AWS_Code_pipeline

Creating Simple Pipeline

(AWS CodeCommit Repository)

The easiest way to get started with AWS CodePipeline is to use the **Create Pipeline** wizard in the AWS CodePipeline console to create a simple pipeline.
In this lab, you use AWS CodePipeline to deploy code that is maintained in an AWS CodeCommit repository to a single Amazon EC2 instance. You will use AWS CodeDeploy as the deployment service.
Not what you're looking for? To create a simple pipeline using a versioned Amazon S3 bucket as a code repository, see Tutorial: Create a Simple Pipeline (Amazon S3 Bucket).
After you complete this tutorial, you should have enough practice with AWS CodeCommit concepts to use it as a repository in your pipelines.
AWS CodePipeline uses Amazon CloudWatch Events to detect changes in your AWS CodeCommit source repository and branch. Using Amazon CloudWatch Events to start your pipeline when changes occur is the default for this source type. When you use the wizard in the console to create a pipeline, the rule is created for you.
Before you begin, make sure you have completed the following tasks:

- Configured an IAM user
- Installed and configured the AWS CLI
- Created your key pair using Amazon EC2

In addition, make sure to complete these service-specific tasks:

- AWS CodeCommit: Install Git and configure credentials
- AWS CodeDeploy: Create an IAM instance profile and an AWS CodeDeploy service role
- AWS CodePipeline: Assign AWS CodePipeline permissions to the IAM user role

Note
If you have already completed the Tutorial: Create a Simple Pipeline (Amazon S3 Bucket) tutorial, but have not yet cleaned up its resources, you must create different names for many of the resources you used in that tutorial. For example, instead of **MyFirstPipeline**, you might name your pipeline **MySecondPipeline**.

Topics

Step 1: Create an Amazon EC2 Linux Instance and Install the AWS 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 AWS CodeDeploy agent on the instance. The AWS CodeDeploy agent is a software package that enables an instance to be used in AWS CodeDeploy deployments.
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 the **Step 1: Choose an Amazon Machine Image (AMI)** page, locate the row for the HVM edition of the Amazon Linux AMI, and then choose **Select**. (This AMI is labeled "Free tier eligible" and can be found at the top of the list.)

Note

> **Note**
>
> These basic configurations, called *Amazon Machine Images (AMIs)*, serve as templates for your instance. This tutorial can be completed with any of the free tier eligible AMIs. For simplicity, we use the HVM edition of the Amazon Linux AMI.

1. 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**.
2. 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 an IAM role that has been configured for use as an IAM instance profile for use with AWS CodeDeploy. If you do not have an IAM instance profile, choose **Create new IAM role** and follow the instructions in Create an IAM Instance Profile for Your Amazon EC2 Instances.

Note
For the purposes of this tutorial, you can use the following unrestricted policy in your IAM instance profile for AWS CodeDeploy. For pipelines you use in your development workflows, you might create a more restrictive bucket policy.

```
{
"Version": "2012-10-17",
"Statement": [
{
"Action": [
"s3:Get*",
"s3:List*"
],
"Effect": "Allow",
"Resource": "*"
}
]
}
```

1. On the **Step 3: Access the command line interface and run following commands:**

Bash

```
[ec2-user@ip-172-31-4-186 ~]$ vim 1.sh
[ec2-user@ip-172-31-4-186 ~]$ chmod 755 1.sh
[ec2-user@ip-172-31-4-186 ~]$ sudo ./1.sh
```

```
#!/bin/bash
yum -y update
yum install -y ruby
yum install -y aws-cli
cd /home/ec2-user
wget https://aws-codedeploy-us-east-2.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

This code installs the AWS CodeDeploy agent on your instance as it is created. If you prefer, you can connect to your Linux instance using SSH and install the AWS CodeDeploy agent manually after the instance 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. On the **Add Tags** page, with **Name** displayed in the **Key** box, type MyCodePipelineDemo in the **Value** box, and then choose **Next: Configure Security Group**.

Important

The **Key** and **Value** boxes are case-sensitive.

1. 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**.
2. Choose **Review and Launch**.
3. 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. For more information, see Amazon EC2 Key Pairs.

Warning
Don't select the **Proceed without a key pair** option. If you launch your instance without a key pair, you can't connect to it if you need to troubleshoot issues with the AWS CodeDeploy agent.
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. Check that your instance has passed its status checks. You can view this information in the **Status Checks** column.

If you want to confirm that the AWS CodeDeploy agent is configured correctly, you can connect to your Linux instance using SSH and then verify the AWS CodeDeploy agent is running.

Step 2: Create an AWS CodeCommit Repository and Local Repo

To start this tutorial, you create a repository in AWS 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 pushing it to the AWS CodeCommit repository.
Important
AWS CodeCommit is currently supported for pipelines in the following regions:

- US East (Ohio) Region (us-east-2)
- US East (N. Virginia) Region (us-east-1)
- US West (Oregon) Region (us-west-2)
- EU (Ireland) Region (eu-west-1)
- South America (São Paulo) Region (sa-east-1)

Be sure to complete all of the steps in this tutorial with one of these AWS regions selected.
Follow the first two procedures in the Git with AWS CodeCommit Tutorial in the *AWS CodeCommit User Guide*:

- Step 1: Create an AWS CodeCommit Repository
- Step 2: Create a Local Repo

For information about connecting to a local repo you create, see Connect to an AWS CodeCommit Repository.
After you complete these two procedures, return to this page and continue to the next step. Do not continue to the third step in the AWS CodeCommit tutorial. You must complete different steps in this tutorial instead.

Step 3: Add Sample Code to Your AWS CodeCommit Repository

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

1. Download the following file:

- SampleApp_Linux.zip.

2. Unzip the files from SampleApp_Linux.zip into the local directory you created in the previous procedure (for example, /tmp/my-demo-repo or c:\temp\my-demo-repo).

Download and extract archive

```
[ec2-user@ip-172-31-4-186 ~]$ cd /tmp
[ec2-user@ip-172-31-4-186 ~]$ wget https://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest
/SampleApp_Linux.zip
[ec2-user@ip-172-31-4-186 ~]$ unzip -q SampleApp_Linux.zip
```

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

```
my-demo-repo
– appspec.yml
– index.html
– LICENSE.txt
`-- scripts
– install_dependencies
– start_server
`-- stop_server
```

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

(For Windows) cd c:\temp\my-demo-repo

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

```
git add -A
```

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

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

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

```
git push
```

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

Step 4: Create an Application in AWS CodeDeploy

In AWS CodeDeploy, an *application* is an identifier, in the form of a name, for the code you want to deploy. AWS CodeDeploy uses this name to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment. You select the name of the AWS CodeDeploy application you create in this step when you create your pipeline later in this tutorial.

To create an application in AWS CodeDeploy

1. Open the AWS 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**.
7. In **Deployment group name**, enter MyDemoDeploymentGroup.
8. Under **Deployment type**, choose **In-place deployment**.

9. Under **Environment configuration**, choose the **Amazon EC2 Instances** tab, and select the Amazon EC2 tag type. Choose **Name** in the **Key** box, and in **Value**, enter MyCodePipelineDotNetDemo.

Important
You must choose the same value for the **Name** key here that you assigned to your Amazon EC2 instance when you created it. If you tagged your instance with something other than MyCodePipelineDemo, be sure to use it here.

1. In **Deployment configuration**, choose CodeDeployDefault.OneAtaTime.
2. In **Service role ARN**, choose an Amazon Resource Name (ARN) for a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in Create a Service Role for AWS CodeDeploy. To get the service role ARN, see Get the Service Role ARN (Console).
3. Choose **Create application**. Stay on the displayed page.

To create a deployment group in AWS CodeDeploy

1. On the page showing your application, choose **Create deployment group**.
2. In **Deployment group name**, enter MyDemoDeploymentGroup.
3. In **Service Role**, choose a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in Create a Service Role for AWS CodeDeploy. To get the service role ARN, see Get the Service Role ARN (Console).
4. Under **Deployment type**, choose **In-place**.
5. Under **Environment configuration**, choose **Amazon EC2 Instances**. Choose **Name** in the **Key** box, and in the **Value** box, enter MyCodePipelineDemo.

Important
You must choose the same value for the **Name** key here that you assigned to your Amazon EC2 instance when you created it. If you tagged your instance with something other than MyCodePipelineDemo, be sure to use it here.

1. Under **Deployment configuration**, choose CodeDeployDefault.OneAtaTime.
2. Under **Load Balancer**, clear **Enable load balancing**.
3. Expand the **Advanced** section. Under **Alarms**, choose **Ignore alarm configuration**.
4. Choose **Create deployment group**.

Step 5: Create Your First Pipeline in AWS CodePipeline

You're now ready to create and run your first pipeline.
To create an AWS CodePipeline automated release process

1. Sign in to the AWS Management Console and open the AWS CodePipeline console at {+}http://console.aws.amazon.com/codesuite/codepipeline+.
2. On the **Welcome** page, **Getting started** page, or the **Pipelines** page, choose **Create pipeline**.
3. In **Step 1: Choose pipeline settings**, in **Pipeline name**, enter **MyFirstPipeline**.

> **Note**
>
> If you choose another name for your pipeline, be sure to use it instead of *MyFirstPipeline* in the remaining steps of this tutorial. After you create a pipeline, you cannot change its name. Pipeline names are subject to some limitations. For more information, see Limits in AWS CodePipeline.

1. In **Role name**, do one of the following:
   - If you do not have a service role, choose **New service role**, and in **Role name**, enter the name for the role. IAM creates the role for you.
   - If you have previously created an AWS-CodePipeline-Service service role, choose **Existing service role**.

Note
Depending on when your service role was created, you might need to update its permissions to support additional AWS services. For information, see Add Permissions for Other AWS Services.

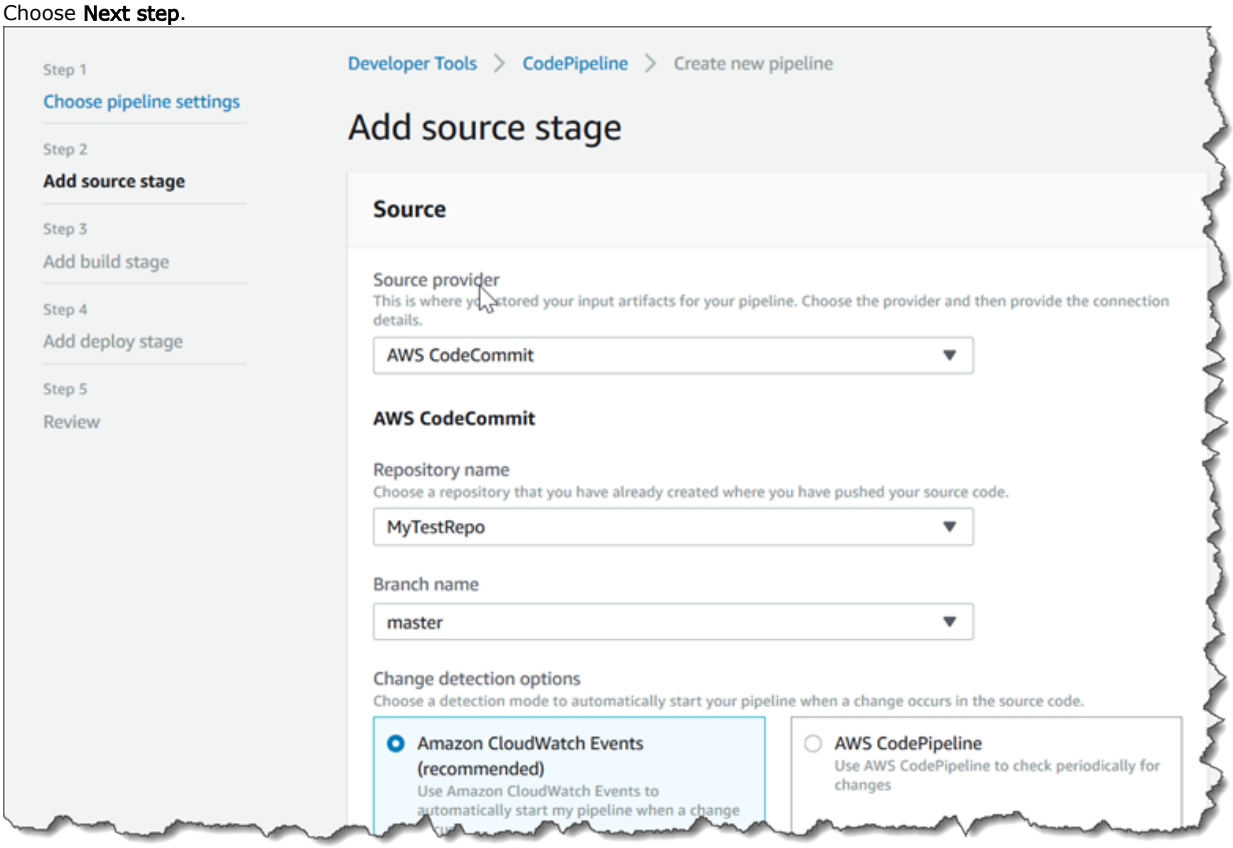
1. In **Artifact store**, do one of the following:
   a. Choose **Default location**. This uses the default artifact store, such as the Amazon S3 artifact bucket designated as the default, for your pipeline in the region you have selected for your pipeline.
   b. Choose **Custom location** if you already have an artifact store, such as an Amazon S3 artifact bucket, in the same region as your pipeline.

> **Note**
>
> This is not the source bucket for your pipeline's source code. This is the artifact store for your pipeline. A separate artifact store, such as an Amazon S3 bucket, is required for each pipeline, in the same region as the pipeline.

Choose **Next**.

1. In **Step 2: Add source stage**, in **Source provider**, choose AWS CodeCommit. In **Repository name**, choose the name of the AWS CodeCommit repository you created in Step 1: Create an AWS CodeCommit Repository and Local Repo. In **Branch name**, choose the name of the branch that contains your latest code update. Unless you created a different branch on your own, only master is available.

Choose **Next step**.

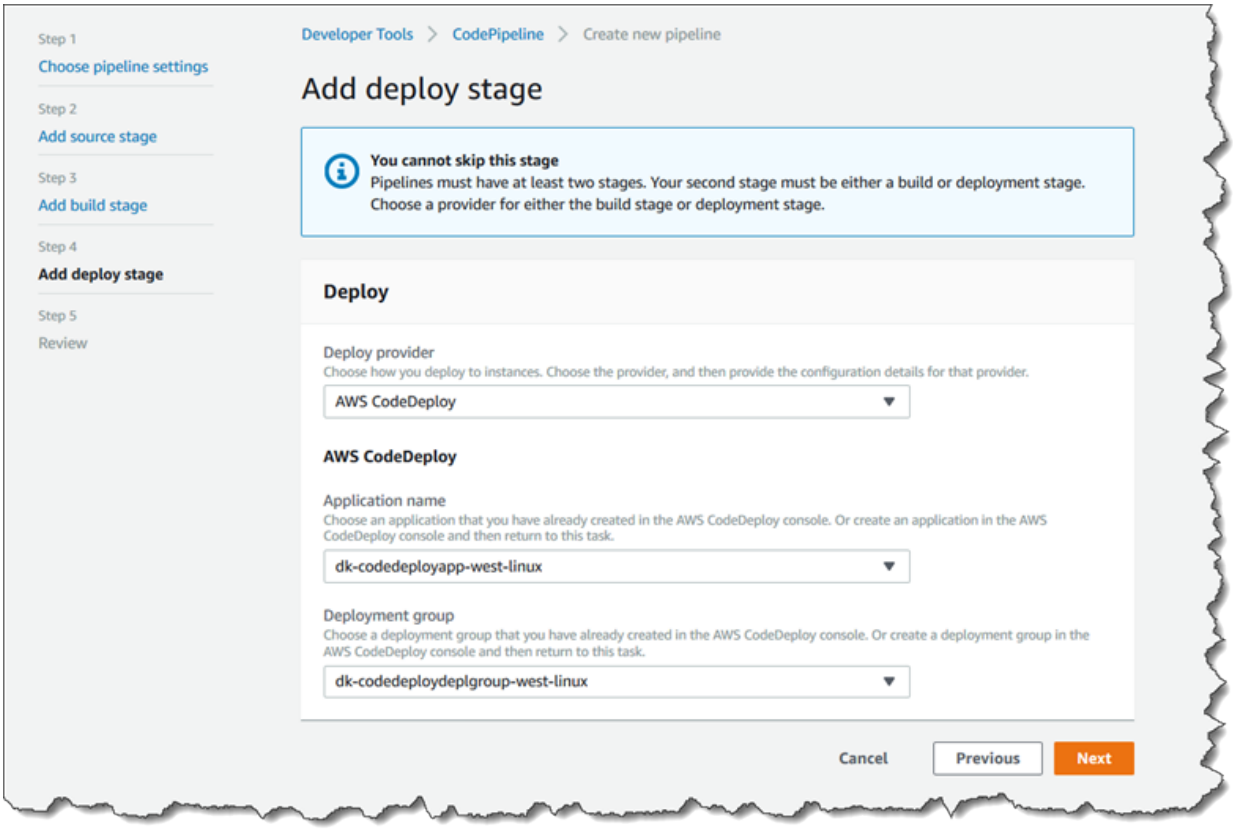


After you select the repository name and branch, a message is displayed showing the Amazon CloudWatch Events rule to be created for this pipeline.
Under **Change detection options**, leave the defaults. This allows AWS CodePipeline to use Amazon CloudWatch Events to detect changes in your source repository.
Choose **Next**.

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

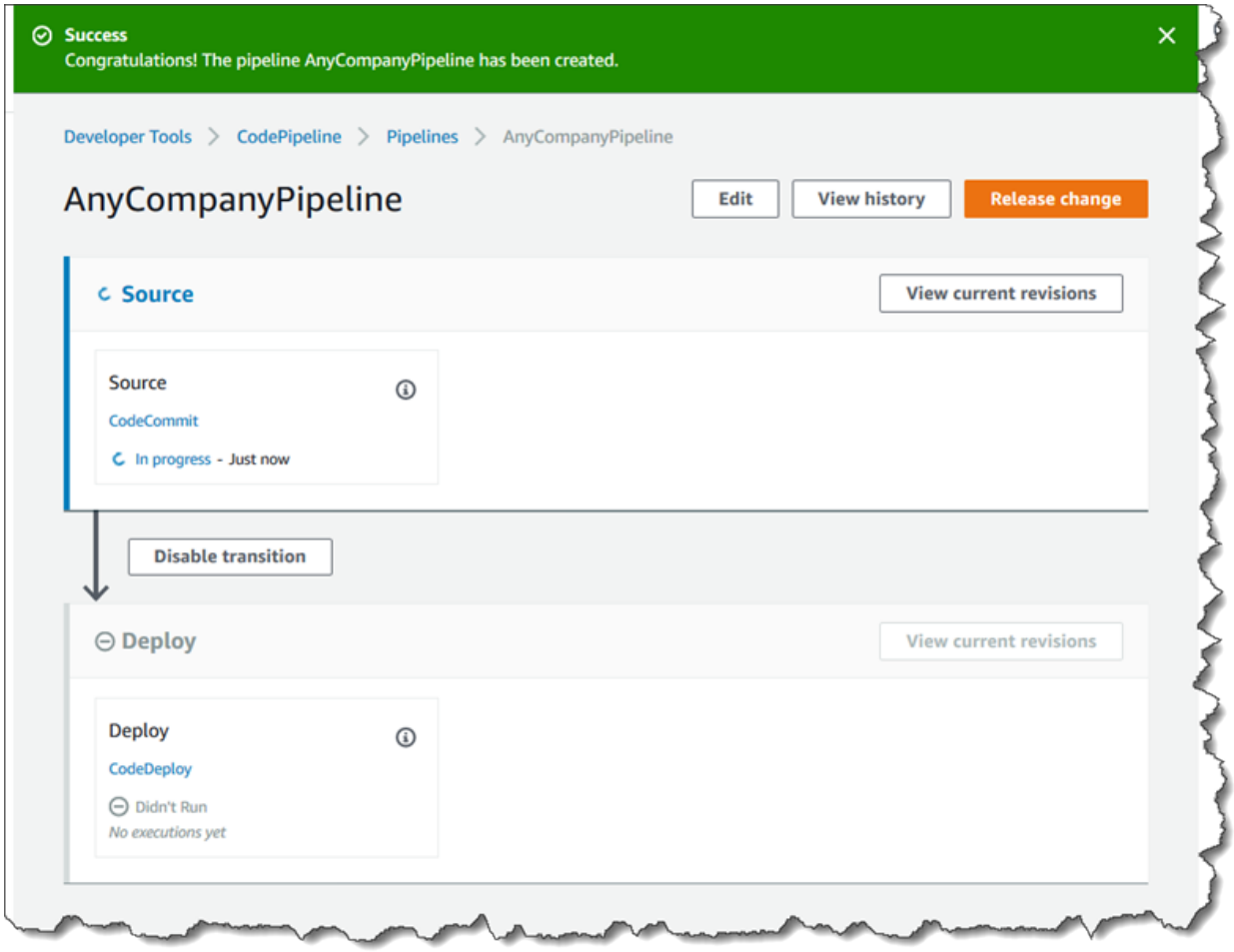
> **Note**
>
> In this tutorial, you are deploying code that requires no build service.

1. In **Step 4: Add deploy stage**, in **Deploy provider**, choose **AWS CodeDeploy**. In **Application name**, enter **MyDemoApplication**, or choose the **Refresh** button, and then choose the application name from the list. In **Deployment group**, enter **MyDemoDeployme ntGroup**, or choose it from the list, and then choose **Next step**.



> **Note**
>
> The name "Deploy" is the name given by default to the stage created in the **Step 4: Deploy** step, just as "Source" is the name given to the first stage of the pipeline.

1. In **Step 5: Review**, review the information, and then choose **Create pipeline**.
2. The pipeline starts to run. You can view progress and success and failure messages as the AWS CodePipeline sample deploys the webpage to the Amazon EC2 instance in the AWS CodeDeploy deployment.



Congratulations! You just created a simple pipeline in AWS CodePipeline. The pipeline has two stages:

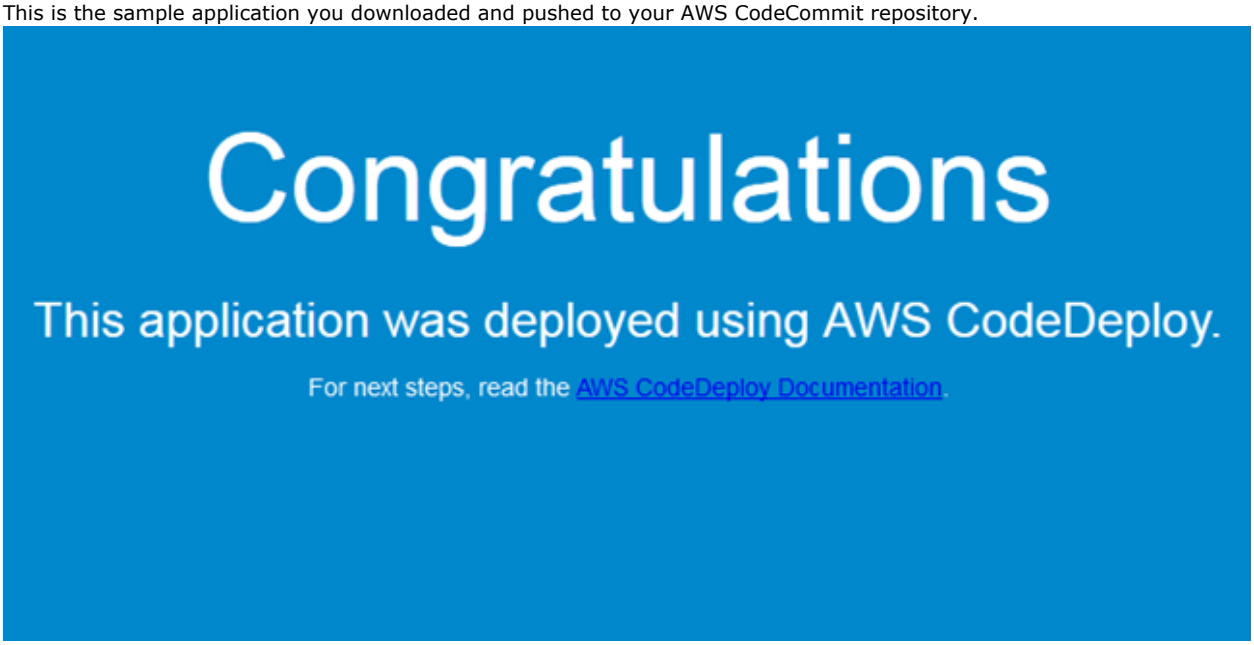
- A source stage (**Source**) that detects changes in the sample application stored in the AWS CodeCommit repository and pulls those changes into the pipeline.
- A deployment stage (**Deploy**) that deploys those changes to the Amazon EC2 instance using AWS CodeDeploy.

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 **Staging** stage, choose **Details**. This opens the AWS CodeDeploy console.
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, and then paste it into the address bar of your web browser.

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



For more information about stages, actions, and how pipelines work, see AWS CodePipeline Concepts.

Step 6: Modify Code in Your AWS CodeCommit Repository

In this step, you make changes to the HTML file that is part of the sample AWS CodeDeploy application you deployed to your Amazon EC2 instance. When your pipeline runs again later in this tutorial, the changes you make are visible at the http://*PublicDNS* URLs.

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

(For Windows) cd c:\temp\my-demo-repo

1. Use a text editor to modify the index.html file:
2. *(For Linux or Unix)*gedit index.html
3. *(For OS X)*open –e index.html

*(For Windows)*notepad index.html

   a. 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;
}
h1 {
font-size: 250%;
font-weight: normal;
margin-bottom: 0;
}
h2 {
font-size: 175%;
font-weight: normal;
margin-bottom: 0;
}
</style>
```

```
</head>
<body>
<div align="center"><h1>Updated Sample Deployment</h1></div>
<div align="center"><h2>This application was updated
using AWS CodePipeline, AWS CodeCommit, and AWS
CodeDeploy.</h2></div>
<div align="center">
<p>Learn more:</p>
<p><a
href="https://docs.aws.amazon.com/codepipeline/latest/userguide/">AWS
 CodePipeline User Guide</a></p>
<p><a href="https://docs.aws.amazon.com/codecommit/latest/userguide/">AWS CodeCommit User Guide</a></p>
<p><a href="https://docs.aws.amazon.com/codedeploy/latest/userguide/">AWS CodeDeploy User Guide</a></p>
</div>
</body>
</html>
```

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

Your pipeline is configured to run whenever code changes are made to your AWS CodeCommit repository.
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 pipeline should complete within a few minutes.
2. After **Succeeded** is displayed for the action status, in the status area for the **Staging** stage, choose **Details**. This opens the AWS CodeDeploy console.
3. 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, and then paste it into the address bar of your web browser.

The updated webpage is displayed:

For more information about stages, actions, and how pipelines work, see AWS CodePipeline Concepts.

Step 7: Optional Stage Management Tasks

If you want to gain more experience working with stages before you end the tutorial, you can follow two additional procedures in the Tutorial: Create a Simple Pipeline (Amazon S3 Bucket).

- Step 6: Add Another Stage to Your Pipeline
- Disable and Enable Transitions Between Stages in AWS CodePipeline

Note

In step 4 of the second procedure, instead of uploading your sample to an Amazon S3 bucket again, make a change to the sample app in your local repo and push it to your AWS CodeCommit repository.