see if your instance has passed its status checks.

## Step 4: Create an Application in CodeDeploy

In CodeDeploy, an [application](https://docs.aws.amazon.com/codedeploy/latest/userguide/applications.html) is a resource that contains the software application you want to deploy. Later, you use this application with CodePipeline to automate deployments of the sample application to your Amazon EC2 instance.

First, you create a role that allows CodeDeploy to perform deployments. Then, you create a CodeDeploy application.

**To create a CodeDeploy service role**

1. Open the IAM console at <https://console.aws.amazon.com/iam/>).
2. From the console dashboard, choose **Roles**.
3. Choose **Create role**.
4. Under **Choose the service that will use this role**, choose **CodeDeploy**. Under **Select your use case** , choose **CodeDeploy**.
5. Choose **Next: Permissions**, **Next: Tags**, and **Next: Review**.
6. Enter a name for the role (for example, **CodeDeployRole**), and then choose **Create role**.

**To create an application in CodeDeploy**

1. Open the CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.
2. If the **Applications** page does not appear, on the AWS CodeDeploy menu, choose **Applications**.
3. Choose **Create application**.
4. Leave **Custom application** selected. In **Application name**, enter **MyDemoApplication**.
5. In **Compute Platform**, choose **EC2/On-premises**.
6. Choose **Create application**.

**To create a deployment group in CodeDeploy**

A [deployment group](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-groups.html) is a resource that defines deployment-related settings like which instances to deploy to and how fast to deploy them.

1. On the page that displays your application, choose **Create deployment group**.
2. In **Deployment group name**, enter **MyDemoDeploymentGroup**.
3. In **Service Role**, choose the service role you created earlier (for example, **CodeDeployRole**).
4. Under **Deployment type**, choose **In-place**.
5. Under **Environment configuration**, choose **Amazon EC2 Instances**. In the **Key** field, enter the tag key you used to tag the instance (for example, **MyCodePipelineDemo**).
6. Under **Deployment configuration**, choose CodeDeployDefault.OneAtaTime.
7. Under **Load Balancer**, clear **Enable load balancing**. You do not need to set up a load balancer or choose a target group for this example.
8. Expand the **Advanced** section. Under **Alarms**, if any alarms are listed, choose **Ignore alarm configuration**.
9. Choose **Create deployment group**.

## Step 5: Create Your First Pipeline in CodePipeline

You're now ready to create and run your first pipeline. In this step, you create a pipeline that runs automatically when code is pushed to your CodeCommit repository.

1. Sign in to the AWS Management Console and open the CodePipeline console at <http://console.aws.amazon.com/codesuite/codepipeline/home>.

Open the CodePipeline console at <https://console.aws.amazon.com/codepipeline/>.

1. Choose **Create pipeline**.
2. In **Step 1: Choose pipeline settings**, in **Pipeline name**, enter **MyFirstPipeline**.
3. In **Service role**, choose **New service role** to allow CodePipeline to create a service role in IAM.
4. In **Artifact store**, choose **Default location**, and then choose **Next**.
5. In **Step 2: Add source stage**, in **Source provider**, choose **AWS CodeCommit**. In **Repository name**, choose the name of the CodeCommit repository you created in [Step 1: Create a CodeCommit Repository](https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-simple-codecommit.html#codecommit-create-repository). In **Branch name**, choose master, and then choose **Next step**.

After you select the repository name and branch, a message displays the Amazon CloudWatch Events rule to be created for this pipeline.

Under **Change detection options**, leave the defaults. This allows CodePipeline to use Amazon CloudWatch Events to detect changes in your source repository.

Choose **Next**.

In **Step 3: Add build stage**, choose **Skip build stage**, and then accept the warning message by choosing **Skip** again. Choose **Next**.

In **Step 4: Add deploy stage**, in **Deploy provider**, choose **AWS CodeDeploy**. In **Application name**, choose **MyDemoApplication**. In **Deployment group**, choose **MyDemoDeploymentGroup**, and then choose **Next step**.

In **Step 5: Review**, review the information, and then choose **Create pipeline**.

The pipeline starts running after it is created. It downloads the code from your CodeCommit repository and creates a CodeDeploy deployment to your Amazon EC2 instance. You can view progress and success and failure messages as the CodePipeline sample deploys the webpage to the Amazon EC2 instance in the CodeDeploy deployment.

Congratulations! You just created a simple pipeline in CodePipeline.

**To verify that your pipeline ran successfully**

1. View the initial progress of the pipeline. The status of each stage changes from **No executions yet** to **In Progress**, and then to either **Succeeded** or **Failed**. The pipeline should complete the first run within a few minutes.
2. After **Succeeded** is displayed for the pipeline status, in the status area for the **Staging** stage, choose **Details**. This opens the CodeDeploy console. If **Succeeded** is not displayed see [Troubleshooting CodePipeline](https://docs.aws.amazon.com/codepipeline/latest/userguide/troubleshooting.html).
3. Choose your application in the list. On the **Deployment group** tab, under **Deployment lifecycle events**, choose the instance ID. This opens the EC2 console.
4. On the **Description** tab, in **Public DNS**, copy the address (for example, ec2-192-0-2-1.us-west-2.compute.amazonaws.com), and then paste it into the address bar of your web browser.

This is the sample application you downloaded and pushed to your CodeCommit repository.

## Step 6: Modify Code in Your CodeCommit Repository

Your pipeline is configured to run whenever code changes are made to your CodeCommit repository. In this step, you make changes to the HTML file that is part of the sample CodeDeploy application in the CodeCommit repository. When you push these changes, your pipeline runs again, and the changes you make are visible at the web address you accessed earlier.