**Create a Continuous Delivery Pipeline in AWS with CodeCommit, CodeDeploy and CodePipeline**

Created by[Zeal Vora](https://deloittedevelopment.udemy.com/user/cybercorp/)

**AWS Workspace**

**60-120 minutes**

In this lab, you will be an engineer at an E-Commerce based startup whose website keeps going down due to mistakes that developers make while manually pushing code and making changes in the production server environments. The architecture group thinks that creating a continuous delivery pipeline might be a good architecture to try out since the automation can save time and also eliminates the requirement for developers to manually log in to the production servers and make changes. Your boss wants you to build a proof-of-concept.

The customer success team at our company has raised an issue. The coupon management server where the company regularly posts coupons for its products keeps going down. The system downtimes are primarily due to the developers making mistakes while pushing new changes to the production systems manually. It is observed that developers do not have experience working with EC2 instances based on Linux. I'd like you to create a continuous delivery pipeline that would automatically deploy any updates by developers on their repository to a set of EC2 instances.

The continuous delivery pipeline should be created using AWS Managed services like AWS CodeCommit, AWS CodeDeploy, and AWS CodePipeline. The deployment should be made on an EC2 instance. Developers have provided you with a sample index.html file that needs to be part of the CodeCommit repository. Your pipeline should detect any new changes made to the index.html file and automatically deploy the updates to an EC2 instance. The installation commands for configuring the coupon system and relevant documentation has also been provided to you as part of the install-system.txt file.

**How you'll work**

Your project has been broken into a set of tasks. To complete these tasks, use the provided workspace. You can launch your workspace by clicking below or using the button in the top right of the screen.

Tasks

1-Set up an AWS codecommit repository to store the application code

2-Add sample code to the codecommit repository

3-Create IAM role for EC2 running codedeploy agent

4-Create EC2 and install codedeploy Agent

5-Create Service Role for codedeploy

6- create an application in codedeploy

7-Create AppSpec File for codedeploy

8-Create a pipeline using AWS Codedeploy

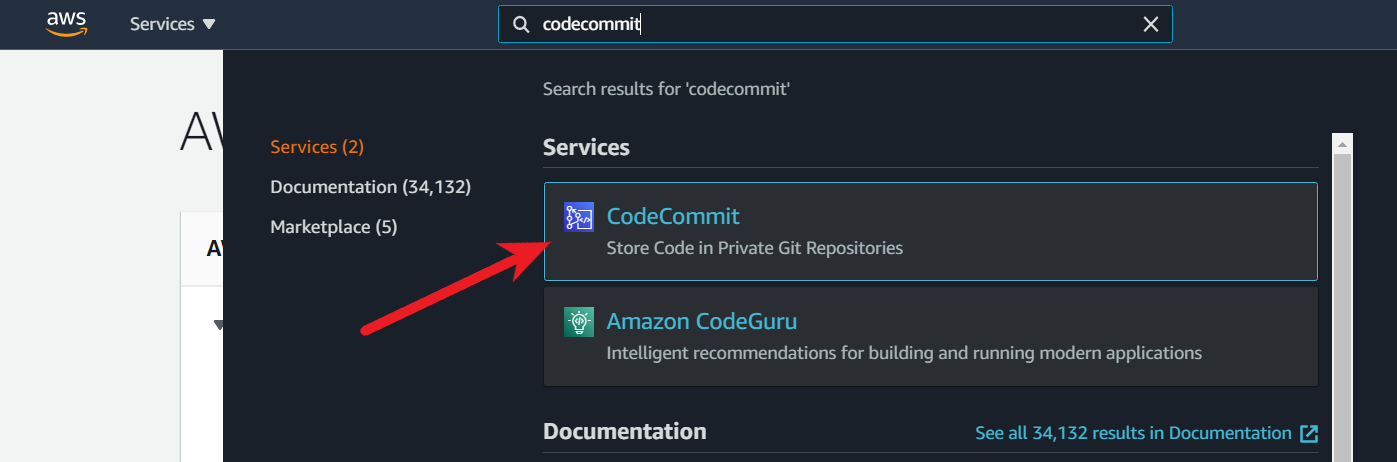
9-Verify the Coupon system Deployment

10-Clean up

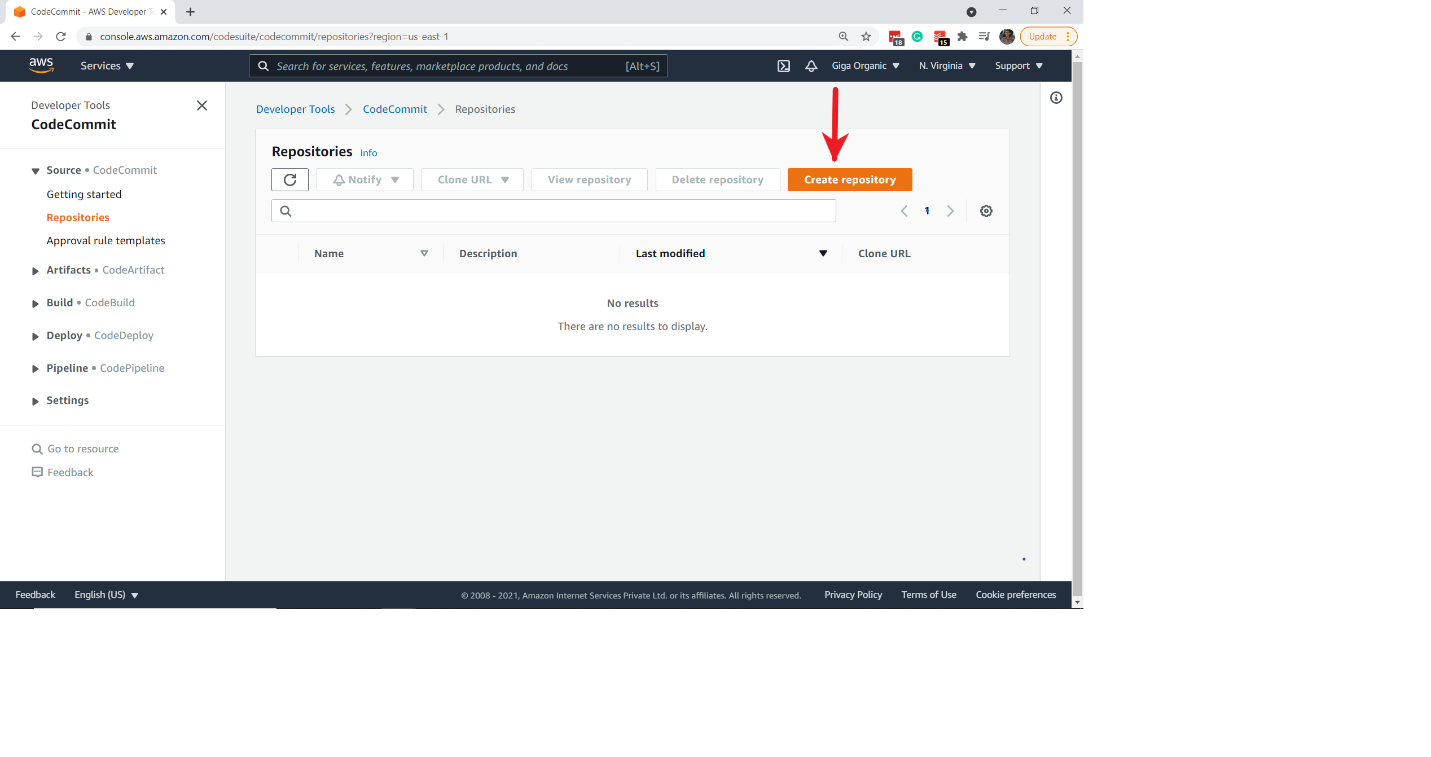
Set up an AWS codecommit repository to store the application code

You need to create a new repository in AWS CodeCommit where the application code can be stored. Eventually, you will have to connect this repository to the pipeline, so any new update pushed to this repository is automatically deployed to a set of EC2 instances.

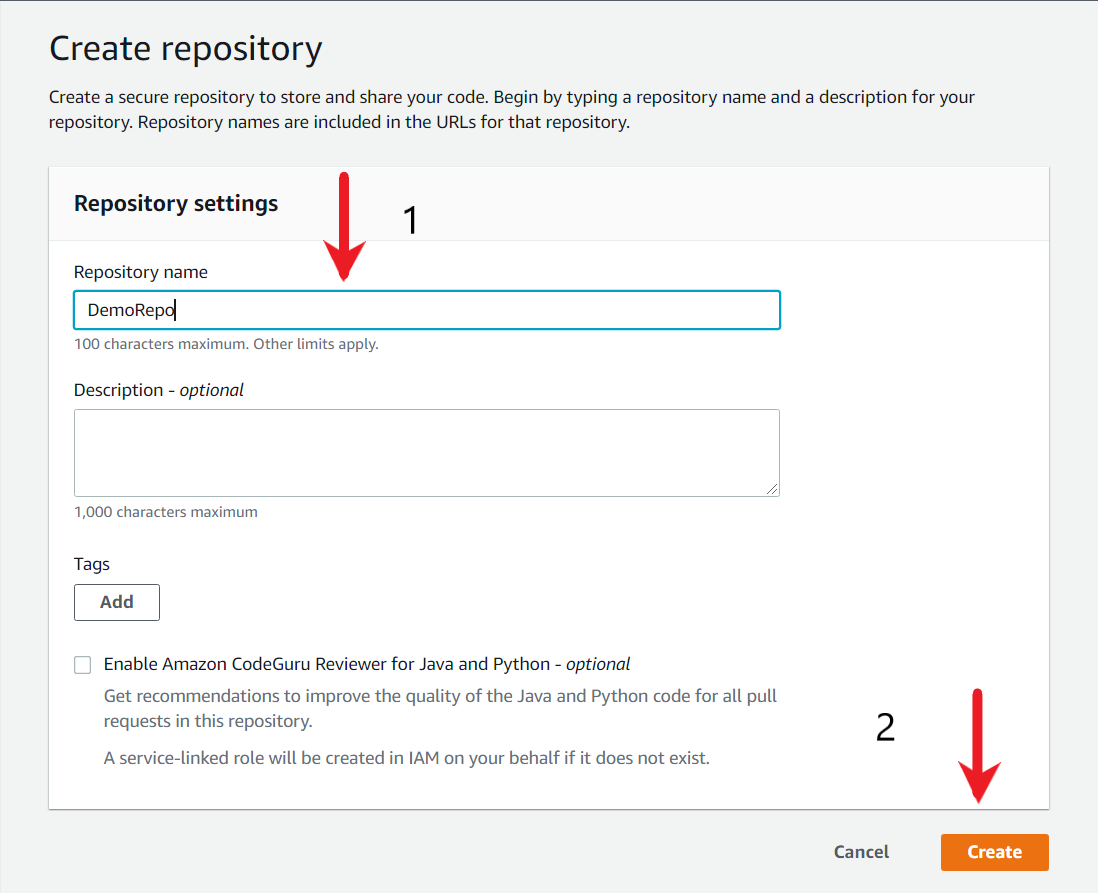
1. From the AWS console, search for the CodeCommit service and click on the available option.



1. On the Repositories page, choose Create repository.



1. On the Create repository page, in Repository name, enter a name for your repository (for example, DemoRepo) and click on Create

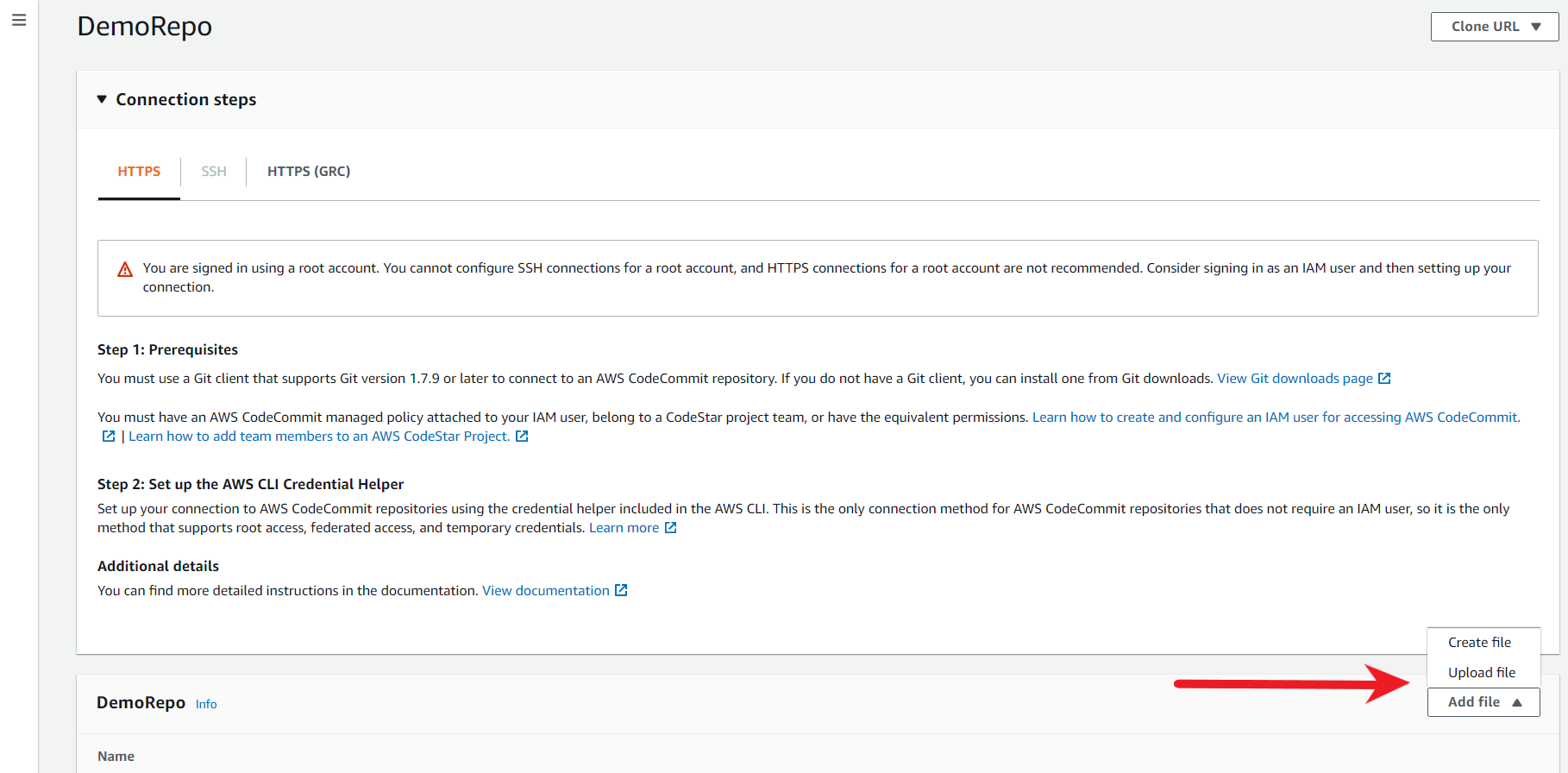


<https://docs.aws.amazon.com/codecommit/latest/userguide/getting-started-cc.html>

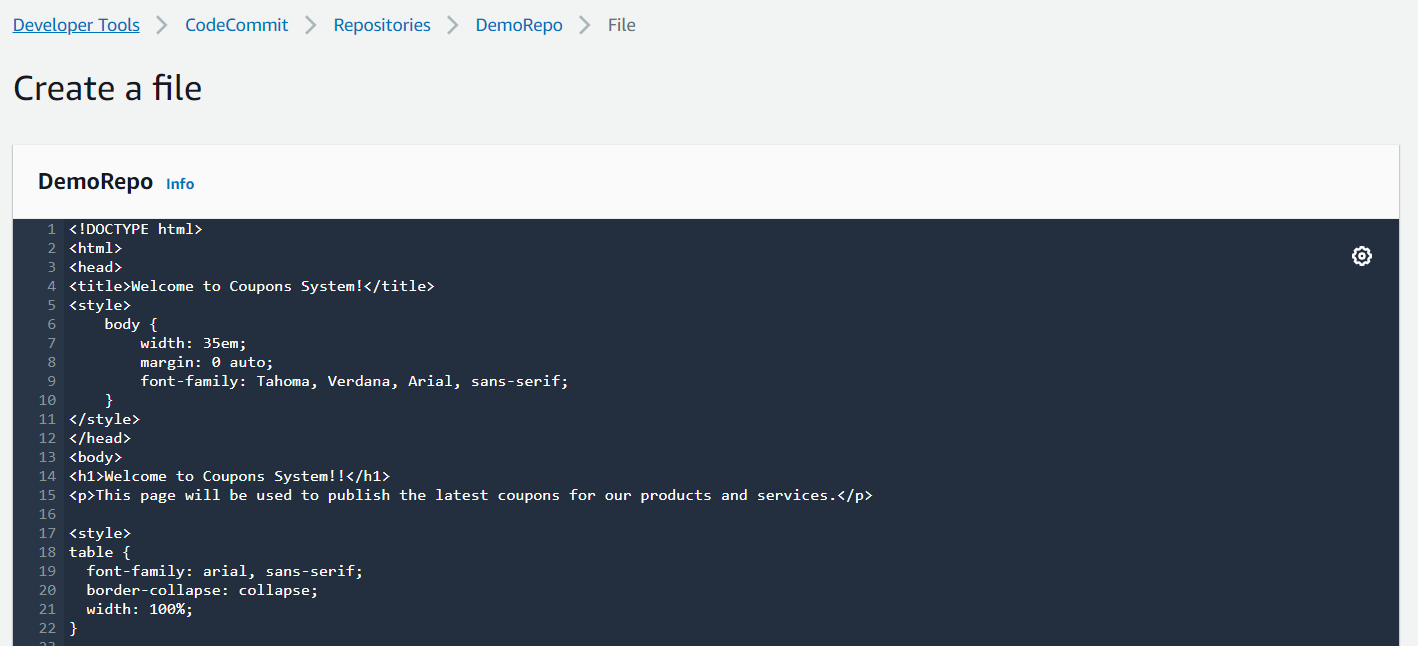
Add sample code to the codecommit repository

The index.html file that sample coupon codes and its webpage design needs to be committed as part of the CodeCommit repository. Within your existing repository, commit the file to it.

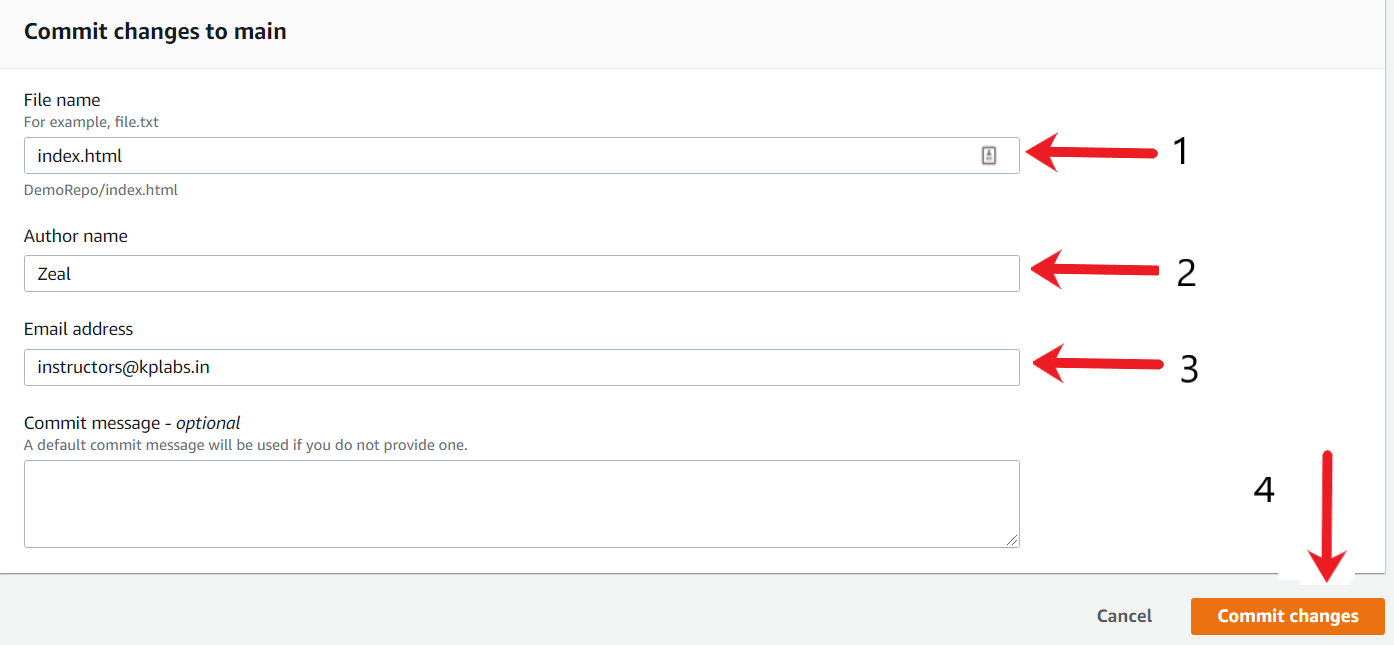
1. Open the CodeCommit console, and choose your repository from the Repositories list.
2. Click on the Add file button, and then choose Create a file.



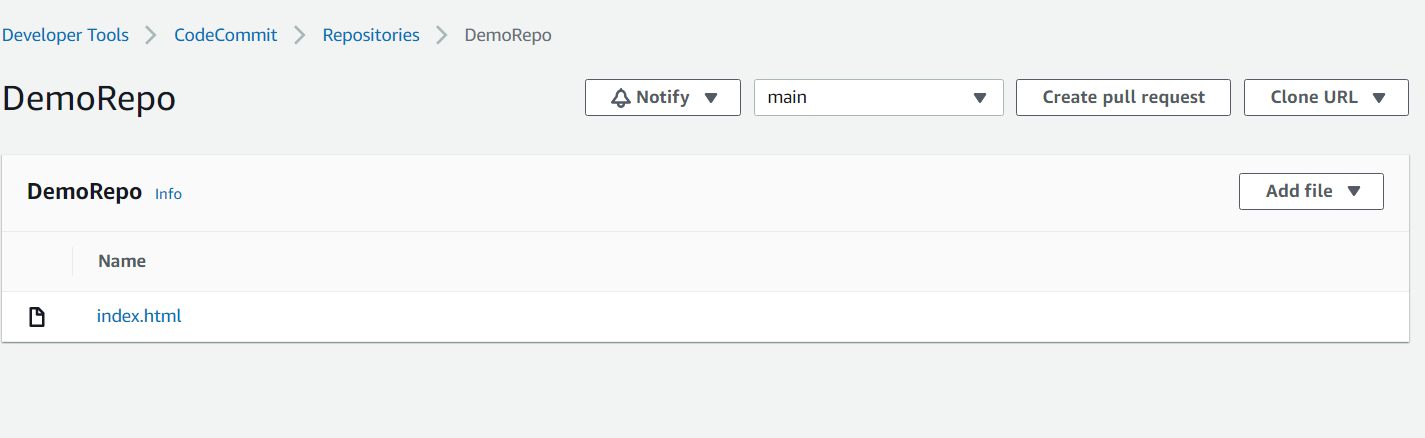
1. Copy the contents of the index.html file from your workstation to CodeCommit code-editor.



1. Scroll to the bottom page and add the File Name as index.html followed by Author name and email address post which click on the Commit changes button. This will add the index.html file inside the repository.



1. After the file has been committed, go back to the DemoRepo repository and the index.html file should be visible on the home page.



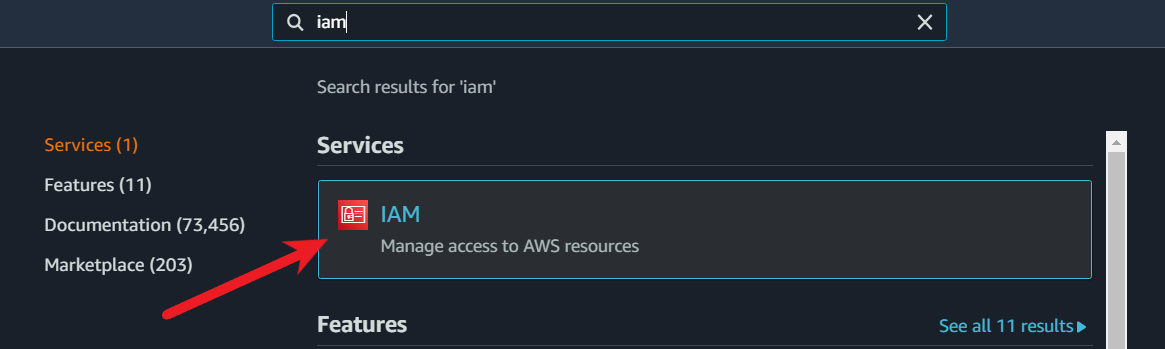
Ref index.html

Create IAM role for EC2 running codedeploy agent

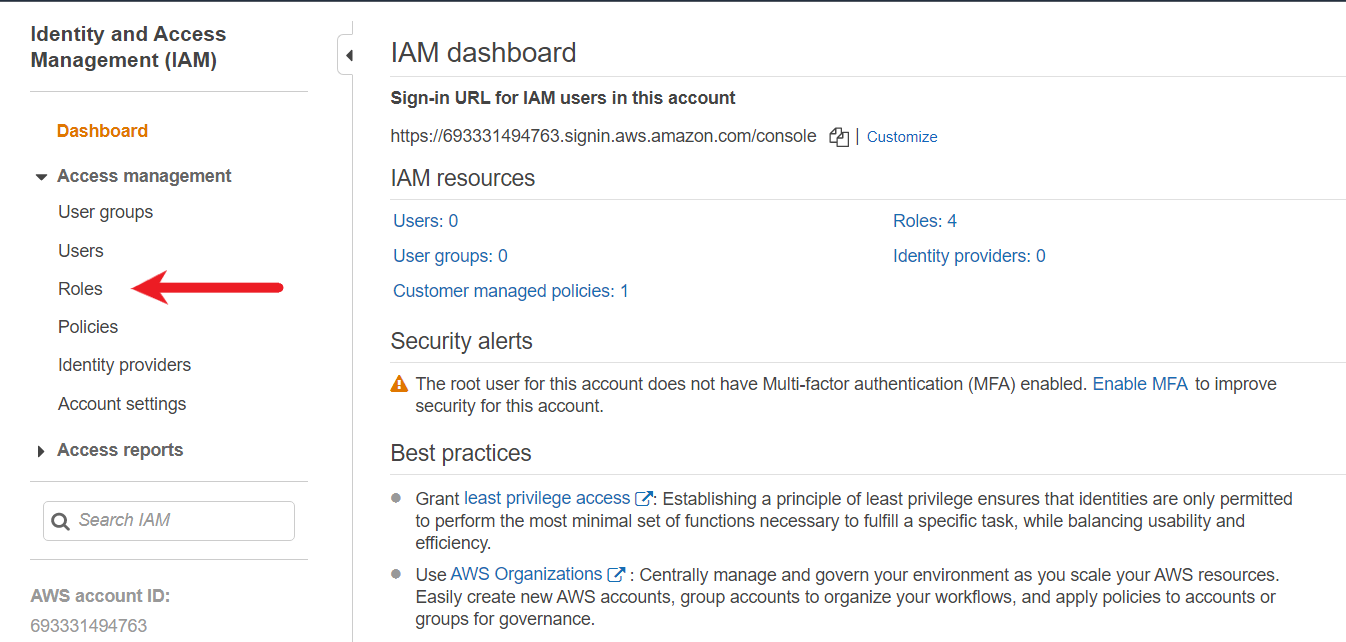
An IAM role is required for EC2 instances that will be running the CodeDeploy agent.

This IAM role should allow the agent in EC2 instance to fetch files that the CodeDeploy agent uses to deploy your application.

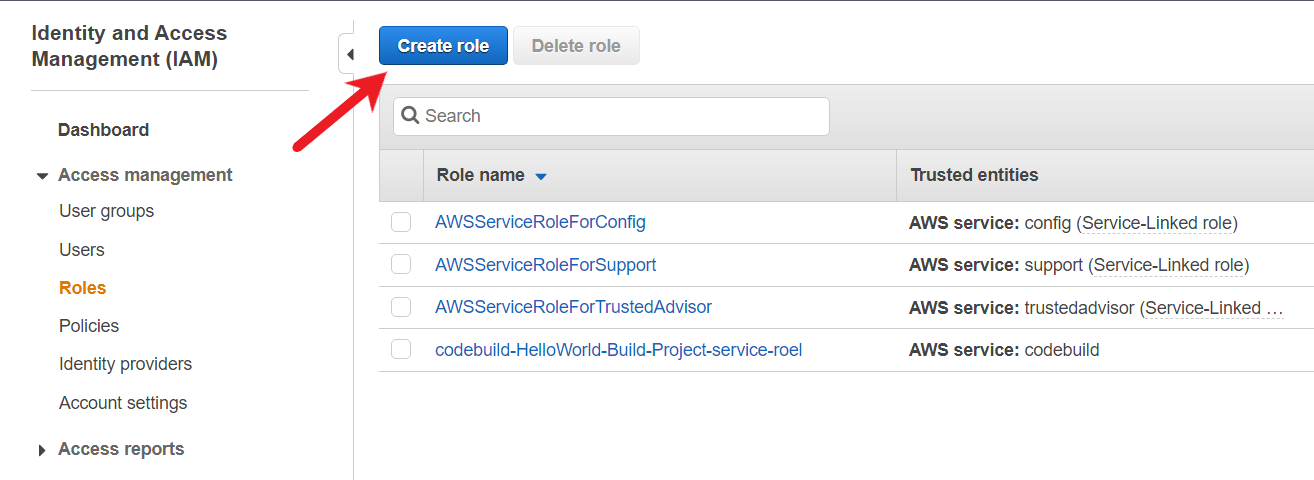
1. To create an instance role, from the AWS console, search for the CodeCommit service and click on the available option.



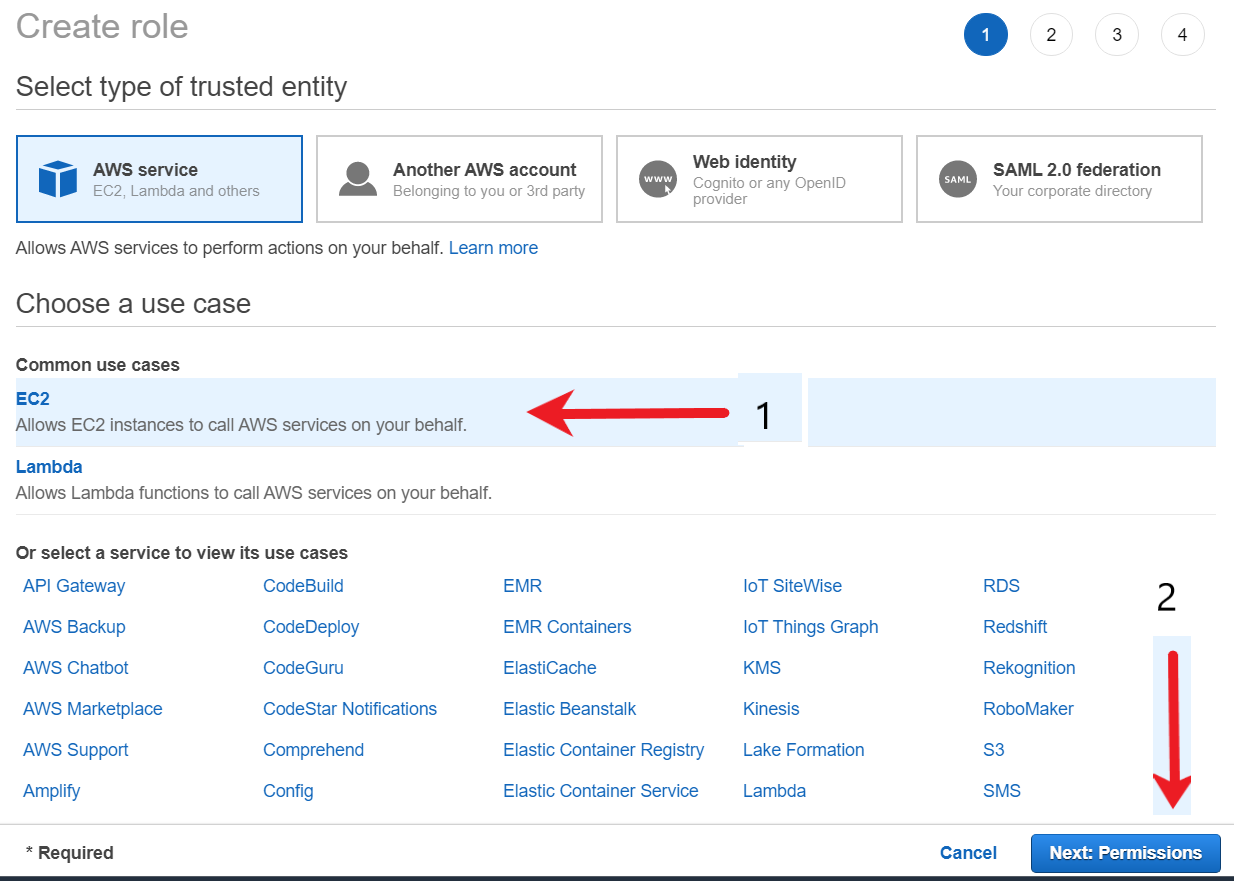
1. From the console dashboard, choose Roles.



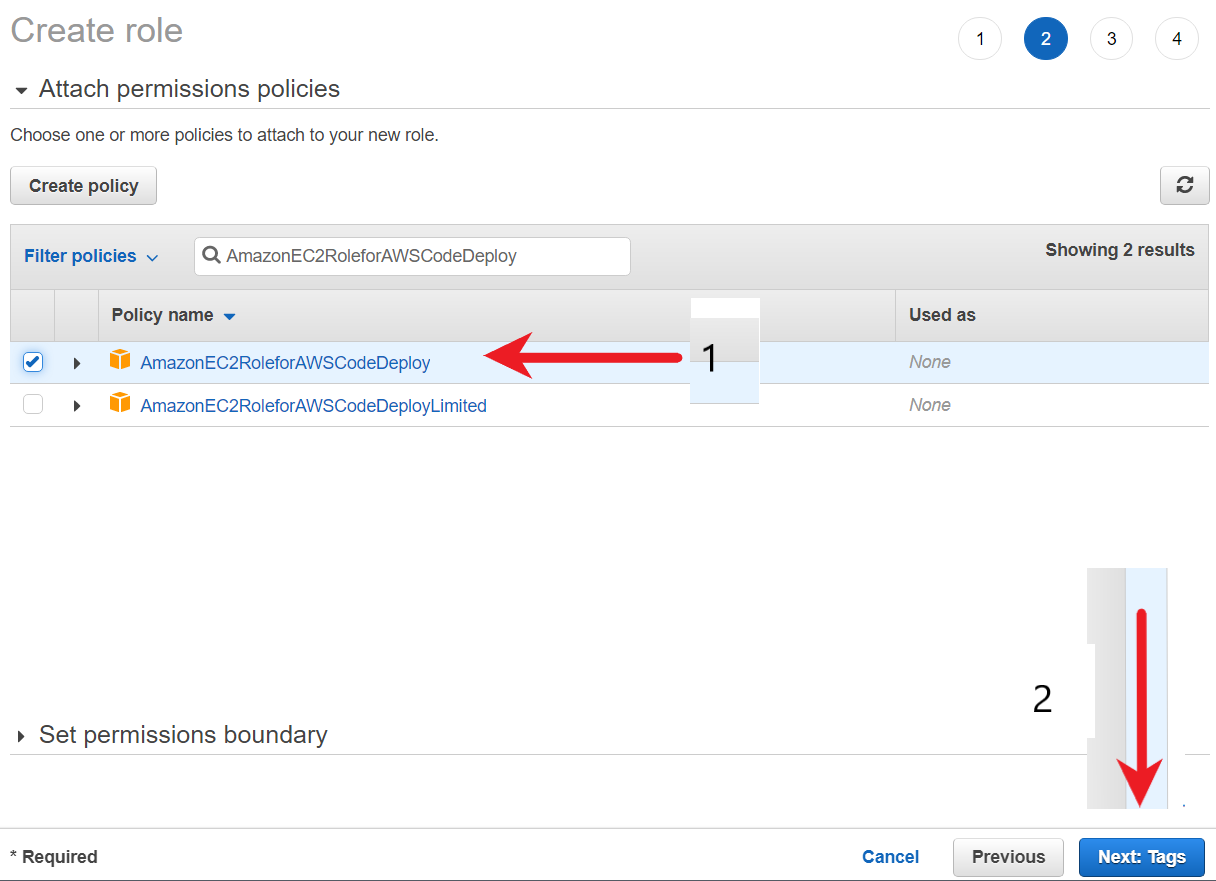
1. Choose Create role.



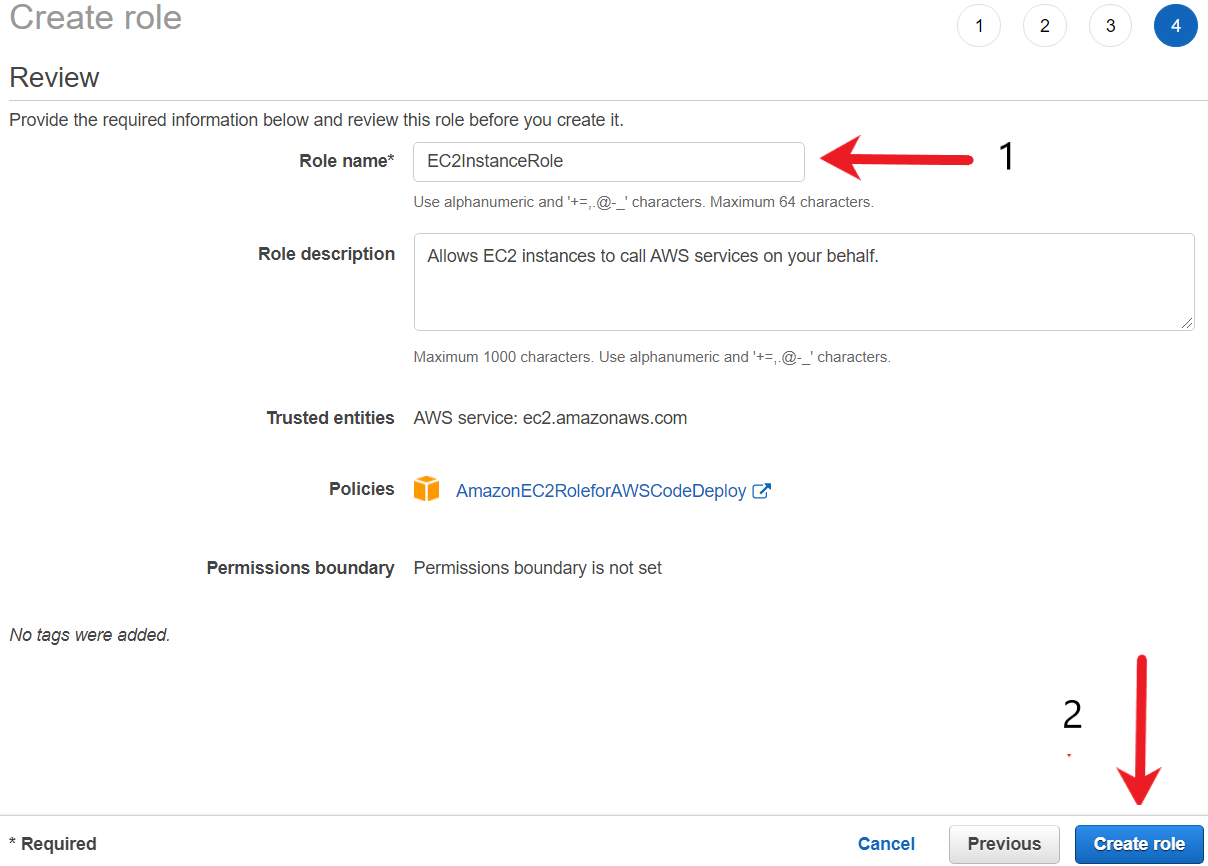
1. Under Select type of trusted entity, select AWS service. Under Choose a use case, select EC2. Under Select your use case, choose EC2. Choose Next: Permissions.



1. Search for and select the policy named AmazonEC2RoleforAWSCodeDeploy, and then choose Next: Tags.



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



1. Click on Create role button to create the IAM role.

<https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html>

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html>

Create EC2 and install codedeploy Agent

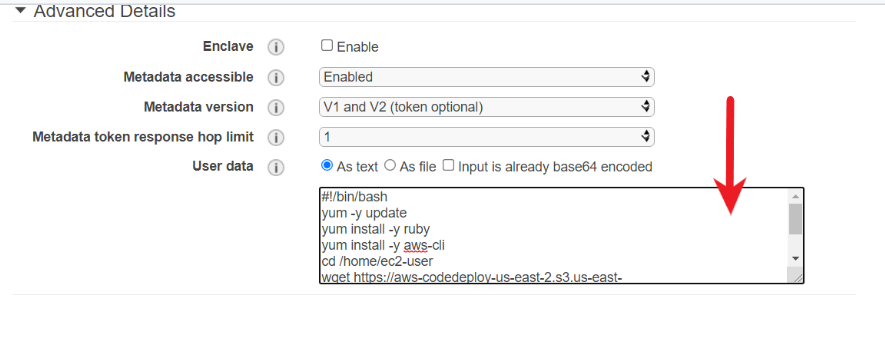
You need to create an EC2 instance which has a CodeDeploy agent installed. This will be the instance where the coupon system will be deployed.

The EC2 instance to be based on Amazon Linux 2 AMI. The Port 80 (HTTP) and SSH (22) should be open for everyone (0.0.0.0/0)

The EC2 instances must be appropriately tagged for it to be used with the AWS CodeDeploy service.

You also need to attach appropriate IAM role for the CodeDeploy agent in EC2 instance to fetch files that the CodeDeploy agent uses to deploy your application.

1. Open the Amazon EC2 console by searching for EC2.
2. From the console dashboard, choose Launch Instance, and select Launch instance from the options that pop up.
3. Choose an Amazon Machine Image (AMI), locate Amazon Linux 2 AMI (HVM), SSD Volume Type, and then choose Select. (This AMI is labeled "Free tier eligible" and can be found at the top of the list.)
4. 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. Configure Instance Details page, do the following:
6. In Number of instances, enter 1.
7. In the IAM role, choose the IAM role you created in the previous procedure (for example, EC2InstanceRole).
8. Expand Advanced Details, and in the User data field, enter the following:
   1. #!/bin/bash
   2. yum -y update
   3. yum install -y ruby
   4. yum install -y aws-cli
   5. cd /home/ec2-user
   6. wget https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install
   7. chmod +x ./install
   8. ./install auto



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

1. Leave the rest of the items. Configure Instance Details page unchanged. Choose Next: Add Storage.
2. Add Storage page unchanged, and then choose Next: Add Tags.
3. Choose Add Tag. In Key, enter Name, and in Value, enter MyCodePipelineDemo. 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.
4. Configure Security Group page, do the following:

Next to Assign a security group, choose Create a new security group. The security group should have following rules:

SSH allowed for 0.0.0.0/0

HTTP allowed for 0.0.0.0/0

In the row for SSH, under Source, make sure it is 0.0.0.0/0

Choose Review and Launch.

1. On the Review Instance Launch page, choose Launch. When prompted for a key pair, choose Proceed without a key pair. When you are ready, select the acknowledgment check box, and then choose Launch Instances.
2. Choose View Instances to close the confirmation page and return to the console.

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.)

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.

<https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install.html>

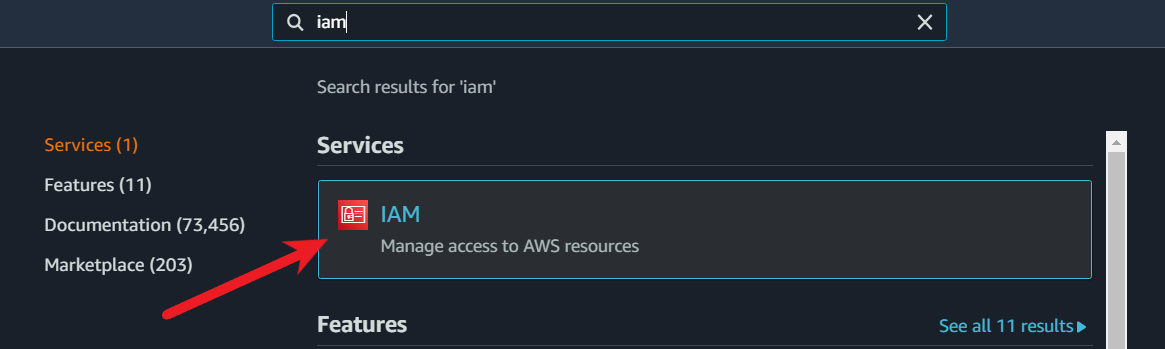
Create Service Role for codedeploy

The CodeDeploy service needs to be able to perform various actions, some of these include but not limited to:

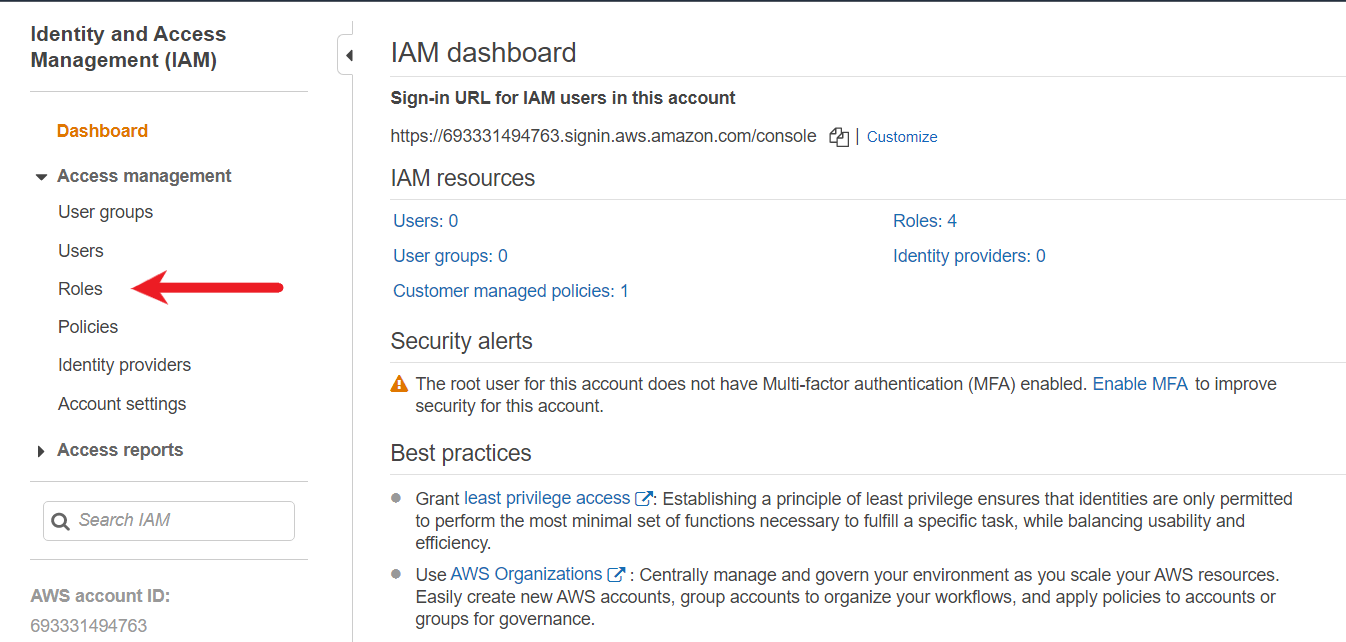
1. Read the tags on your instances or identify your Amazon EC2 instances.
2. Retrieve information about CloudWatch alarms.

You need to create a service role for Code Deploy. This service role should have the AWSCodeDeployRole policy attached to it.

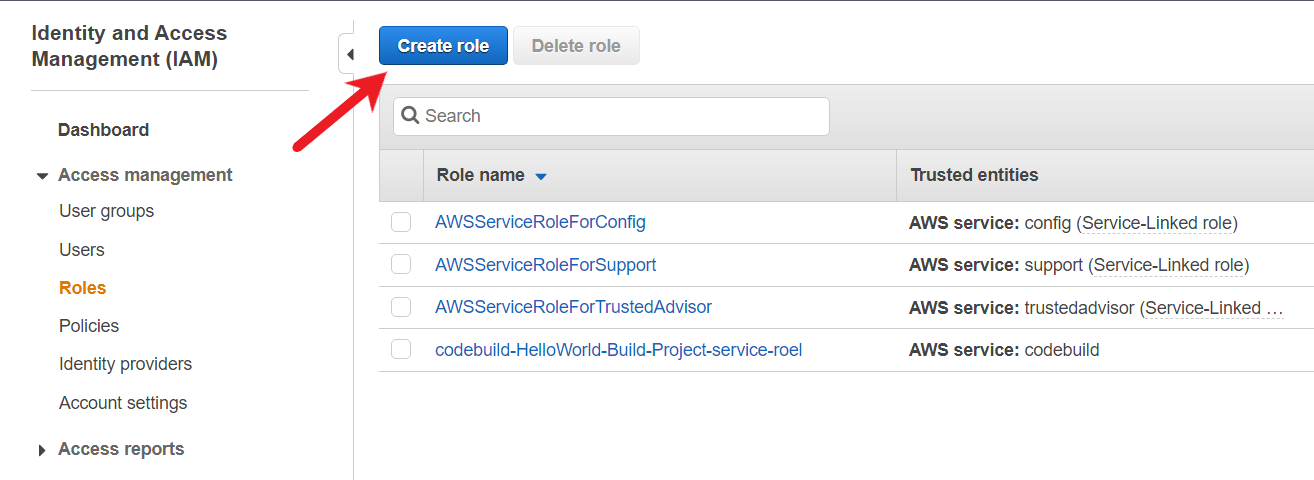
1. To create an instance role, from the AWS console, search for the CodeCommit service and click on the available option.



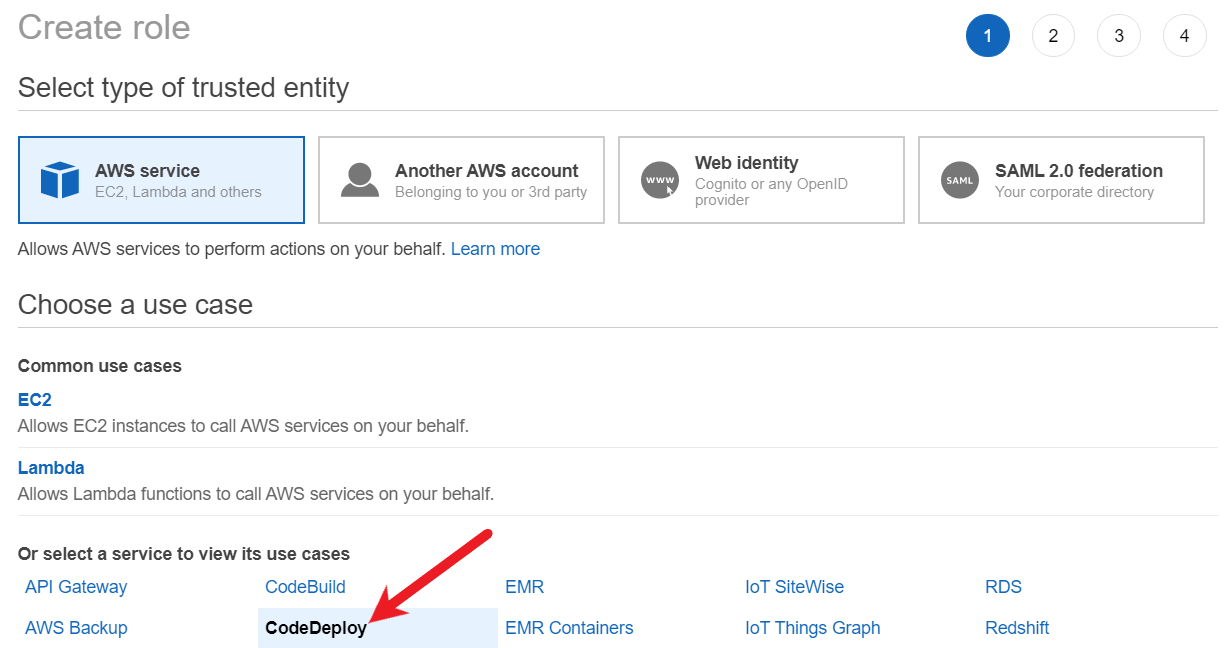
1. From the console dashboard, choose Roles.

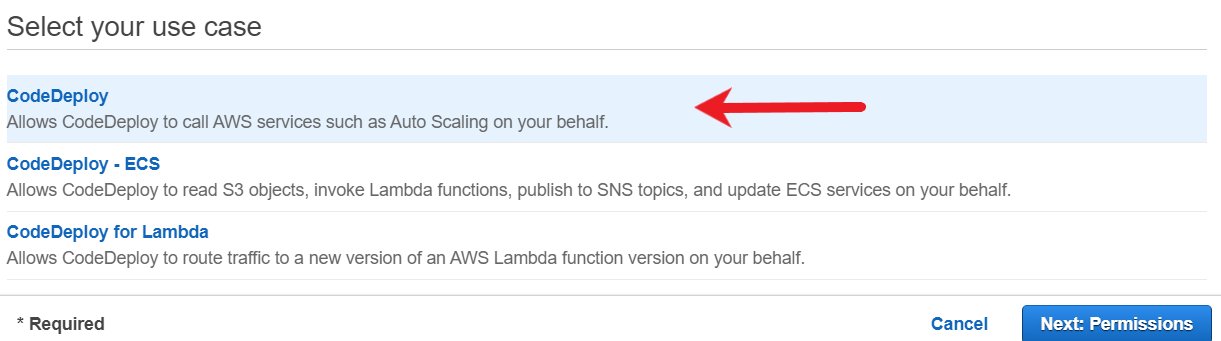


1. Choose Create role.

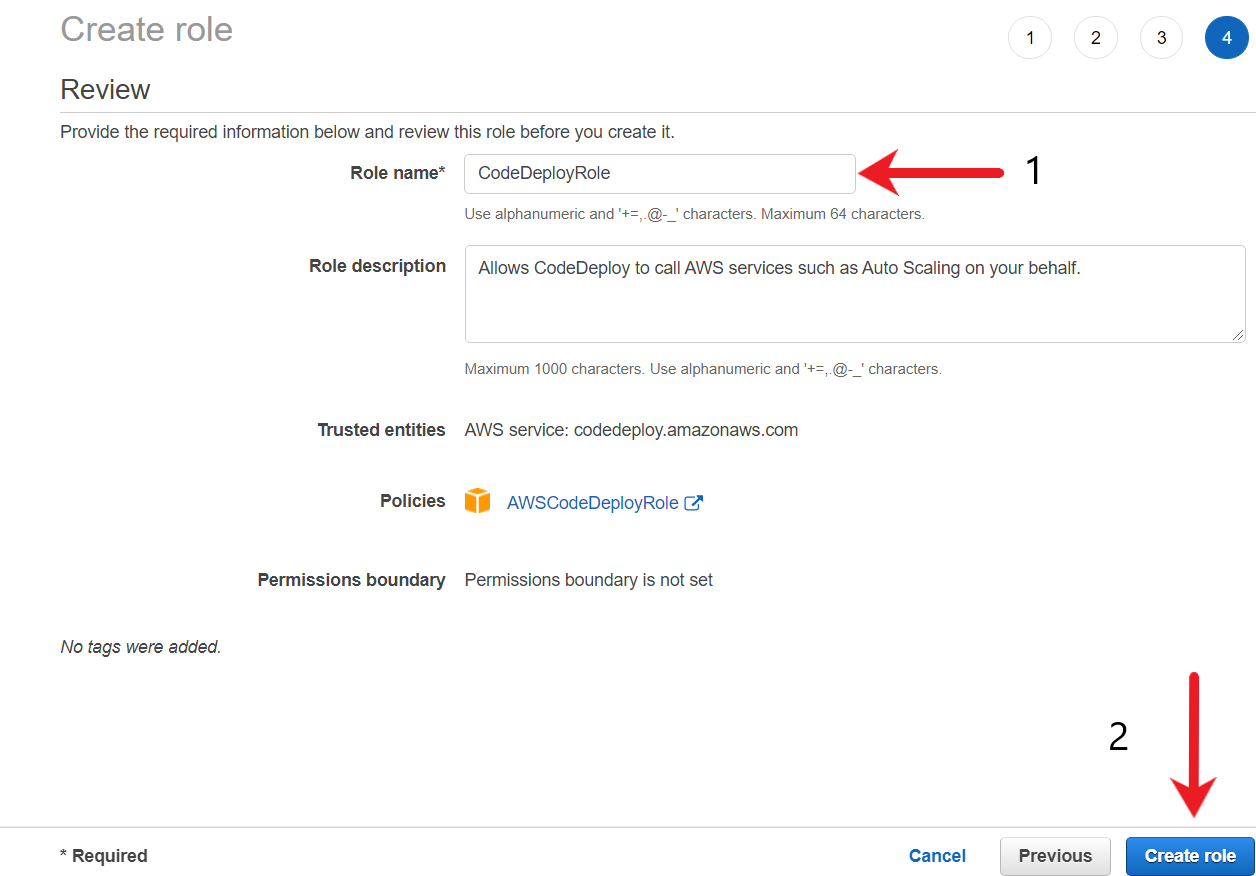


1. Under Select type of trusted entity, select AWS service. Under Choose a use case, select CodeDeploy. Under Select your use case, choose CodeDeploy.





1. Choose Next: Permissions. The AWSCodeDeployRole managed policy is already attached to the role
2. Choose Next: Tags, and Next: Review.
3. Enter a name for the role (for example, CodeDeployRole), and then choose Create role.



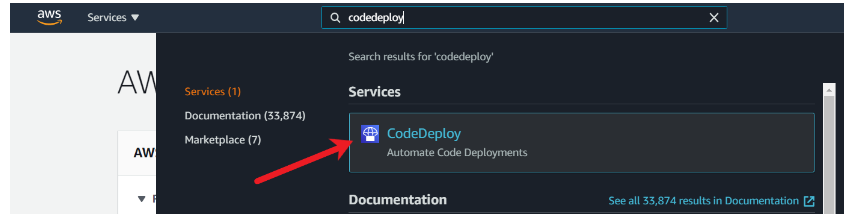
create an application in codedeploy

A CodeDeploy application and its associated deployment group needs to be created. The deployment group should be configured to push updates to the EC2 instance created in the previous steps.

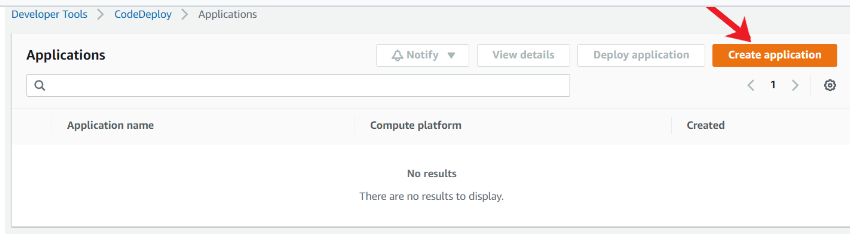
Since you are performing a POC, following deployment configuration can be used:

* Deployment Setting of OneAtATime
* Deployment Type of In-Place
* Load Balancer: Disabled (Not required)

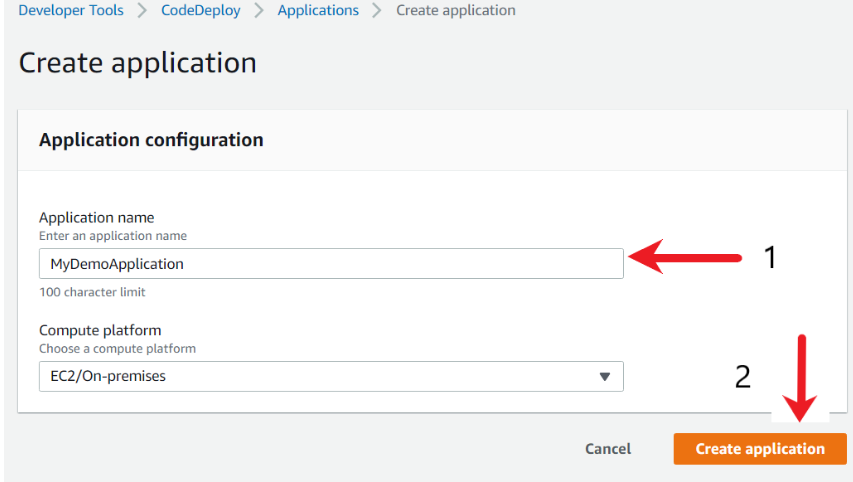
1. Search for the CodeDeploy service and click on the available option.



1. If the Applications page does not appear, on the AWS CodeDeploy menu, choose Applications.
2. Choose Create application.

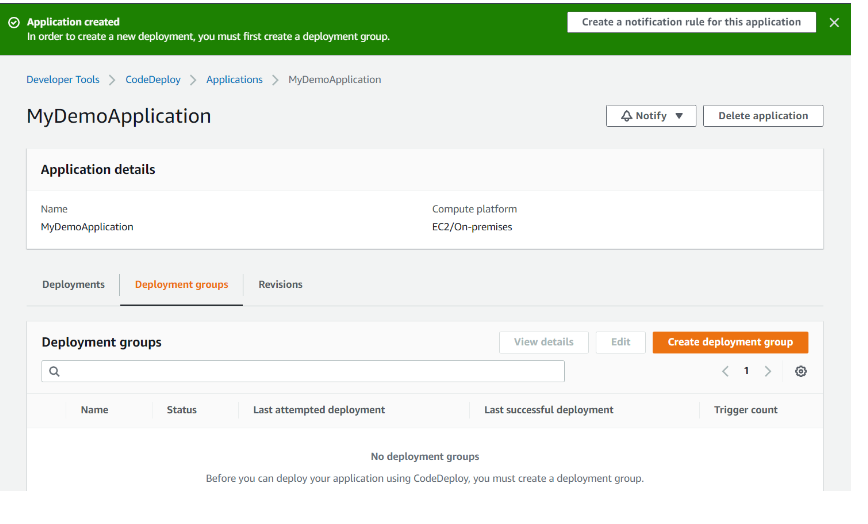


1. In Application name, enter MyDemoApplication.
2. In Compute Platform, choose EC2/On-premises.



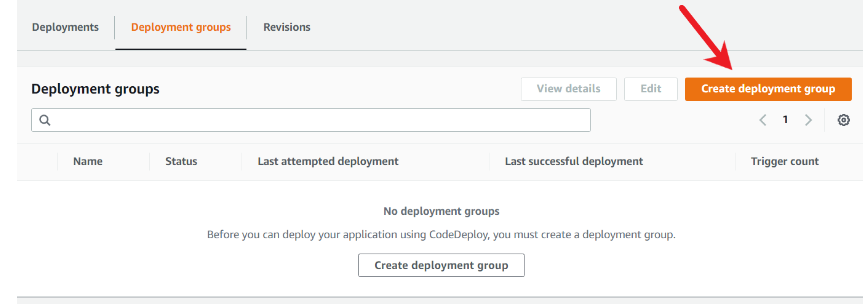
1. Choose Create application.

To create a deployment group in CodeDeploy. On successful creation, you will see the application configuration screen along with a success message.

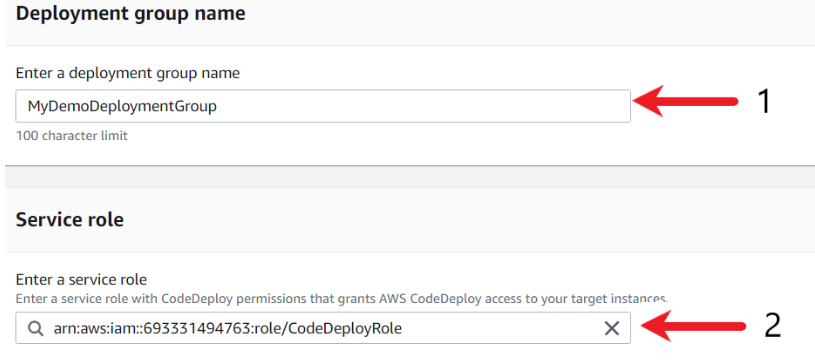


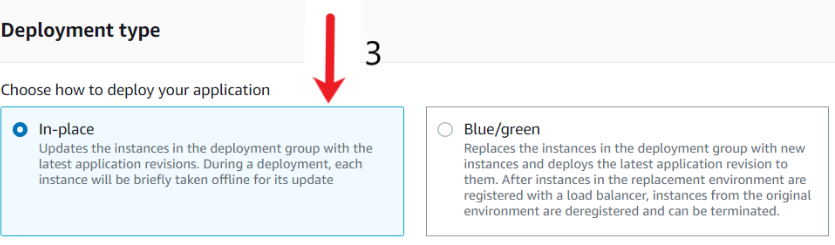
A deployment group is a resource that defines deployment-related settings like which instances to deploy to and how fast to deploy them.

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

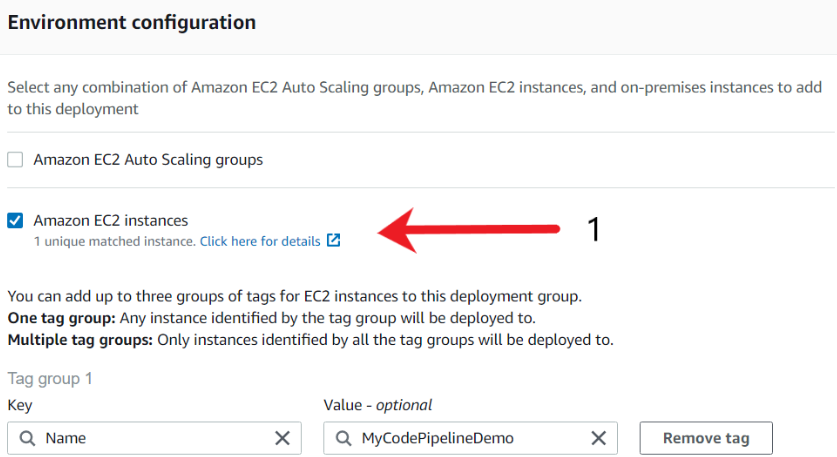


1. In Deployment group name, enter MyDemoDeploymentGroup
2. In Service Role, choose the service role you created earlier (for example, CodeDeployRole).
3. Under Deployment type, choose In-place.

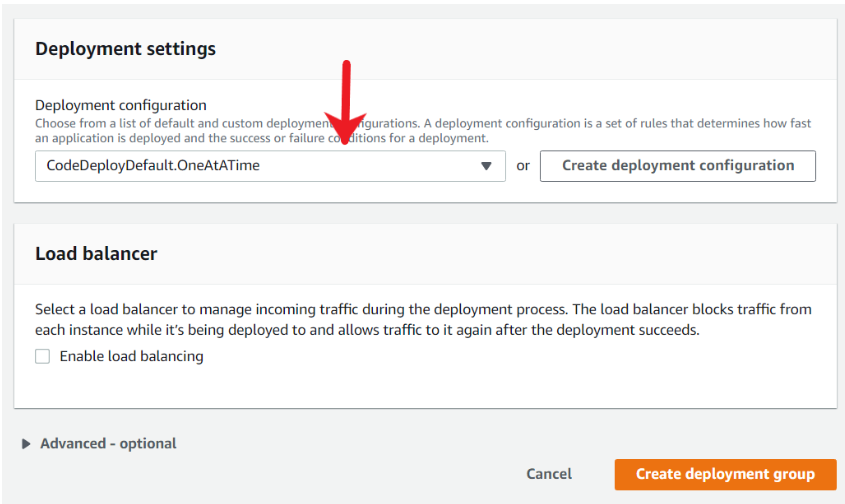




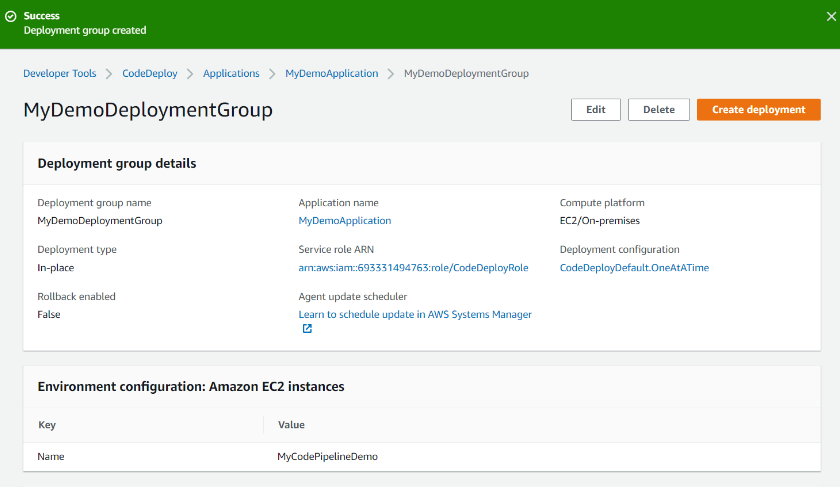
1. Under Environment configuration, choose Amazon EC2 Instances. In the Key field, enter Name. In the Value field, enter the name you used to tag the instance (for example, MyCodePipelineDemo).



1. Under Deployment configuration, choose CodeDeployDefault.OneAtaTime.



1. 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.
2. Choose Create deployment group. Once the deployment group is created, you will see a success message similar to the following screenshot.



Create AppSpec File for codedeploy

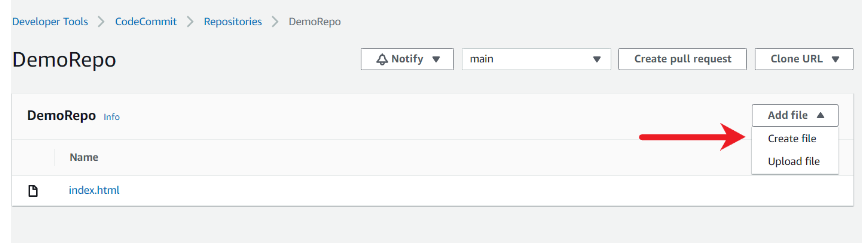
The development team has provided the following commands and instructions that are used to create a coupon based system. All the commands must run as a root user.

1. yum -y install httpd
2. systemctl start httpd

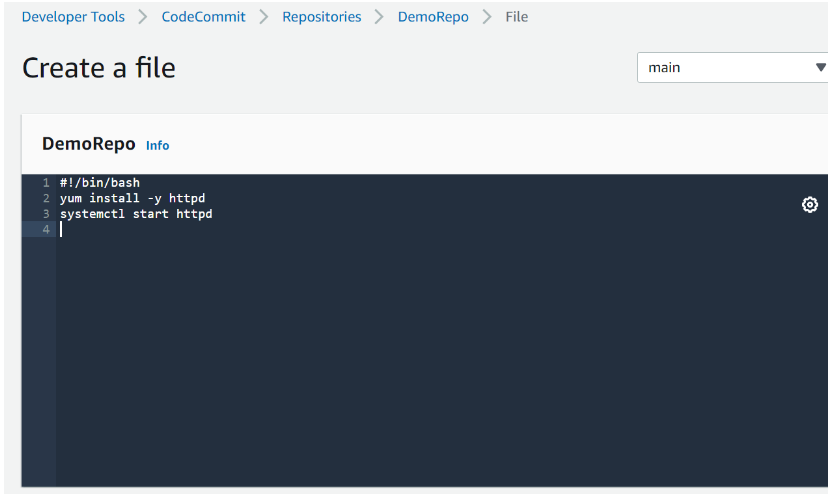
The index.html file from CodeCommit repository must be stored at /var/www/html location.

You have to translate the above requirements to create a AppSec file to instruct CodeDeploy to implement the system.

1. Create a file named install\_httpd.sh in your existing CodeCommit repository. This file contains the command to install the coupon system. Following is the content of the install\_httpd.sh file. In order to do that, go to your Code Commit repository and click on Add FIle and select Create file

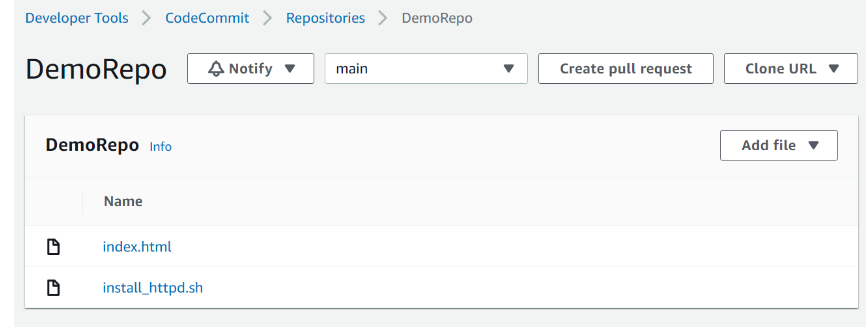


1. Following is the contents that needs to be added as part of this file. Do note copy paste as it might lead to an error at the later stage.. Instead, manually type it in the CodeCommit editor.
   1. #!/bin/bash
   2. yum install -y httpd
   3. systemctl start httpd



1. Scroll to the bottom of the screen and fill in the details related to file name, author and email. In the demo, I have named the file as install\_httpd.sh

At this stage, there should be two files within the CodeCommit repository, namely the index.html and install\_httpd.sh

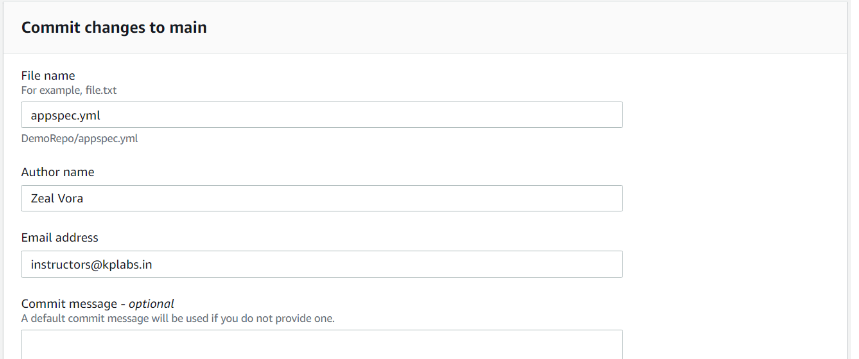


1. Based on the requirements given by the developer, the appspec file should be able to do the following tasks
   1. Take the source file of index.html and move it to a destination of /var/www/html
   2. Use the Hooks to run the commands that are part of the install\_httpd.sh file

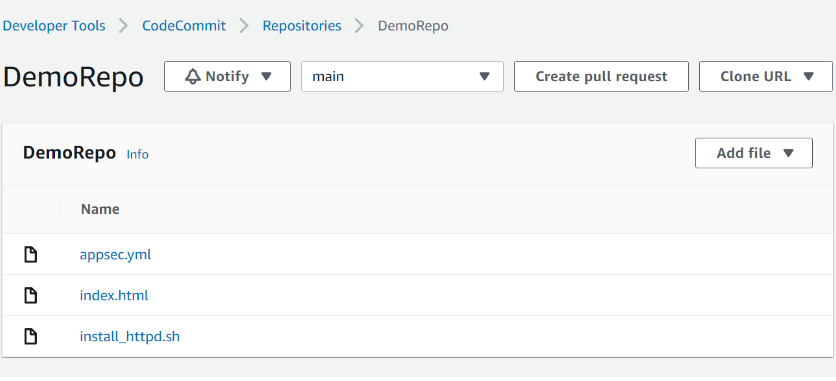
Based on the above requirements, you can find the sample appsec.yaml file that can be used.

1. We will create a new file in our existing code commit repository and paste the contents of the created appspec.yml





At this stage, there should be a total of 3 files in the CodeCommit repository, namely index.html, install\_httpd.sh and appspec.yml



Ref install-system.txt

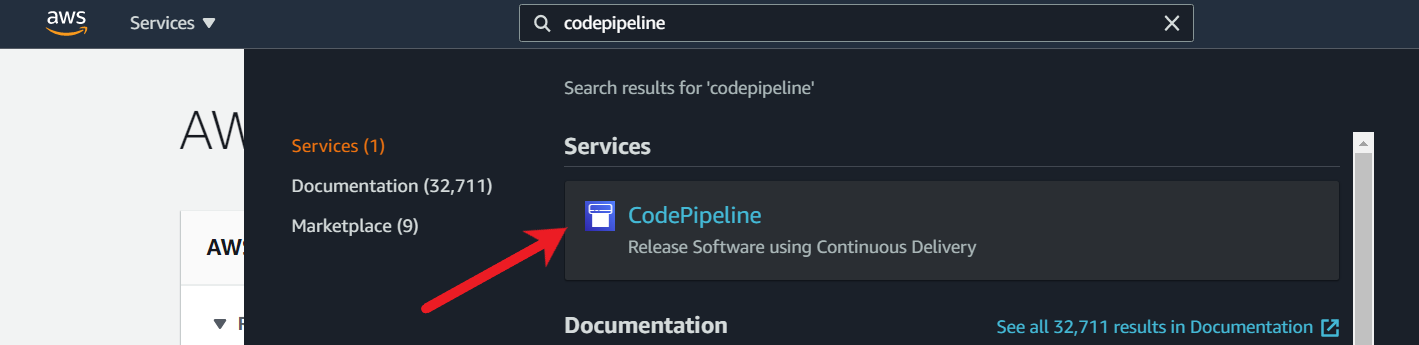
<https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file.html>

Create a pipeline using AWS Codedeploy

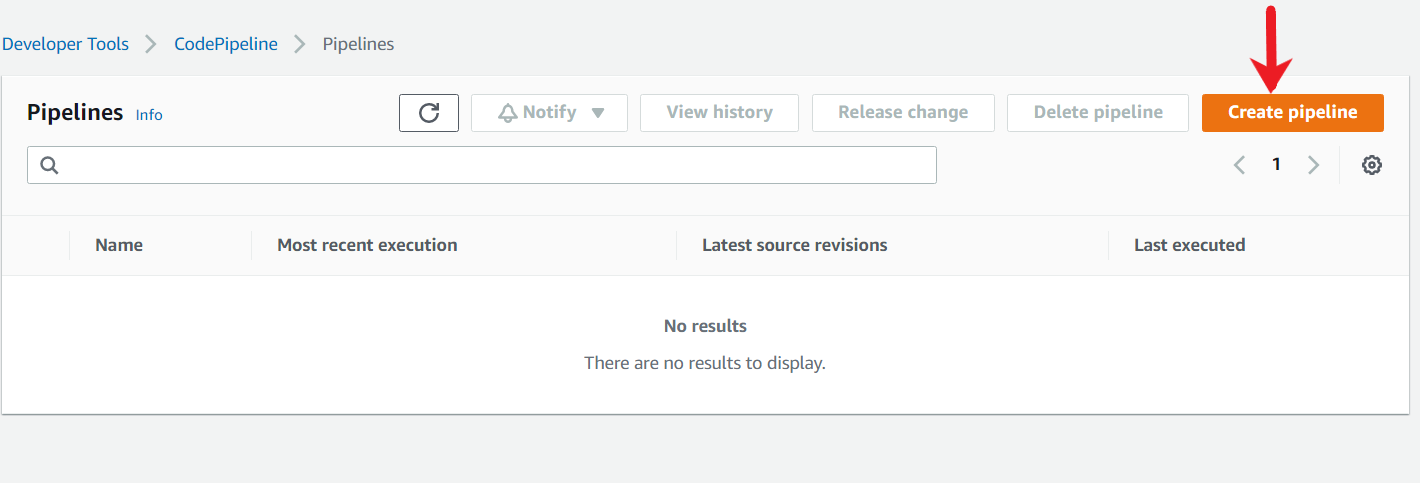
You have to use the AWS CodePipeline service to create a continuous delivery pipeline with appropriate source, and deploy stages.

The pipeline will detect changes in the code stored in your CodeCommit repository and then deploy the newer updates to the EC2 instances.

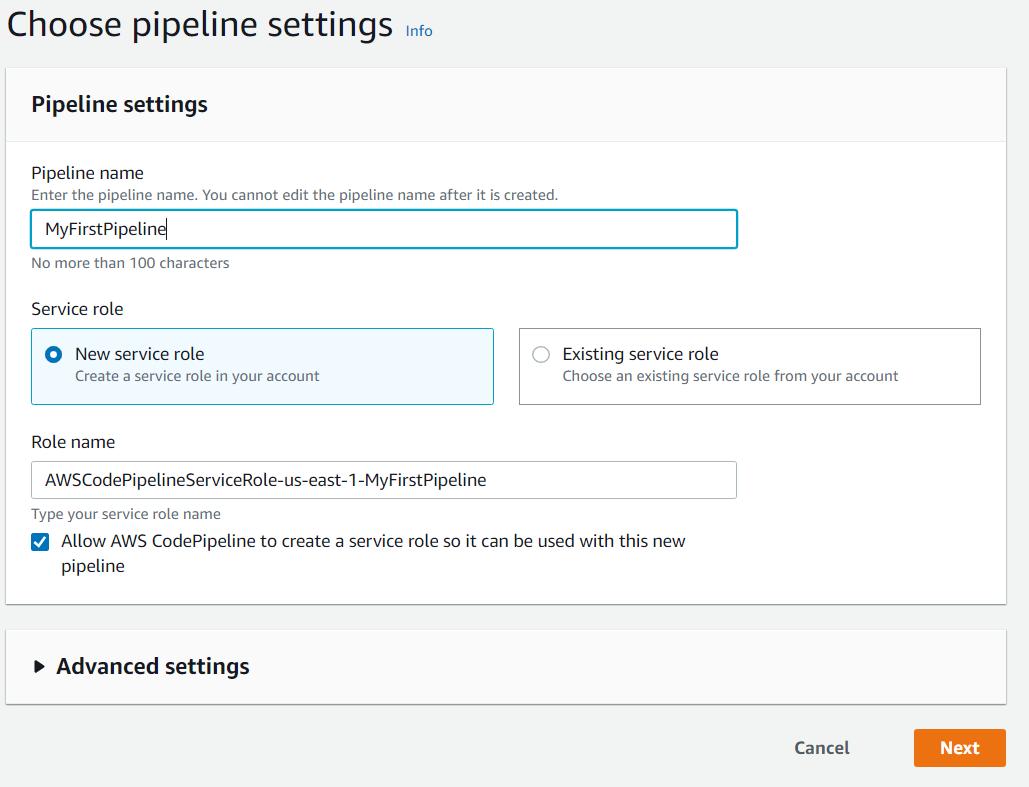
1. To create a CodePipeline pipeline, search for the CodePipeline service and click on the available option.



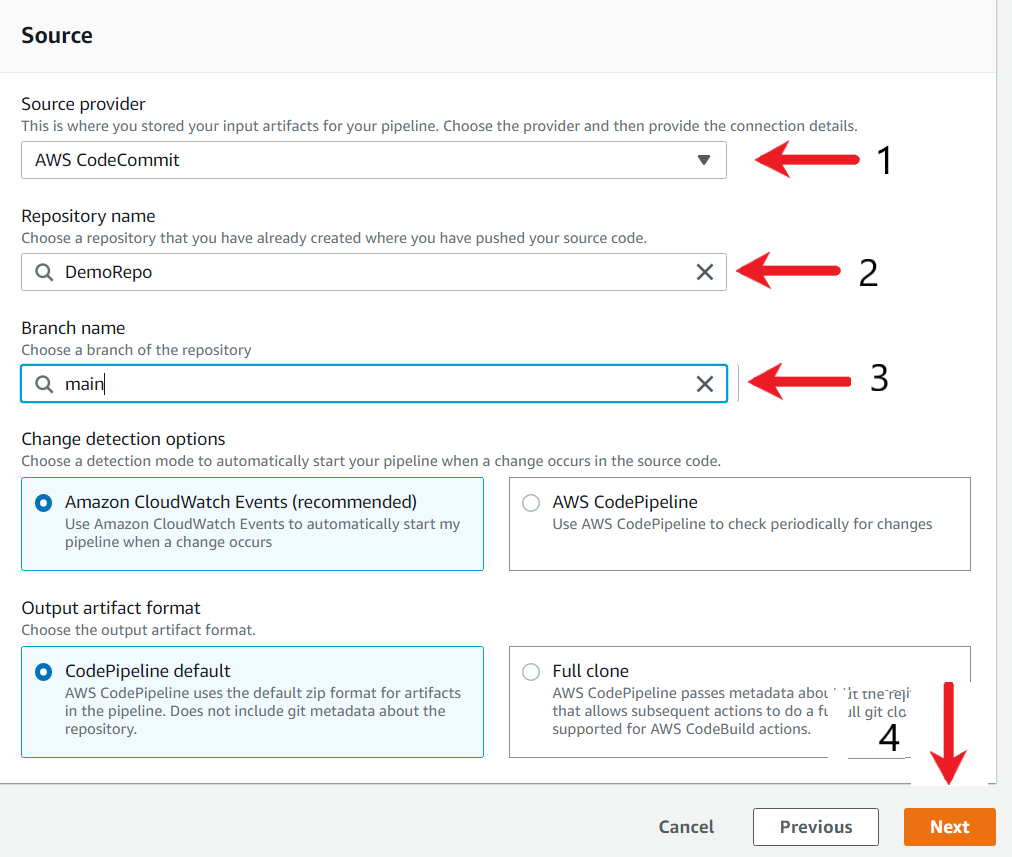
1. Click on the Create pipeline button.



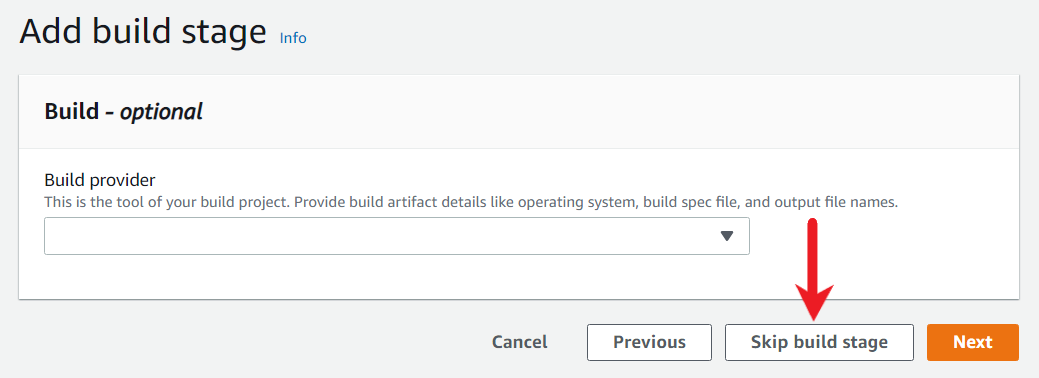
1. Under the pipeline settings, in Pipeline name, enter MyFirstPipeline
2. In Service role, choose New service role to allow CodePipeline to create a service role in IAM.



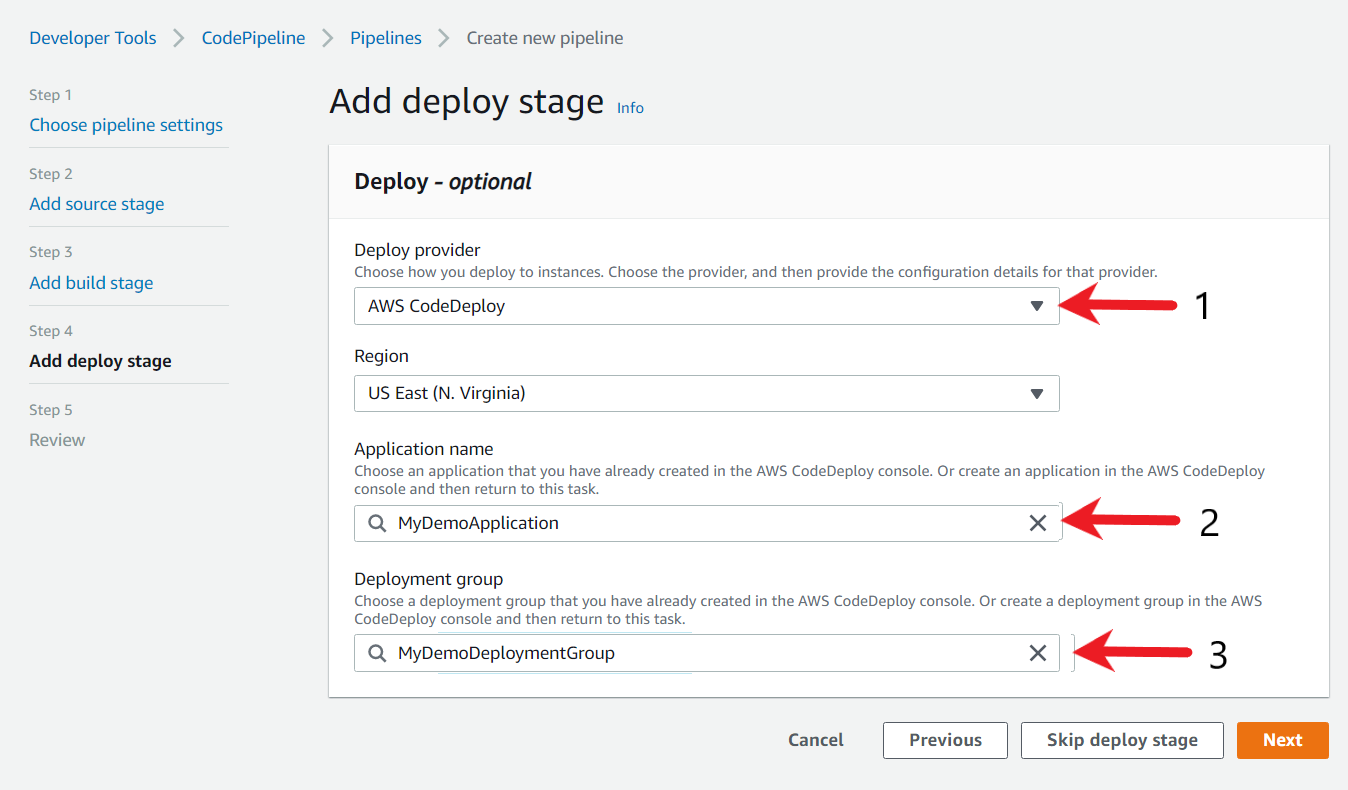
1. Leave the settings under Advanced settings at their defaults, and then choose Next.
2. Add source stage, in Source provider, choose AWS CodeCommit. In the Repository name, choose the name of the CodeCommit repository you created (DemoRepo, for this example). In Branch name, choose main, and then choose Next step.



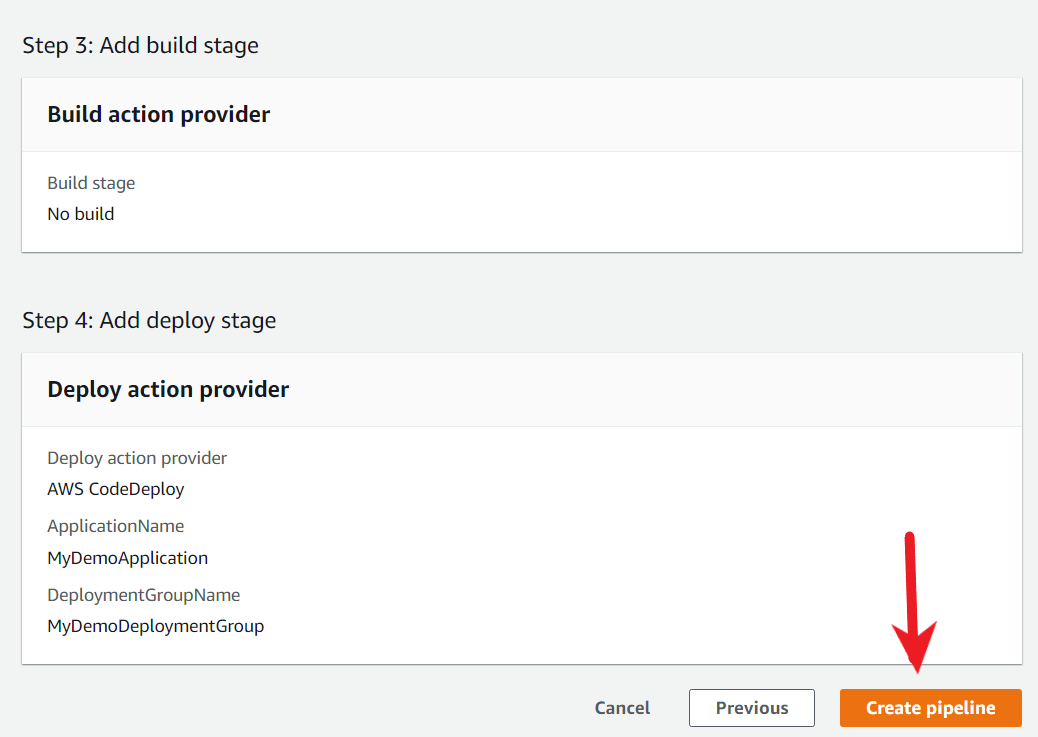
1. After you select the repository name and branch, a message displays the Amazon CloudWatch Events rule to be created for this pipeline. Under Change detection options, leave the defaults. This allows CodePipeline to use Amazon CloudWatch Events to detect changes in your source repository. Choose Next.
2. In the build stage screen, choose Skip build stage, and then accept the warning message by choosing Skip again. Choose Next.



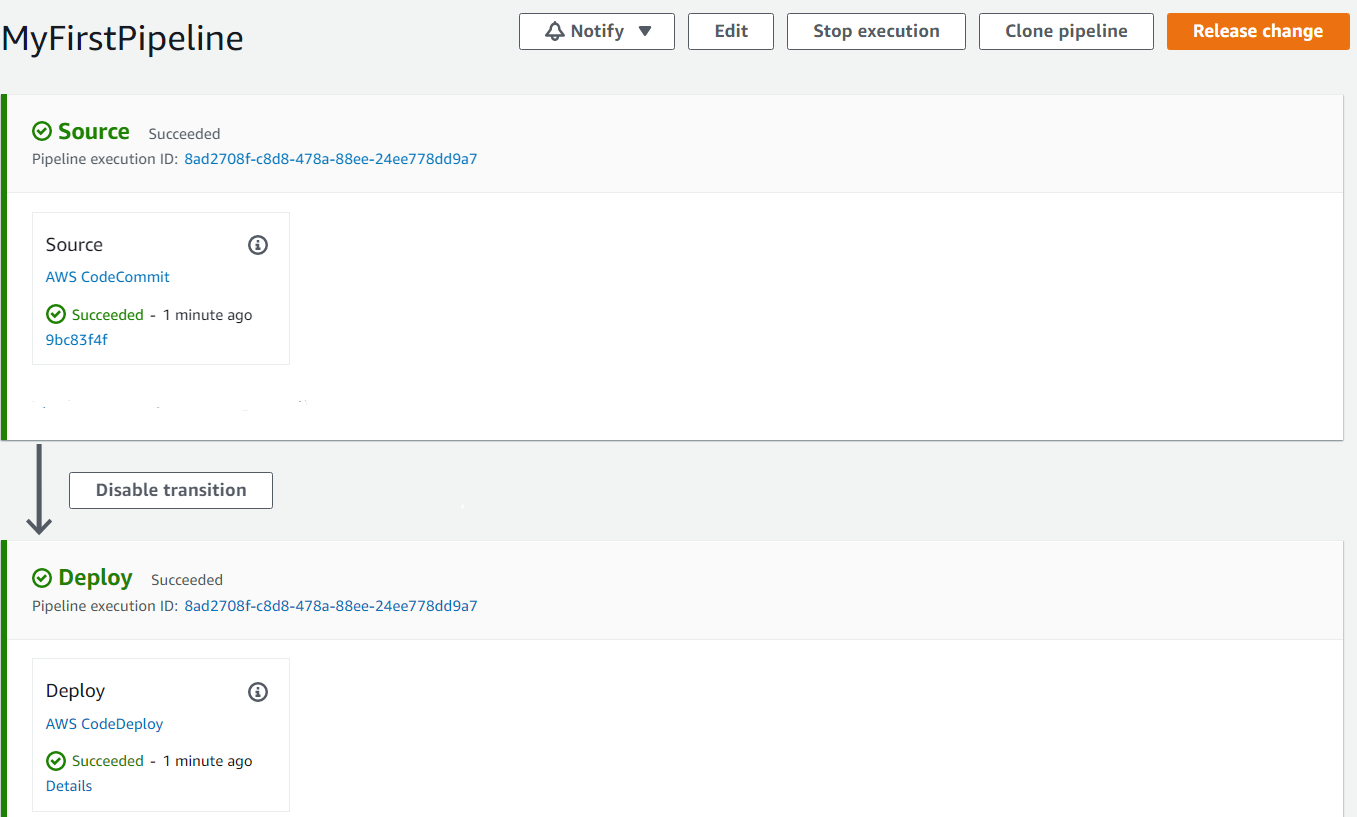
1. Add deploy stage, in Deploy provider, choose AWS CodeDeploy. In Application name, choose MyDemoApplication. In the Deployment group, choose MyDemoDeploymentGroup, and then choose Next step.



1. 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 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.

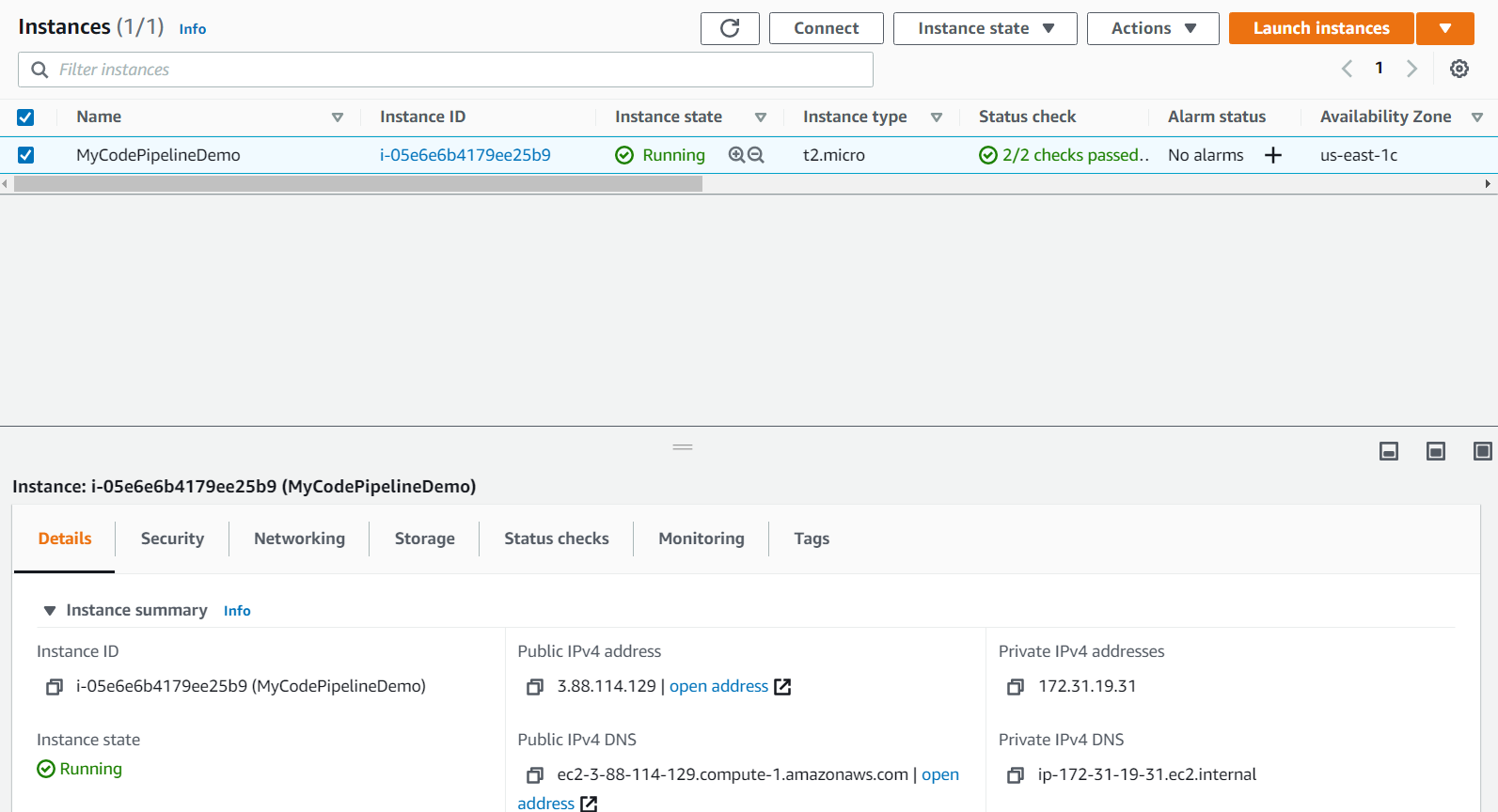


Verify the Coupon system Deployment

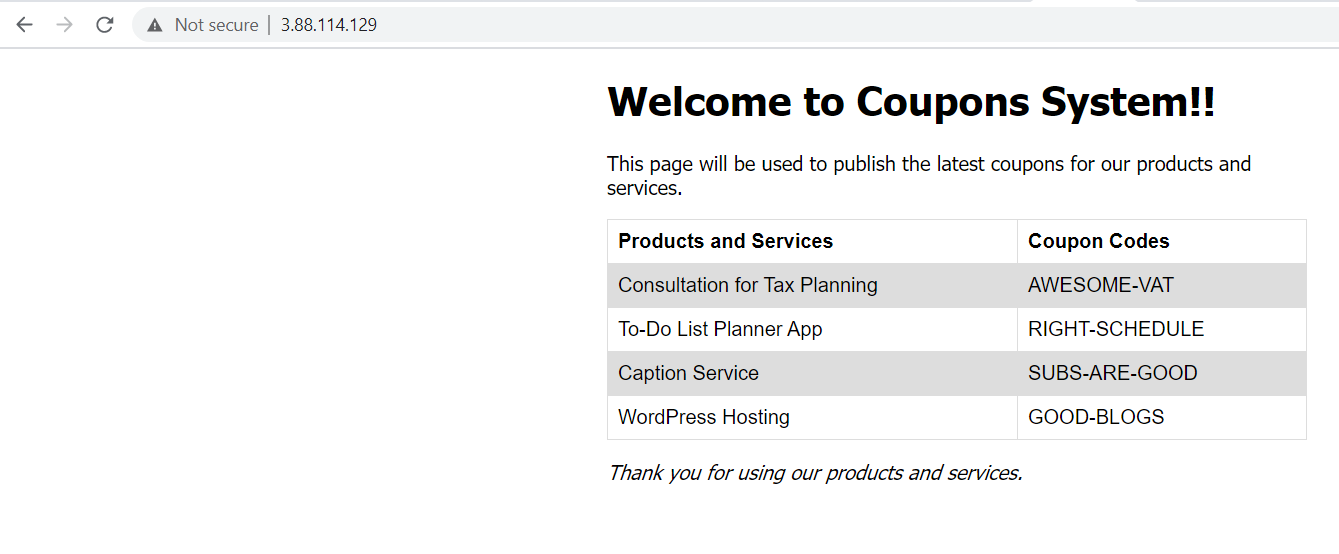
After successful execution of CodePipeline steps, it is necessary to verify if the coupon systems webpage is up and running. Load the Public IP of EC2 instance in the web-browser and verify if the coupon page loads.

Once the page loads successfully, make an update to the index.html file and edit the coupon-code associated with the Wordpress Hosting service. Replace the coupon-code associated with any of the service in the repository and verify if the new update is detected and deployed in the pipeline and subsequently if the coupon-page is updated.

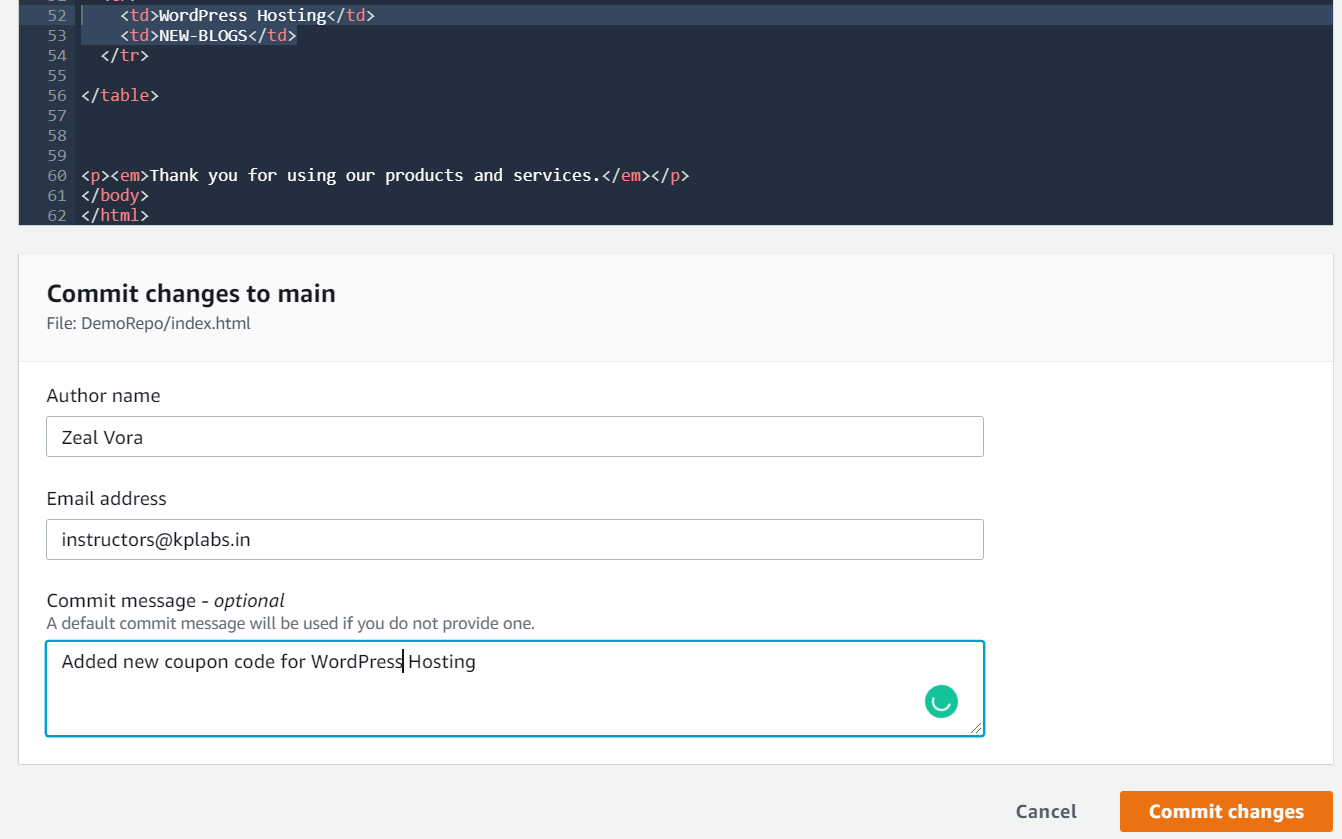
1. Open the Amazon EC2 console.
2. From the console dashboard, choose Instances
3. Click on the EC2 instance to expand its details



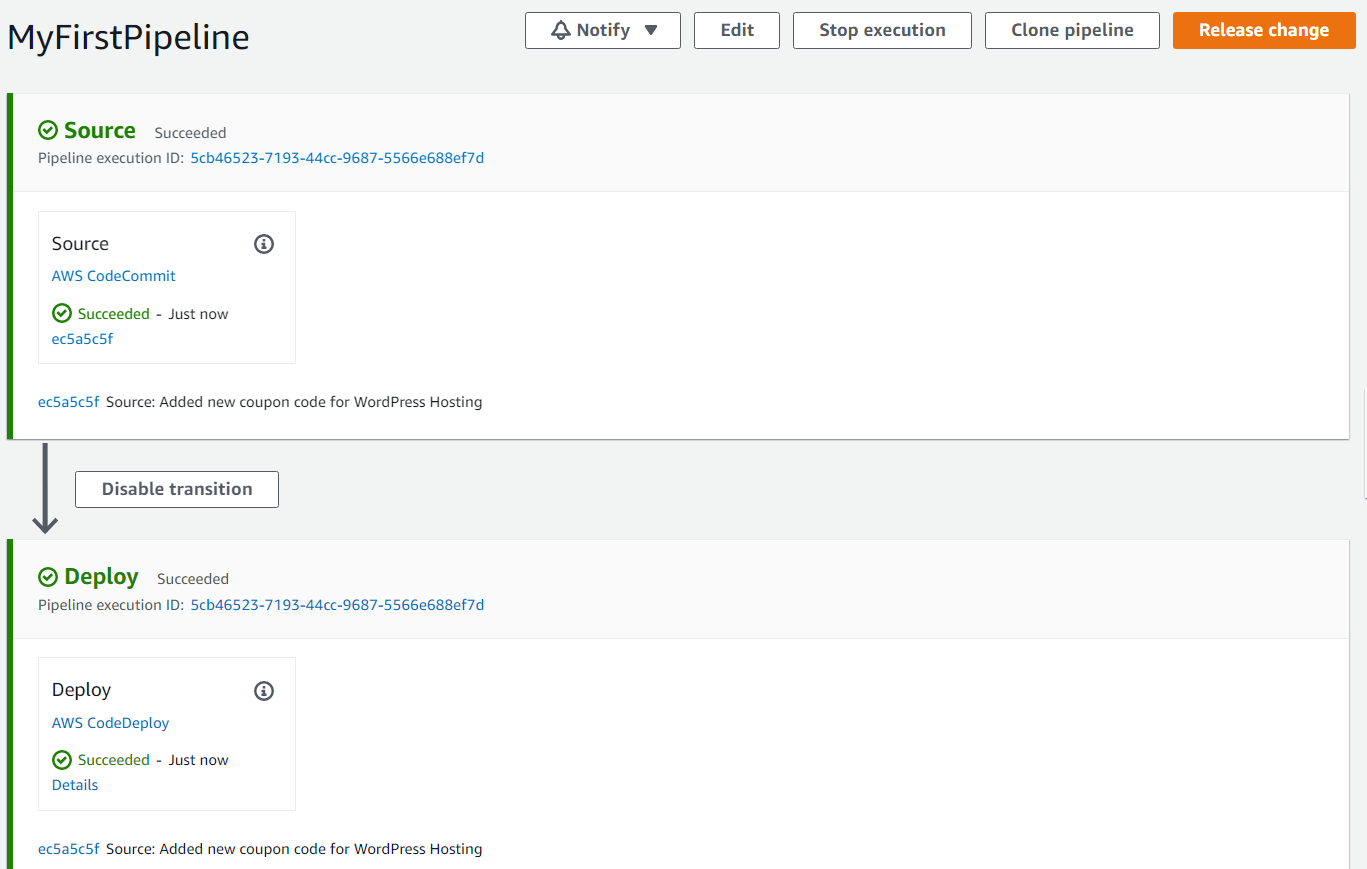
1. Copy the Public IP Address of the EC2 instance and paste it within the browser. You should see the webpage.



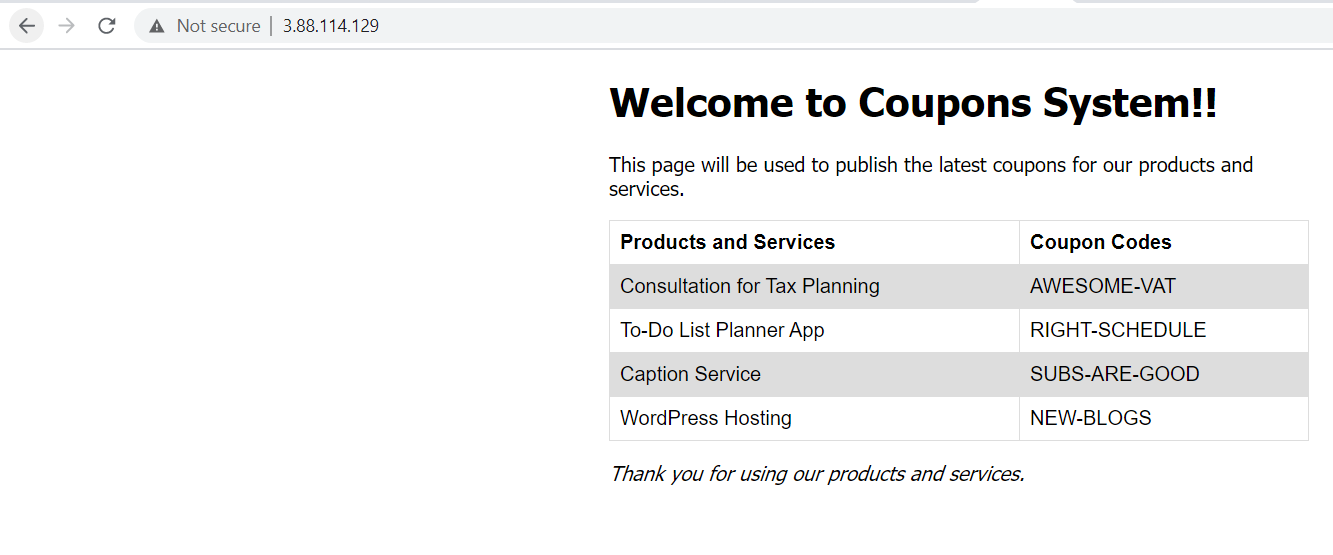
1. To verify if the pipeline is running, go back to the CodeCommit repository and modify the index.html page and edit the coupon-code associated with the Wordpress Hosting service. We will replace the content of Line 53 from GOOD-BLOGS to NEW-BLOGS. Once done, commit the file back to the repository by clicking on Commit Changes button.



1. On the modification of the index.html file, the CodePipeline should trigger the release accordingly. You can verify this by going to the AWS CodePipeline service and you should see the new release successful.



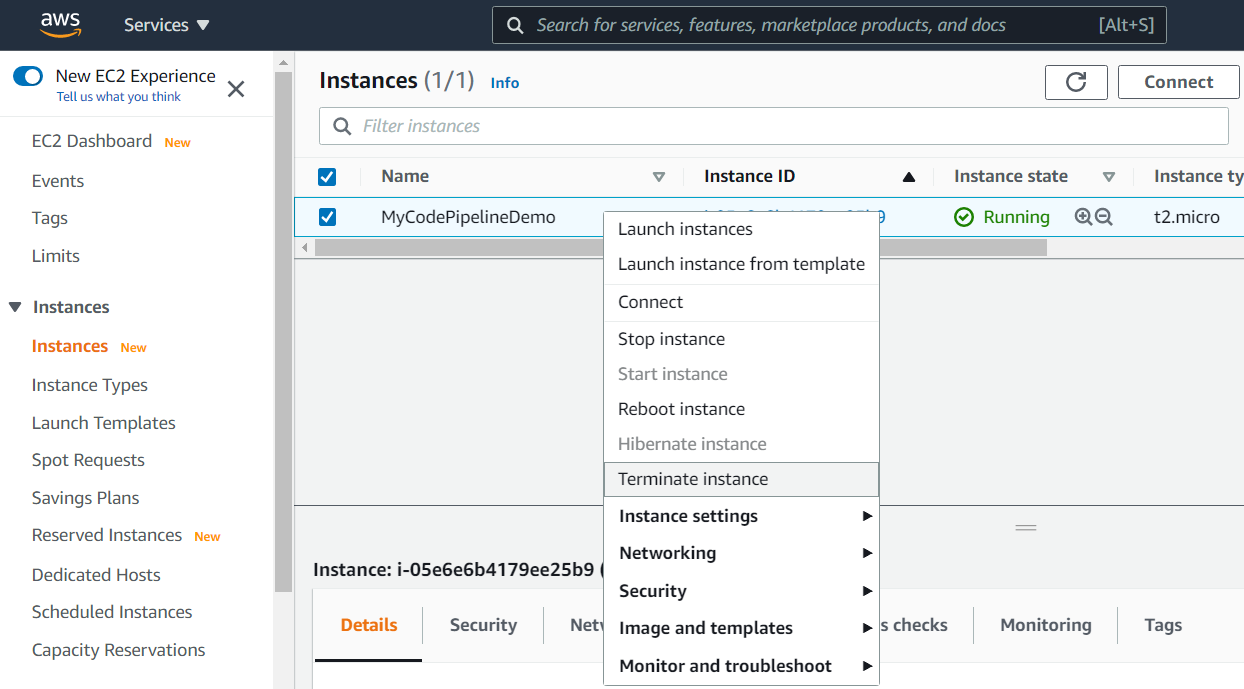
1. You can even verify by refreshing the browser page and the updated coupon code for WordPress Hosting should be reflected.



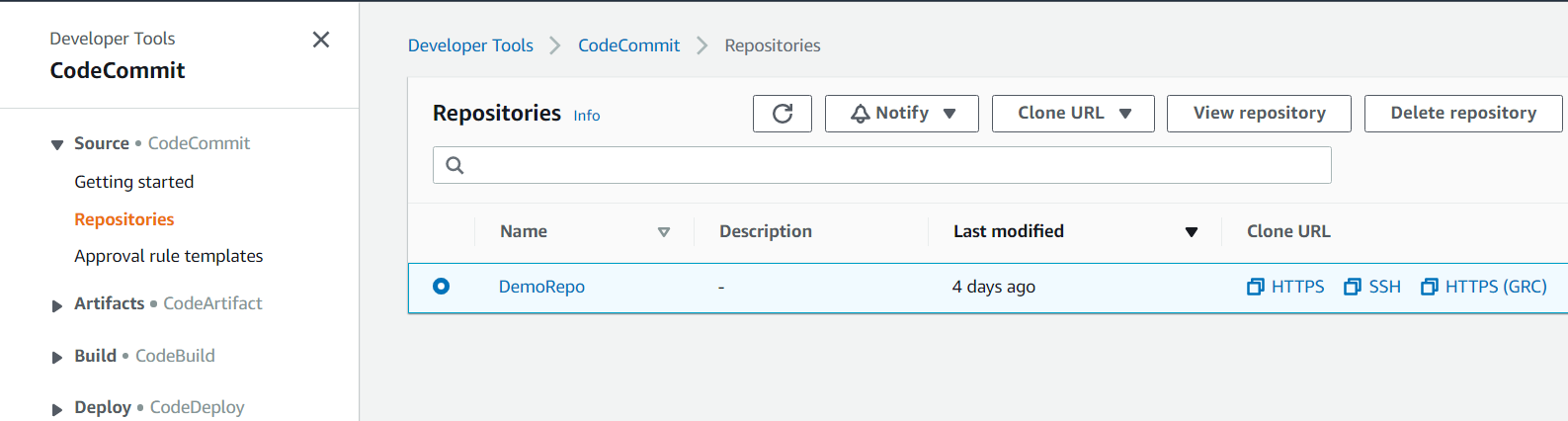
Clean up

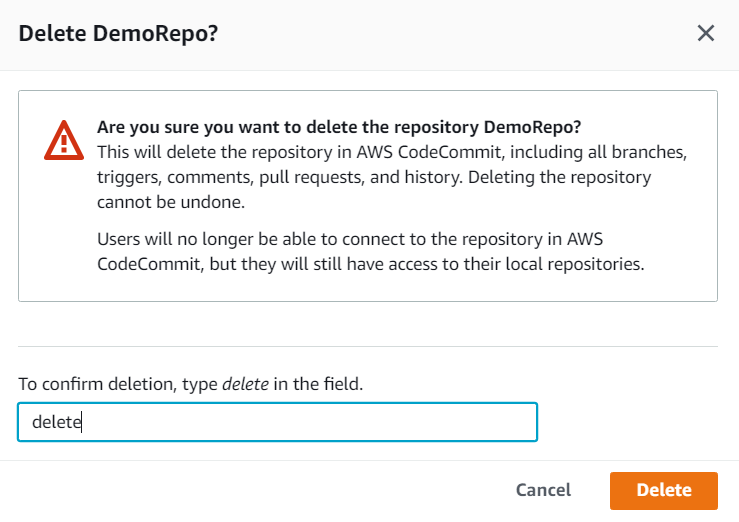
As we have successfully demonstrated a proof of concept for a continuous delivery pipeline, we'll clean up our AWS resources at this time. Delete all the resources created for this lab.

1. To terminate an EC2 instance, go to the Amazon EC2 console and locate your instance. Right click on the EC2 instance and select Terminate instances option and click on Terminate.

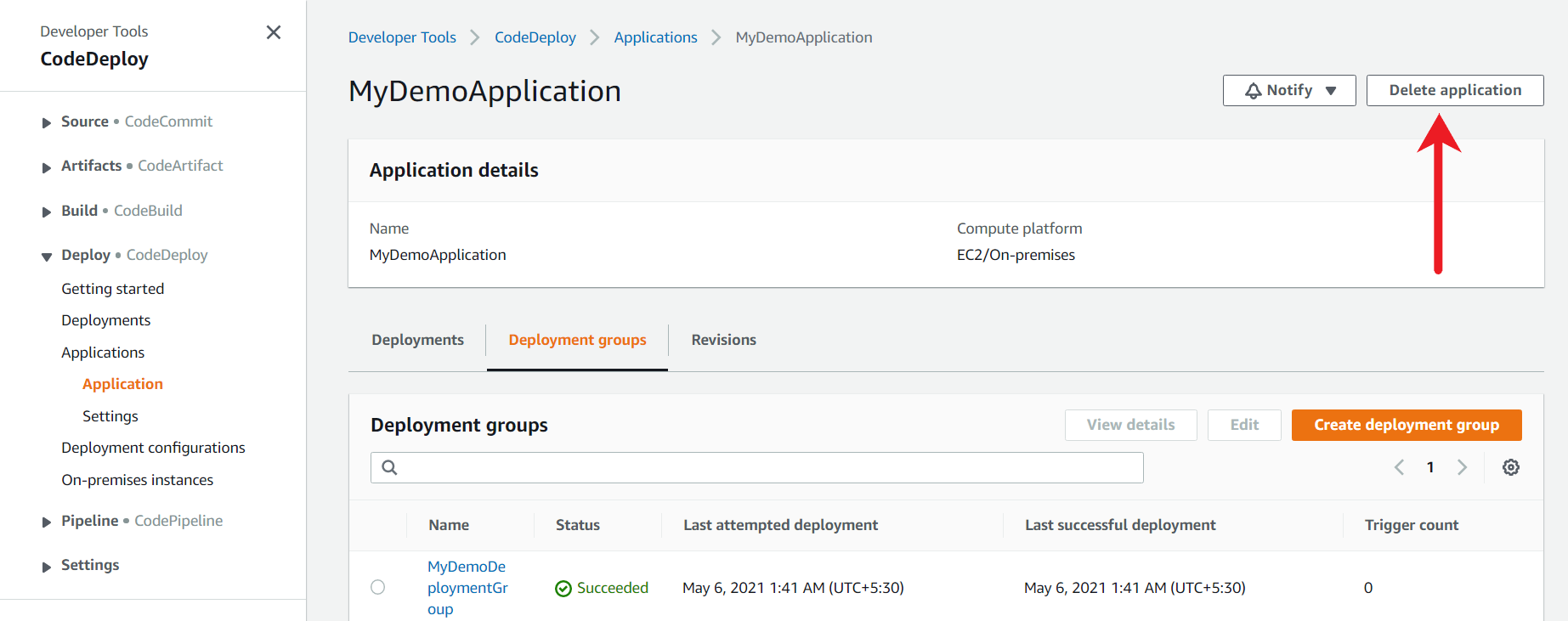


1. To delete the CodeCommit repository, go to the CodeCommit console and select your repository. Click on the Delete Repository option. In the popup window, enter delete, and then choose Delete. The repository is permanently deleted.

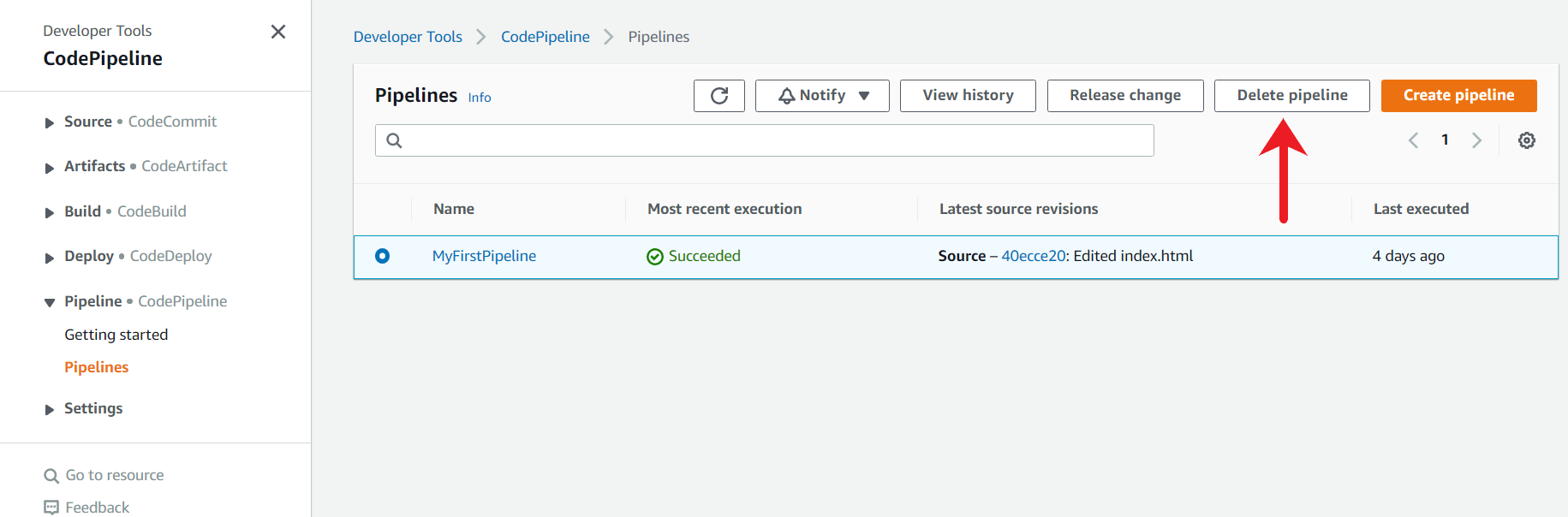




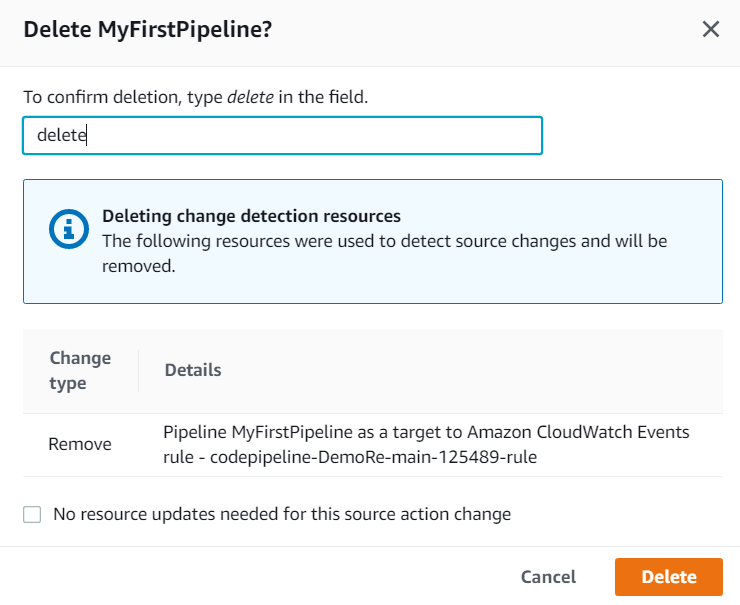
1. To Delete CodeDeploy application, open the AWS CodeDeploy console. Go to the Applications tab and select your application. A page appears containing details about the application. Click on the Delete Application button and enter delete on the confirmation window.



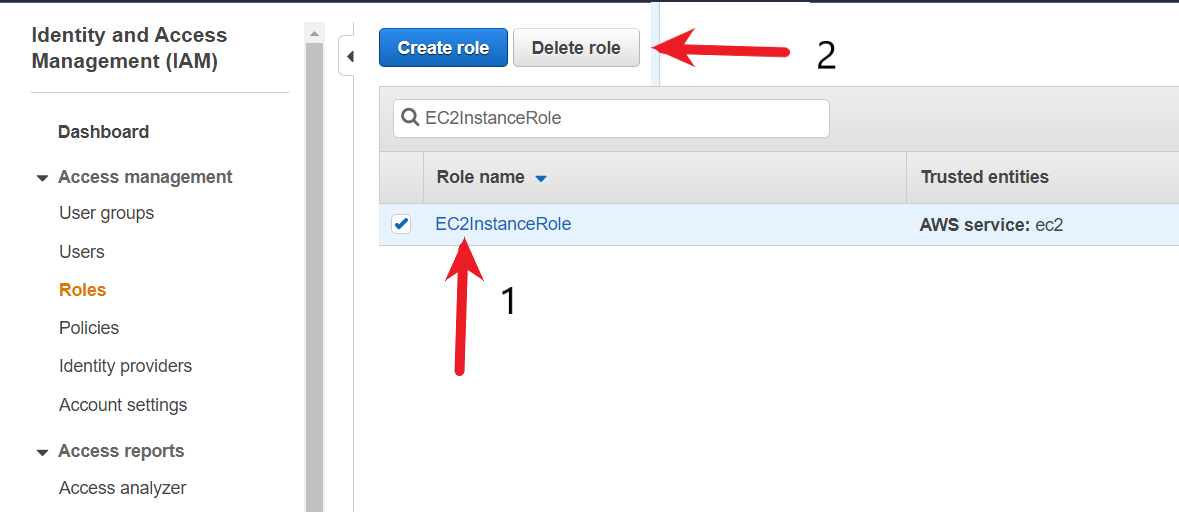
1. To delete CodePipeline, go to the AWS CodePipeline console and select the Pipeline sub-tab. Select your pipeline and click on Delete.



On the confirmation window, type delete and press and click on the Delete button.



1. To delete the IAM role, go to the IAM console. In the navigation pane, choose Roles, and then select the check box next to the role name that you want to delete. Following roles need to be deleted
   * EC2InstanceRole (Created for EC2 instance)
   * CodeDeployRole (Created for CodeDeploy application)
   * Role for CodePipeline (name starts with AWSCodePipelineServiceRole\*)



In the confirmation dialog box, review the last accessed information, which shows when each of the selected roles last accessed an AWS service. This helps you to confirm whether the role is currently active. If you want to proceed, choose Yes, Delete. If you are sure, you can proceed with the deletion even if the last accessed information is still loading.

