DevOps Project:

Create a simple pipeline (CodeCommit repository)

In this tutorial, 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.

The easiest way to get started with AWS CodePipeline is to use the **Create Pipeline** wizard in the CodePipeline console.

Note

Before you begin, make sure you've set up your Git client to work with CodeCommit. For instructions, see Setting up for CodeCommit.

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.
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**.

Note

The remaining steps in this tutorial use MyDemoRepo for the name of your CodeCommit repository. If you choose a different name, be sure to use it throughout this tutorial.

To set up a local repository

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

Note

You are not required to set up a local repository. You can also use the console to upload files as described in Step 2

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
2. Unzip the files from SampleApp_Linux.zip into the local directory you created earlier (for example, /tmp/MyDemoRepo or c:\temp\MyDemoRepo).

Be sure to place the files directly into your local repository. Do not include a SampleApp_Linux folder. On your local Linux, macOS, or Unix machine, for example, your directory and file hierarchy should look like this:

```
/tmp
    └-- MyDemoRepo
        |-- appspec.yml
        |-- index.html
        |-- LICENSE.txt
        └-- scripts
            |-- install_dependencies
            |-- start_server
            └-- stop_server
```

3. To upload files to your repository, use one of the following methods.
    a. To use the CodeCommit console to upload your files:
        i. Open the CodeCommit console, and choose your repository from the **Repositories** list.
        ii. Choose **Add file**, and then choose **Upload file**.
        iii. Select **Choose file**, and then browse for your file. To add a file under a folder, choose **Create file** and then enter the folder name with the file name, such as scripts/install_dependencies. Paste the file contents into the new file.

        Commit the change by entering your user name and email address.

        Choose **Commit changes**.
        iv. Repeat this step for each file.

        Your repository contents should look like this:

```
        |-- appspec.yml
        |-- index.html
```

```
|-- LICENSE.txt
└-- scripts
   |-- install_dependencies
   |-- start_server
   └-- stop_server
```

b. To use git commands to upload your files:
   i. Change directories to your local repo:
   ii. *(For Linux, macOS, or Unix)* cd /tmp/MyDemoRepo

   (For Windows) cd c:\temp\MyDemoRepo

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

   ```
   git add -A
   ```

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

   ```
   git commit -m "Add sample application files"
   ```

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

   ```
   git push
   ```

4. The files you downloaded and added to your local repo have now been added to the main 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, create an instance role that allows install and management of the CodeDeploy agent on the instance. The CodeDeploy agent is a software package that enables an instance to be used in CodeDeploy deployments. You also attach policies that allow the instance to fetch files that the CodeDeploy agent uses to deploy your application and to allow the instance to be managed by SSM.

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 **Select type of trusted entity**, select **AWS service**. Under **Choose a use case**, select **EC2**. Under **Select your use case**, choose **EC2**. Choose **Next: Permissions**.
5. Search for and select the policy named **AmazonEC2RoleforAWSCodeDeploy.**
6. Search for and select the policy named **AmazonSSMManagedInstanceCore**. Choose **Next: Tags**.
7. Choose **Next: Review**. Enter a name for the role (for example, **EC2InstanceRole**).

   Note

   Make a note of your role name for the next step. You choose this role when you are creating your instance.

   Choose **Create role**.

To launch an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the side navigation, choose **Instances**, and select **Launch instances** from the top of the page.
3. In **Name**, enter **MyCodePipelineDemo**. This assigns the instance a tag **Key** of **Name** and a tag **Value** of **MyCodePipelineDemo**. Later, you create a CodeDeploy application that deploys the sample application to this instance. CodeDeploy selects instances to deploy based on the tags.
4. Under **Application and OS Images (Amazon Machine Image)**, locate the **Amazon Linux** AMI option with the AWS logo, and make sure it is selected. (This AMI is described as the Amazon Linux 2 AMI (HVM) and is labeled "Free tier eligible".)
5. Under **Instance type**, choose the free tier eligible t2.micro type as the hardware configuration for your instance.
6. Under **Key pair (login)**, choose a key pair or create one.

   You can also choose **Proceed without a key pair**.

   Note

   For the purposes of this tutorial, you can proceed without a key pair. To use SSH to connect to your instances, create or use a key pair.

7. Under **Network settings**, do the following.

   In **Auto-assign Public IP**, make sure the status is **Enable**.
   - Next to **Assign a security group**, choose **Create a new security group**.
   - In the row for **SSH**, under **Source type**, choose **My IP**.
   - Choose **Add security group**, choose **HTTP**, and then under **Source type**, choose **My IP**.
8. Expand **Advanced details**. In **IAM instance profile**, choose the IAM role you created in the previous procedure (for example, EC2InstanceRole).
9. Under **Summary**, under **Number of instances**, enter 1..
10. Choose **Launch instance**.
11. 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**.)

Step 4: Create an application in CodeDeploy

In CodeDeploy, an *application* 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 **Select trusted entity**, choose **AWS service**. Under **Use case**, choose **CodeDeploy**. Choose **CodeDeploy** from the options listed. Choose **Next**. The AWSCodeDeployRole managed policy is already attached to the role.
5. Choose **Next**.
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 menu, choose **Applications**.
3. Choose **Create application**.
4. 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* 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 Name. In the **Value** field, enter the name you used to tag the instance (for example, MyCodePipelineDemo).
6. Under **Agent configuration with AWS Systems Manager**, choose **Now and schedule updates**. This installs the agent on the instance. The Linux instance is already configured with the SSM agent and will now be updated with the CodeDeploy agent.
7. Under **Deployment configuration**, choose CodeDeployDefault.OneAtaTime.
8. Under **Load Balancer**, make sure **Enable load balancing** is not selected. You do not need to set up a load balancer or choose a target group for this example.
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.

To create a CodePipeline pipeline

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/.
2. Choose **Create pipeline**.
3. In **Step 1: Choose pipeline settings**, in **Pipeline name**, enter MyFirstPipeline.
4. In **Service role**, choose **New service role** to allow CodePipeline to create a service role in IAM.
5. Leave the settings under **Advanced settings** at their defaults, and then choose **Next**.
6. In **Step 2: Add source stage**, in **Source provider**, choose **CodeCommit**. In **Repository name**, choose the name of the CodeCommit repository you created in Step 1: Create a CodeCommit repository. In **Branch name**, choose main, 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.

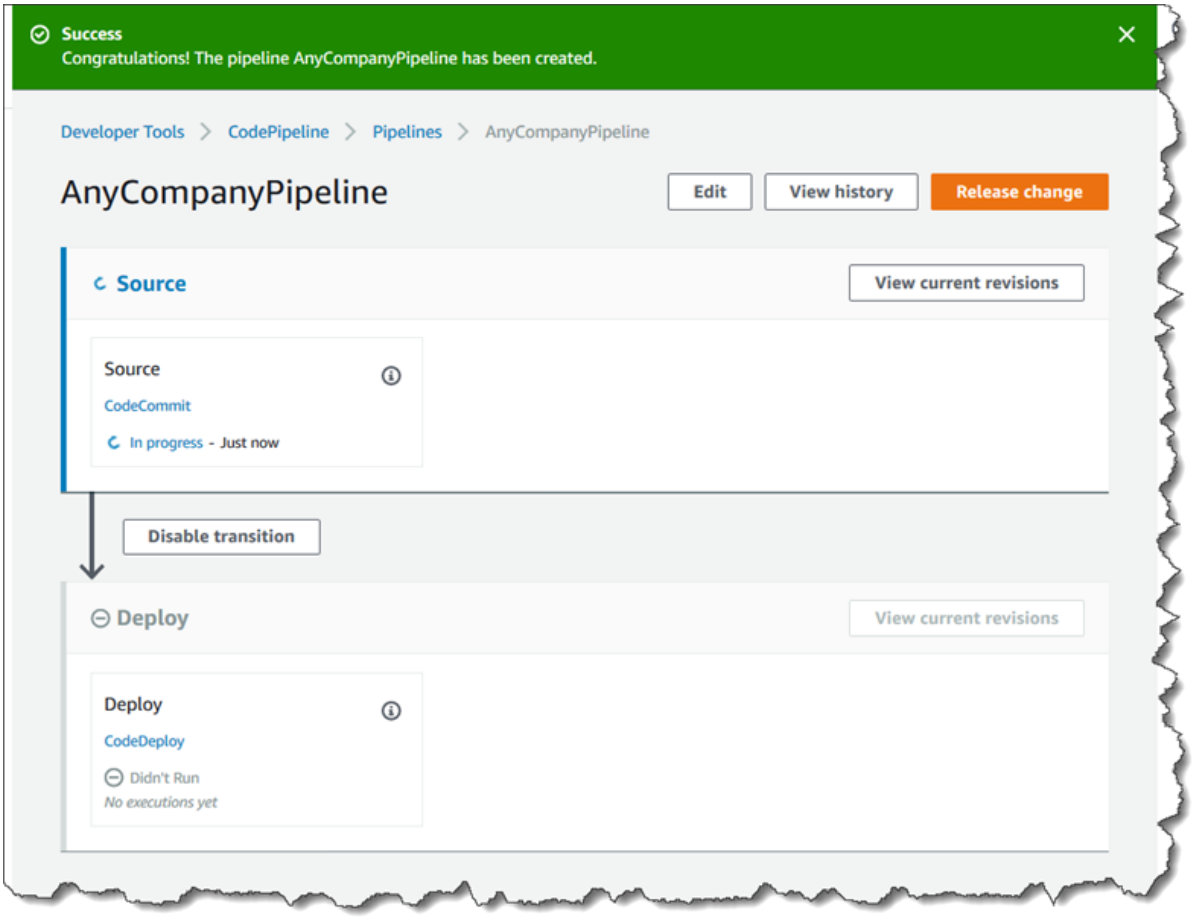
Choose **Next**.

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

   Note

   In this tutorial, you are deploying code that requires no build service, so you can skip this step. However, if your source code needs to be built before it is deployed to instances, you can configure CodeBuild in this step.

8. In **Step 4: Add deploy stage**, in **Deploy provider**, choose **CodeDeploy**. In **Application name**, choose **MyDemoApplication**. In **Deployment group**, choose **MyDemoDeploymentGroup**, and then choose **Next step**.
9. In **Step 5: Review**, review the information, and then choose **Create pipeline**.
10. The pipeline starts running after it is created. It downloads the code from your CodeCommit repository and creates a CodeDeploy deployment to your 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



Congratulations! You just created a simple pipeline in CodePipeline.

Next, you verify the results.

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 **Deploy** stage, choose **CodeDeploy**. This opens the CodeDeploy console. If**Succeeded** is not displayed see Troubleshooting CodePipeline.
3. On the **Deployments** tab, choose the deployment ID. On the page for the deployment, 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 or the public IPv4 with http), and then paste it into the address bar of your web browser.

   The web page displays for 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.

1. Change directories to your local repo:
2. *(For Linux, macOS, or Unix)* cd /tmp/**MyDemoRepo**

   (For Windows) cd c:\temp**MyDemoRepo**
3. Use a text editor to modify the index.html file:
4. *(For Linux or Unix)*gedit index.html
5. *(For OS X)*open –e index.html

   *(For Windows)*notepad index.html
6. Revise the contents of the index.html file to change the background color and some of the text on the webpage, and then save the file.

```
<!DOCTYPE html>
<html>
<head>
<title>Updated Sample Deployment</title>
<style>
  body {
    color: #000000;
    background-color: #CCFFCC;
    font-family: Arial, sans-serif;
    font-size:14px;
```

```
17.    }
18.
19.    h1 {
20.      font-size: 250%;
21.      font-weight: normal;
22.      margin-bottom: 0;
23.    }
24.
25.    h2 {
26.      font-size: 175%;
27.      font-weight: normal;
28.      margin-bottom: 0;
29.    }
30.  </style>
31. </head>
32. <body>
33.   <div align="center"><h1>Updated Sample Deployment</h1></div>
34.   <div align="center"><h2>This application was updated using CodePipeline, CodeCommit, and
      CodeDeploy.</h2></div>
35.   <div align="center">
36.     <p>Learn more:</p>
37.     <p><a href="https://docs.aws.amazon.com/codepipeline/latest/userguide/">CodePipeline User
      Guide</a></p>
```

```
38.    <p><a href="https://docs.aws.amazon.com/codecommit/latest/userguide/">CodeCommit User
       Guide</a></p>

39.    <p><a href="https://docs.aws.amazon.com/codedeploy/latest/userguide/">CodeDeploy User
       Guide</a></p>

40.  </div>

41. </body>

    </html>
```

42. Commit and push your changes to your CodeCommit repository by running the following commands, one at a time:

```
git commit -am "Updated sample application files"

git push
```

To verify 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 running of the pipeline should be complete within a few minutes.
2. After **Succeeded** is displayed for the action status, refresh the demo page you accessed earlier in your browser.

   The updated webpage is displayed.