




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.

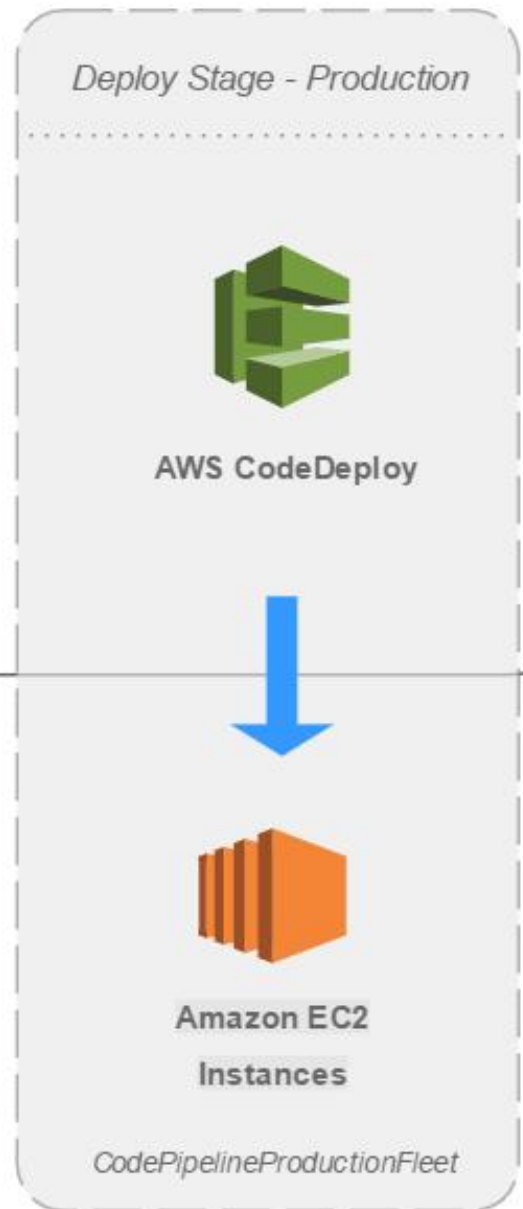
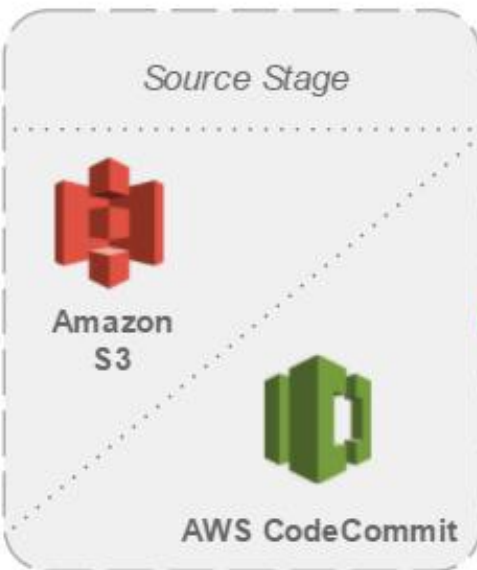
The easiest way to get started with AWS CodePipeline is to use the **Create Pipeline** wizard in the CodePipeline console.

Note

Before you begin, make sure you've set up your Git client to work with CodeCommit.



AWS CodePipeline



Setting up for AWS CodeCommit

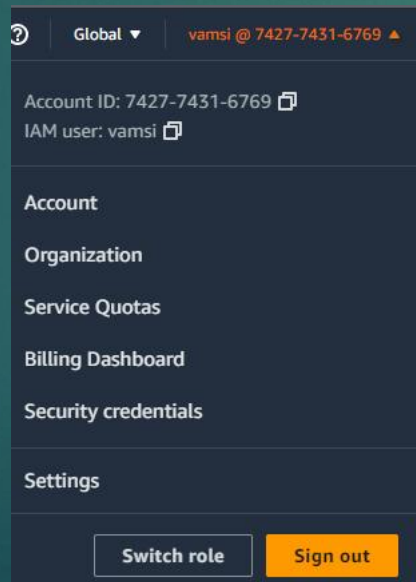
View and manage your credentials

You can view and manage your CodeCommit credentials from the AWS console through **My Security Credentials**.

Note

This option is not available for users using federated access, temporary credentials, or a web identity provider.

1. Sign in to the AWS Management Console and open the IAM console.
2. In the navigation bar on the upper right, choose your user name, and then choose **My Security Credentials**.



3.Choose the **AWS CodeCommit credentials** tab.

AWS IAM credentials

AWS CodeCommit credentials

Amazon Keyspaces credentials

SSH public keys for AWS CodeCommit (0)

User SSH public keys to authenticate access to AWS CodeCommit repositories. You can have a maximum of five SSH public keys (active or inactive) at a time. [Learn more](#)

Actions

Upload SSH public key

SSH Key ID	Uploaded	Status
No SSH public keys		
<div>Upload SSH public key</div>		

HTTPS Git credentials for AWS CodeCommit (0)

Generate a user name and password you can use to authenticate HTTPS connections to AWS CodeCommit repositories. You can have a maximum of 2 sets of credentials (active or inactive) at a time. [Learn more](#)

Actions

Generate credentials

User name	Created	Status
No credentials		
<div>Generate credentials</div>		

Generate credentials

✔ Your new credentials are available.

Save your user name and password or download the credentials file.

This is the only time you can view the password or download it. You cannot recover it later. However, you can reset your password at any time.

You can use these credentials when connecting from your local computer, or from tools that require a static user name and password. [Learn more](#)

User name
vamsi-at-742774316769

Password
e+YY1R30xthBQGoUE39jHRc8iYiLRWl3ZSA445oLAig= [Hide](#)

Download credentials

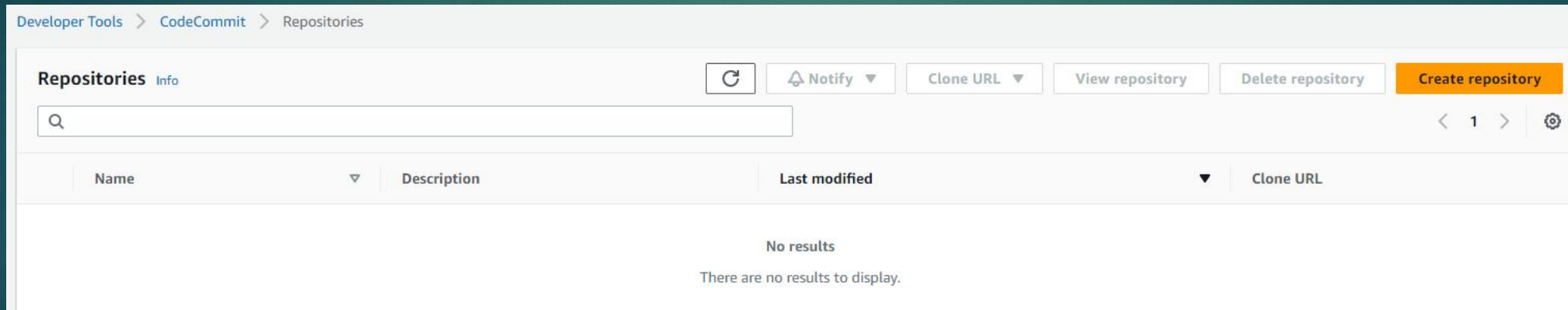
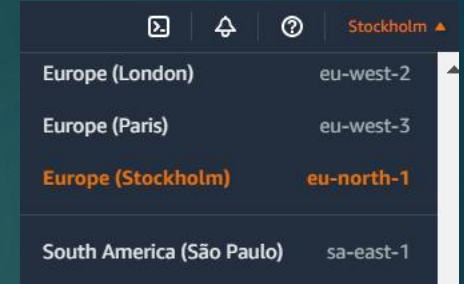
Close

Step 1: Create a CodeCommit repository

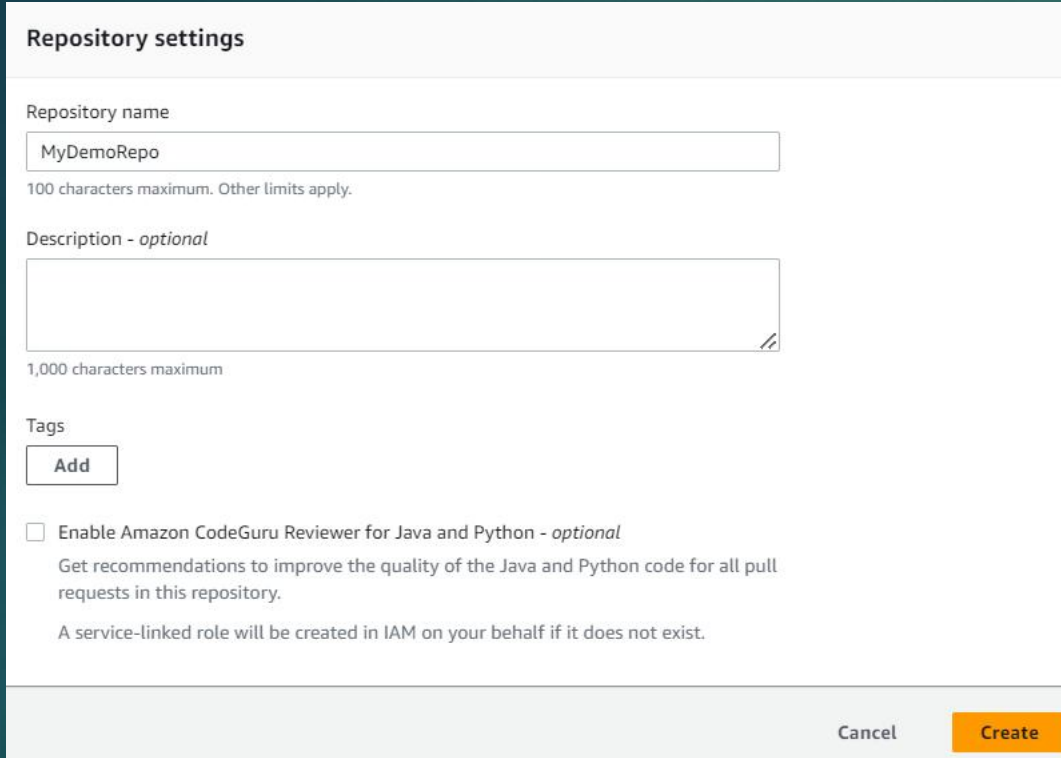
First, you create a repository in CodeCommit. Your pipeline gets source code from this repository when it runs. You also create a local repository where you maintain and update code before you push it to the CodeCommit repository.

To create a CodeCommit repository

1. Open the CodeCommit console.
2. In the Region selector, choose the AWS Region where you want to create the repository and pipeline.
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**.



The screenshot shows the 'Repository settings' form in AWS CodeCommit. It includes a text input for 'Repository name' with the value 'MyDemoRepo' and a note '100 characters maximum. Other limits apply.' Below this is an optional 'Description' text area with a note '1,000 characters maximum'. There is a 'Tags' section with an 'Add' button. At the bottom, there is an unchecked checkbox for 'Enable Amazon CodeGuru Reviewer for Java and Python - optional', with a description: 'Get recommendations to improve the quality of the Java and Python code for all pull requests in this repository. A service-linked role will be created in IAM on your behalf if it does not exist.' The form has 'Cancel' and 'Create' buttons at the bottom right.

Repository settings

Repository name

MyDemoRepo

100 characters maximum. Other limits apply.

Description - *optional*

1,000 characters maximum

Tags

Add

☐ Enable Amazon CodeGuru Reviewer for Java and Python - *optional*

Get recommendations to improve the quality of the Java and Python code for all pull requests in this repository.

A service-linked role will be created in IAM on your behalf if it does not exist.

Cancel 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 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 https://git-codecommit.eu-north-1.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
```

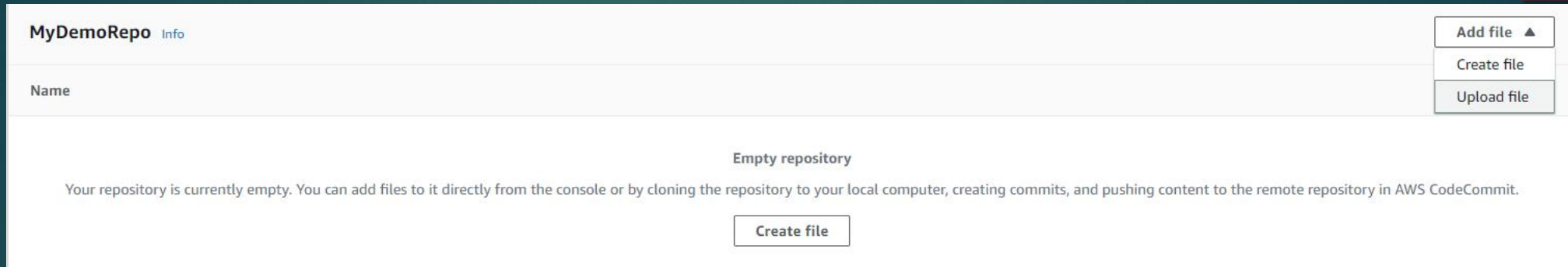
```
PS C:\tmp\MyDemoRepo> tar -xvf '.\SampleApp_Linux (1).zip'
x scripts/install_dependencies
x scripts/start_server
x scripts/stop_server
x appspec.yml
x index.html
x LICENSE.txt
PS C:\tmp\MyDemoRepo> |
```

3.To upload files to your repository, use one of the following methods.

a. To use the CodeCommit console to upload your files:

1.Open the CodeCommit console, and choose your repository from the **Repositories** list.

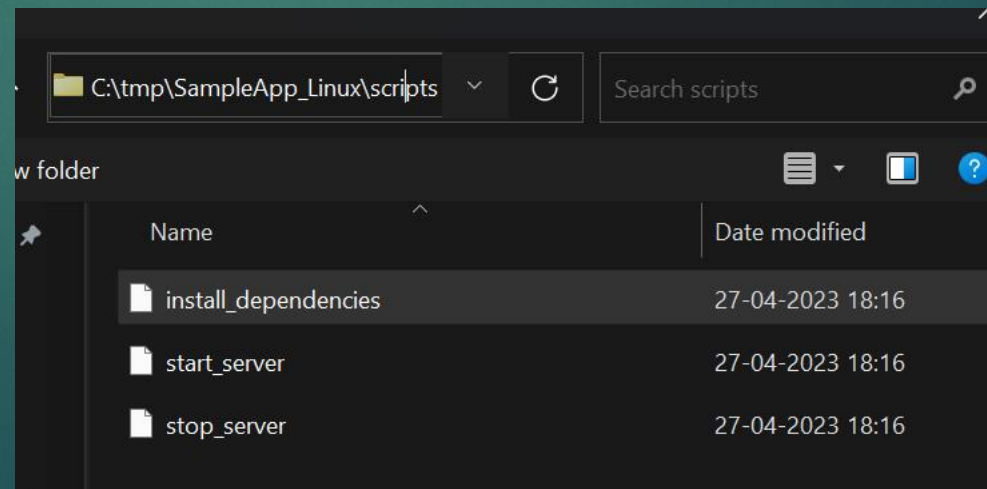
2.Choose **Add file**, and then choose **Upload file**.



3.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**.



Commit changes to main

File: MyDemoRepo/install_dependencies

Author name

Email address

Commit message - *optional*

A default commit message will be used if you do not provide one.

Cancel

Commit changes

4.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
```

5.To use git commands to upload your files:

1.Change directories to your local repo:

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

(For Windows) `cd c:\temp\MyDemoRepo`

```
PS C:\tmp\MyDemoRepo> ls
```

Directory: C:\tmp\MyDemoRepo

Mode	LastWriteTime		Length	Name
----	-----		-----	----
-a----	27-04-2023	20:08	377	appspec.yml
-a----	27-04-2023	20:08	752	index.html
-a----	27-04-2023	20:08	37	install_dependencies
-a----	27-04-2023	20:08	11085	LICENSE.txt
-a----	27-04-2023	20:08	36	start_server
-a----	27-04-2023	20:08	111	stop_server

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

`git add -A`

```
PS C:\tmp\MyDemoRepo> git add -A
warning: in the working copy of 'LICENSE.txt', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'appspec.yml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'scripts/install_dependencies', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'scripts/start_server', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'scripts/stop_server', LF will be replaced by CRLF the next time Git touches it
PS C:\tmp\MyDemoRepo> git add -A
PS C:\tmp\MyDemoRepo> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   LICENSE.txt
    new file:   SampleApp_Linux (1).zip
    new file:   appspec.yml
    new file:   index.html
    new file:   scripts/install_dependencies
    new file:   scripts/start_server
    new file:   scripts/stop_server

PS C:\tmp\MyDemoRepo> |
```

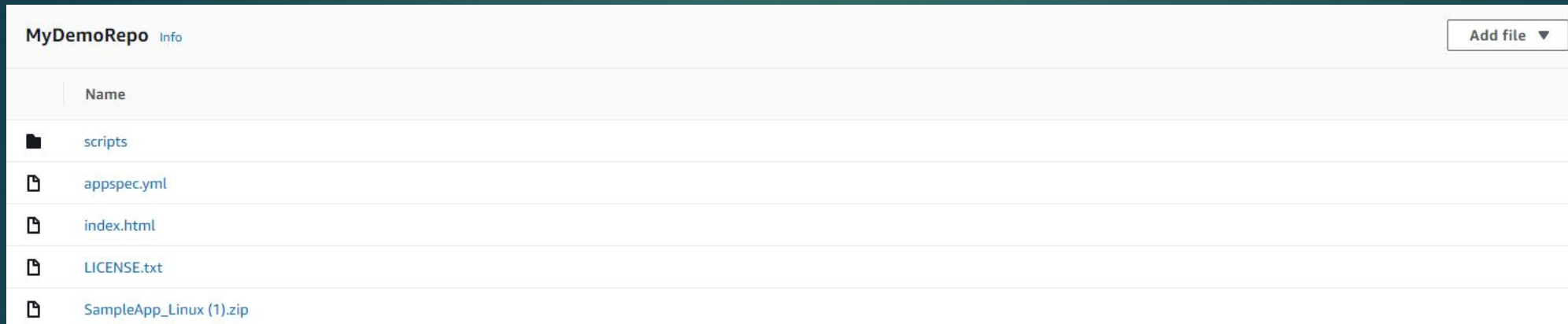

3.Run the following command to commit the files with a commit message:
git commit -m "Add sample application files"

```
PS C:\tmp\MyDemoRepo> git commit -m "Add sample application files"
[master 1b2352a] Add sample application files
 7 files changed, 266 insertions(+)
 create mode 100644 LICENSE.txt
 create mode 100644 SampleApp_Linux (1).zip
 create mode 100644 appspec.yml
 create mode 100644 index.html
 create mode 100644 scripts/install_dependencies
 create mode 100644 scripts/start_server
 create mode 100644 scripts/stop_server
PS C:\tmp\MyDemoRepo> |
```

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

```
PS C:\tmp\MyDemoRepo> git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (10/10), 9.98 KiB | 2.50 MiB/s, done.
Total 10 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-north-1.amazonaws.com/v1/repos/MyDemoRepo
   b4c423a..1b2352a  master -> master
PS C:\tmp\MyDemoRepo> |
```

5. 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, 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**.

Select trusted entity Info

Trusted entity type

☒ AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

☒ EC2
Allows EC2 instances to call AWS services on your behalf.

☐ Lambda
Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

Choose a service to view use case

Cancel

Next

5.Search for and select the policy named **AmazonEC2RoleforAWSCodeDeploy**.
6.Search for and select the policy named **AmazonSSMManagedInstanceCore**.

Permissions policy summary		
Policy name <small>🔗</small>	Type	Attached as
AmazonSSMManagedInstanceCore	AWS managed	Permissions policy
AmazonEC2RoleforAWSCodeDeploy	AWS managed	Permissions policy

7. Choose **Next: Tags**.

8. Choose **Next: Review**. Enter a name for the role (for example, **EC2InstanceRole**).

Role name

Enter a meaningful name to identify this role.

EC2InstanceRole

Maximum 64 characters. Use alphanumeric and '+=,@-_' characters.

To launch instances

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. Under **Name and tags**, in **Name**, enter **MyCodePipelineDemo**. This assigns the instances a tag **Key** of **Name** and a tag **Value** of **MyCodePipelineDemo**. Later, you create a CodeDeploy application that deploys the sample application to the instances. CodeDeploy selects instances to deploy based on the tags.

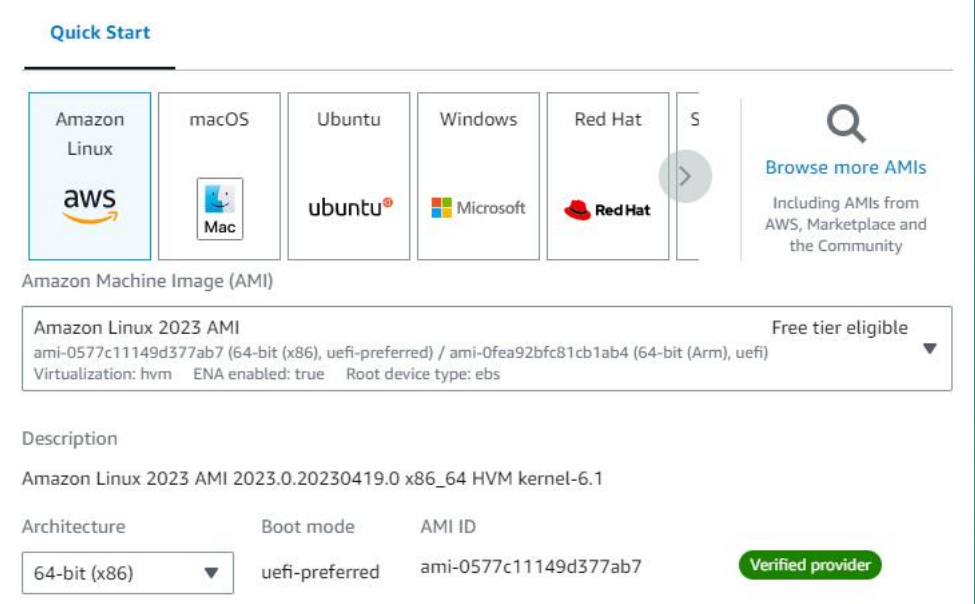
Name and tags [Info](#)

Name

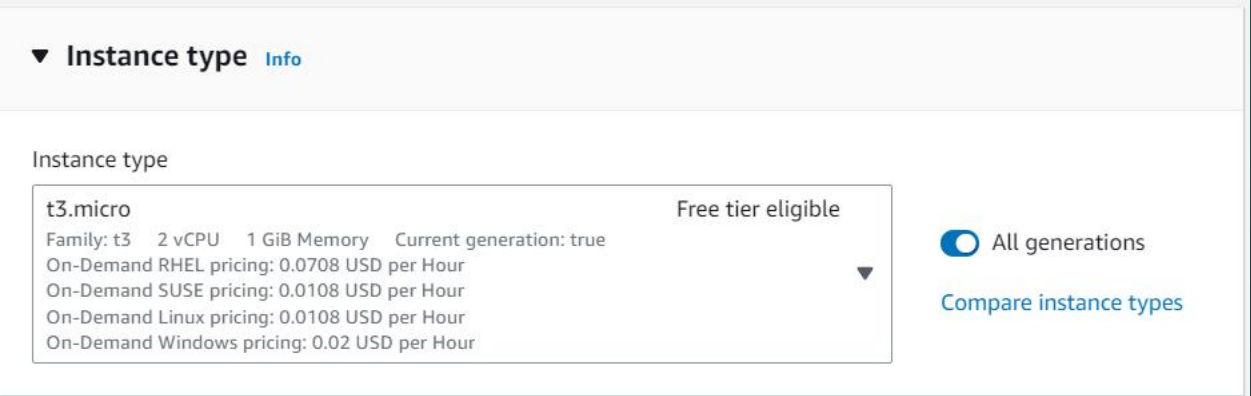
MyCodePipelineDemo

[Add additional tags](#)

1.4. 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 t3.micro type as the hardware configuration for your instance.




6. Under **Key pair (login)**, choose a key pair or create one or continue with the existing Key Pair.

▼ **Key pair (login)** [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

stockholm_keypair ▼

 [Create new key pair](#)

For Windows instances, you use a key pair to decrypt the administrator password. You then use the decrypted password to connect to your instance.

7. Under **Network settings**, do the following.
In **Auto-assign Public IP**, make sure the status is **Enable**.

Auto-assign public IP [Info](#)

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**.

▼ Security group rule 1 (TCP, 22, 117.99.197.88/32) Remove

Type [Info](#)
ssh ▼

Protocol [Info](#)
TCP

Port range [Info](#)
22

Source type [Info](#)
My IP ▼

Name [Info](#)

Q Add CIDR, prefix list or security

117.99.197.88/32 X

Description - optional [Info](#)
e.g. SSH for admin desktop

▼ Security group rule 2 (TCP, 80, 117.99.197.88/32) Remove

Type [Info](#)
HTTP ▼

Protocol [Info](#)
TCP

Port range [Info](#)
80

Source type [Info](#)
My IP ▼

Name [Info](#)

Q Add CIDR, prefix list or security


117.99.197.88/32 X

Description - optional [Info](#)
e.g. SSH for admin desktop

8. Expand **Advanced details**. In **IAM instance profile**, choose the IAM role you created in the previous procedure (for example, **EC2InstanceRole**).

IAM instance profile [Info](#)

EC2InstanceRole
arn:aws:iam::742774316769:instance-profile/EC2InstanceRole

 [Create new IAM profile](#)

9. Under **Summary**, under **Number of instances**, enter 2..
10. Choose **Launch instance**.

▼ **Summary**

Number of instances [Info](#)

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.0.2...[read more](#)
ami-0577c11149d377ab7

Virtual server type (instance type)



t3.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

 **Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet. 

Cancel

Launch instance

[Review commands](#)

11. Choose **View all instances** to close the confirmation page and return to the console.

12. 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**.)

13. 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.

Instances (1/1) [Info](#)

🔄

Connect

Instance state ▼

Actions ▼

Launch instances ▼

🔍 Find instance by attribute or tag (case-sensitive)

< 1 > ⚙️

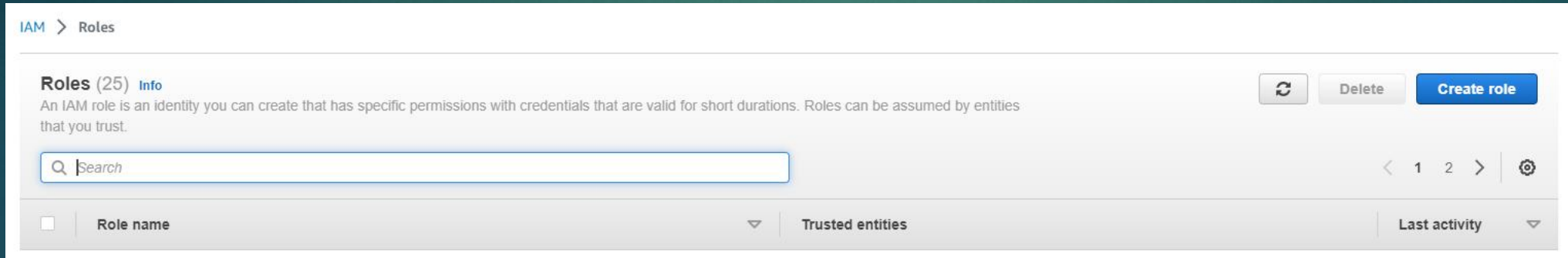
<input checked="" type="checkbox"/>	Name ▼	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status	Availability Zone ▼	Public IPv4 DNS ▼	Public IPv4 ... ▼	Elastic IP
<input checked="" type="checkbox"/>	MyCodePipelineDemo	i-035046c528331bd95	🟢 Running 🔍	t3.micro	🕒 Initializing	No alarms +	eu-north-1c	ec2-16-16-192-249.eu-...	16.16.192.249	-

Step 3: Create an application in CodeDeploy

You first create a service role for CodeDeploy to use. If you have already created a service role, you do not need to create another one.

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**.

Select trusted entity Info

Trusted entity type

☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

☐ **EC2**
Allows EC2 instances to call AWS services on your behalf.

☐ **Lambda**
Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

CodeDeploy ▼

☒ **CodeDeploy**
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.


☐ **CodeDeploy for Lambda**
Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.

☐ **CodeDeploy - ECS**
Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS services on your behalf.

Cancel Next

Permissions policies (1) [Info](#)

The type of role that you selected requires the following policy.

Policy name ↗	Type	Attached entities
 AWSCodeDeployRole	AWS m...	1

6. Enter a name for the role (for example, **CodeDeployRole**), and then choose **Create role**

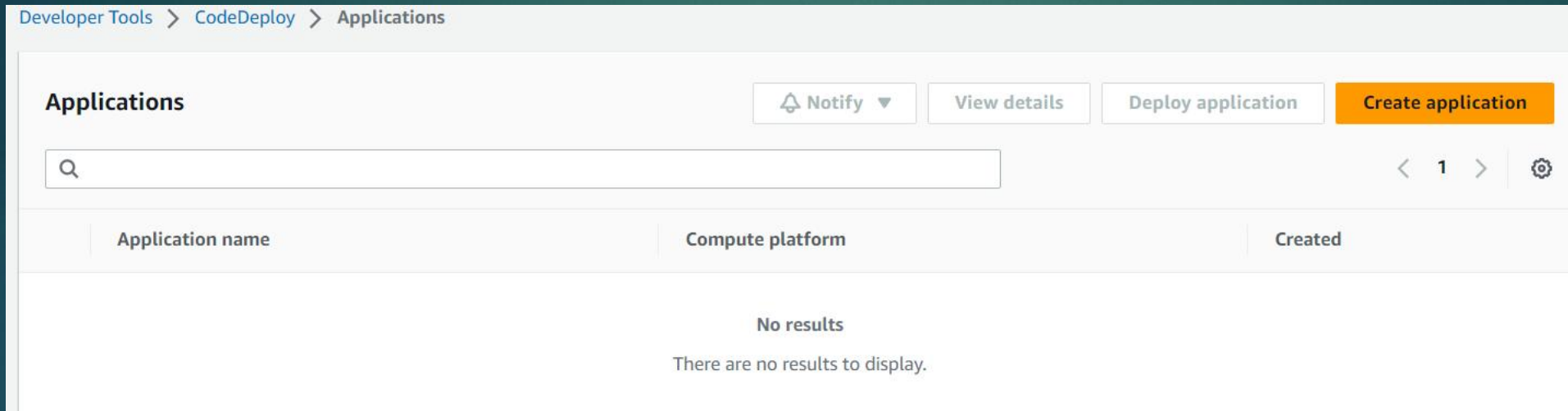
Role name

Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+=, @-_' characters.

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. In **Application name**, enter MyDemoApplication.
5. In **Compute Platform**, choose **EC2/On-premises**.
6. Choose **Create application**.

Create application

Application configuration

Application name

Enter an application name

100 character limit

Compute platform

Choose a compute platform

EC2/On-premises ▲

EC2/On-premises ✓

AWS Lambda

Amazon ECS

Cancel

Create application

To create a deployment group in CodeDeploy

1. On the page that displays your application, choose **Create deployment group**.

Deployments

Deployment groups

Revisions

Deployment groups

View details

Edit


Create deployment group

Q

<

1

>



Name	Status	Last attempted deployment	Last successful deployment	Trigger count
<div>No deployment groups</div> <div>Before you can deploy your application using CodeDeploy, you must create a deployment group.</div> <div>Create deployment group</div>				

2. In **Deployment group name**, enter **MyDemoDeploymentGroup**.

Deployment group name

Enter a deployment group name

MyDemoDeploymentGroup

100 character limit

3. In **Service role**, choose the service role you created earlier.

Service role

Enter a service role

Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

CodeDeployRole

arn:aws:iam::742774316769:role/CodeDeployRole

4. Under **Deployment type**, choose **In-place**.

Deployment type

Choose how to deploy your application

☒ **In-place**
Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update

☐ **Blue/green**
Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

5.Under **Environment configuration**, choose **Amazon EC2 Instances**. Choose **Name** in the **Key** field, and in the **Value** field, enter **MyCodePipelineDemo**.

Environment configuration

Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment

☐ Amazon EC2 Auto Scaling groups

☒ Amazon EC2 instances

2 unique matched instances. [Click here for details](#)

You can add up to three groups of tags for EC2 instances to this deployment group.
One tag group: Any instance identified by the tag group will be deployed to.
Multiple tag groups: Only instances identified by all the tag groups will be deployed to.

Tag group 1

Key	Value - optional	
<input type="text" value="Name"/>	<input type="text" value="MyCodePipelineDemo"/>	<input type="button" value="Remove tag"/>

☐ On-premises instances



Matching instances

2 unique matched instances. [Click here for details](#)

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

6. Under **Agent configuration with AWS Systems Manager**, choose **Now and schedule updates**. This installs the agent on the instance. The Windows instance is already configured with the SSM agent and will now be updated with the CodeDeploy agent.

Agent configuration with AWS Systems Manager [Info](#)

 **Complete the required prerequisites before AWS Systems Manager can install the CodeDeploy Agent.**
Make sure the AWS Systems Manager Agent is installed on all instances and attach the required IAM policies to them. [Learn more](#) 

Install AWS CodeDeploy Agent

☐ Never

☐ Only once

☒ Now and schedule updates

Basic scheduler Cron expression

14 Days ▼

7. Under **Deployment settings**, choose CodeDeployDefault.OneAtATime.

Deployment settings

Deployment configuration

Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.

CodeDeployDefault.OneAtATime ▲

 or

Create deployment configuration

CodeDeployDefault.OneAtATime ✓

CodeDeployDefault.HalfAtATime

CodeDeployDefault.AllAtOnce

8. Under **Load Balancer**, make sure the **Enable load balancing** box is not selected. You do not need to set up a load balancer or choose a target group for this example. After you de-select the checkbox, the load balancer options do not display.

Load balancer

Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

☐ Enable load balancing

-
- 9.In the **Advanced** section, leave the defaults.
- 10.Choose **Create deployment group**.

MyDemoDeploymentGroup

Edit

Delete

Create deployment

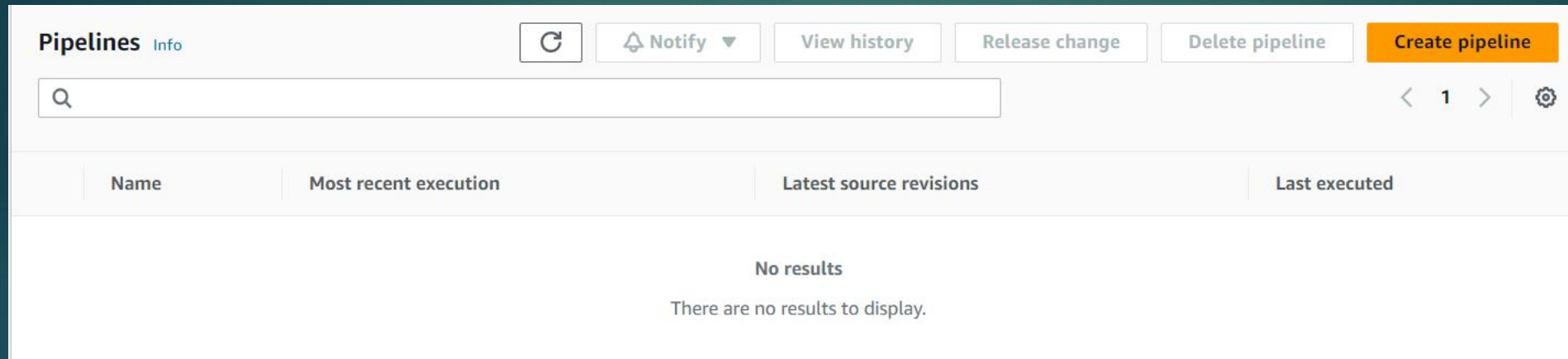
Deployment group details

Deployment group name	Application name	Compute platform
MyDemoDeploymentGroup	MyDemoApplication	EC2/On-premises
Deployment type	Service role ARN	Deployment configuration
In-place	arn:aws:iam::742774316769:role/CodeDeployRole	CodeDeployDefault.OneAtATime
Rollback enabled	Agent update scheduler	
False	Learn to schedule update in AWS Systems Manager ↗	

Step 4: Create your first pipeline in CodePipeline

To create a CodePipeline automated release process

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

Pipeline name

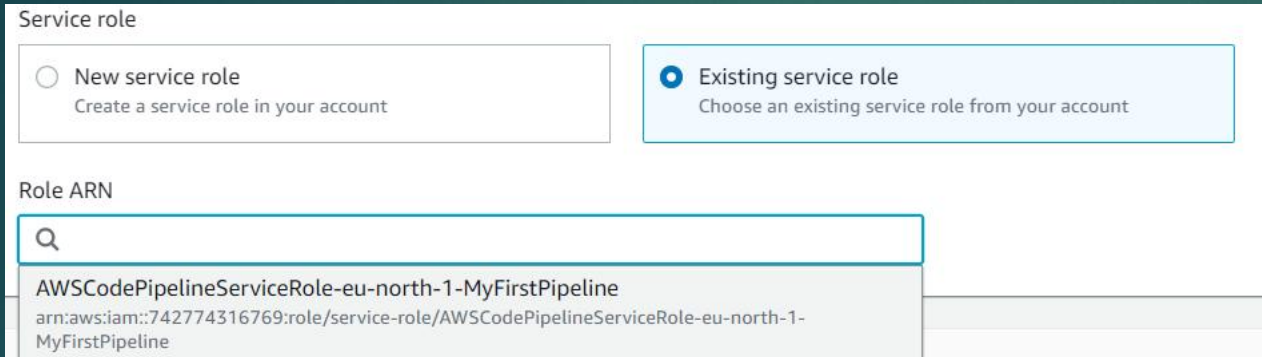
Enter the pipeline name. You cannot edit the pipeline name after it is created.

MyFirstPipeline

No more than 100 characters

4. In **Service role**, do one of the following:

- Choose **New service role** to allow CodePipeline to create a new service role in IAM.
- Choose **Existing service role** to use a service role already created in IAM. In **Role name**, choose your service role from the list.



Service role

☐ New service role
Create a service role in your account

☒ Existing service role
Choose an existing service role from your account

Role ARN

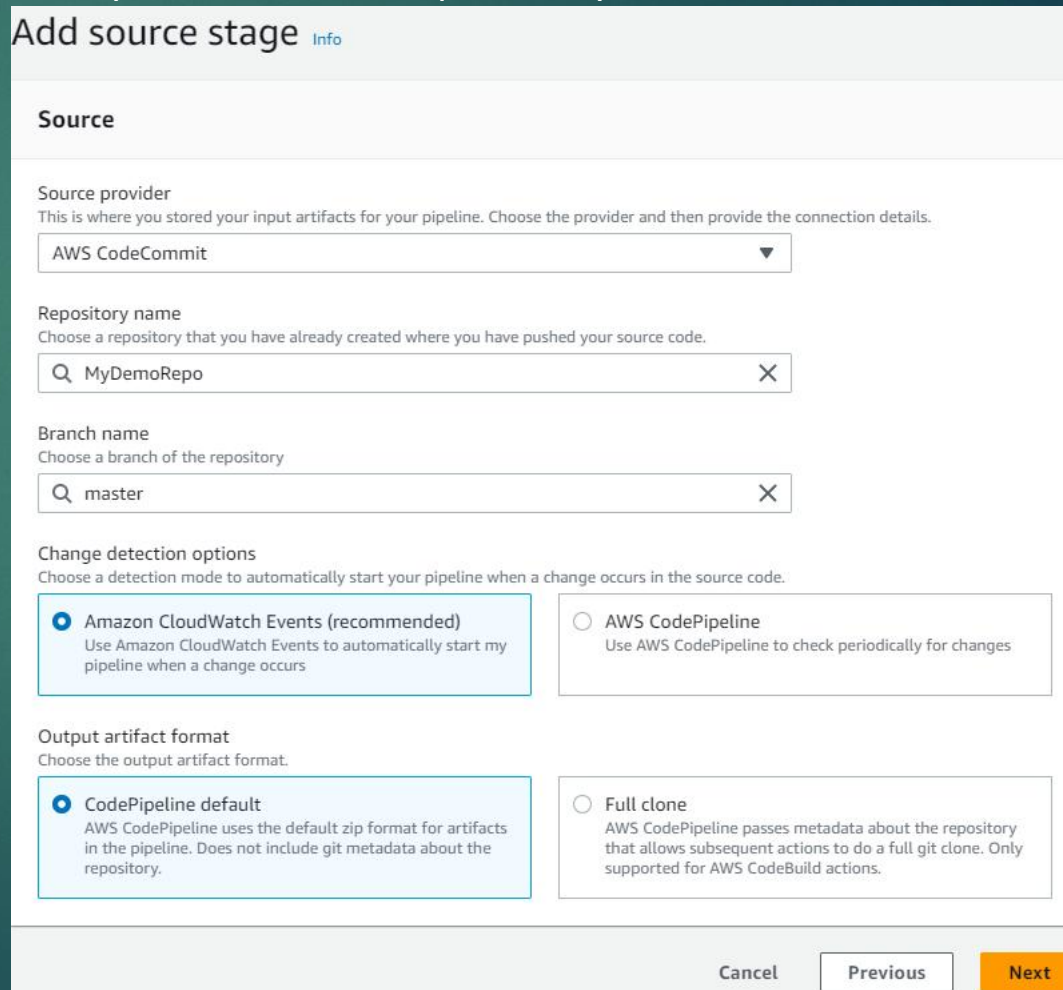
Q

AWSCodePipelineServiceRole-eu-north-1-MyFirstPipeline
arn:aws:iam::742774316769:role/service-role/AWSCodePipelineServiceRole-eu-north-1-MyFirstPipeline

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. 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**.



The screenshot shows the 'Add source stage' configuration page in AWS CodePipeline. The page is titled 'Add source stage' with an 'Info' link. It contains several sections for configuring the source stage:

- Source**: The main section for configuring the source provider.
- Source provider**: A dropdown menu set to 'AWS CodeCommit'. Below it is a description: 'This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.'
- Repository name**: A text input field containing 'MyDemoRepo'. Below it is a description: 'Choose a repository that you have already created where you have pushed your source code.'
- Branch name**: A text input field containing 'master'. Below it is a description: 'Choose a branch of the repository.'
- Change detection options**: Two radio button options. The first, 'Amazon CloudWatch Events (recommended)', is selected. Its description is 'Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs'. The second option is 'AWS CodePipeline', with the description 'Use AWS CodePipeline to check periodically for changes'.
- Output artifact format**: Two radio button options. The first, 'CodePipeline default', is selected. Its description is 'AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include git metadata about the repository.' The second option is 'Full clone', with the description 'AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full git clone. Only supported for AWS CodeBuild actions.'

At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Next'.

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

Build - optional

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

Cancel

Previous

Skip build stage

Next

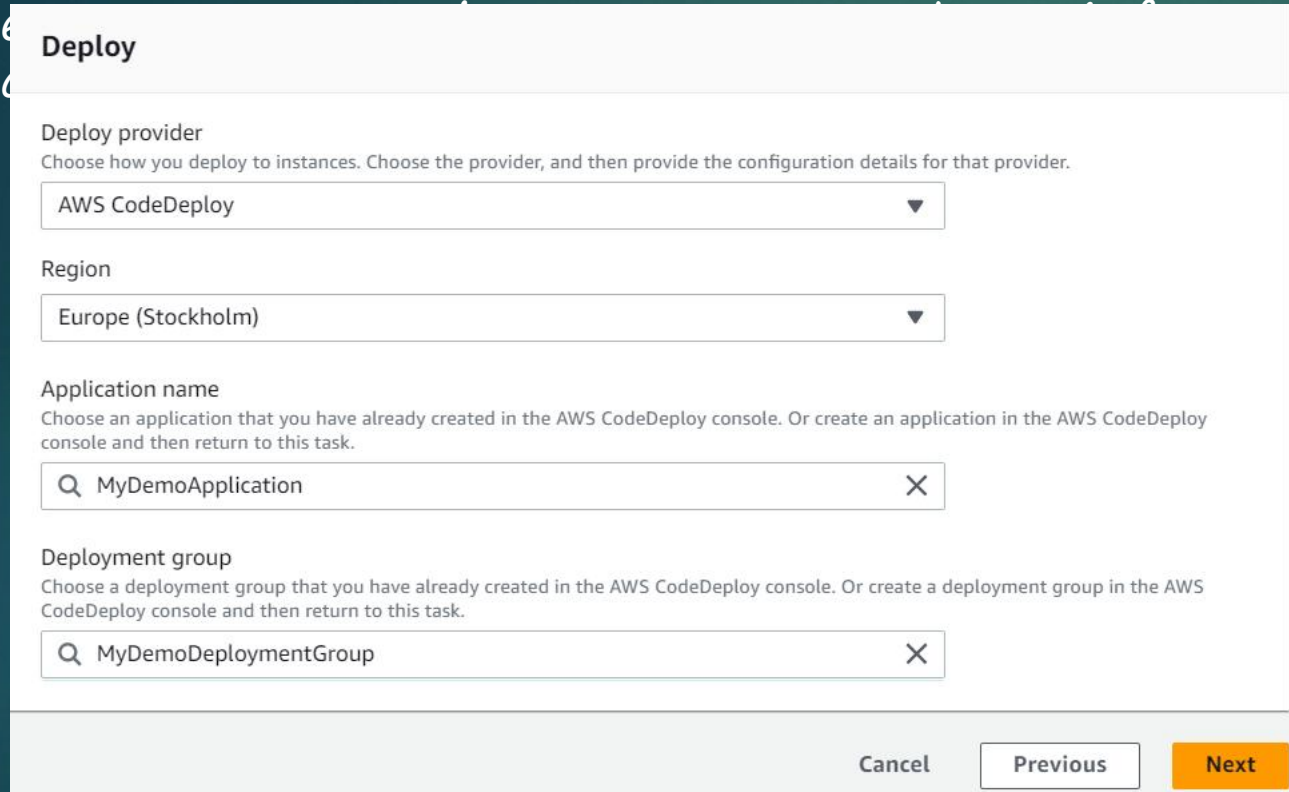
Skip build stage ×

Your pipeline will not include a build stage. Are you sure you want to skip this stage?

Cancel

Skip

8. In **Step 4: Add deploy stage**, in **Deploy provider**, choose **CodeDeploy** . The **Region** field defaults to the same AWS Region as your pipeline. In **Application name**, enter **MyDemoApplication**, or choose the **Refresh** button, and then choose the application name from the list. In **Deployment group**, choose the deployment group from the list, and then



The screenshot shows the 'Deploy' configuration window in the AWS CodePipeline console. It has a title bar 'Deploy' and a subtitle 'Deploy provider'. Below the subtitle is a description: 'Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.' There are four main sections: 'Deploy provider' with a dropdown menu showing 'AWS CodeDeploy'; 'Region' with a dropdown menu showing 'Europe (Stockholm)'; 'Application name' with a search box containing 'MyDemoApplication' and a description 'Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this task.'; and 'Deployment group' with a search box containing 'MyDemoDeploymentGroup' and a description 'Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this task.' At the bottom are three buttons: 'Cancel', 'Previous', and 'Next'.

Deploy

Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

AWS CodeDeploy ▼

Region
Europe (Stockholm) ▼

Application name
Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this task.

Q MyDemoApplication X

Deployment group
Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this task.

Q MyDemoDeploymentGroup X

Cancel Previous Next

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

10. The pipeline starts to run. You can view progress and success and failure messages as the CodePipeline sample deploys a webpage to each of the Amazon EC2 instances in the CodeDeploy deployment.

MyFirstPipeline

Notify ▼

Edit

Stop execution

Clone pipeline

Release change

Source Succeeded

Pipeline execution ID: d5a9ffa8-c018-433e-97f0-858998e1ef66

Source ⓘ

AWS CodeCommit

✓ Succeeded - 3 minutes ago

1b2352af

1b2352af Source: Add sample application files

Disable transition

Deploy Succeeded

Pipeline execution ID: d5a9ffa8-c018-433e-97f0-858998e1ef66

Deploy ⓘ

AWS CodeDeploy

✓ Succeeded - 2 minutes ago

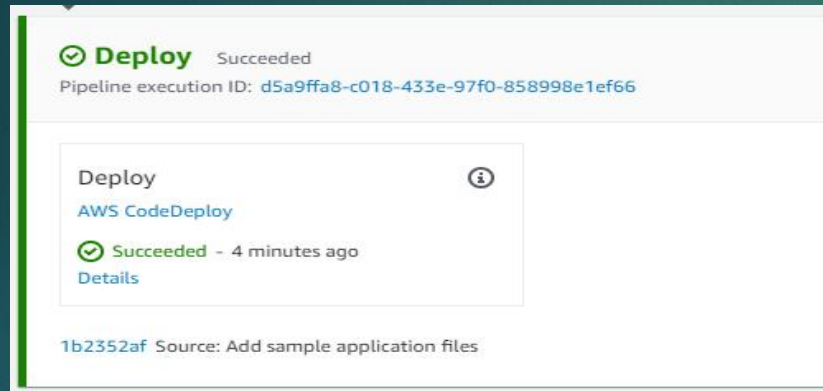
Details

1b2352af Source: Add sample application files

Congratulations! You just created a simple pipeline in CodePipeline.

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 the first run within a few minutes.
2. After **Succeeded** is displayed for the action status, in the status area for the **Deploy** stage, choose **Details**. This opens the CodeDeploy console.



3. In the **Deployment group** tab, under **Deployment lifecycle events**, choose an instance ID. This opens the EC2 console.

Deployment lifecycle events						
<input type="text"/>						
Instance ID	Duration	Status	Most recent event	Events	Start time	End time
i-035046c528331bd95 🔗	9 seconds	🟢 Succeeded	ValidateService	View events	Apr 28, 2023 4:09 PM (UTC+5:30)	Apr 28, 2023 4:09 PM (UTC+5:30)

4. On the **Description** tab, in **Public DNS**, copy the address, and then paste it into the address bar of your web browser. View the index page for the sample application you uploaded to your S3 bucket.

Instance: i-035046c528331bd95 (MyCodePipelineDemo)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

▼ Instance summary Info

Instance ID i-035046c528331bd95 (MyCodePipelineDemo)	Public IPv4 address 16.16.192.249 open address	Private IPv4 addresses 172.31.9.41
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-16-16-192-249.eu-north-1.compute.amazonaws.com open address
Hostname type IP name: ip-172-31-9-41.eu-north-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-9-41.eu-north-1.compute.internal	

5. The web page displays for the sample application you uploaded to your S3 bucket.



Congratulations

This application was deployed using AWS CodeDeploy.

For next steps, read the [AWS CodeDeploy Documentation](#).

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:

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

(For Windows) `cd c:\temp\MyDemoRepo`

2. Use a text editor to modify the index.html file:

(For Linux or Unix) `gedit index.html`

(For OS X) `open -e index.html`

(For Windows) `notepad index.html`

3.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 CodePipeline,
CodeCommit, and CodeDeploy.</h2></div> <div align="center"> <p>Learn
more:</p> <p><a
href="https://docs.aws.amazon.com/codepipeline/latest/userguide/">CodePipeline
User Guide</a></p> <p><a
href="https://docs.aws.amazon.com/codecommit/latest/userguide/">CodeCommit
User Guide</a></p> <p><a
href="https://docs.aws.amazon.com/codedeploy/latest/userguide/">CodeDeploy
User Guide</a></p> </div> </body> </html>
```


4.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"`

```
PS C:\tmp\MyDemoRepo> git commit -am "Updated sample application files"
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
[master 011ba86] Updated sample application files
 1 file changed, 15 insertions(+), 14 deletions(-)
PS C:\tmp\MyDemoRepo> |
```

git push

```
PS C:\tmp\MyDemoRepo> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 686 bytes | 686.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-north-1.amazonaws.com/v1/repos/MyDemoRepo
 1b2352a..011ba86  master -> master
PS C:\tmp\MyDemoRepo>
```

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.

