Hands-on tutorials

Set Up a Continuous Deployment Pipeline Using AWS CodePipeline

Set Up a Continuous Deployment Pipeline Using AWS CodePipeline: Hands-on tutorials

Table of Contents

Set Up a Continuous Deployment Pipeline Using AWS CodePipeline

AWS experience	Beginner
Minimum time to complete	30 minutes
Cost to complete	Free Tier eligible
Services used	AWS CodePipeline AWS Elastic Beanstalk
Last updated	February 14, 2023

Overview

In this tutorial, you will learn how to create an automated software release pipeline that deploys a live sample app. You will create the pipeline using AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change. You will use your GitHub account, an Amazon Simple Storage Service (Amazon S3) bucket, or an AWS CodeCommit repository as the source location for the sample app's code. You will also use AWS Elastic Beanstalk as the deployment target for the sample app. Your completed pipeline will be able to detect changes made to the source repository containing the sample app and then automatically update your live sample app.

Continuous deployment allows you to deploy revisions to a production environment automatically without explicit approval from a developer, making the entire software release process automated.

Everything done in this tutorial is Free Tier eligible.

What you will accomplish

In this tutorial, you will:

- create an automated software release pipeline that deploys a live sample app

- create the pipeline using AWS CodePipeline
- use AWS Elastic Beanstalk as the deployment target for the sample app

Prerequisites

Before starting this tutorial, you will need an AWS account. If you don't already have one, follow the Setting Up Your AWS Environment getting started guide for a quick overview.

Implementation

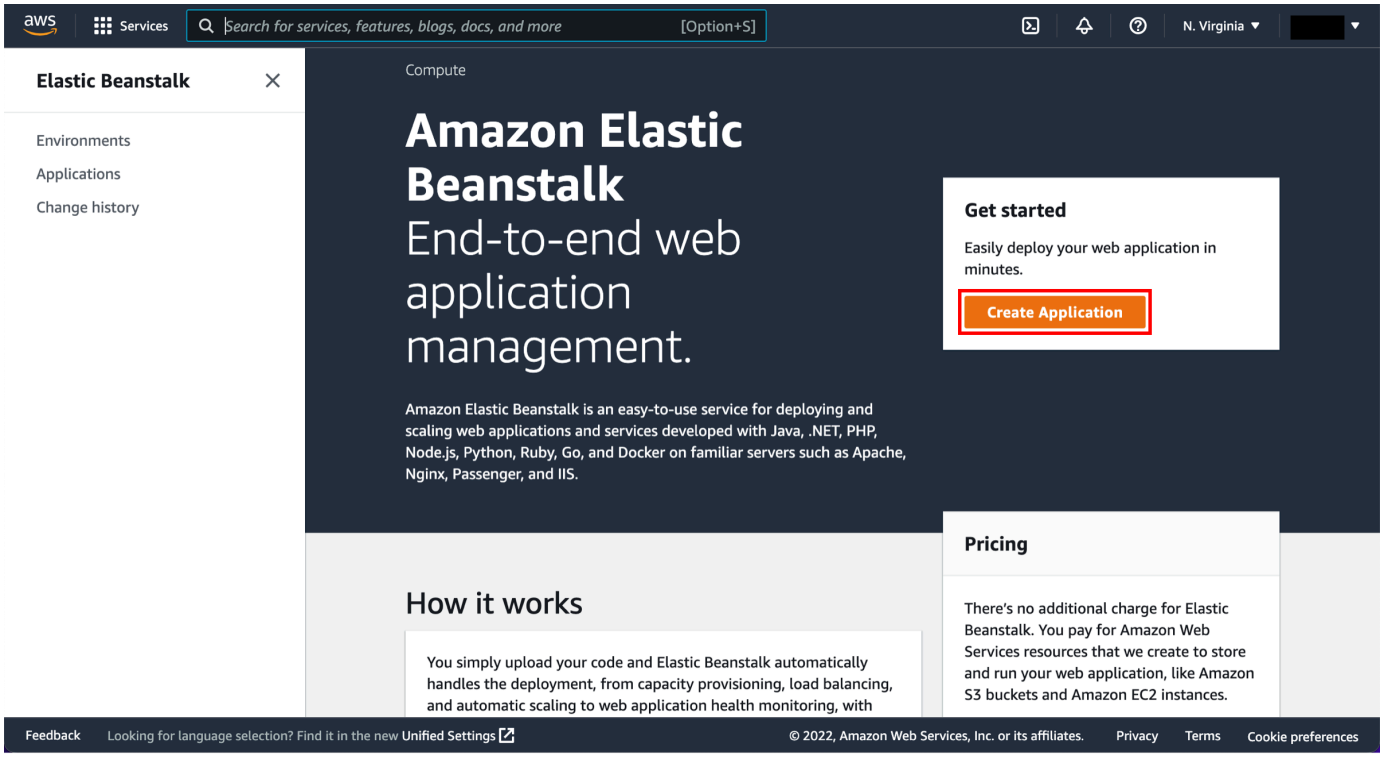
Step 1: Create a deployment environment

Your continuous deployment pipeline will need a target environment containing virtual servers, or Amazon EC2 instances, where it will deploy sample code. You will prepare this environment before creating the pipeline.

To simplify the process of setting up and configuring EC2 instances for this tutorial, you will spin up a sample environment using AWS Elastic Beanstalk. With Elastic Beanstalk you can easily host web applications without needing to launch, configure, or operate virtual servers on your own. It automatically provisions and operates the infrastructure (such as virtual servers and load balancers) and provides the application stack (such as OS, language and framework, and web and application server) for you.

1. Create an application

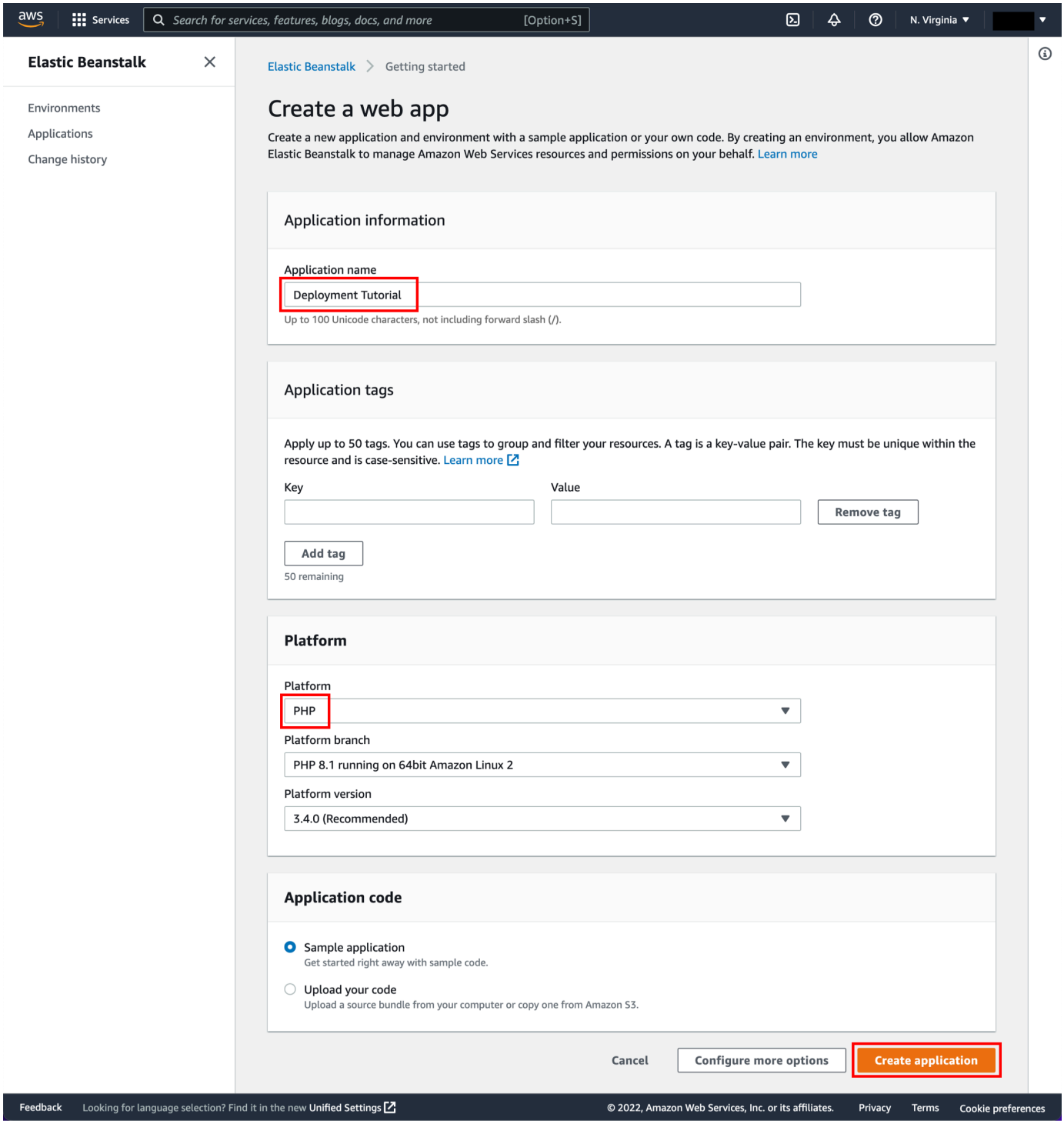
   To start, open the Elastic Beanstalk console and choose **Create Application.**



2. Configure the application

For **Application name,** enter **Deployment Tutorial.** Select **PHP** from the dropdown menu under **Platform**, and choose **Create application.**

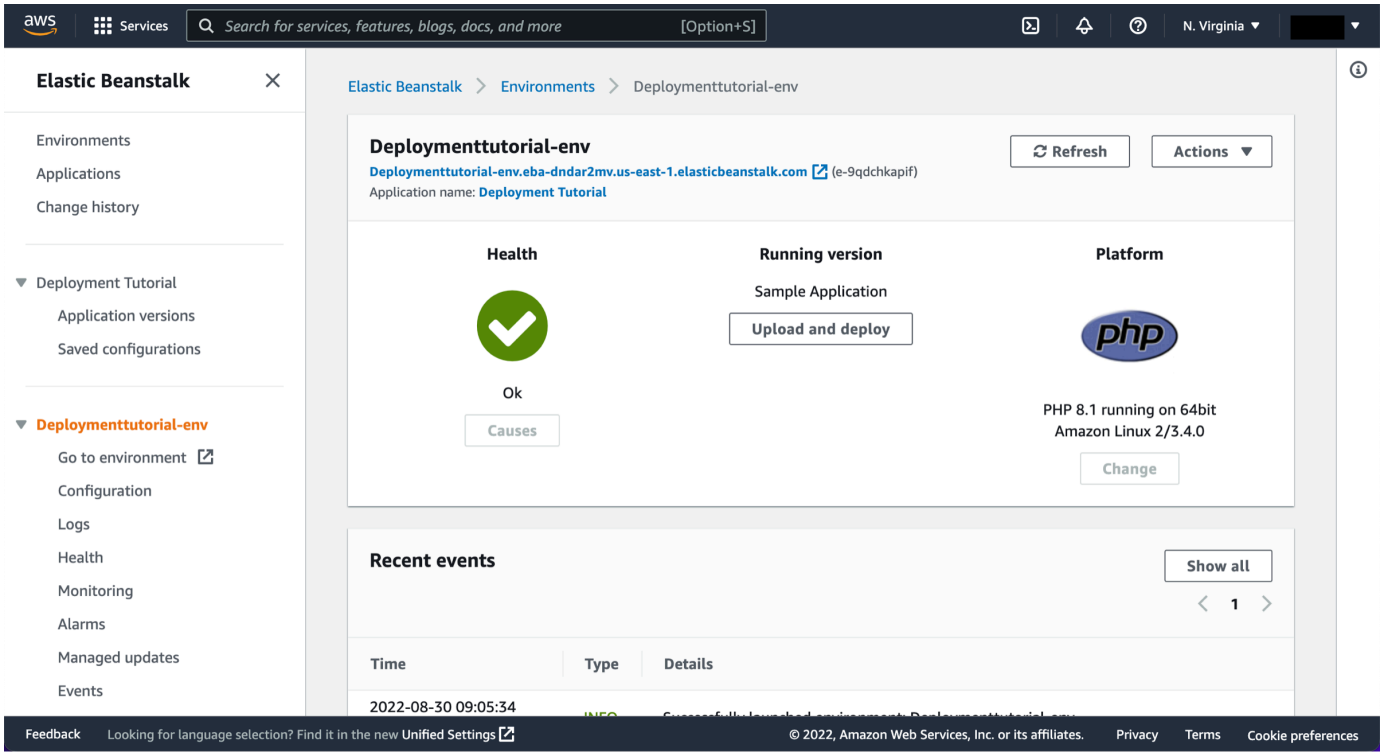
> **Note**
>
> If you have created an Elastic Beanstalk application before, choose **Create New Application** on the upper-right corner. Name your application and create a new **web server environment**. Select **PHP** as your **Platform** and **Single Instance** as your **Environment type.** If you are planning to remote login to your instances, select a key pair. Otherwise, leave default values for the remaining options and create the environment for your continuous deployment pipeline.



3. Create a sample environment

   Elastic Beanstalk will begin creating a sample environment for you to deploy your application to. It will create an Amazon EC2 instance, a security group, an Auto Scaling group, an Amazon S3 bucket, Amazon CloudWatch alarms, and a domain name for your application.

> **Note**
>
> This will take several minutes to complete.



Step 2: Get a copy of the sample code

In this step, you will retrieve a copy of the sample app's code and choose a source to host the code. The pipeline takes code from the source and then performs actions on it.

You can use one of three options as your source: a GitHub repository, an Amazon S3 bucket, or an AWS CodeCommit repository. Select your preference and follow the steps.

GitHub

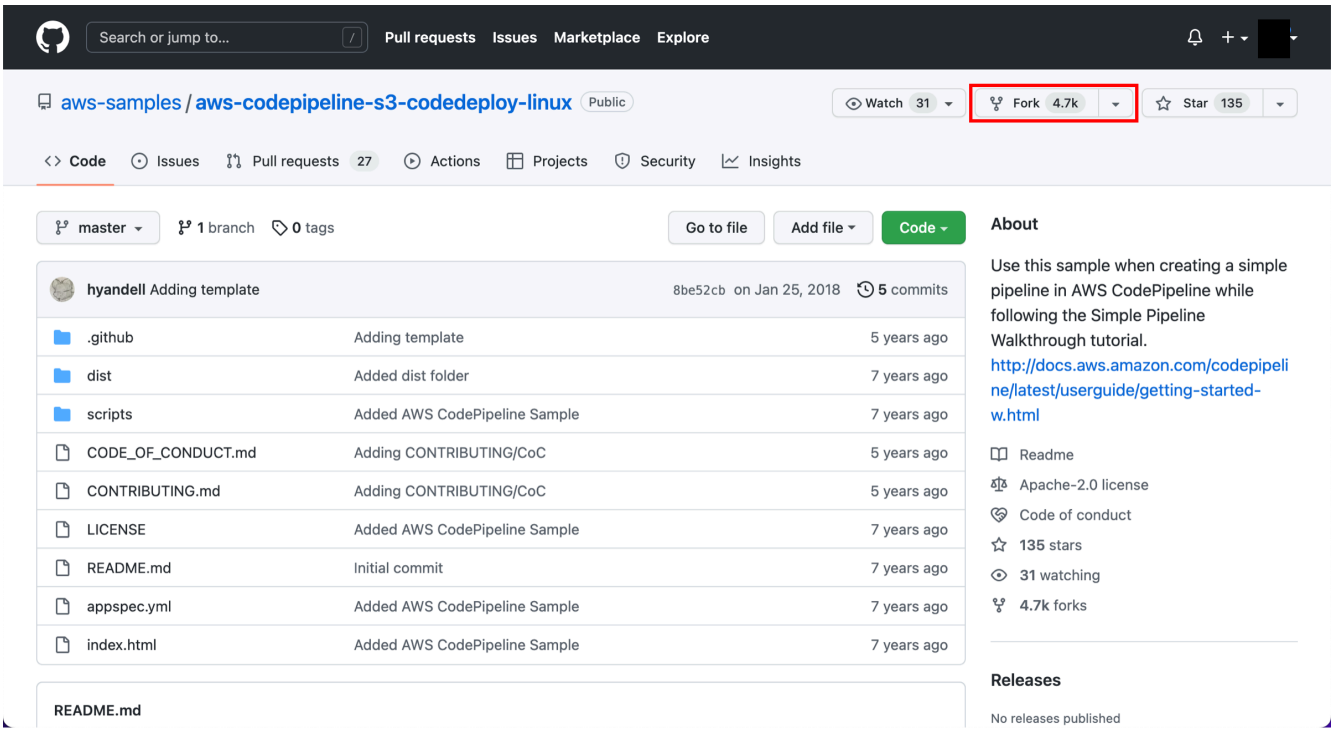
Use this procedure if you would like to use your GitHub account as your source.

- Fork the repository

  If you would like to use your GitHub account:

- Visit our GitHub repository containing the sample code at https://github.com/aws-samples/aws-codepipeline-s3-codedeploy-linux.
- Fork a copy of the repository to your own GitHub account by choosing the **Fork** button in the upper-right corner.

Then, go to **Create your pipeline**.



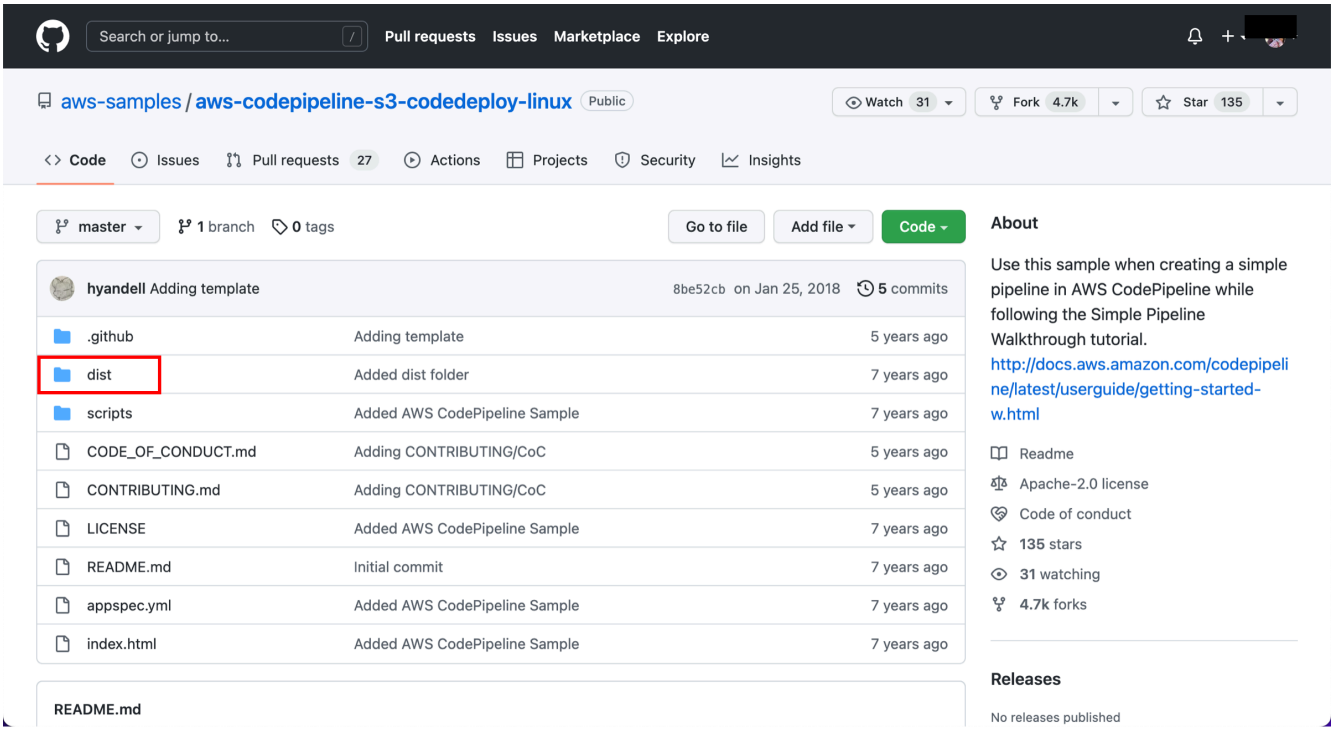
Amazon S3

Use this procedure if you would like to use Amazon S3 as your source.

1. Navigate to the sample code

   If you plan to use Amazon S3 as your source, you will retrieve the sample code from the AWS GitHub repository, save it to your computer, and upload it to an Amazon S3 bucket.

   - Visit our GitHub repository containing the sample code at https://github.com/aws-samples/aws-codepipeline-s3-codedeploy-linux
   - Select the **dist** folder.



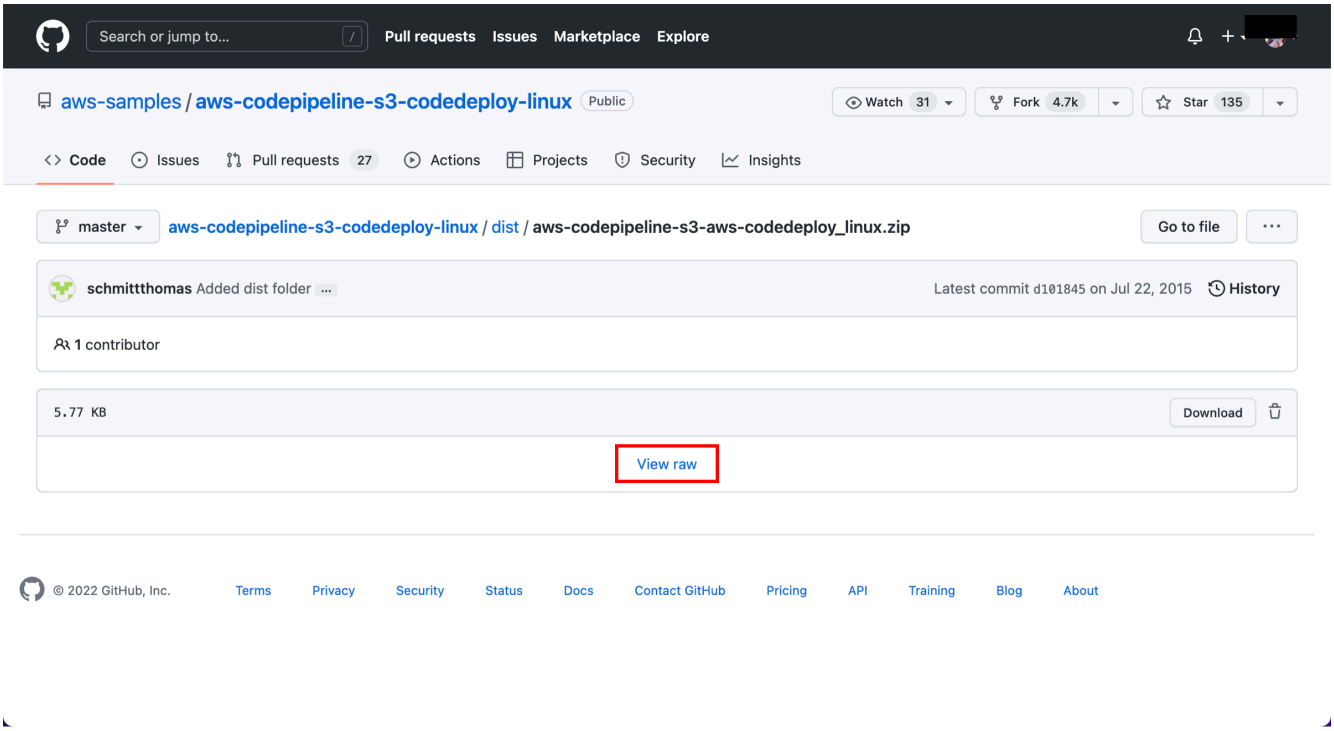
2. Download the files

   Save the source files to your computer:

   a. Select the file named **aws-codepipeline-s3-aws-codedeploy_linux.zip**.

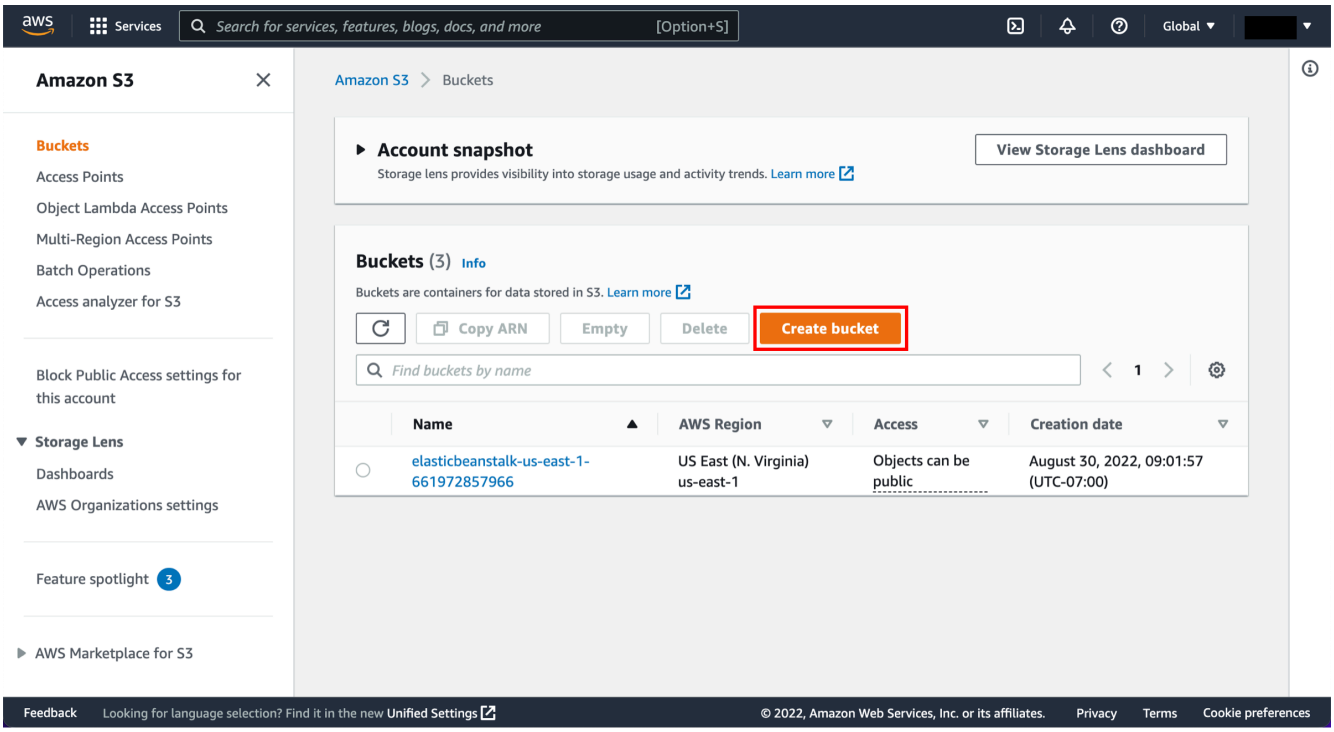
   b. Choose **View raw**.

   c. Save the sample file to your local computer.



3. Create a bucket

Open the Amazon S3 console and choose **Create bucket**.

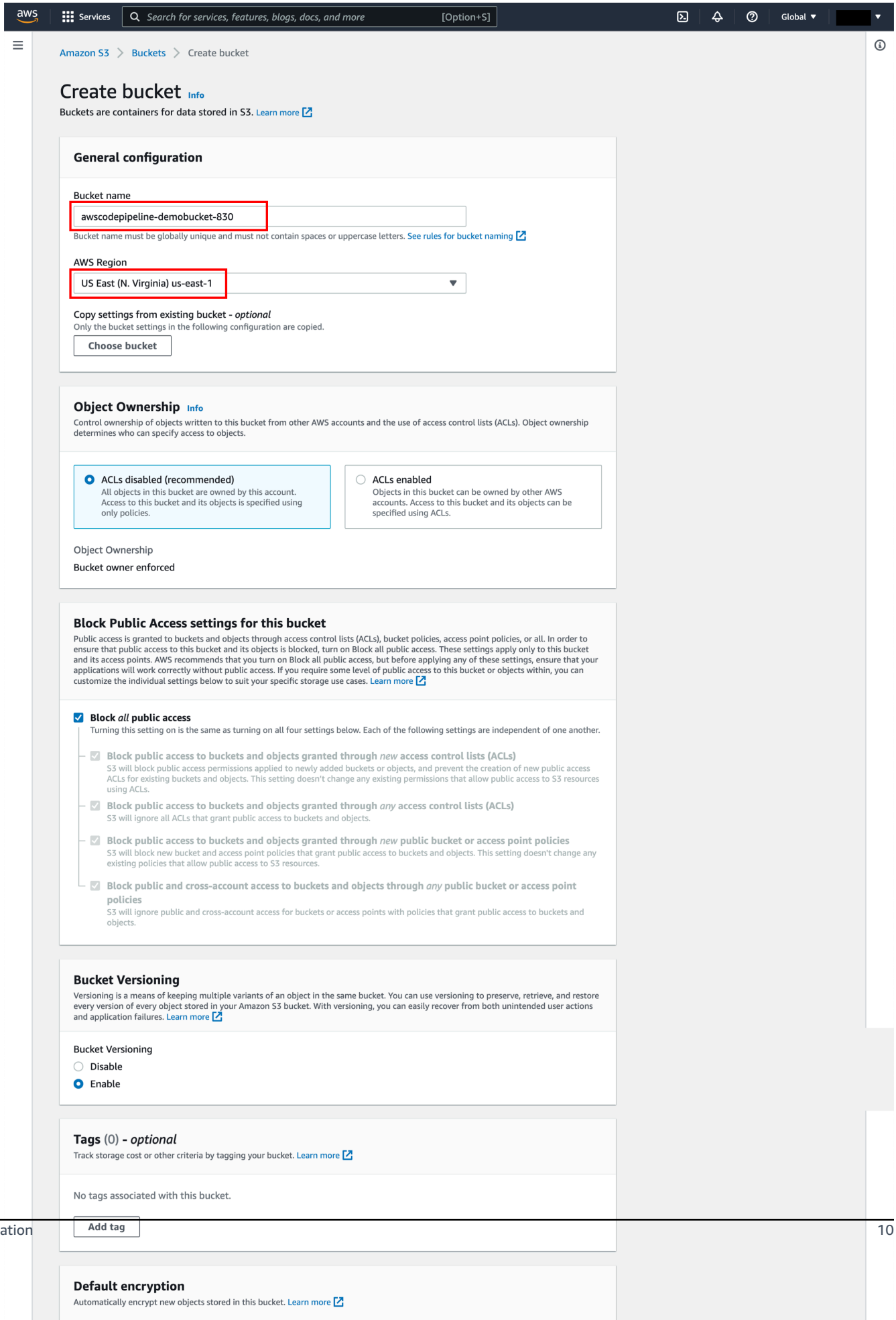


4. Configure bucket details

Bucket name: Enter a unique name for your bucket, such as **awscodepipeline-demobucket-variables**. All bucket names in Amazon S3 must be unique, so use one of your own, not one with the name shown in the example.

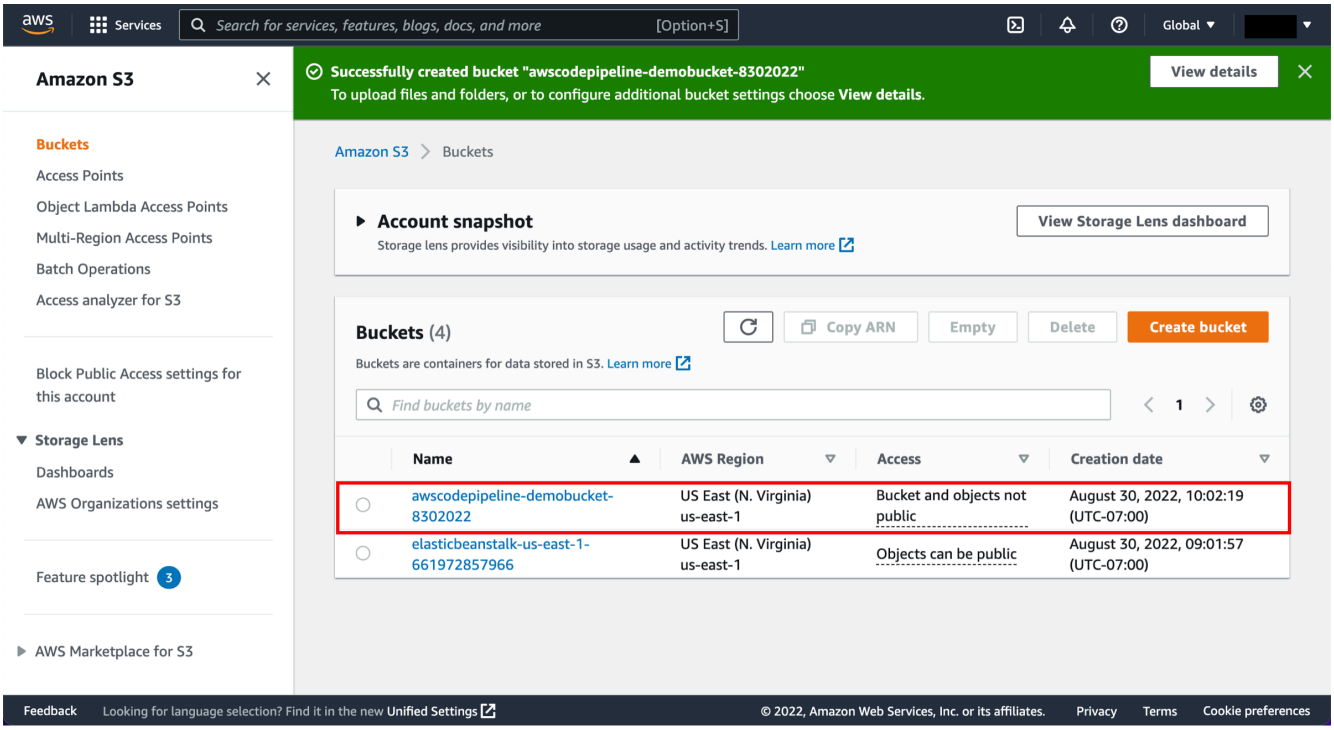
Region: In the dropdown, select the Region where you will create your pipeline, such as US East (N. Virginia).

Choose **Create bucket**.


aws
Services
Search for services, features, blogs, docs, and more [Option+S]
Global
Amazon S3 > Buckets > Create bucket
Create bucket Info
Buckets are containers for data stored in S3. Learn more
General configuration
Bucket name
awscodepipeline-demobucket-830
Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming
AWS Region
US East (N. Virginia) us-east-1
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.
ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.
Object Ownership
Bucket owner enforced
Block Public Access settings for this bucket
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more
Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
Block public access to buckets and objects granted through new access control lists (ACLs)
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
Block public access to buckets and objects granted through any access control lists (ACLs)
S3 will ignore all ACLs that grant public access to buckets and objects.
Block public access to buckets and objects granted through new public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
Block public and cross-account access to buckets and objects through any public bucket or access point policies
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.
Bucket Versioning
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more
Bucket Versioning
Disable
Enable
Tags (0) - optional
Track storage cost or other criteria by tagging your bucket. Learn more
No tags associated with this bucket.
Add tag
Default encryption
Automatically encrypt new objects stored in this bucket. Learn more

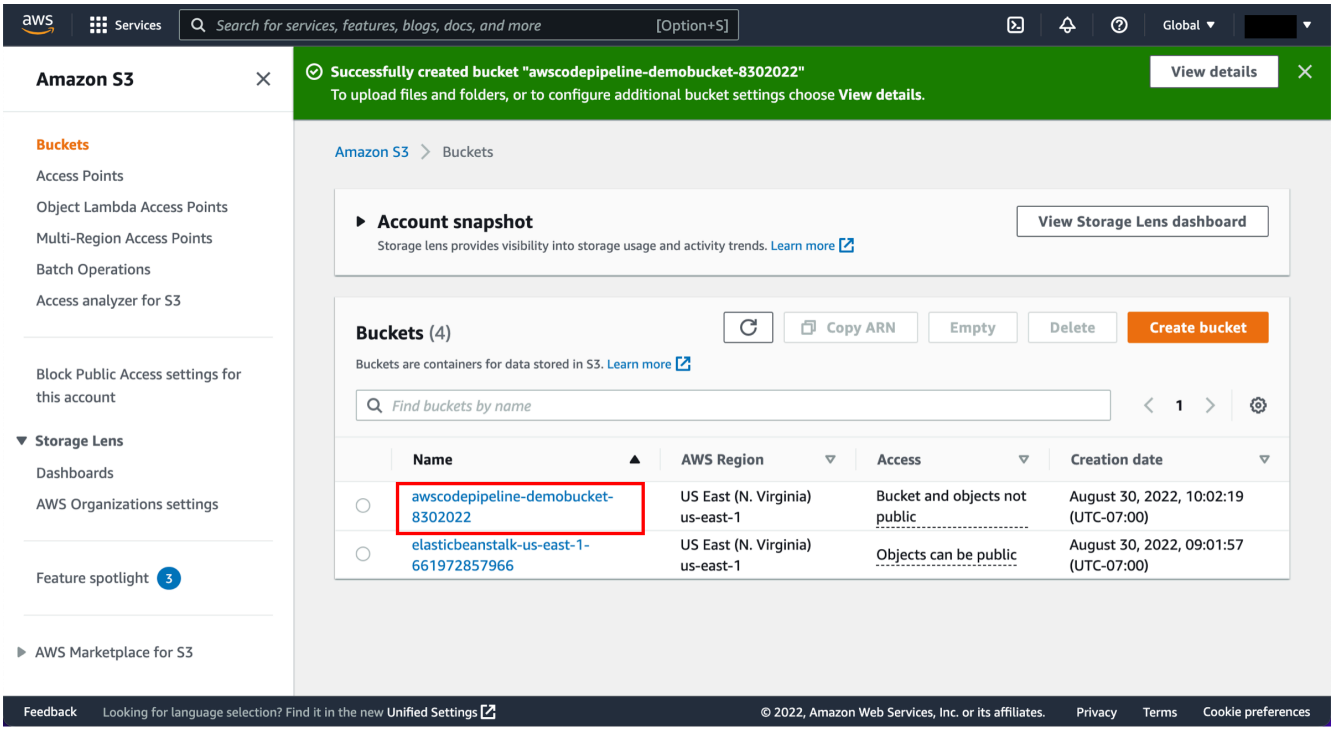
5. View the created bucket

   The console displays the newly created bucket, which is empty.



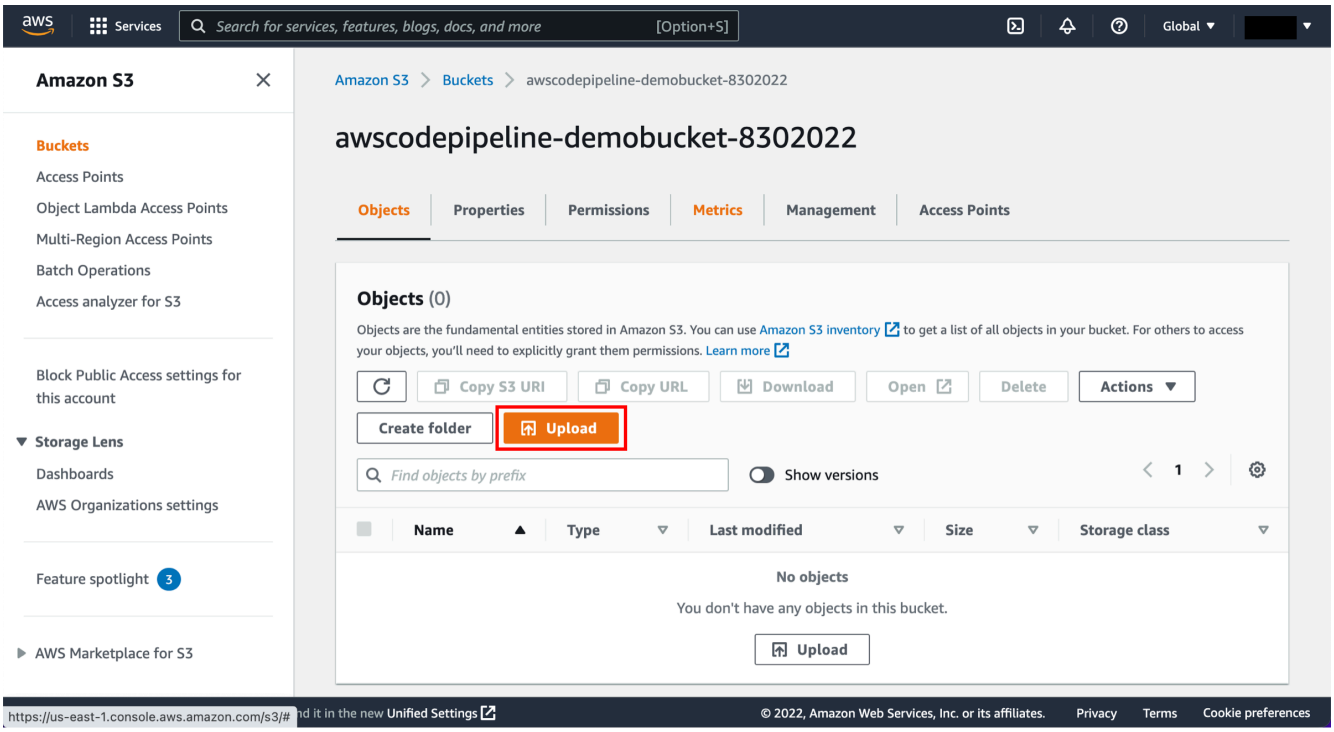
6. Open the bucket

   You will now upload the sample code to the Amazon S3 bucket. Select the Amazon S3 bucket.



7. Upload the sample code

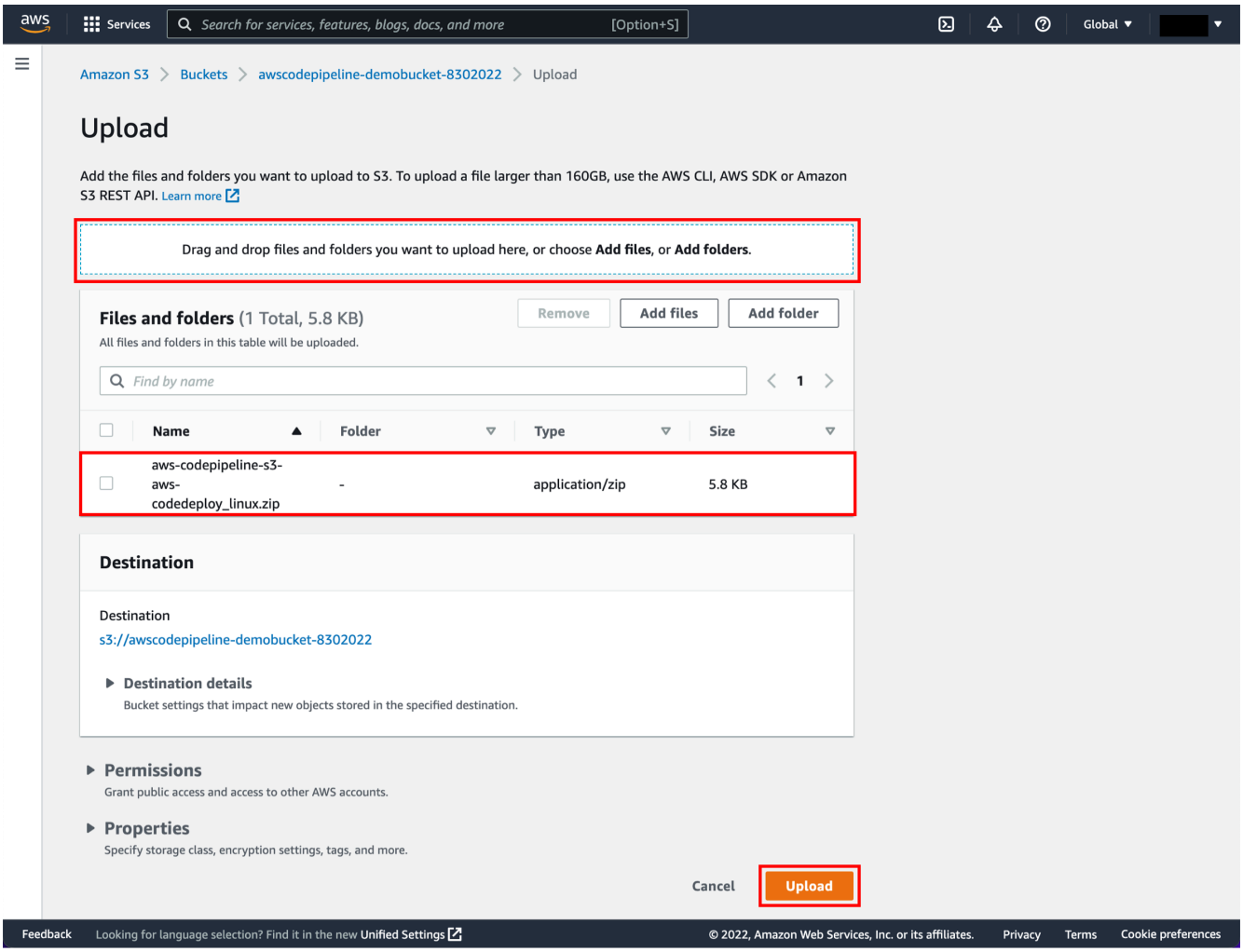
   Select **Upload**.



8. Add files

Select **Add files** to upload the zip file you downloaded earlier or drag and drop the file. Then select **Upload**.

Then, go to **Create your pipeline**.



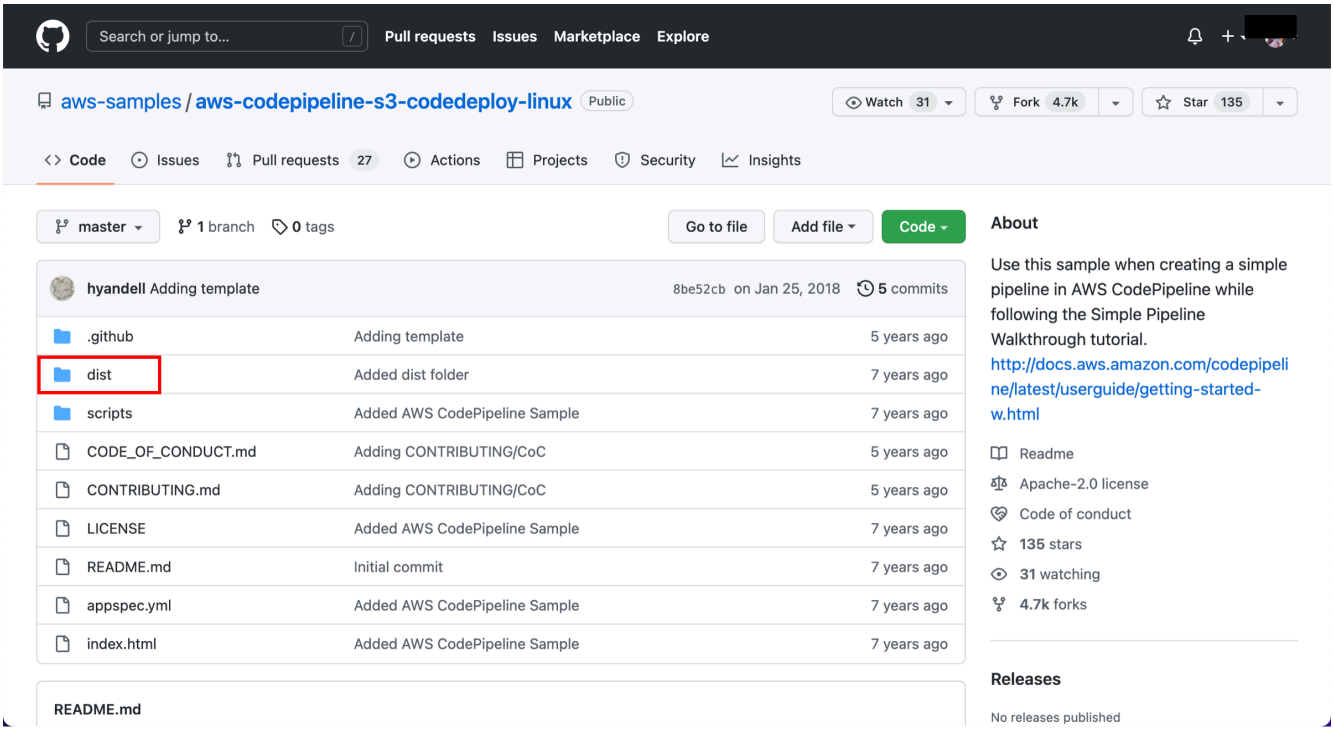
AWS CodeCommit

Use this procedure if you would like to use AWS CodeCommit as your source.

1. Navigate to the sample code

   If you plan to use AWS CodeCommit as your source, you will retrieve the sample code from the AWS GitHub repository, save it to your computer, and upload it to an AWS CodeCommit repository.

- Visit our GitHub repository containing the sample code at https://github.com/aws-samples/aws-codepipeline-s3-codedeploy-linux
- Select the **dist** folder.



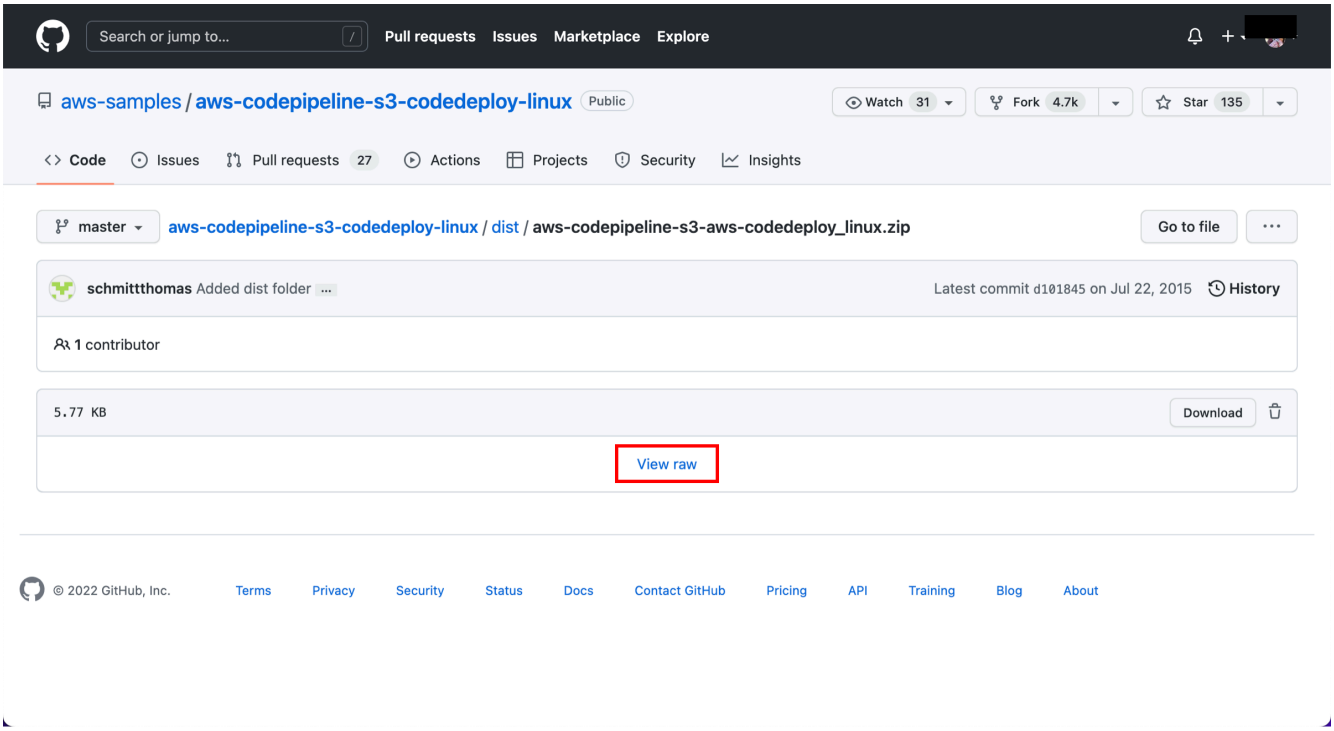
2. Download the files

   Save the source files to your computer:

   a. Select the file named **aws-codepipeline-s3-aws-codedeploy_linux.zip**.

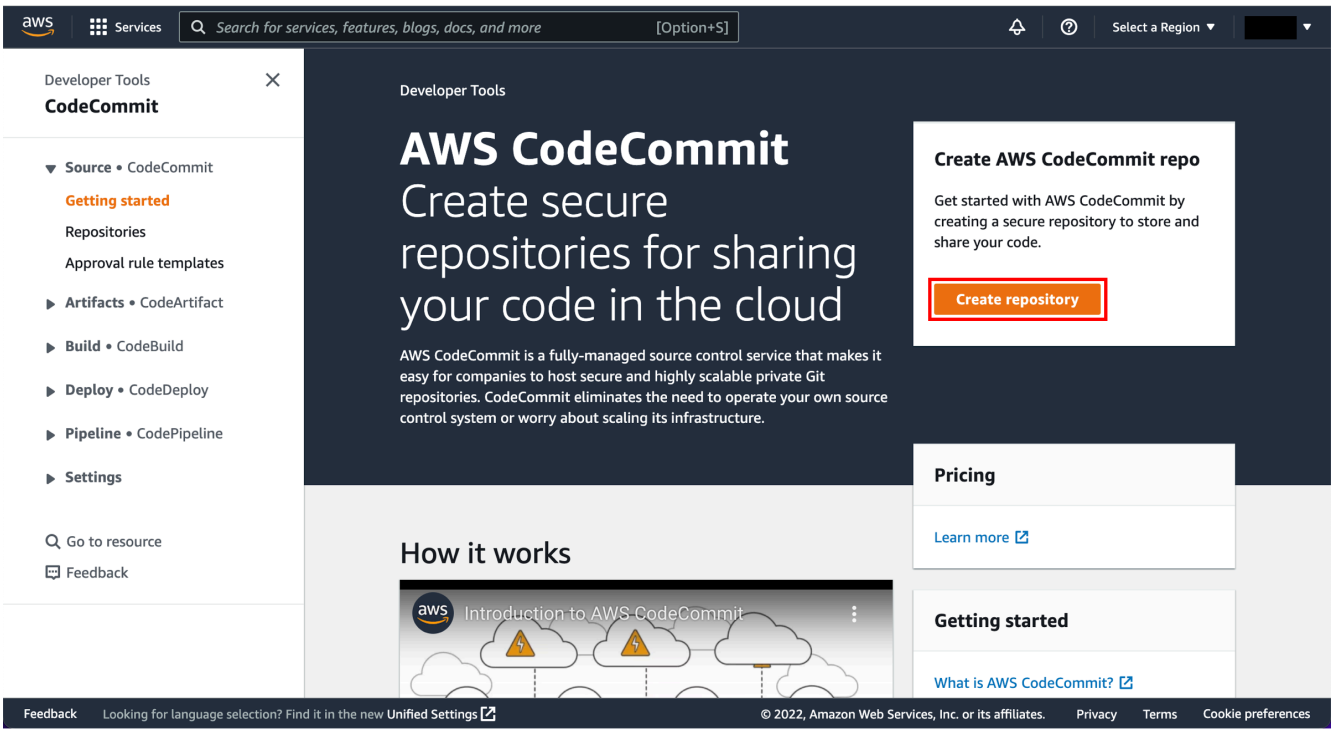
   b. Choose **View raw**.

   c. Save the sample file to your local computer.



3. Create a repository

Open the AWS CodeCommit console and choose **Create repository**.

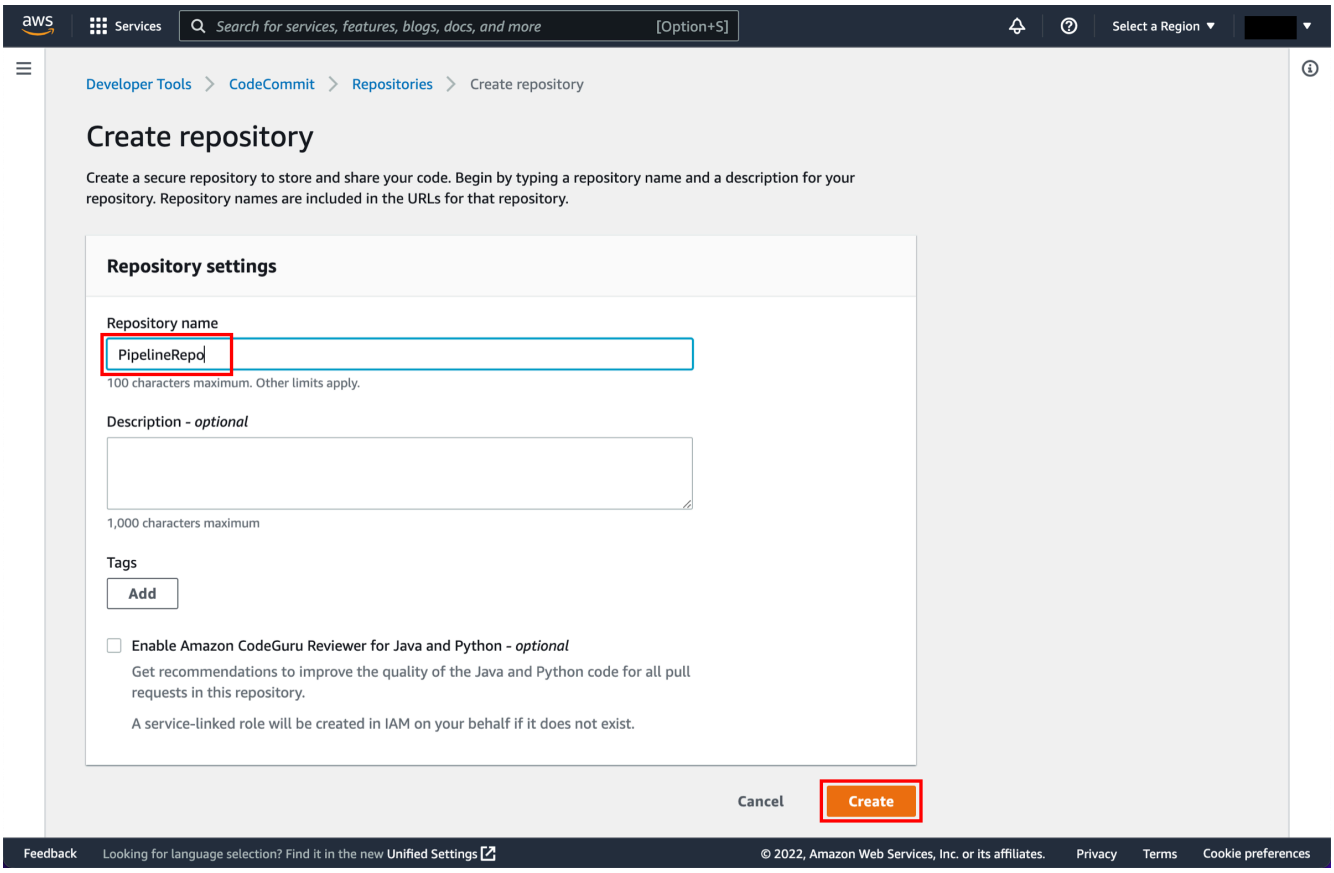


4. Configure repository settings

On the **Create repository** page:

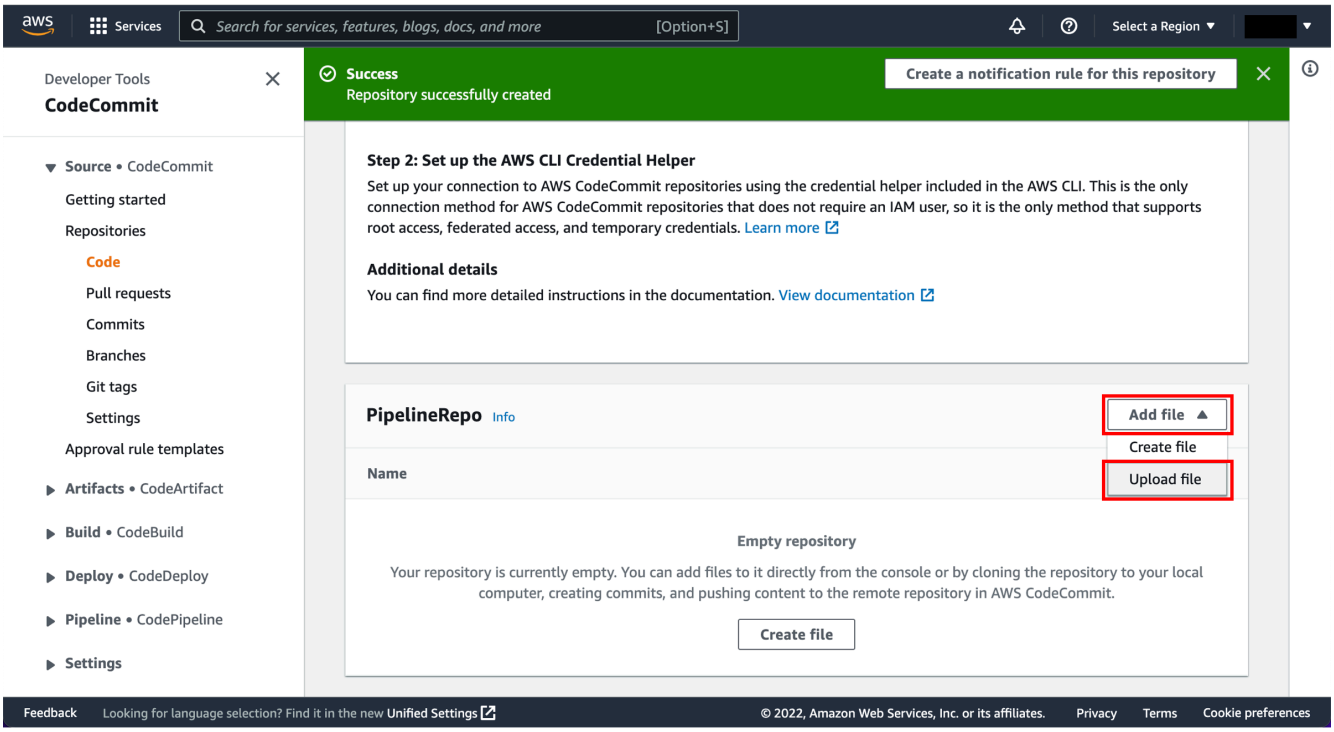
Enter **PipelineRepo** for **Repository name**.

Choose **Create**.



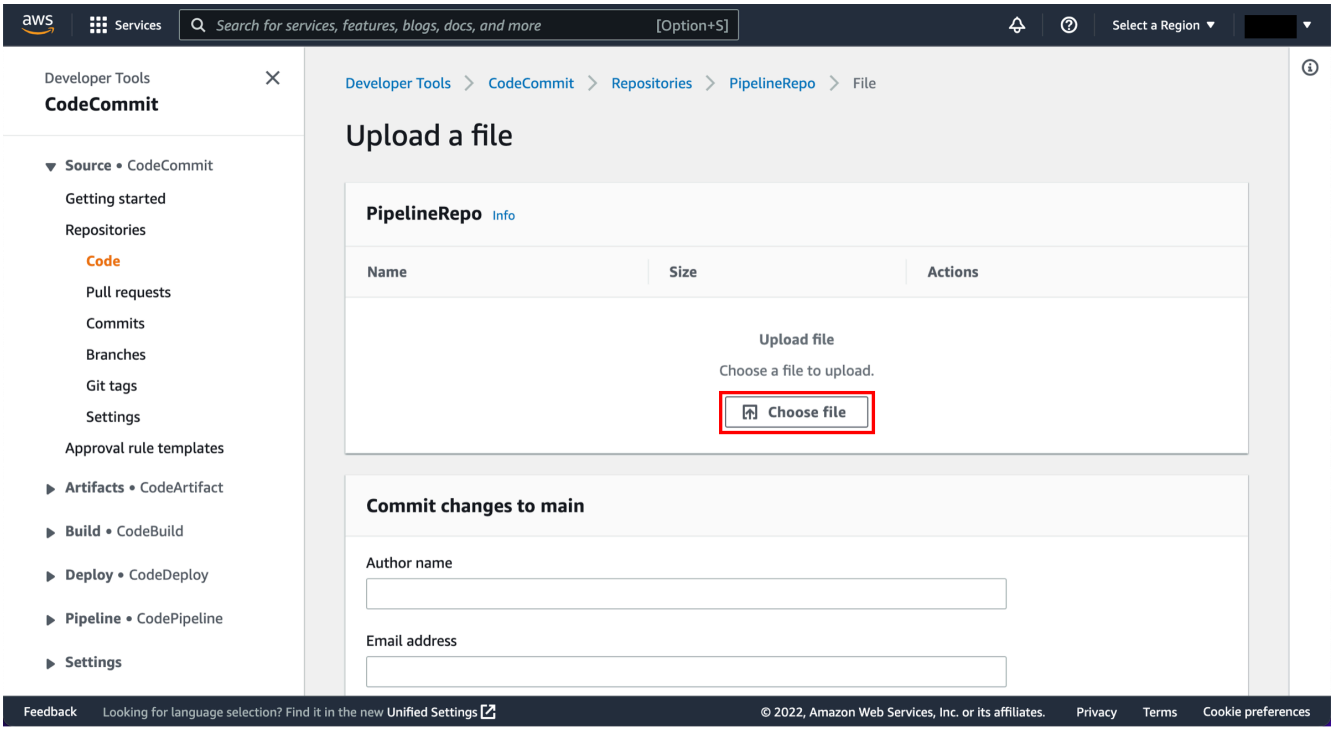
5. Upload sample code

Once the repository is successfully created, scroll down to the **PipelineRepo** section and select **Add file**, then choose **Upload file**.



6. Select the sample code file

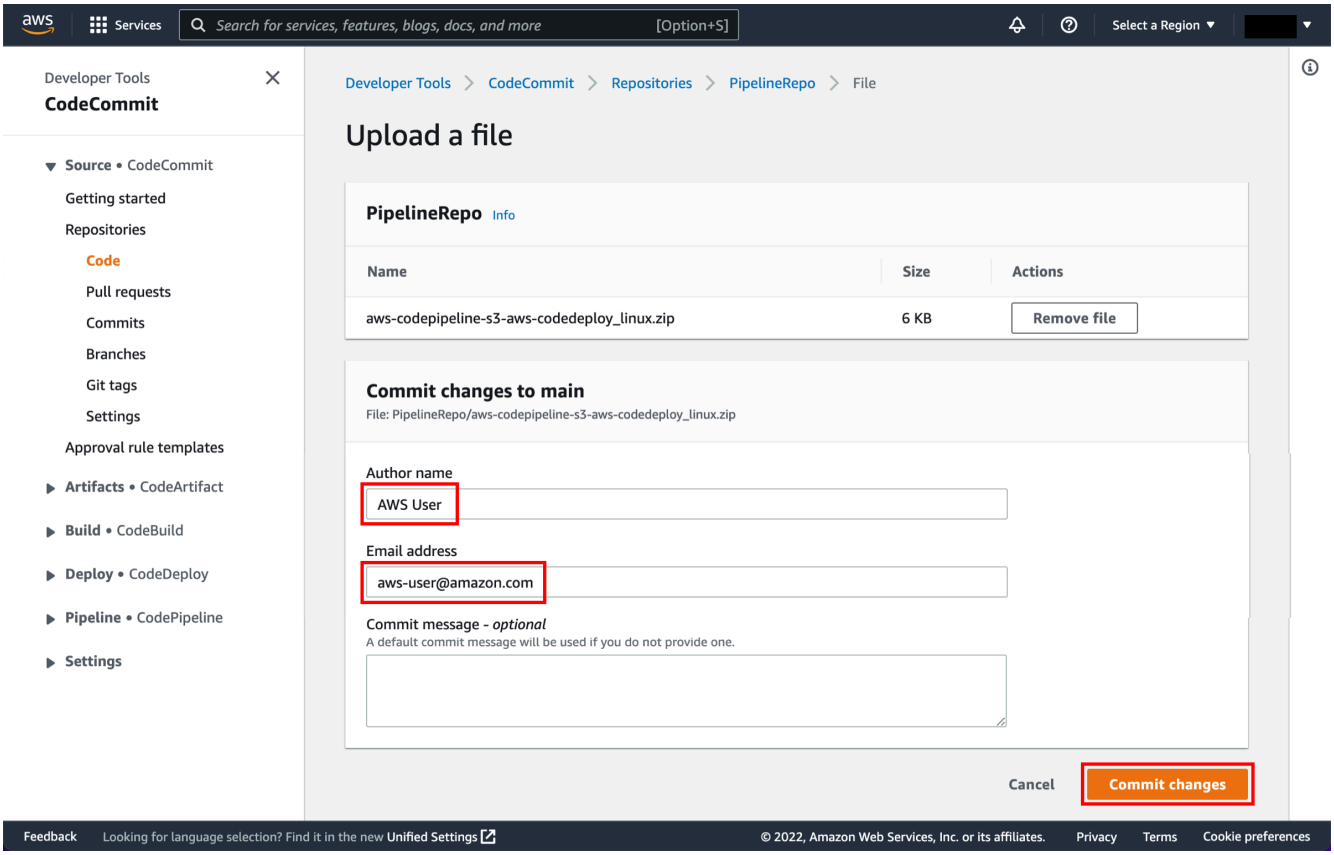
   On the **Upload a file** page, choose the **Choose file** button and select the downloaded aws-codepipeline-s3-aws-codedeploy_linux.zip file.



7. Commit changes to main

Enter an **Author name** and **Email address**, then choose **Commit changes**.

Then, go to **Create your pipeline**.



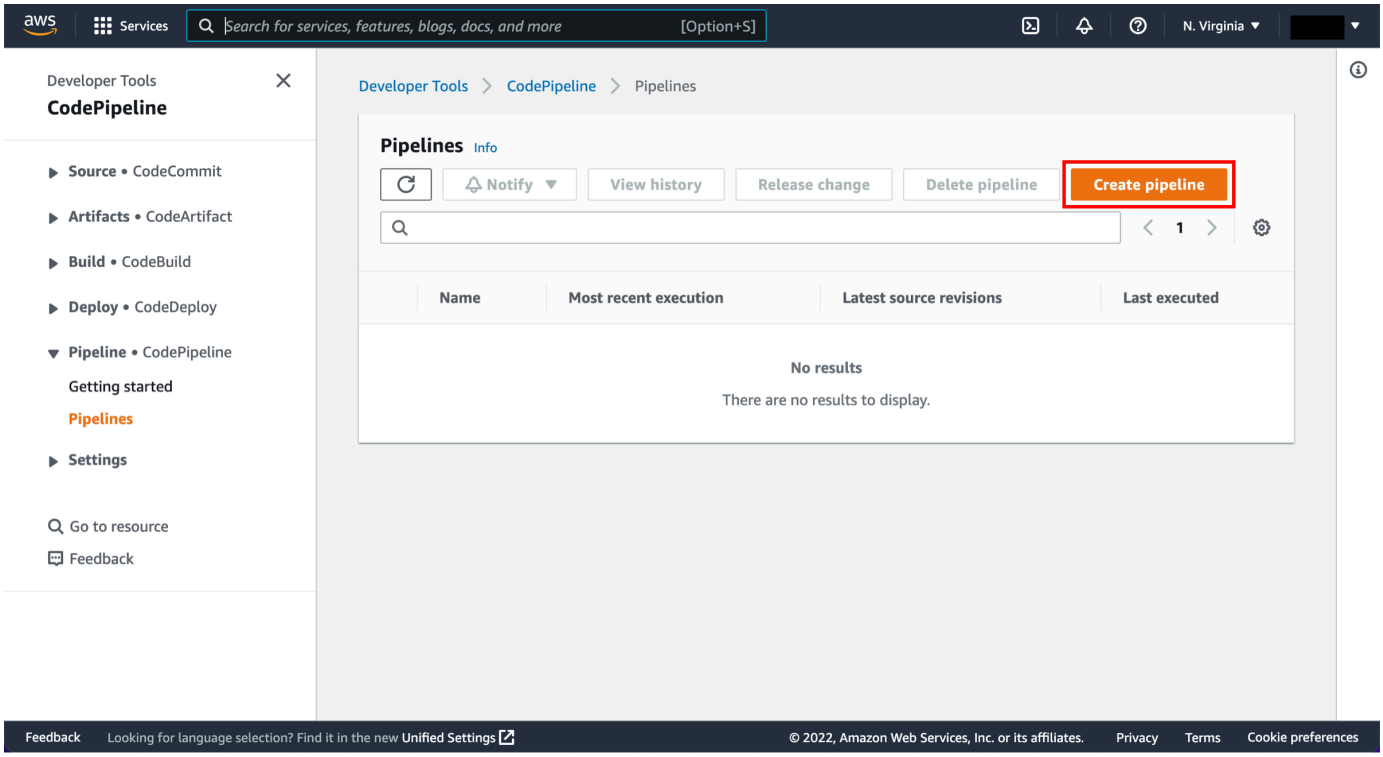
Step 3: Create your pipeline

In this step, you will create and configure a simple pipeline with two actions: source and deploy. You will provide CodePipeline with the locations of your source repository and deployment environment.

1. Create a pipeline

   Open the AWS CodePipeline console.

   On the **Welcome** page, choose **Create pipeline.**

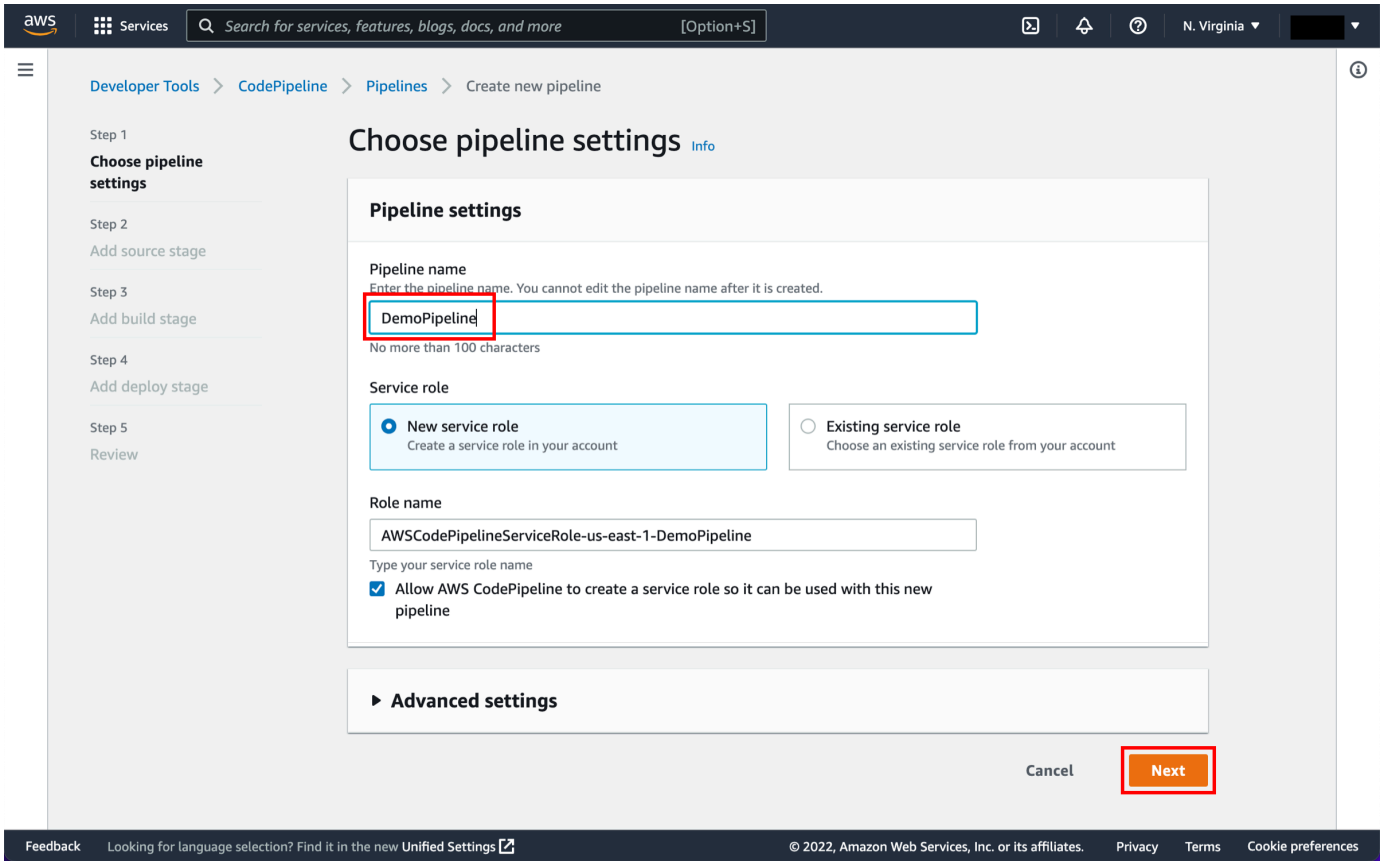


2. Configure pipeline settings

   On the **Step 1: Choose pipeline settings** page:

   - **Pipeline name:** Enter the name for your pipeline, **DemoPipeline**.
   - Choose **Next.**

> **Note**
>
> After you create a pipeline, you cannot change its name.



3. Choose your source

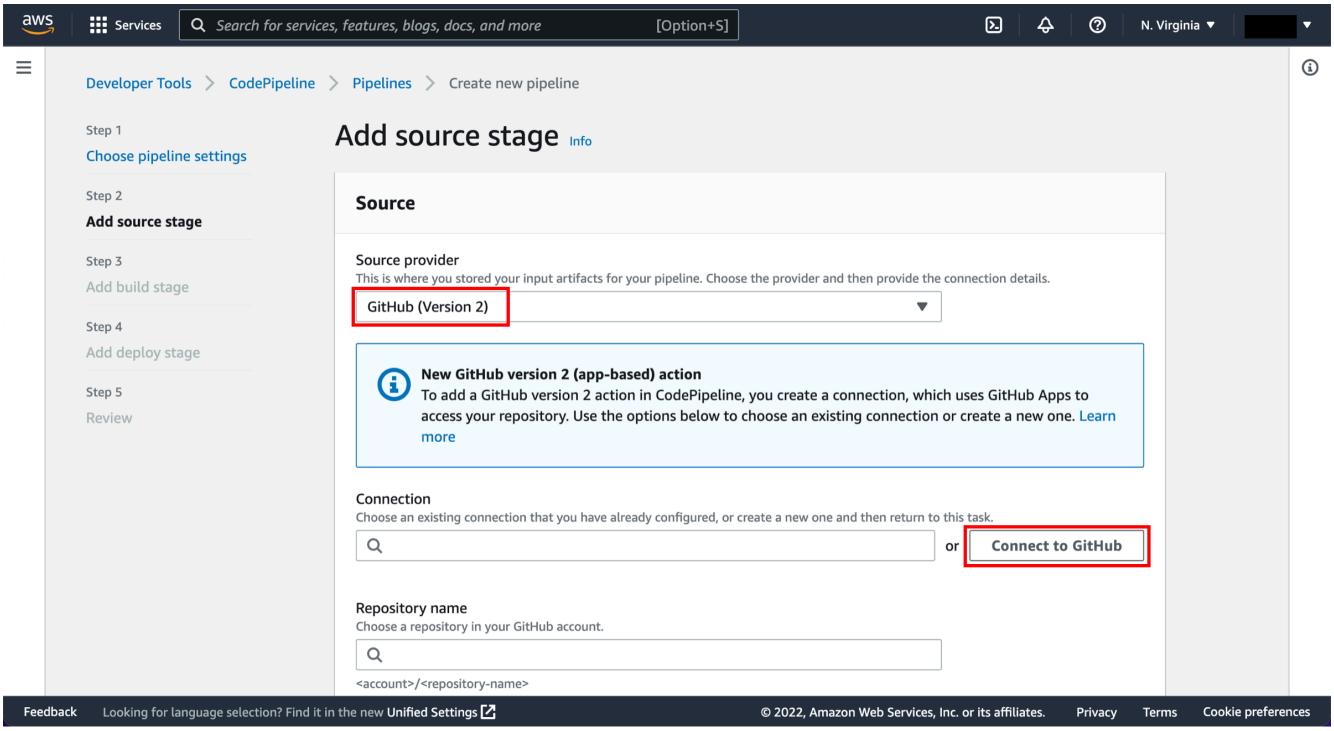
   On the **Step 2: Add source stage** page, select the location of the source you selected using the following instructions.

GitHub

1. Add source

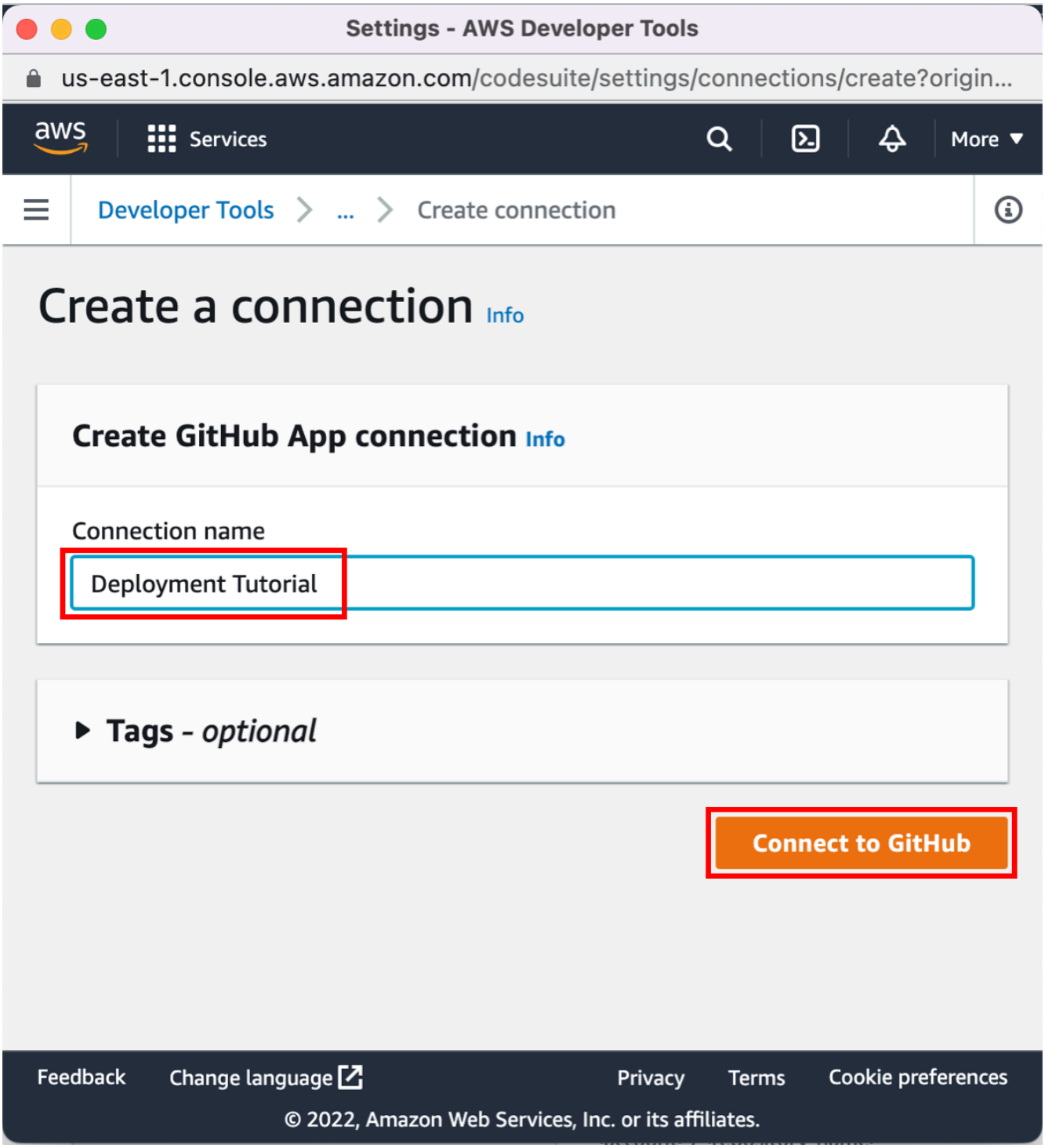
   Select **GitHub (Version 2)** for the **Source provider**.

   Choose **Connect to GitHub**.



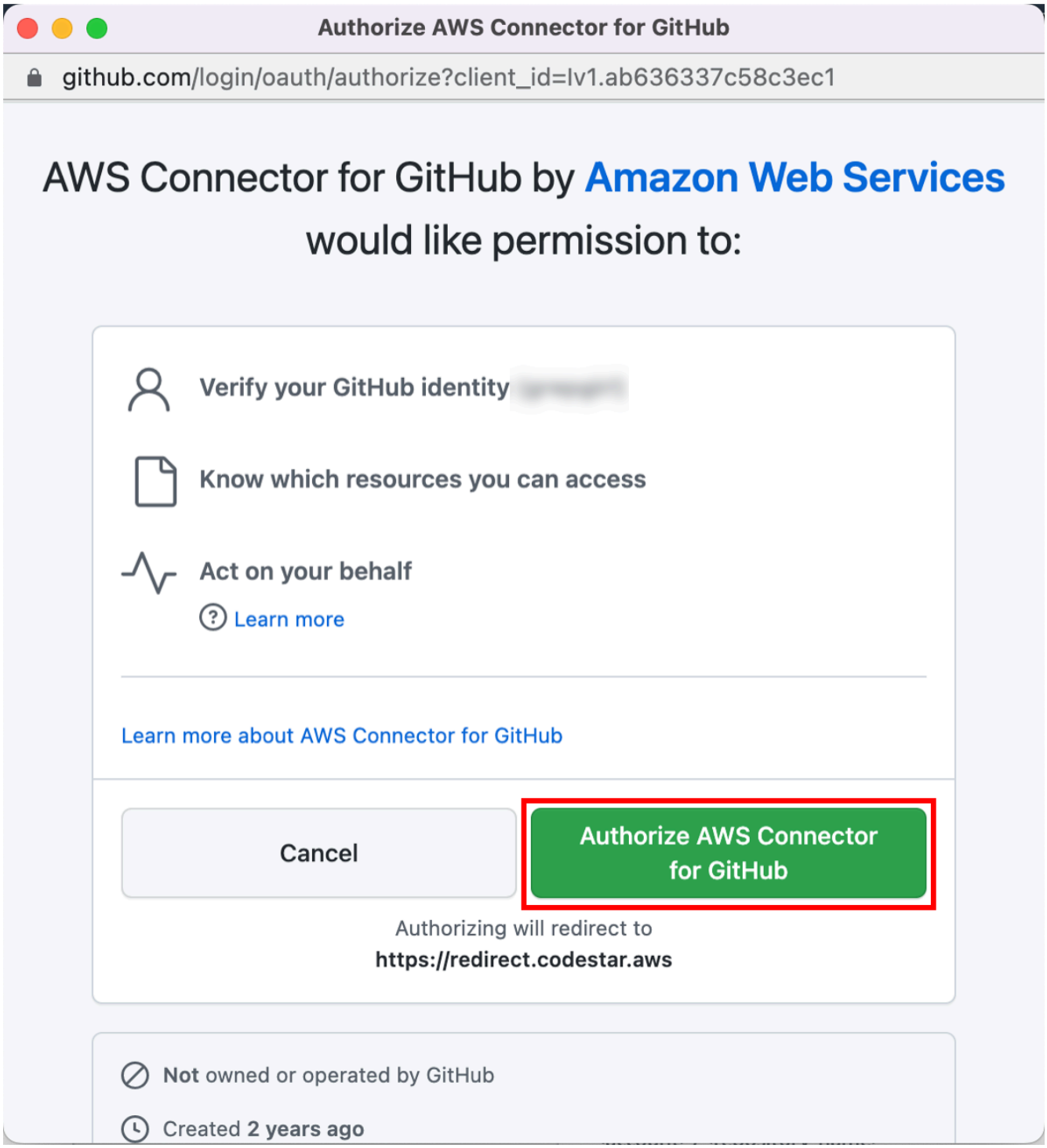
2. Enter a connection name

   Enter **Deployment Tutorial** for **Connection name** and choose **Connect to GitHub**.



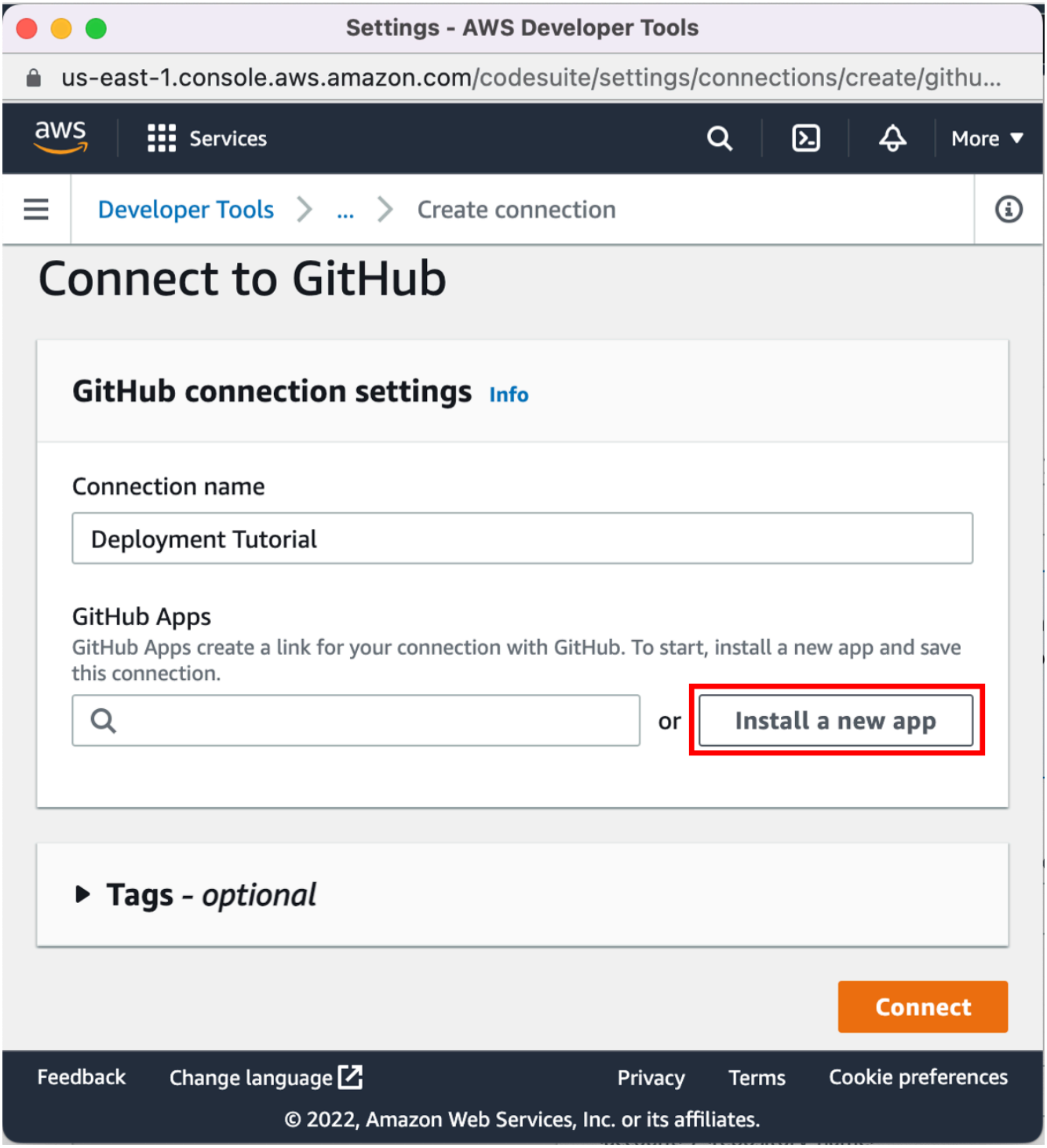
3. Grant permissions

   Select **Authorize AWS Connector for GitHub**.



4. Install a new app

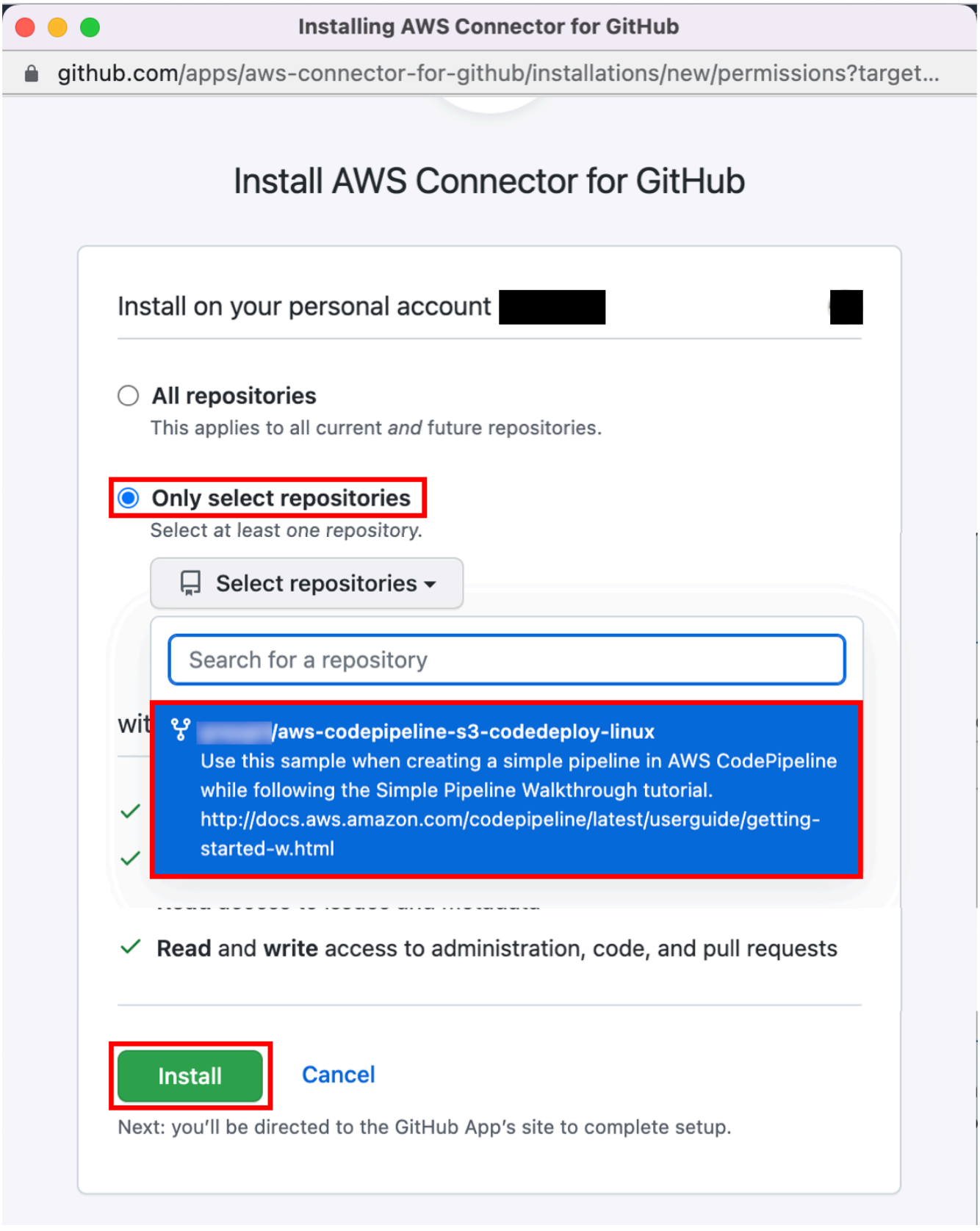
   When redirected back to the connection screen, choose **Install a new app**.



5. Install the forked repository

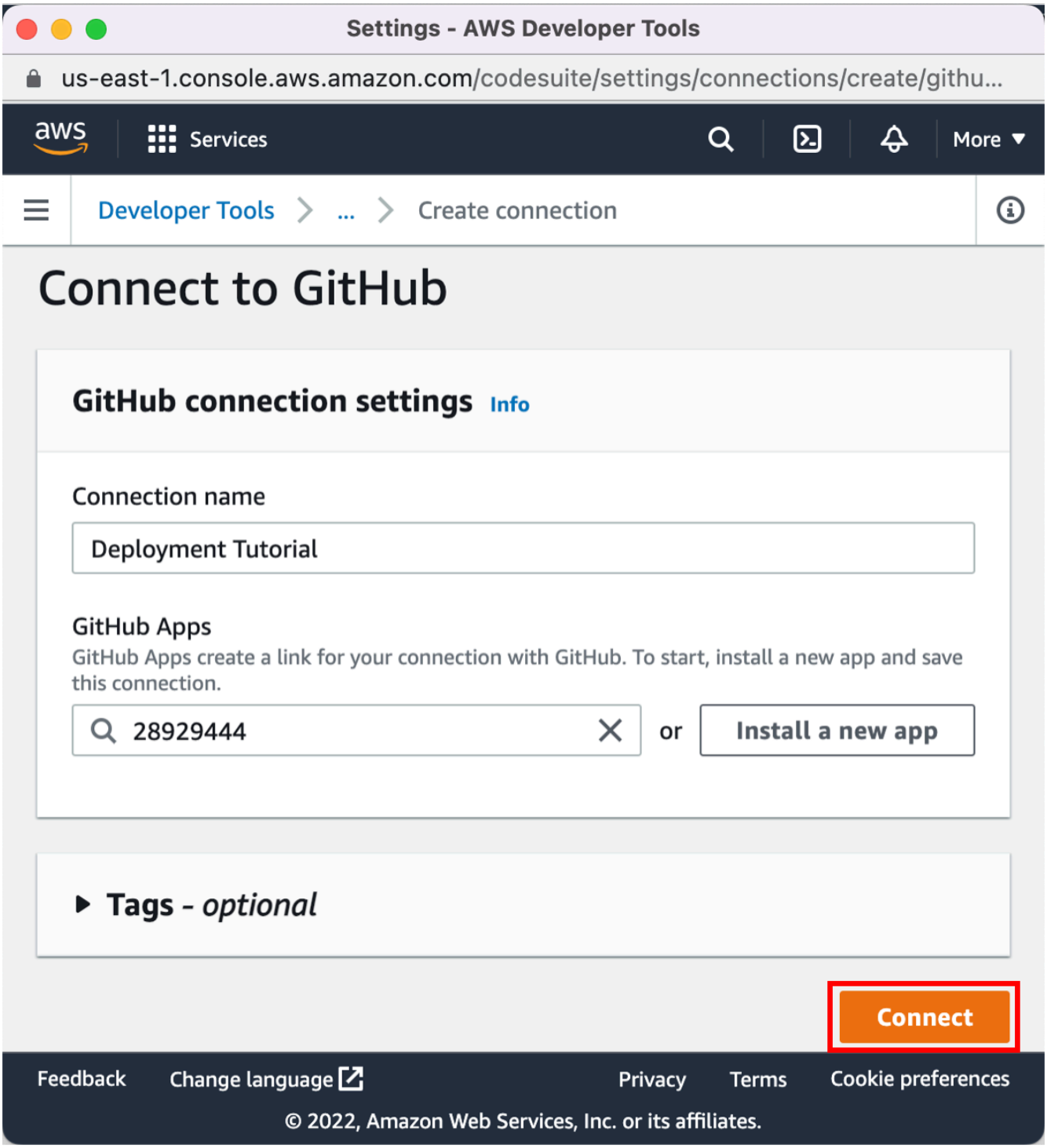
   On the **Install AWS Connector for GitHub** page, choose **Only select repositories** and select the **aws-codepipeline-s3-codedeploy-linux** repository forked in the previous step.

   Choose **Install**.



6. Connect to GitHub

Once redirected back to the **Connect to GitHub** page, choose **Connect**.



7. Specify a repository and branch

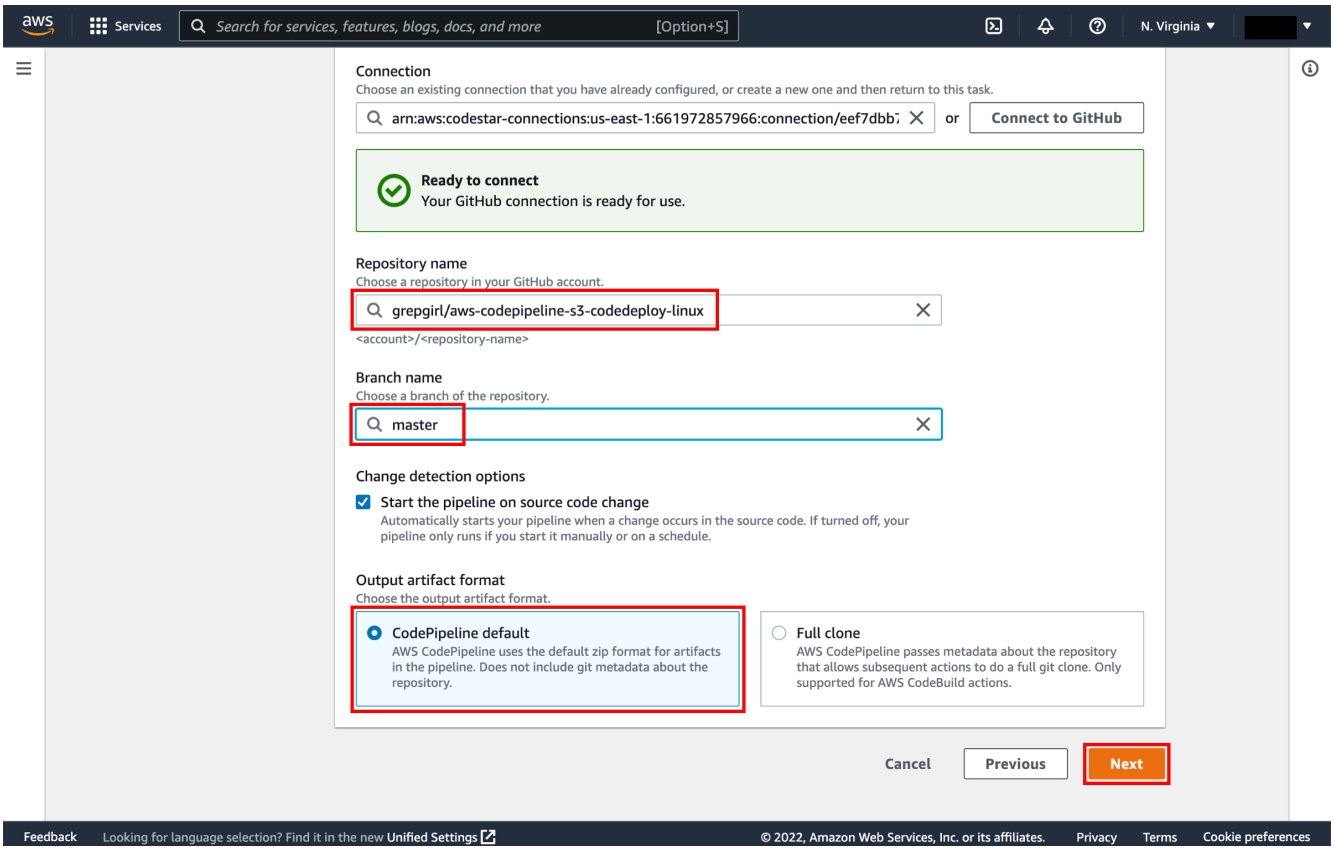
   The **Add source** page will be updated to reflect GitHub is ready to connect. Specify the repository and branch:

Repository name: In the dropdown list, select the GitHub repository you want to use as the source location for your pipeline. Select the forked repository in your GitHub account named aws-codepipeline-s3-codedeploy-linux.

Branch name: In the dropdown list, select the branch you want to use, **master**.

Output artifact format: Select **CodePipeline default**.

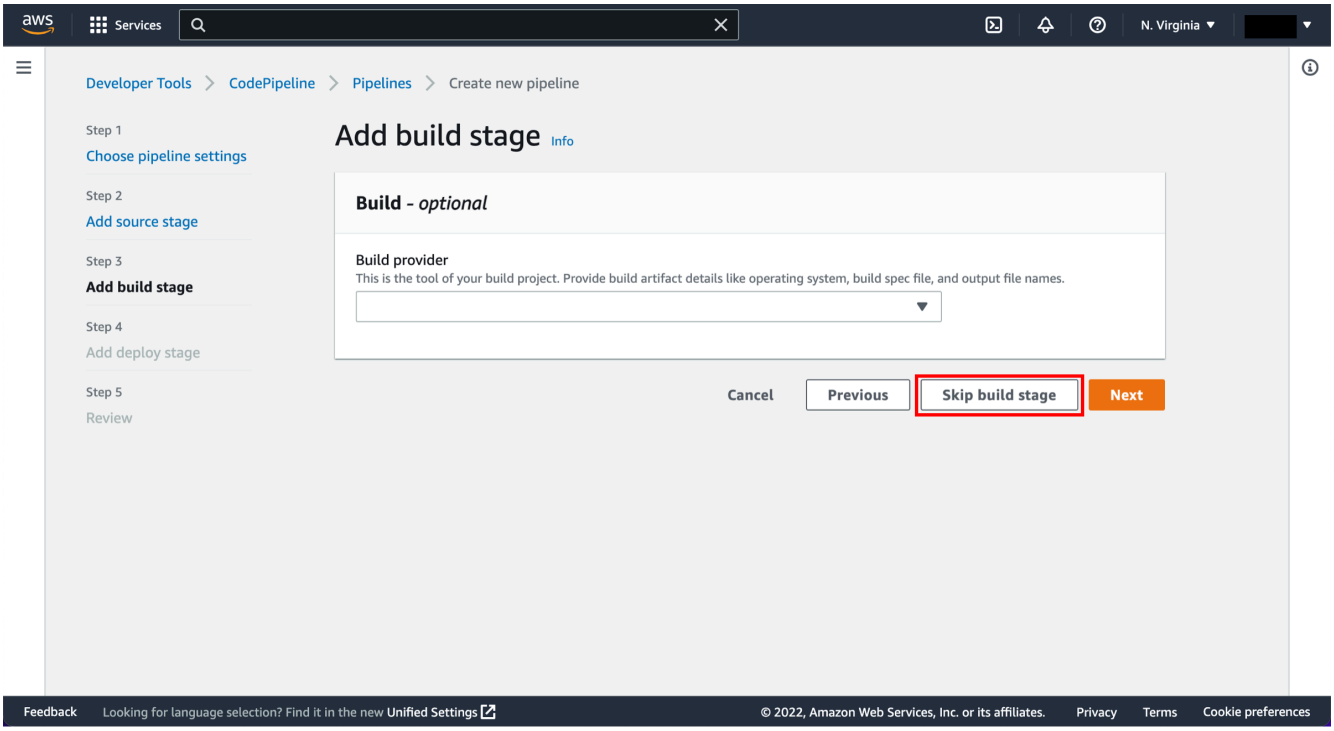
Choose **Next**.



8. Skip build stage

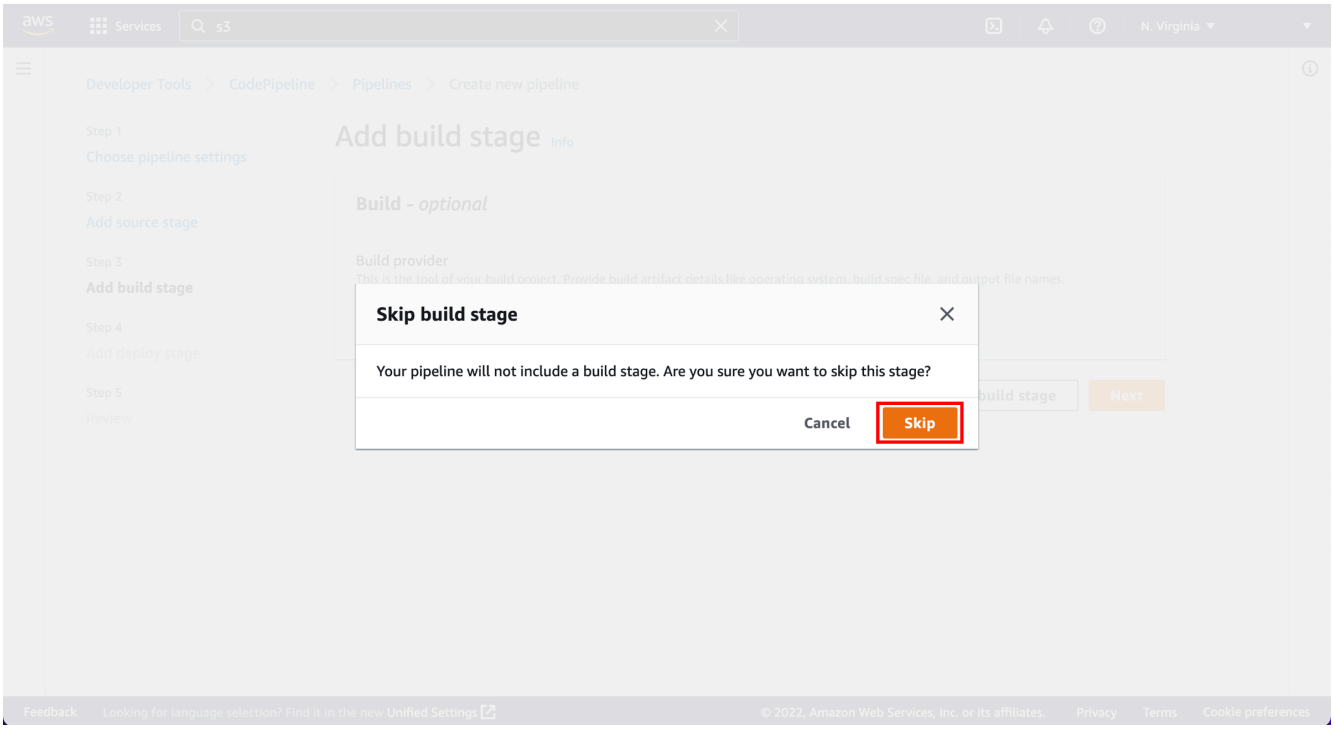
A true continuous deployment pipeline requires a build stage, where code is compiled and unit tested. CodePipeline lets you plug your preferred build provider into your pipeline. However, in this tutorial you will skip the build stage.

In **Step 3: Add build stage**, choose **Skip build stage**.



9. Choose Skip

   In the confirmation dialog, select **Skip**.



10. Configure deploy stage

    In the **Step 4: Add deploy stage** page:

Deploy provider: Select **AWS Elastic Beanstalk**.

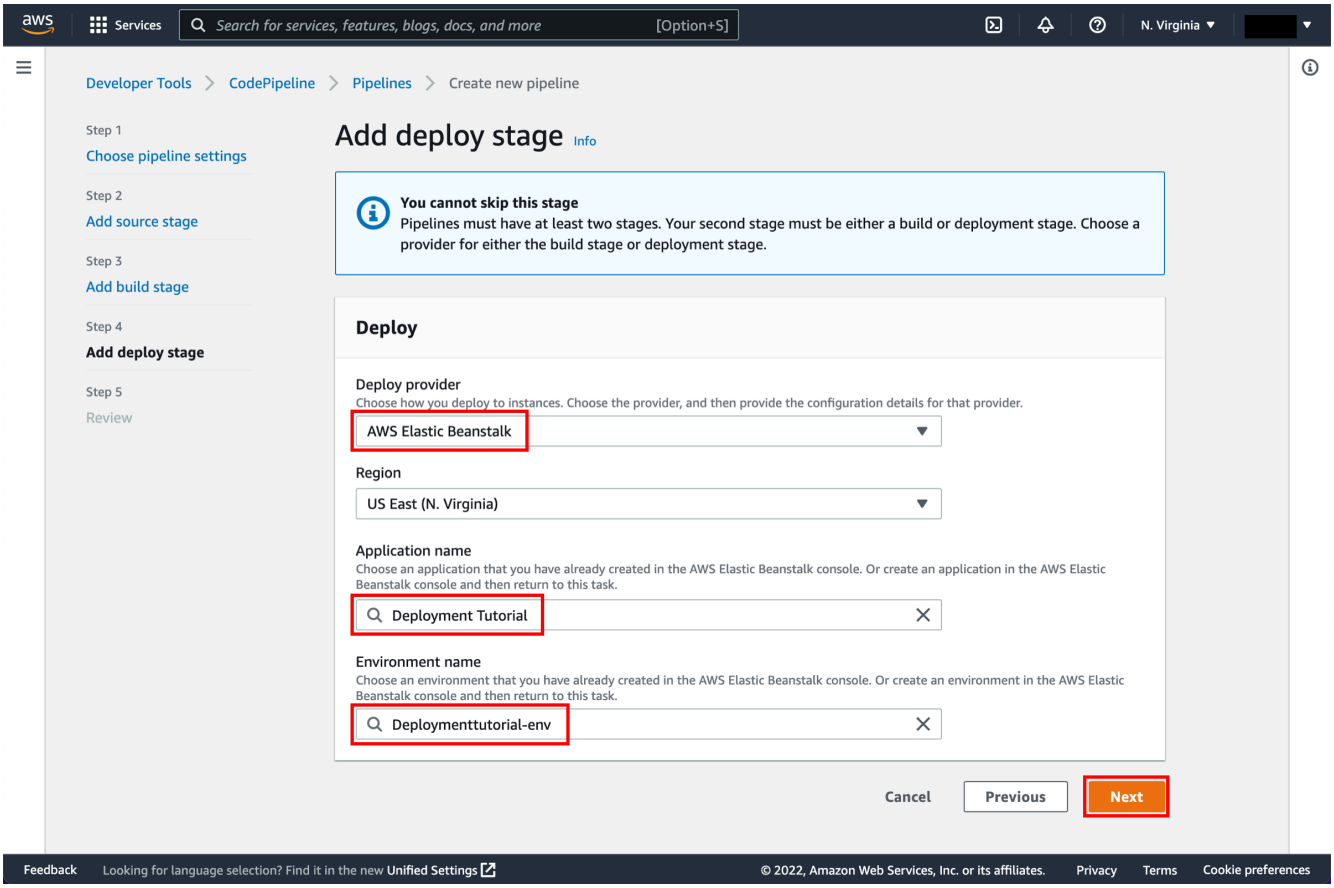
Region: Retain the default region.

Application name: Select **Deployment Tutorial**.

Environment name: Select **Deploymenttutorial-env**.

Click **Next**.

Continue to **Activate your pipeline to deploy your code**.

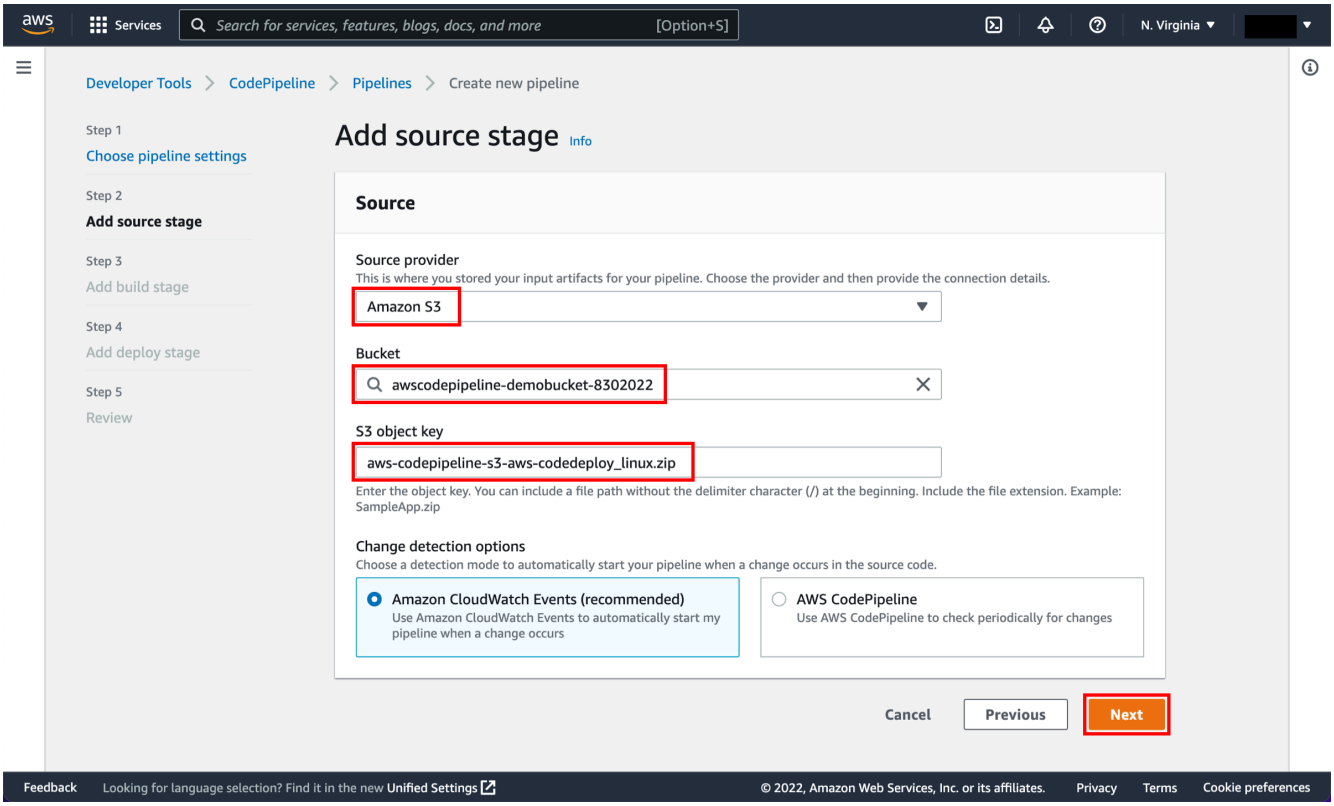


Amazon S3

1. Add source

   Select **Amazon S3** for the **Source provider**, select the Amazon S3 bucket you created, and then enter the S3 object key for the file uploaded, for example: **aws-codepipeline-s3-aws-codedeploy_linux.zip**.

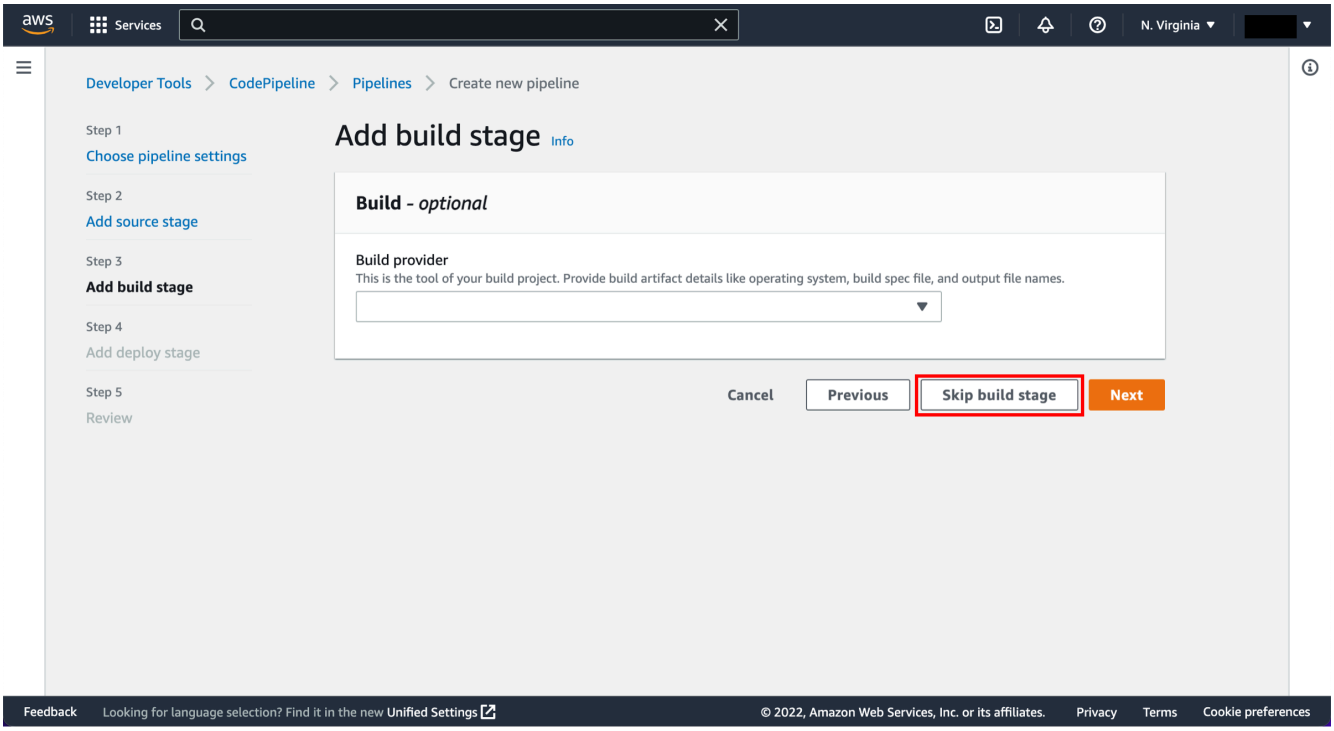
Choose **Next**.



2. Skip build stage

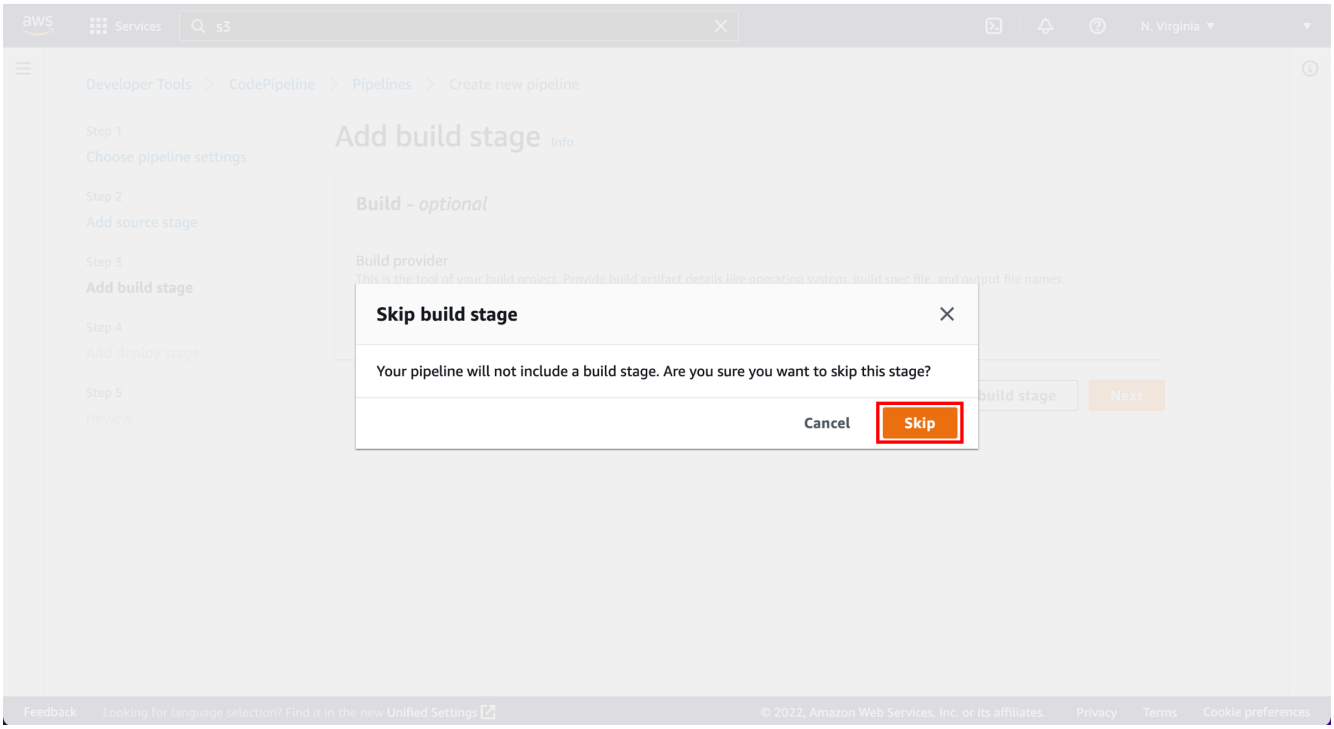
A true continuous deployment pipeline requires a build stage, where code is compiled and unit tested. CodePipeline lets you plug your preferred build provider into your pipeline. However, in this tutorial you will skip the build stage.

In **Step 3: Add build stage**, choose **Skip build stage**.



3. Choose Skip

   In the confirmation dialog, select **Skip**.



4. Configure deploy stage

   In the **Step 4: Add deploy stage** page:

Deploy provider: Select **AWS Elastic Beanstalk**.

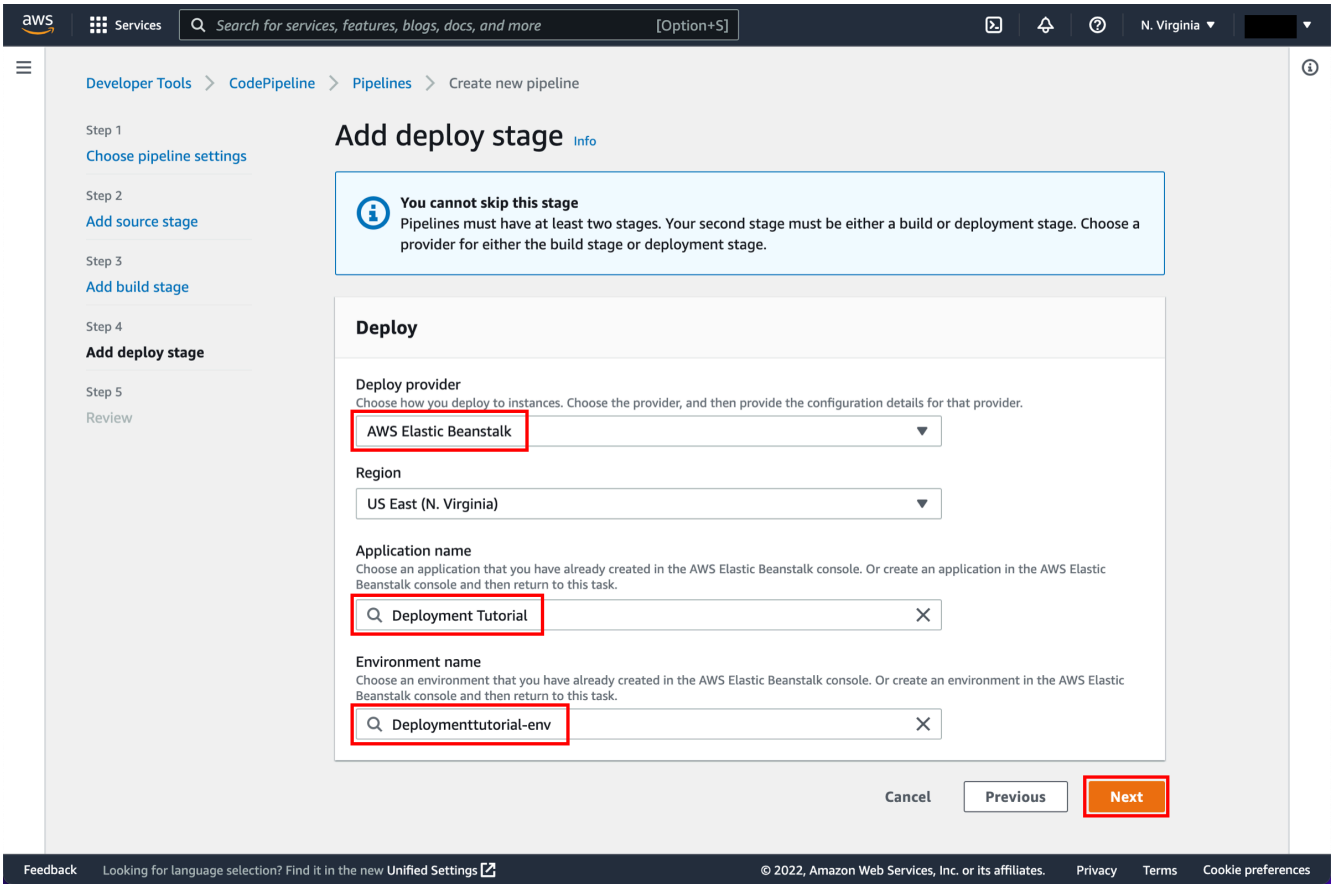
Region: Retain the default region.

Application name: Select **Deployment Tutorial**.

Environment name: Select **Deploymenttutorial-env**.

Click **Next**.

Continue to **Activate your pipeline to deploy your code**.



AWS CodeCommit

1. Add source

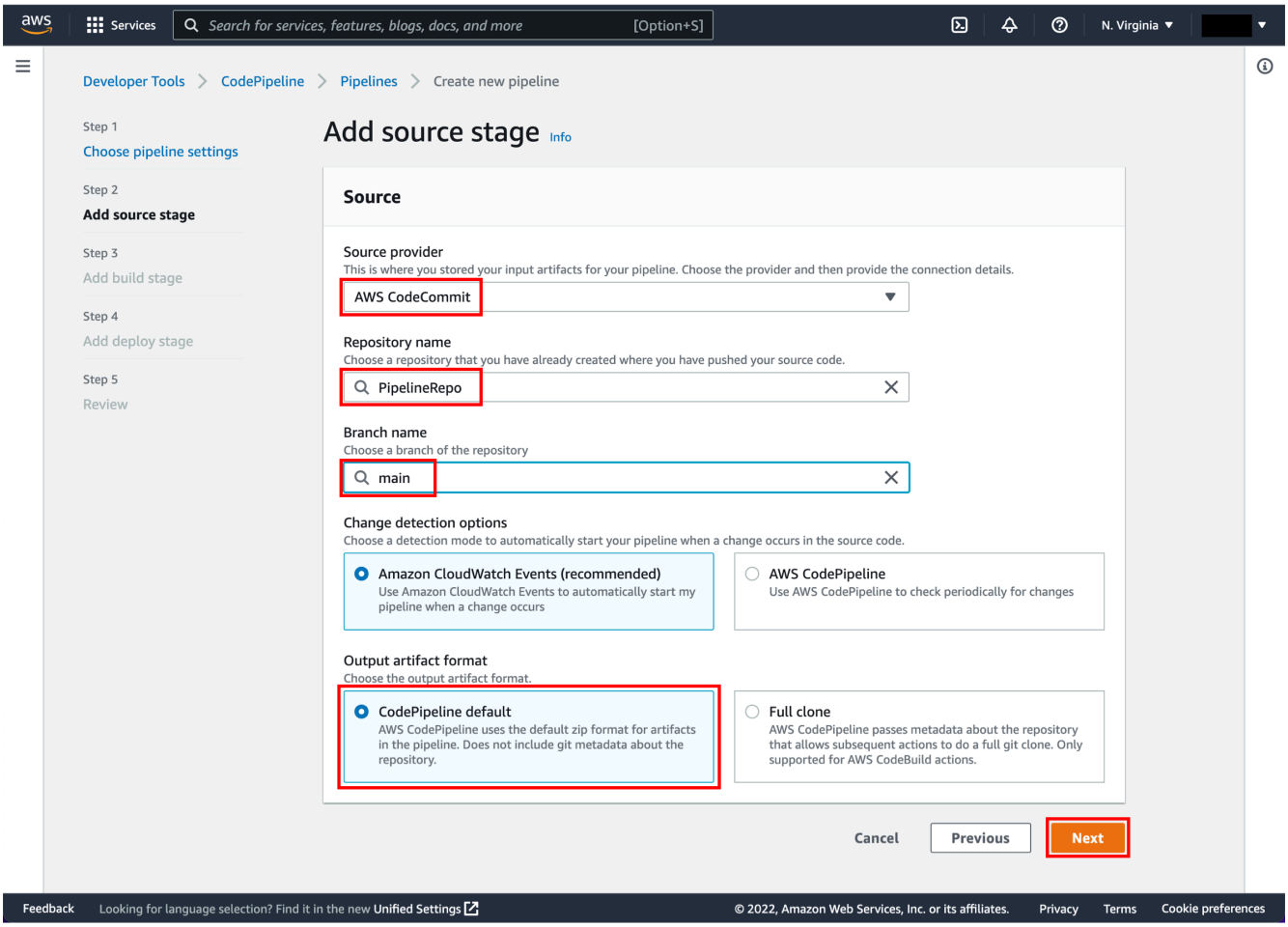
   Select **AWS CodeCommit** for the **Source provider**.

Repository name: In the dropdown list, choose the **PipelineRepo repository** you created to use as the source location for your pipeline.

Branch name: In the dropdown list, choose the branch you want to use, **main**.

Output artifact format: Choose **CodePipeline default**.

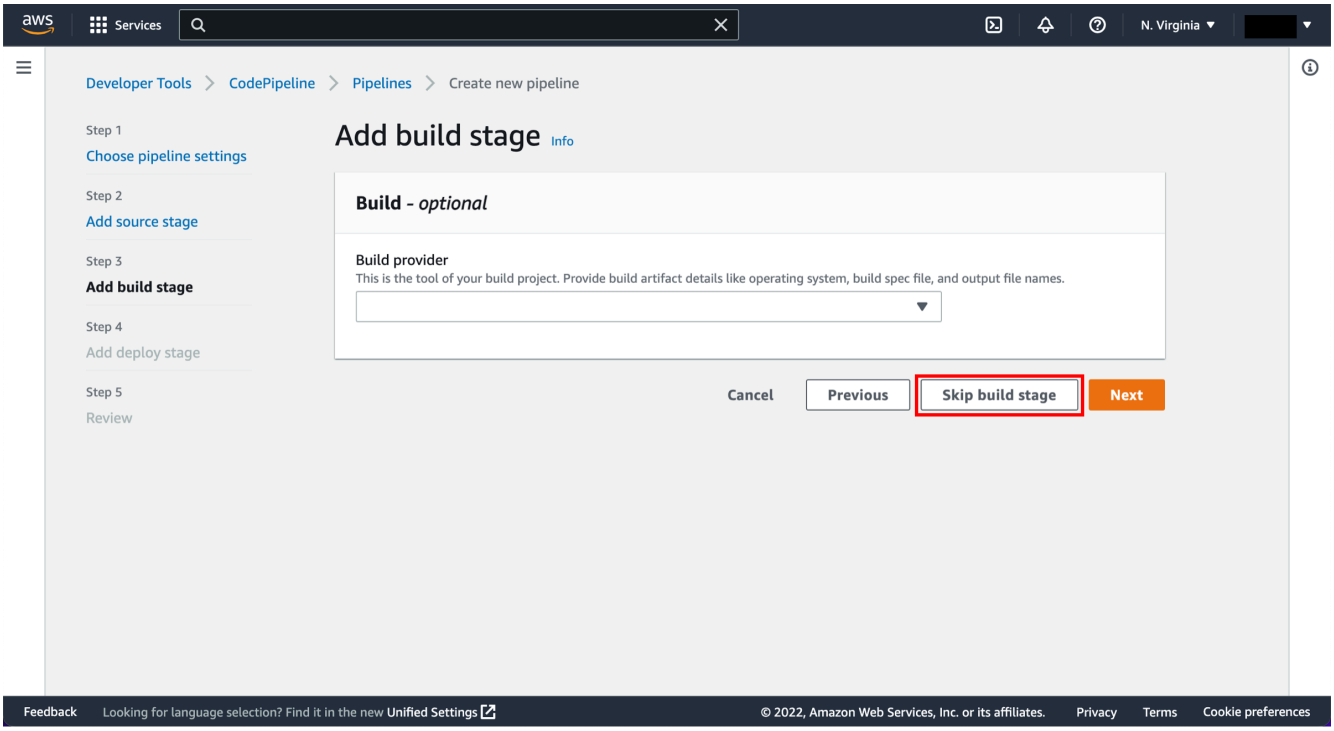
Choose **Next**.



2. Skip build stage

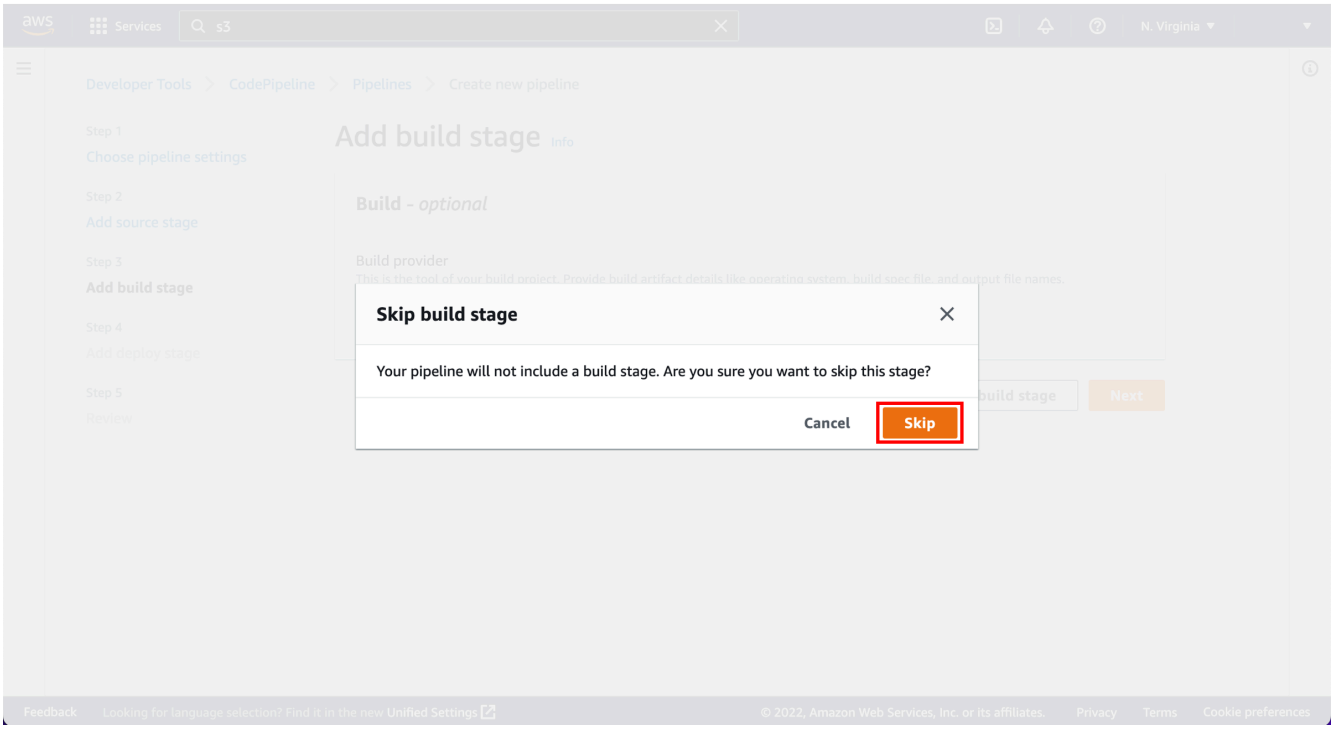
A true continuous deployment pipeline requires a build stage, where code is compiled and unit tested. CodePipeline lets you plug your preferred build provider into your pipeline. However, in this tutorial you will skip the build stage.

In **Step 3: Add build stage**, choose **Skip build stage**.



3. Choose Skip

   In the confirmation dialog, select **Skip**.



4. Configure deploy stage

   In the **Step 4: Add deploy stage** page:

Deploy provider: Select **AWS Elastic Beanstalk**.

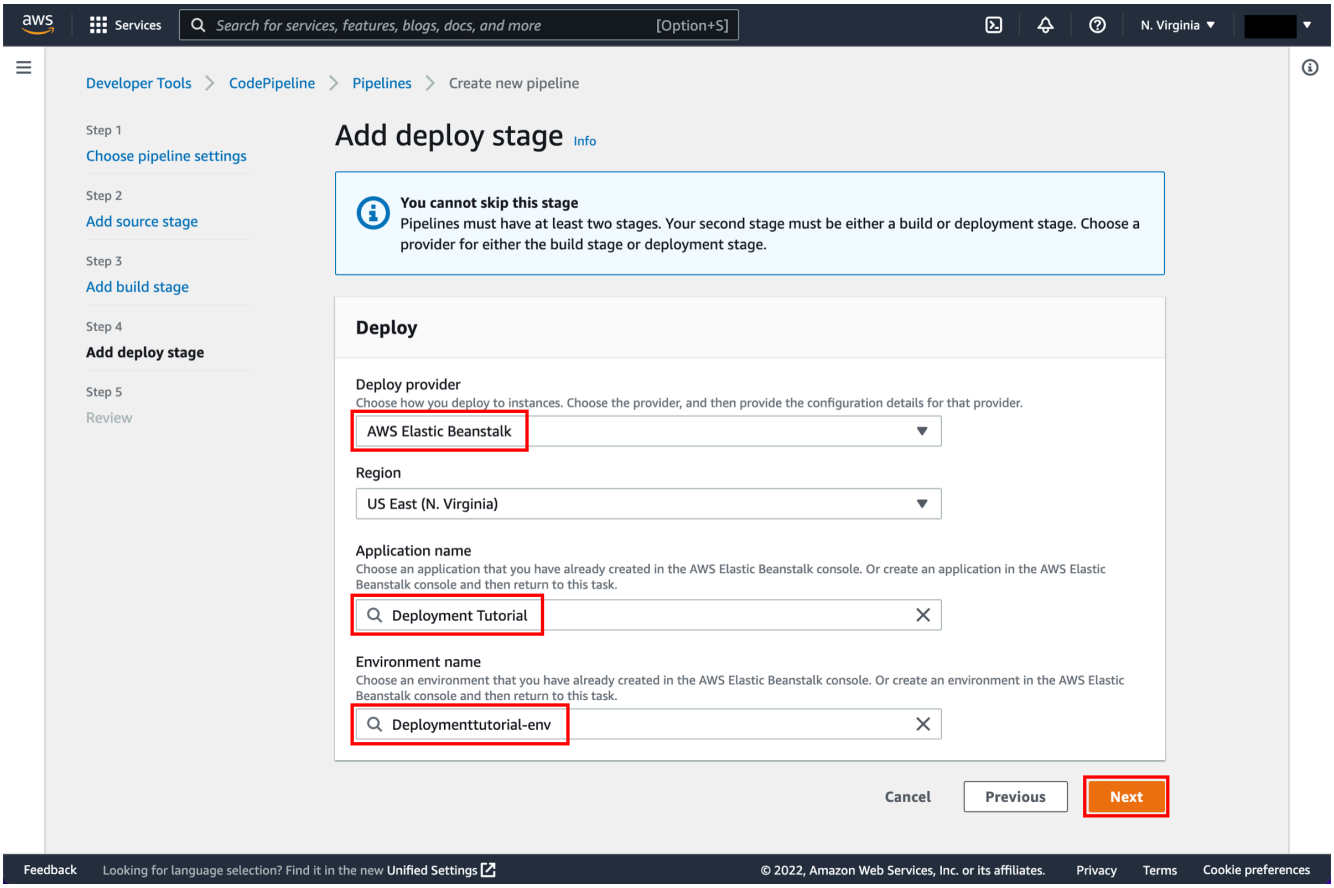
Region: Retain the default region.

Application name: Select **Deployment Tutorial**.

Environment name: Select **Deploymenttutorial-env**.

Click **Next**.

Continue to **Activate your pipeline to deploy your code**.

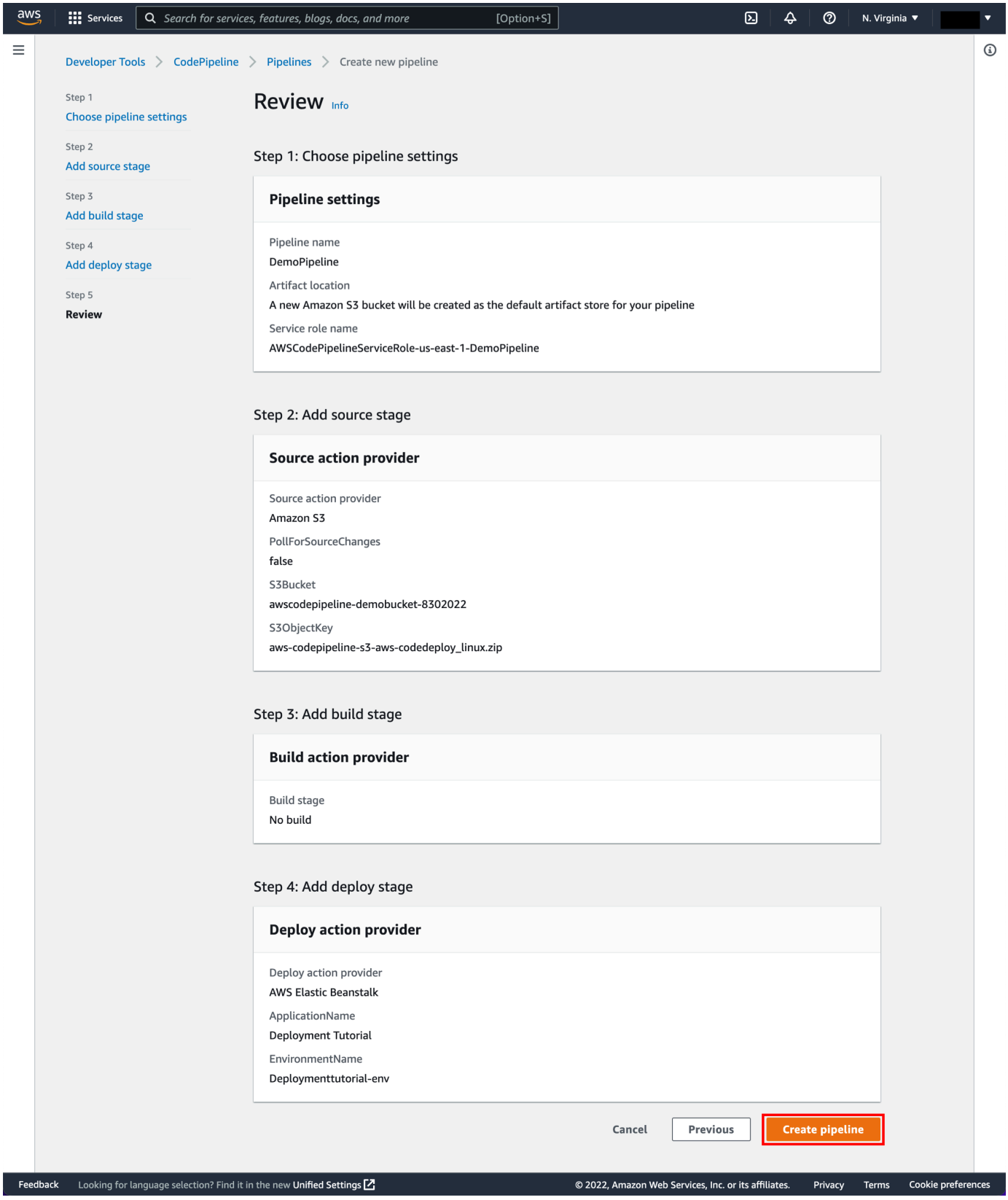


Step 4: Activate your pipeline to deploy your code

In this step, you will launch your pipeline. Once your pipeline has been created, it will start to run automatically. First, it detects the sample app code in your source location, bundles up the files, and then moves them to the second stage that you defined. During this stage, it passes the code to Elastic Beanstalk, which contains the EC2 instance that will host your code. Elastic Beanstalk handles deploying the code to the EC2 instance.

1. Review configuration and create pipeline

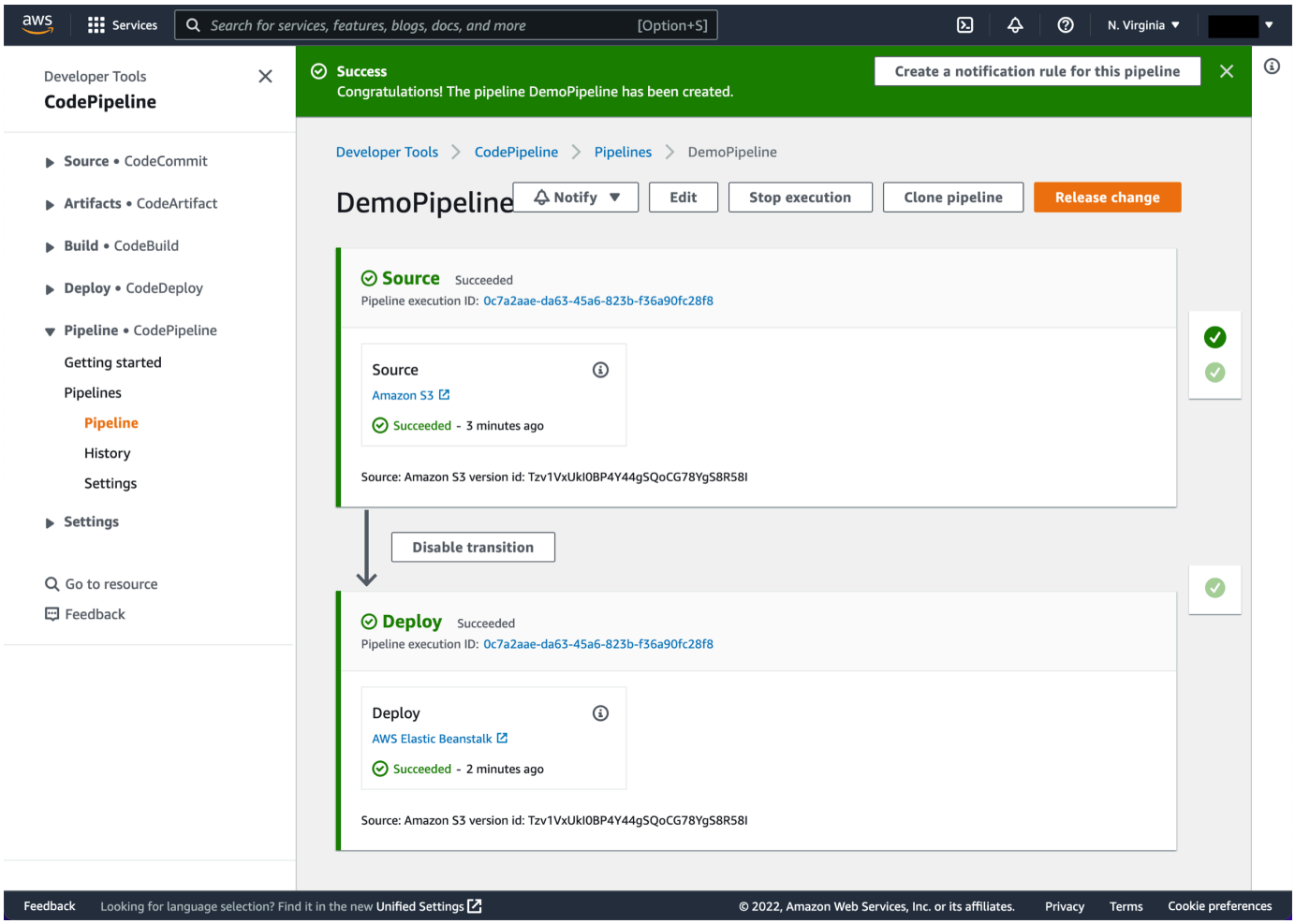
   In the **Step 5: Review** page, review the information and choose **Create pipeline**.



2. Monitor the pipeline status

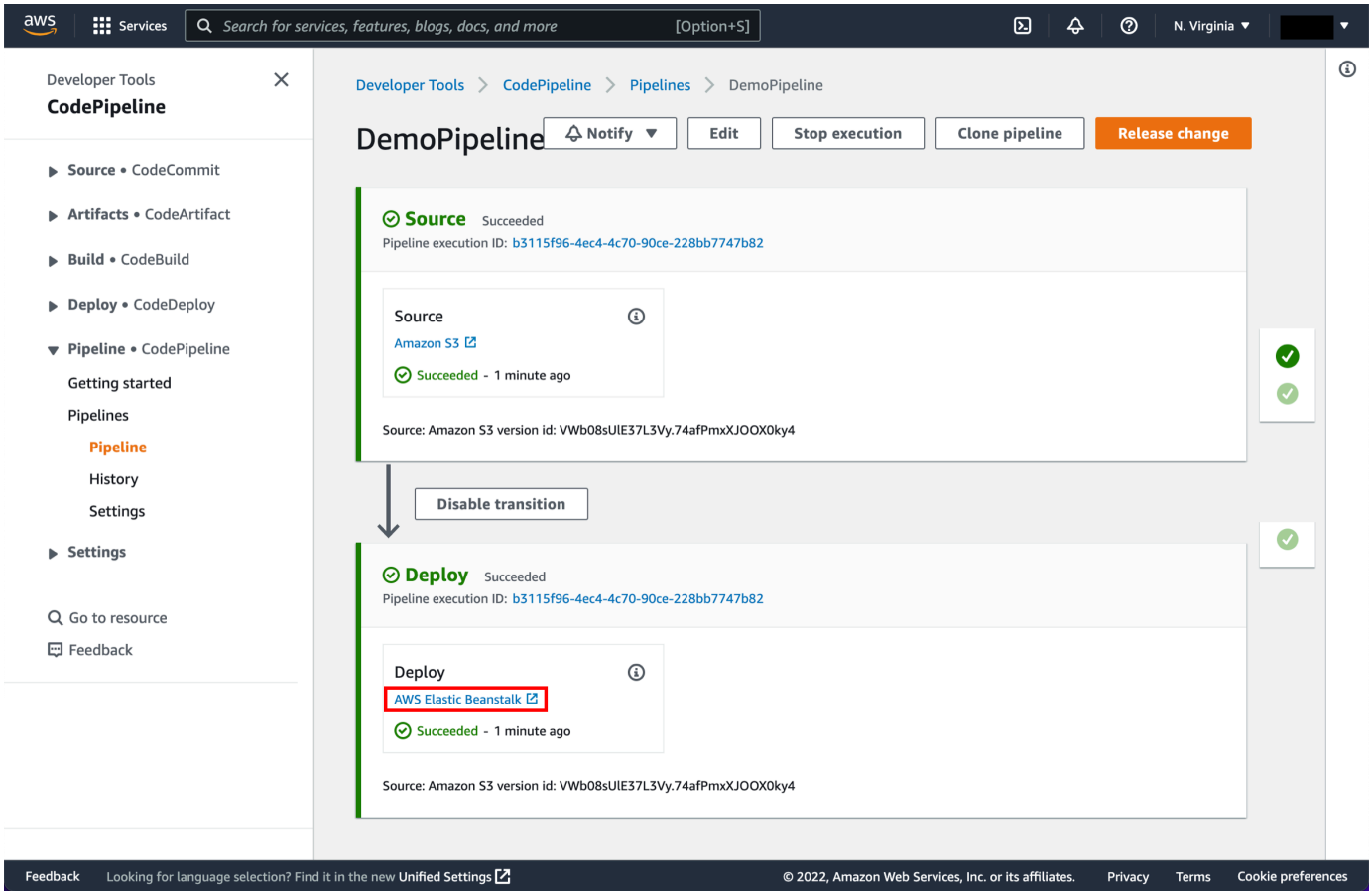
After your pipeline is created, the pipeline status page appears and the pipeline automatically starts to run. You can view progress as well as success and failure messages as the pipeline performs each action.

To verify your pipeline ran successfully, monitor the progress of the pipeline as it moves through each stage. The status of each stage will change 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.



3. Select Elastic Beanstalk

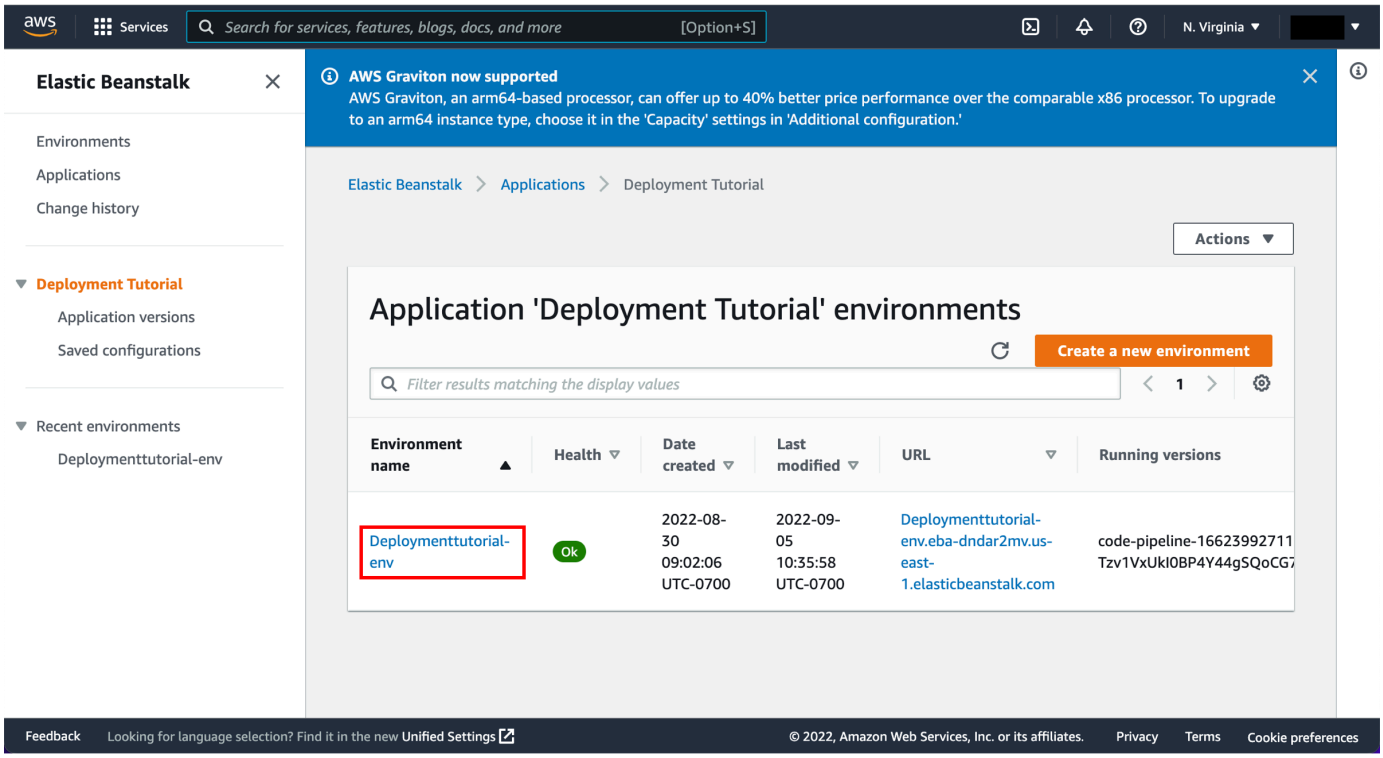
   In the status area for the Beta stage, select **AWS Elastic Beanstalk**.



4. Select the environment

   The AWS Elastic Beanstalk console opens with the details of the deployment.

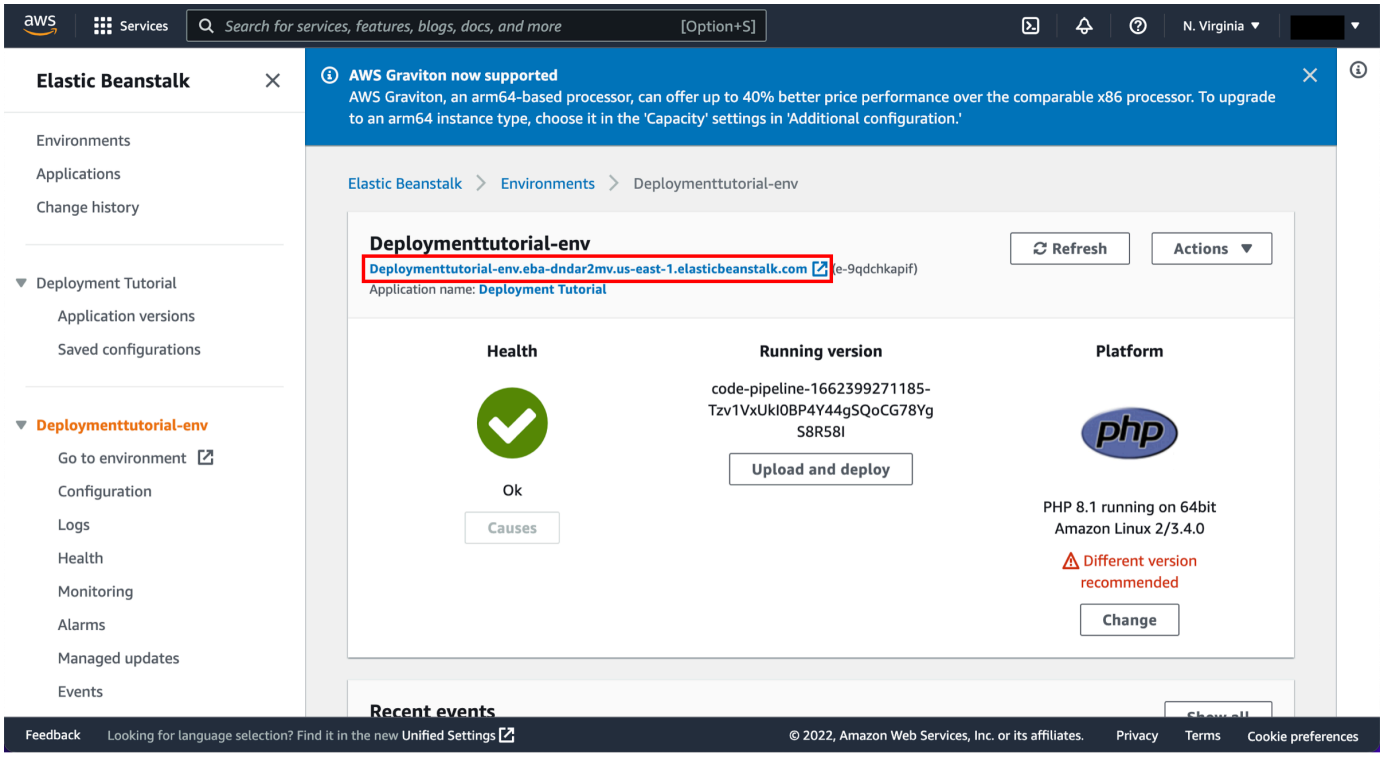
   Select the environment you created earlier, called **Default-Environment Deploymenttutorial-env**.



5. Select the URL of the sample website

   Select the URL to view the sample website you deployed.

   A webpage with a congratulations message indicating you successfully created a pipeline from your source to Amazon EC2 will open.



Step 5: Commit a change and then update your app

In this step, you will revise the sample code and commit the change to your repository. CodePipeline will detect your updated sample code and then automatically initiate deploying it to your EC2 instance by way of Elastic Beanstalk.

Note that the sample web page you deployed refers to AWS CodeDeploy, a service that automates code deployments. In CodePipeline, CodeDeploy is an alternative to using Elastic Beanstalk for deployment actions. Let's update the sample code so that it correctly states that you deployed the sample using Elastic Beanstalk.

Choose the appropriate tab based on the code source you used.

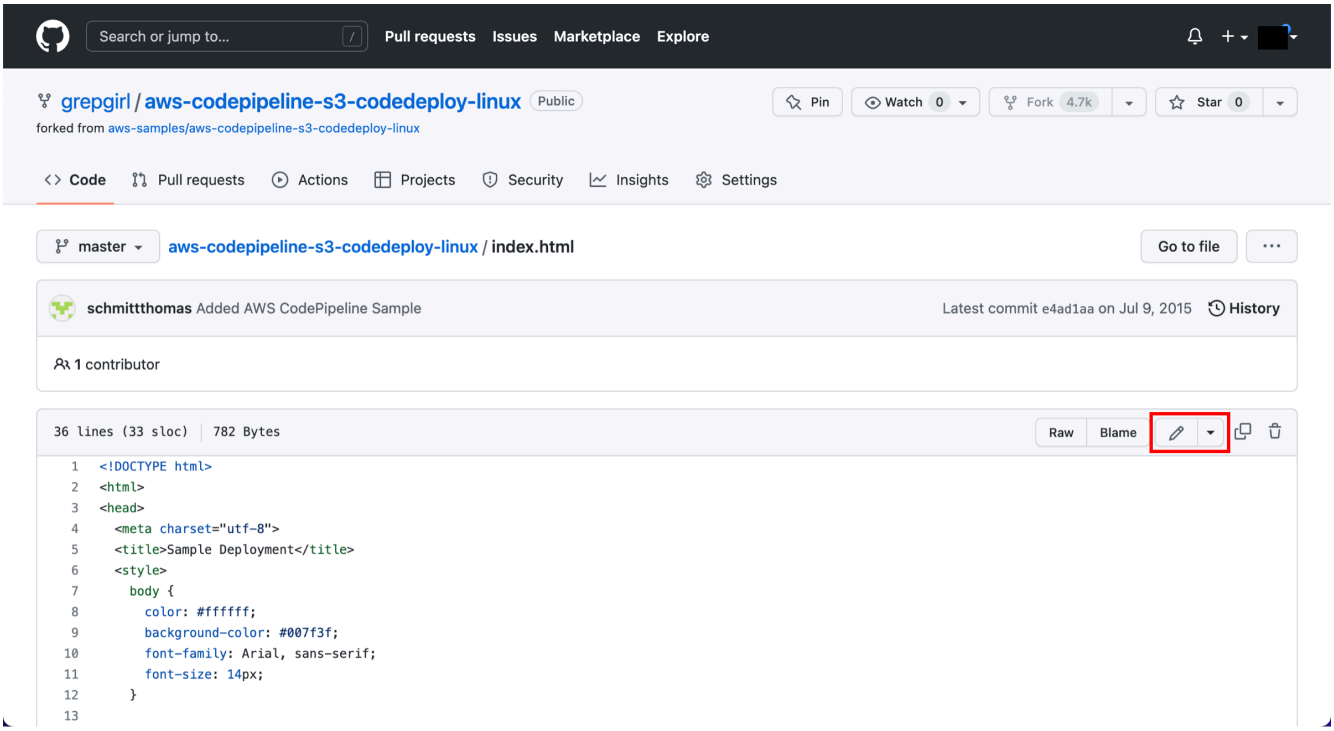
GitHub

1. Edit the code

   Visit your own copy of the repository that you forked in GitHub.

   Open **index.html**.

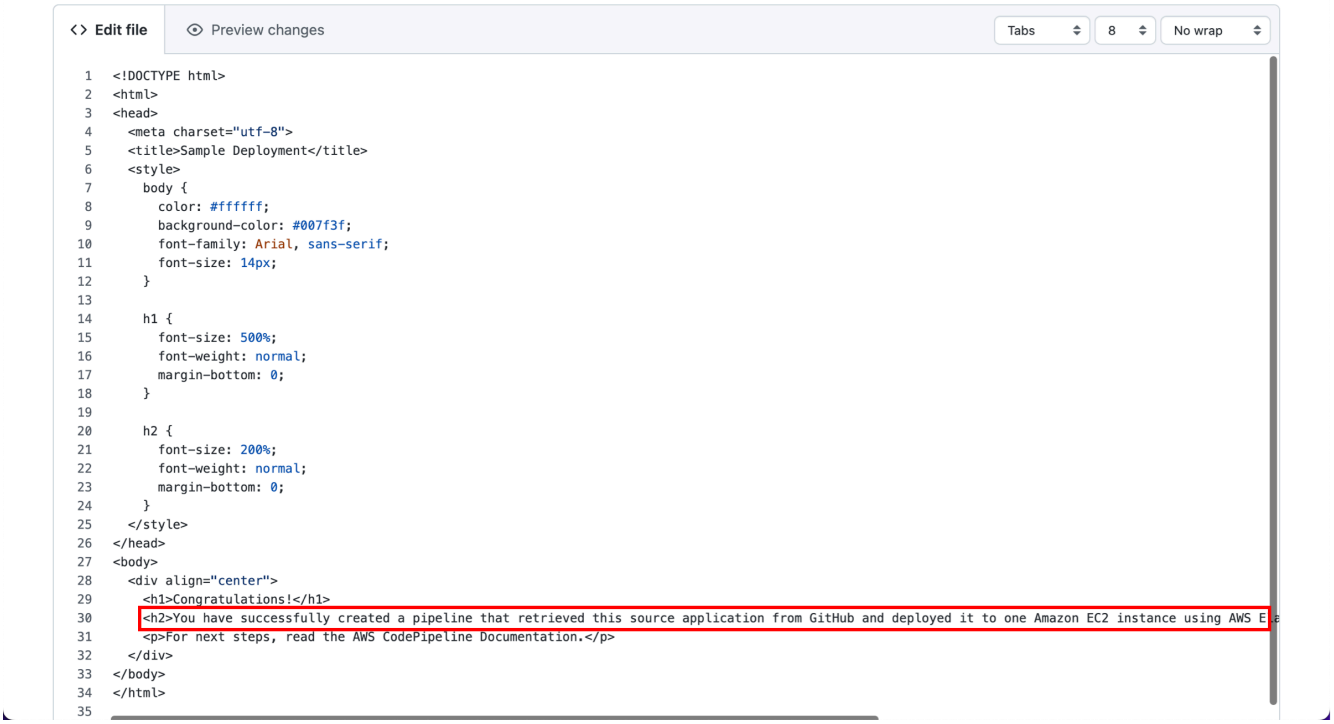
   Select the **Edit icon**.



2. Insert text

   Update the webpage by copying and pasting the following text on line 30:

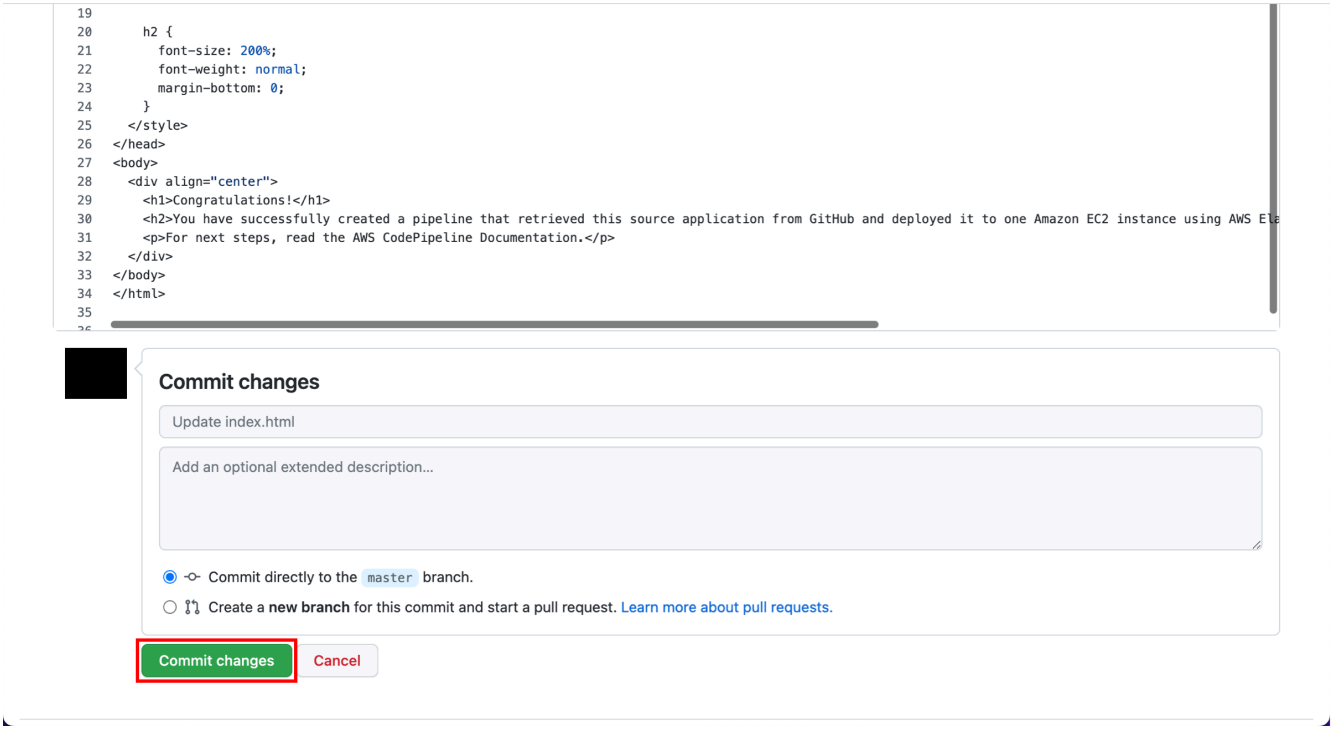
   You have successfully created a pipeline that retrieved this source application from GitHub and deployed it to one Amazon EC2 instance using AWS Elastic Beanstalk. You're one step closer to practicing continuous deployment!



3. Commit the change

   Commit the change to your repository.

   Then, go to View the page you updated with GitHub.



Amazon S3

1. Edit the code

   On your desktop, visit the zip file you downloaded called **aws-codepipeline-s3-aws-codedeploy_linux.zip**.

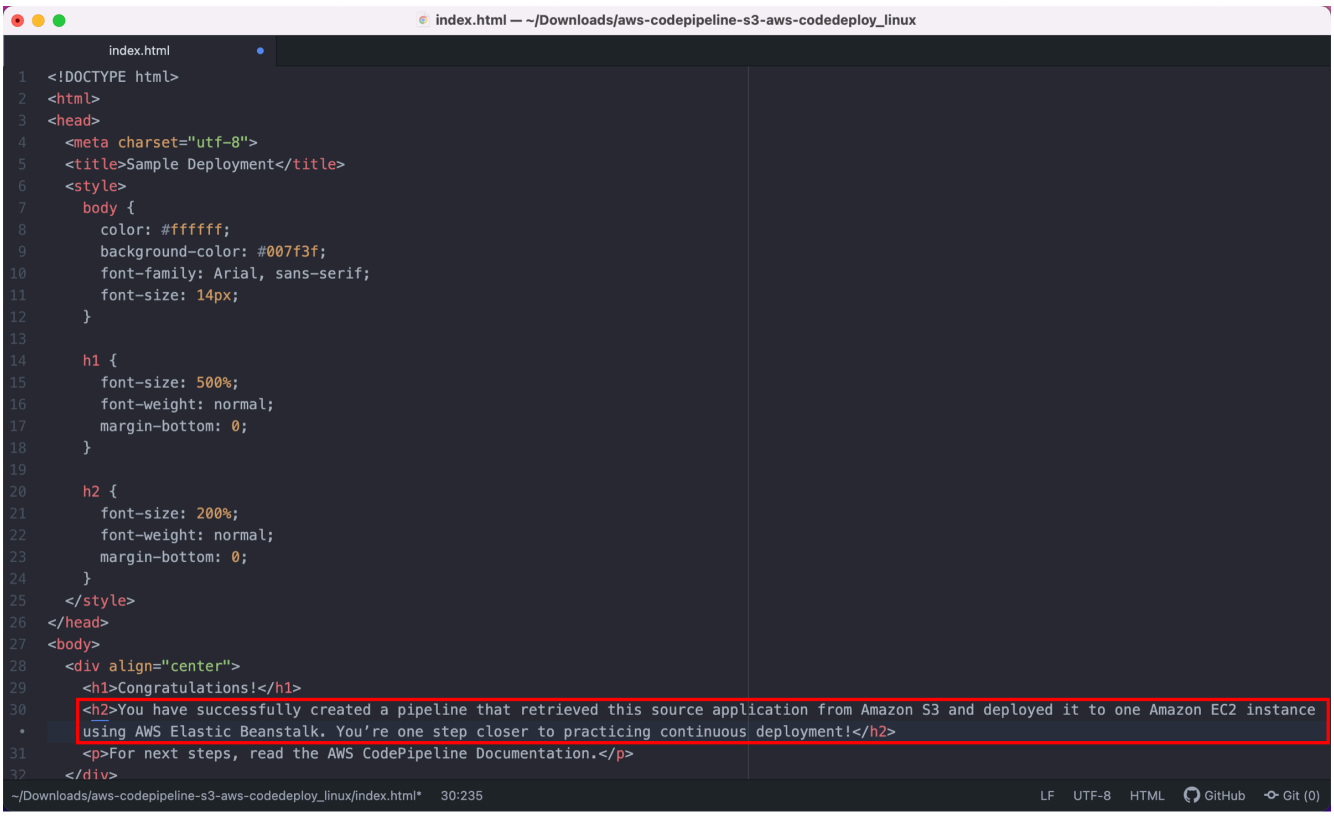
   Edit the sample web app code:

   a. Extract index.html from the zip file and open it using your preferred text editor.

   b. Update the header text that comes after **Congratulations!** so that it reads:

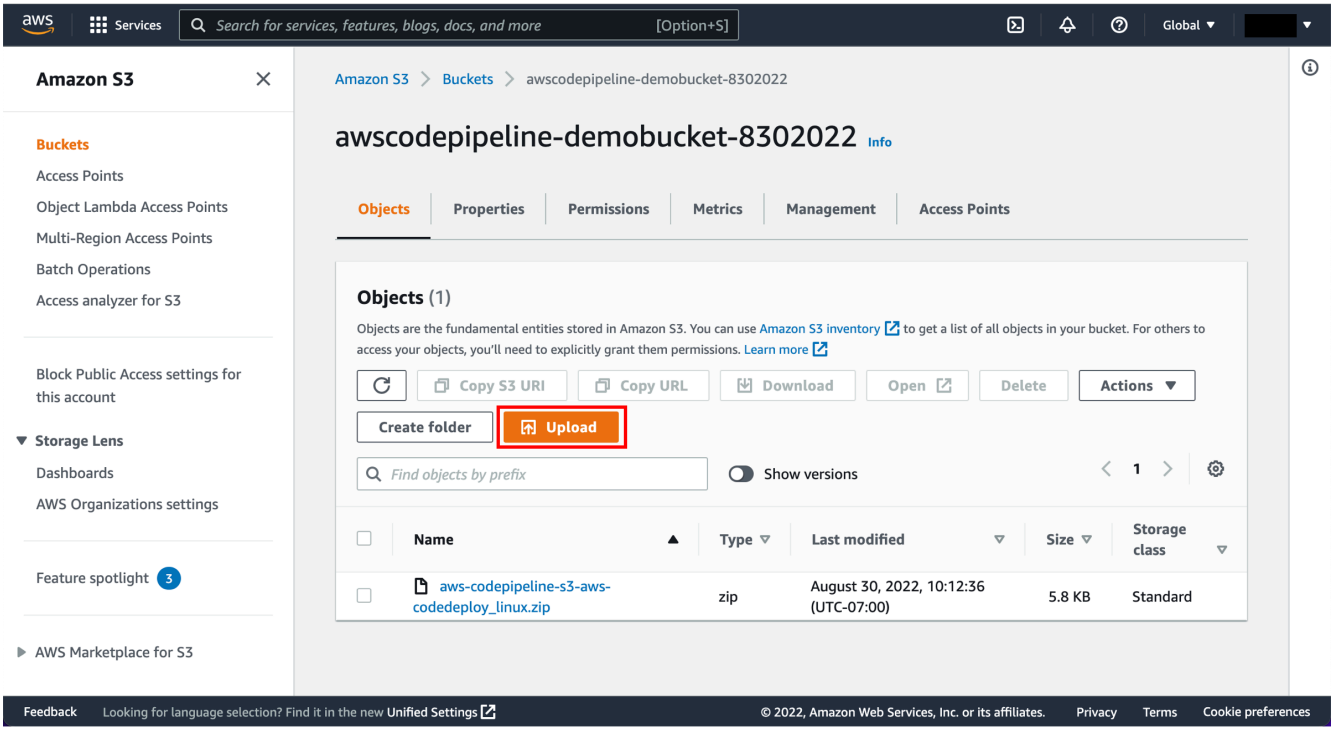
      You have successfully created a pipeline that retrieved this source application from Amazon S3 and deployed it to one Amazon EC2 instance using AWS Elastic Beanstalk. You're one step closer to practicing continuous deployment!

   c. Copy the updated index.html file back into **aws-codepipeline-s3-aws-codedeploy_linux.zip** and replace the older version of index.html.



2. Upload the file to your bucket

Return to the S3 bucket that you created earlier and select **Upload**.

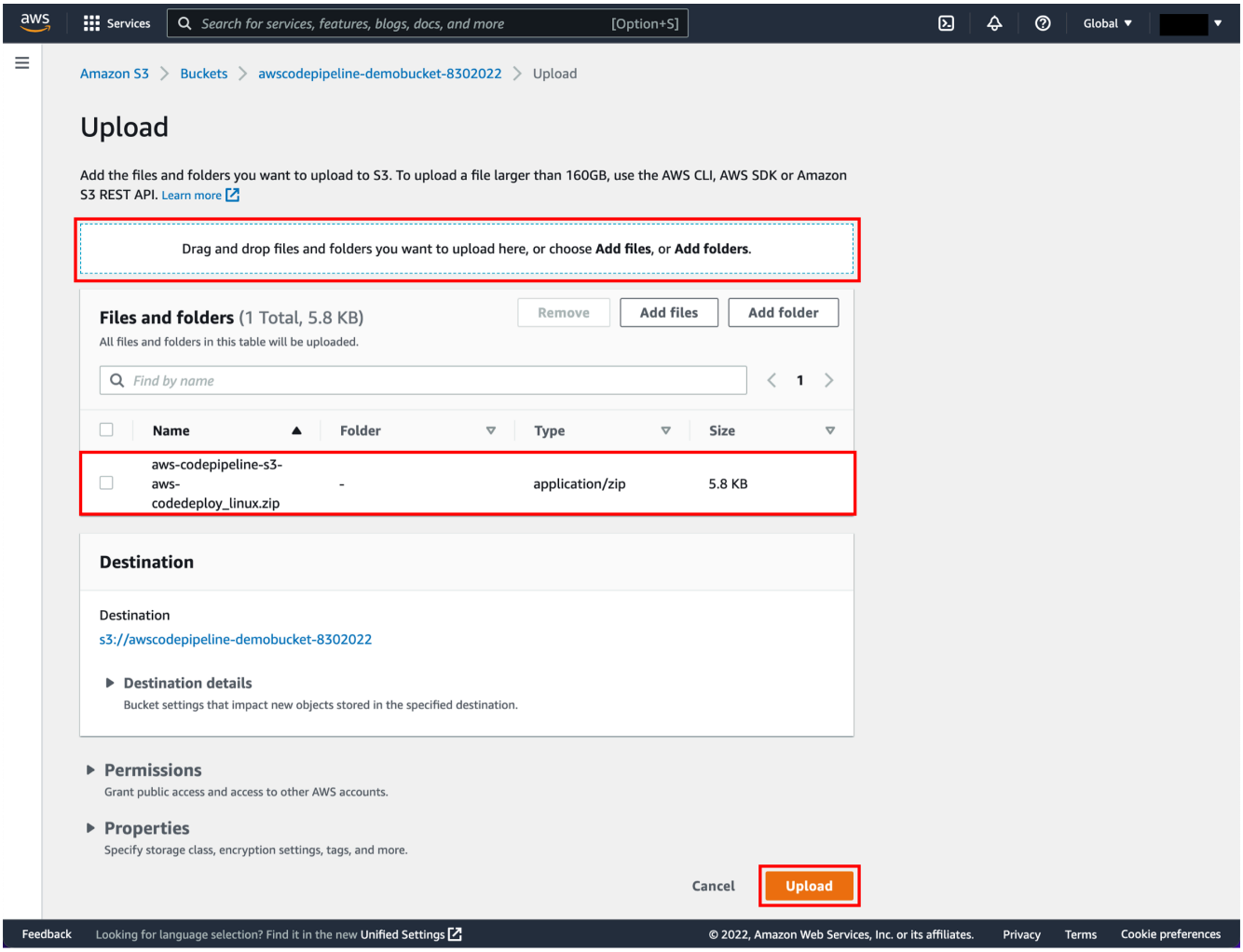


3. Upload the file to your bucket

Select **Add files** to upload the updated **aws-codepipeline-s3-aws-codedeploy_linux.zip** file or drag and drop the file. Then choose **Upload**.

> **Note**
>
> Because you enabled versioning when you first created the S3 bucket, S3 will save a copy of every version of your files.

Then, go to View the page you updated in Amazon S3.



AWS CodeCommit

1. Edit the code

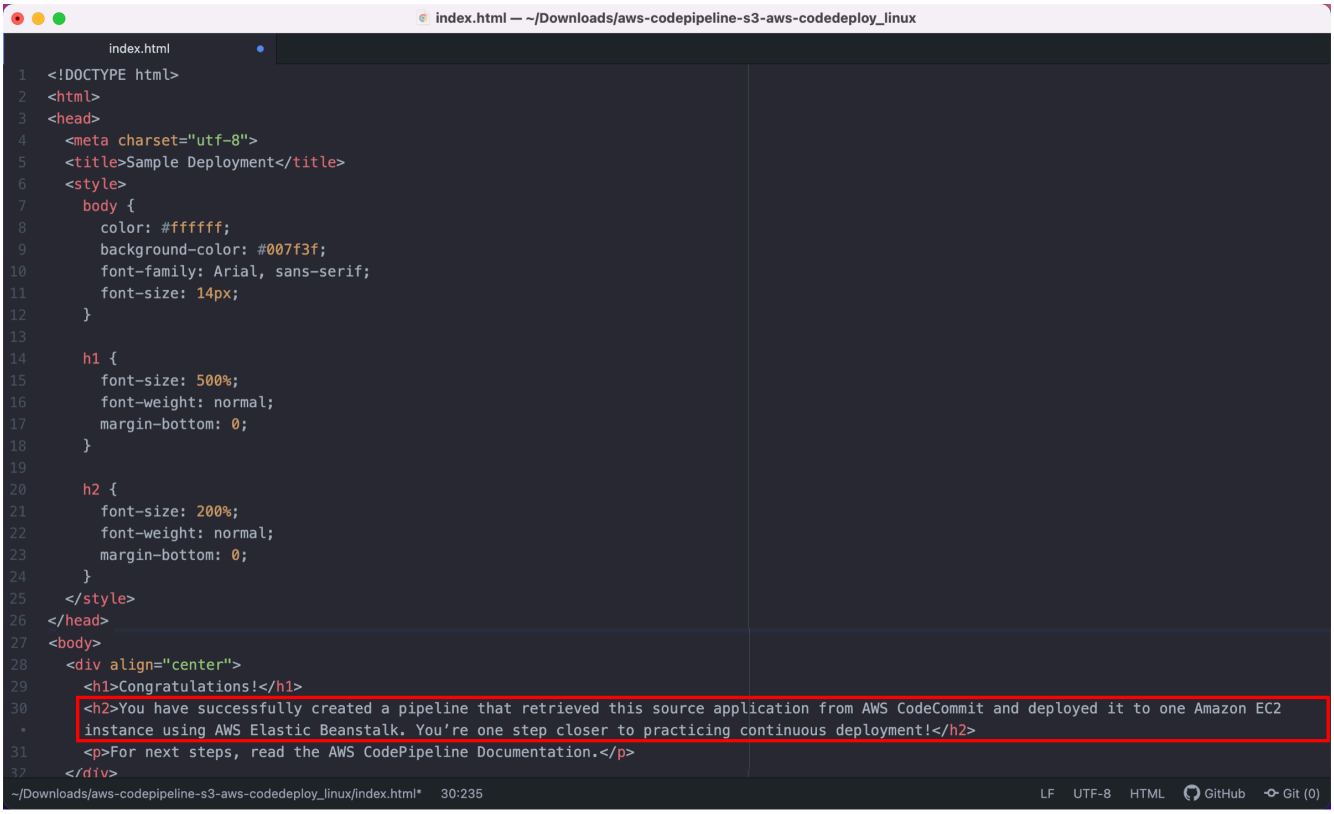
   On your desktop, visit the zip file you downloaded called **aws-codepipeline-s3-aws-codedeploy_linux.zip**.

   Edit the sample web app code:

   a. Extract index.html from the zip file and open it using your preferred text editor.

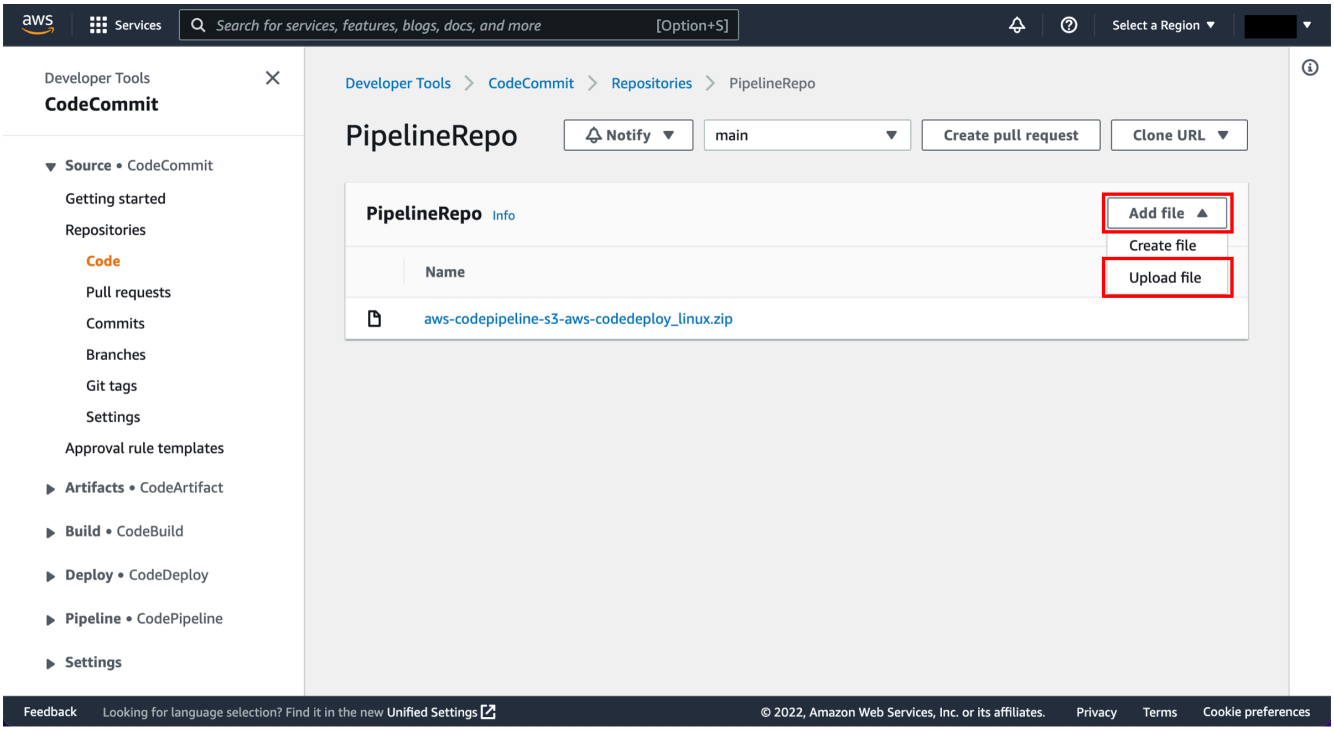
   b. Update the header text that comes after **Congratulations!** so that it reads:

      You have successfully created a pipeline that retrieved this source application from AWS CodeCommit and deployed it to one Amazon EC2 instance using AWS Elastic Beanstalk. You're one step closer to practicing continuous deployment!



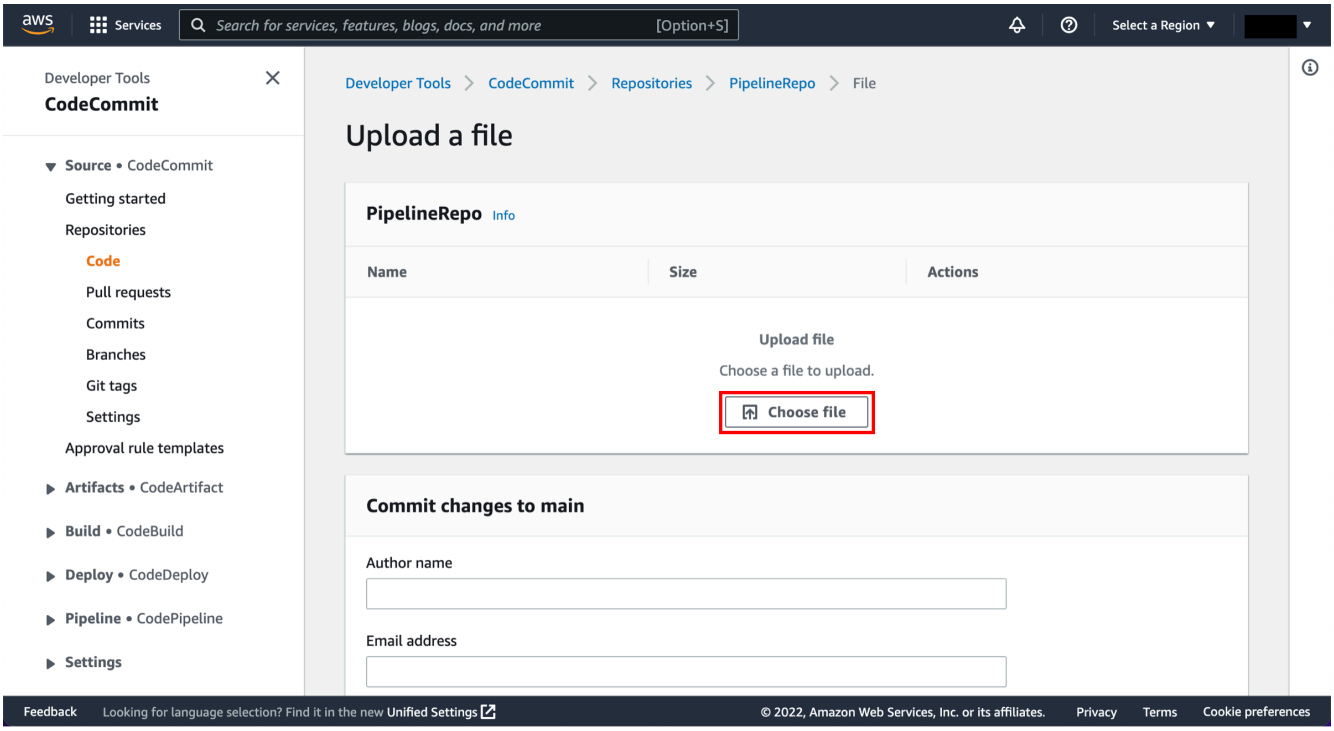
2. Upload the file

   From the **CodeCommit PipelineRepo** page, choose **Add file** and select **Upload file**.



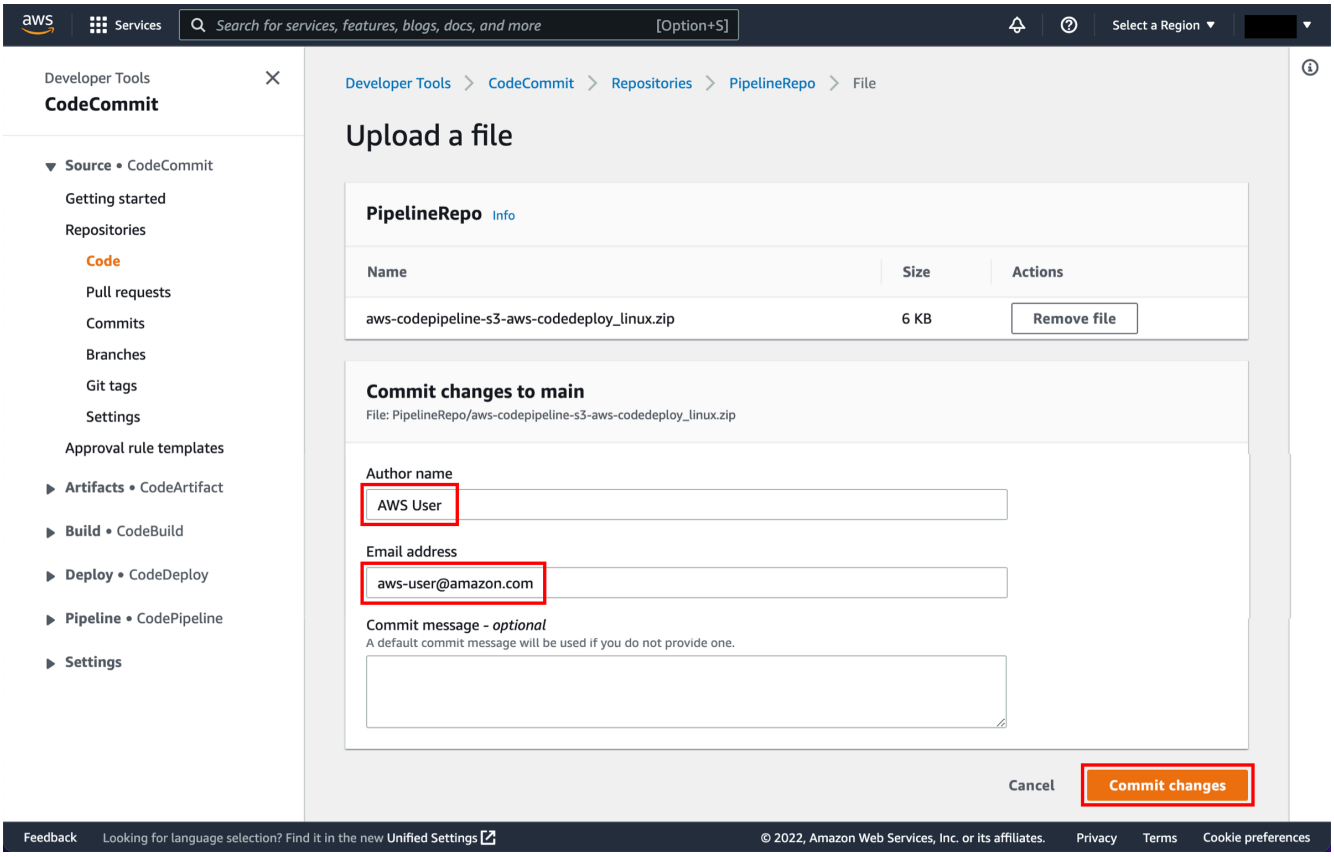
3. Upload the file

On the **Upload a file** page, choose the **Choose file** button and select the updated **aws-codepipeline-s3-aws-codedeploy_linux.zip** file.



4. Commit changes

Enter an Author name and Email address, then choose **Commit changes**.



Step 6: View the page you updated

In this step, you will view the page you updated.

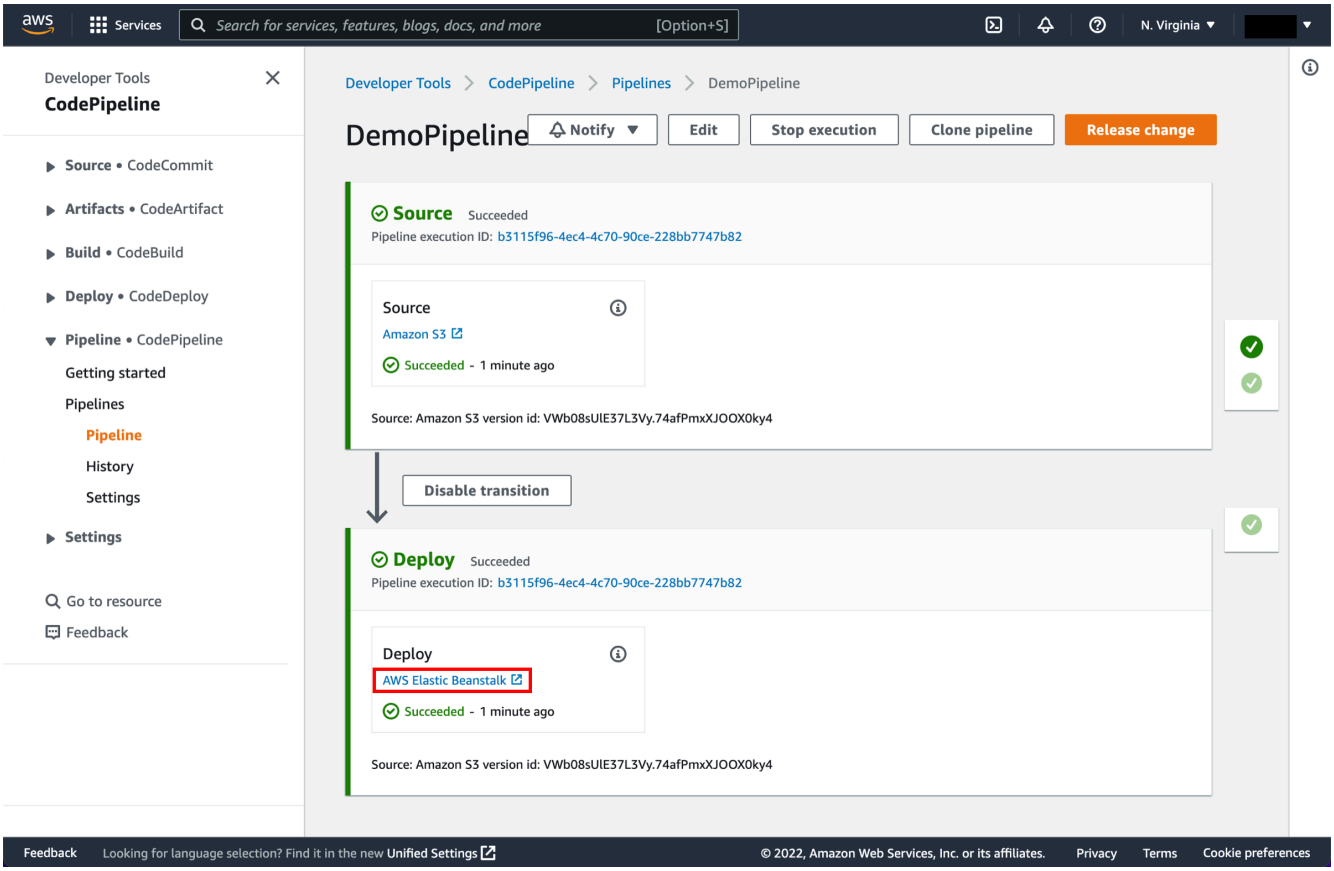
Choose the appropriate tab based on the code source you used.

GitHub

1. Choose Elastic Beanstalk

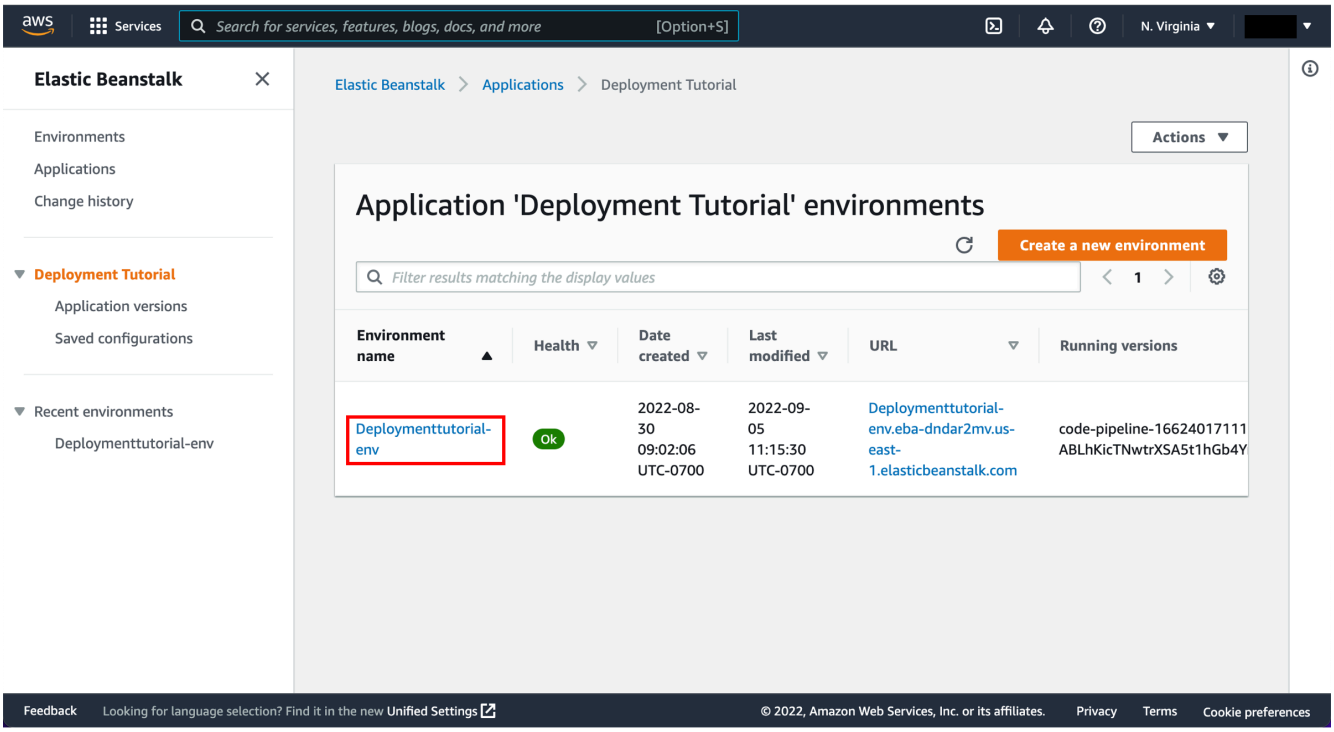
   Return to your pipeline in the CodePipeline console. In a few minutes, you should see the Source change to blue, indicating that the pipeline has detected the changes you made to your source repository. Once this occurs, it will automatically move the updated code to Elastic Beanstalk.

   After the pipeline status displays **Succeeded**, in the status area for the Beta stage, choose **AWS Elastic Beanstalk**.



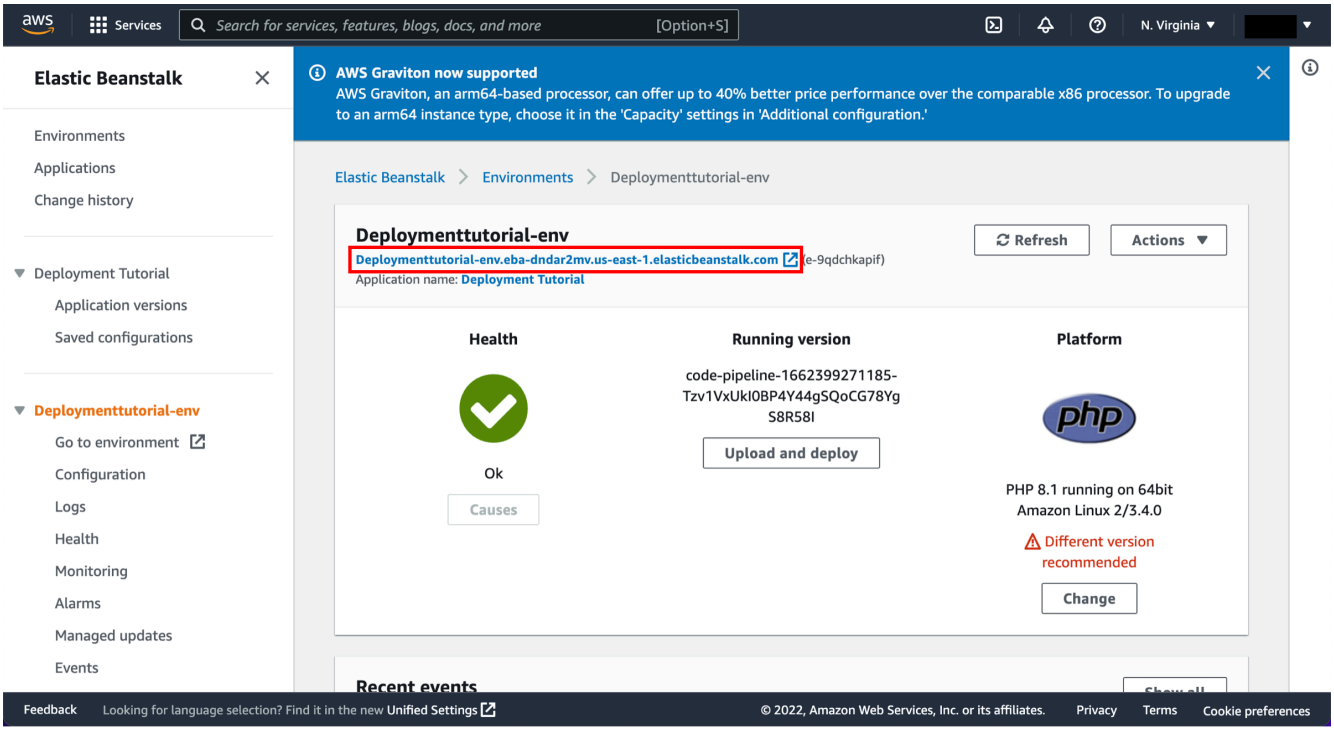
2. Select the environment

   The AWS Elastic Beanstalk console opens with the details of the deployment. Select the environment you created earlier, called **Deploymenttutorial-env**.



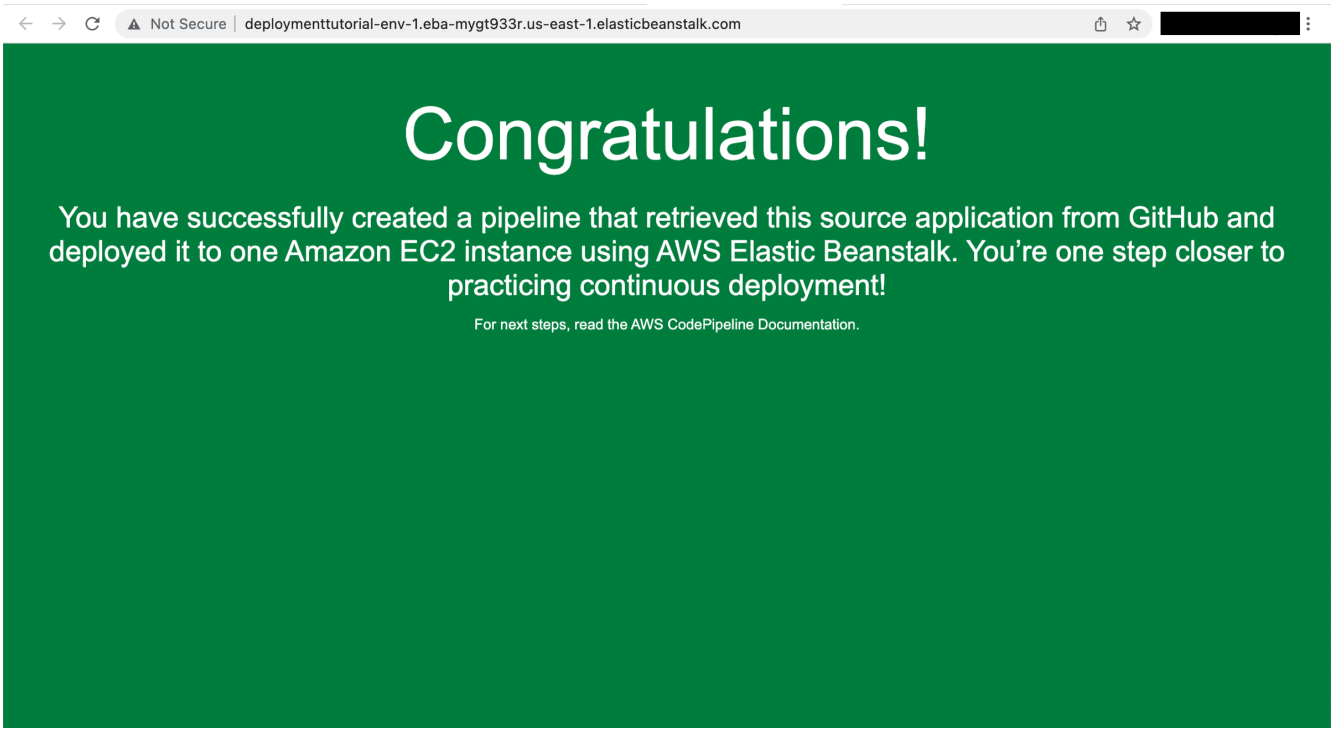
3. Select the URL

   Select the URL to view the sample website again.



4. View the page

   Confirm that the updated text appears on the webpage.

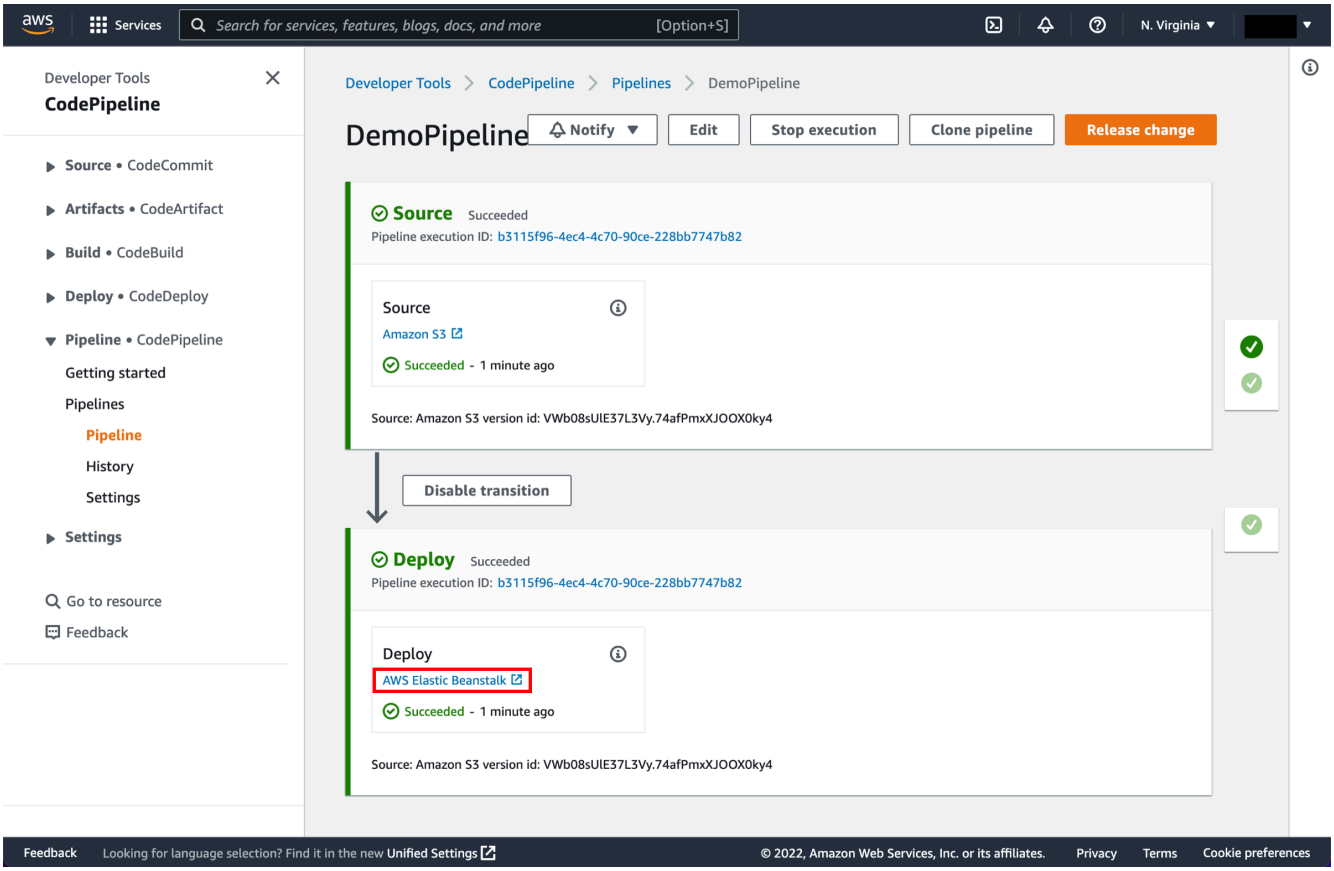


Amazon S3

1. Choose Elastic Beanstalk

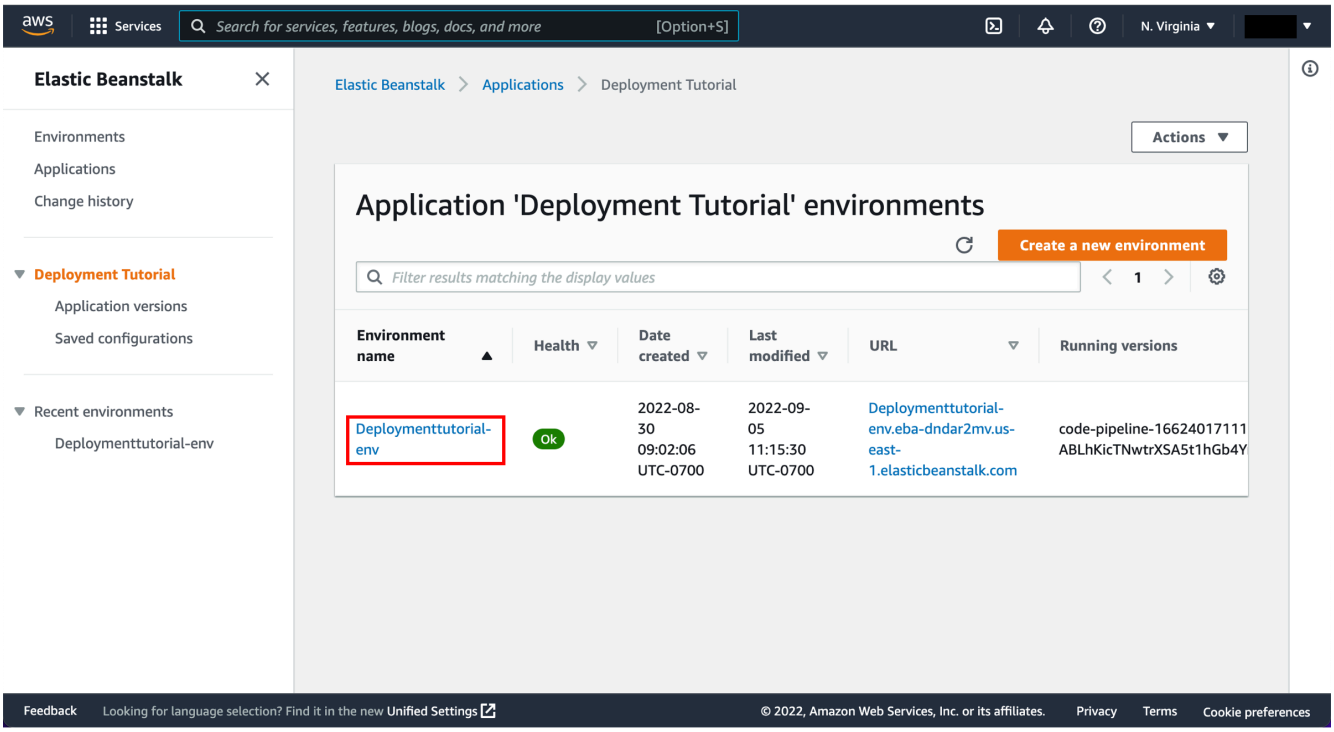
   Return to your pipeline in the CodePipeline console. In a few minutes, you should see the Source change to blue, indicating that the pipeline has detected the changes you made to your source repository. Once this occurs, it will automatically move the updated code to Elastic Beanstalk.

   After the pipeline status displays **Succeeded**, in the status area for the Beta stage, choose **AWS Elastic Beanstalk**.



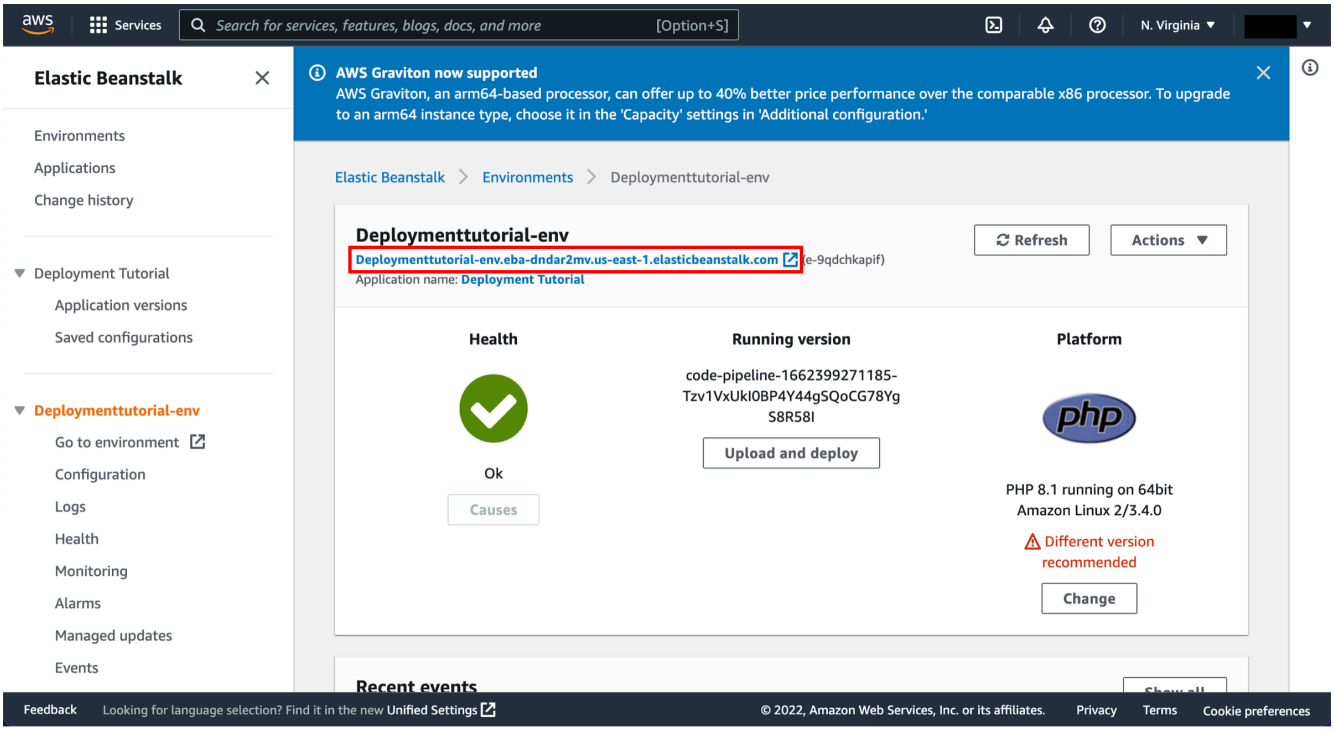
2. Select the environment

   The AWS Elastic Beanstalk console opens with the details of the deployment. Select the environment you created earlier, called **Deploymenttutorial-env**.



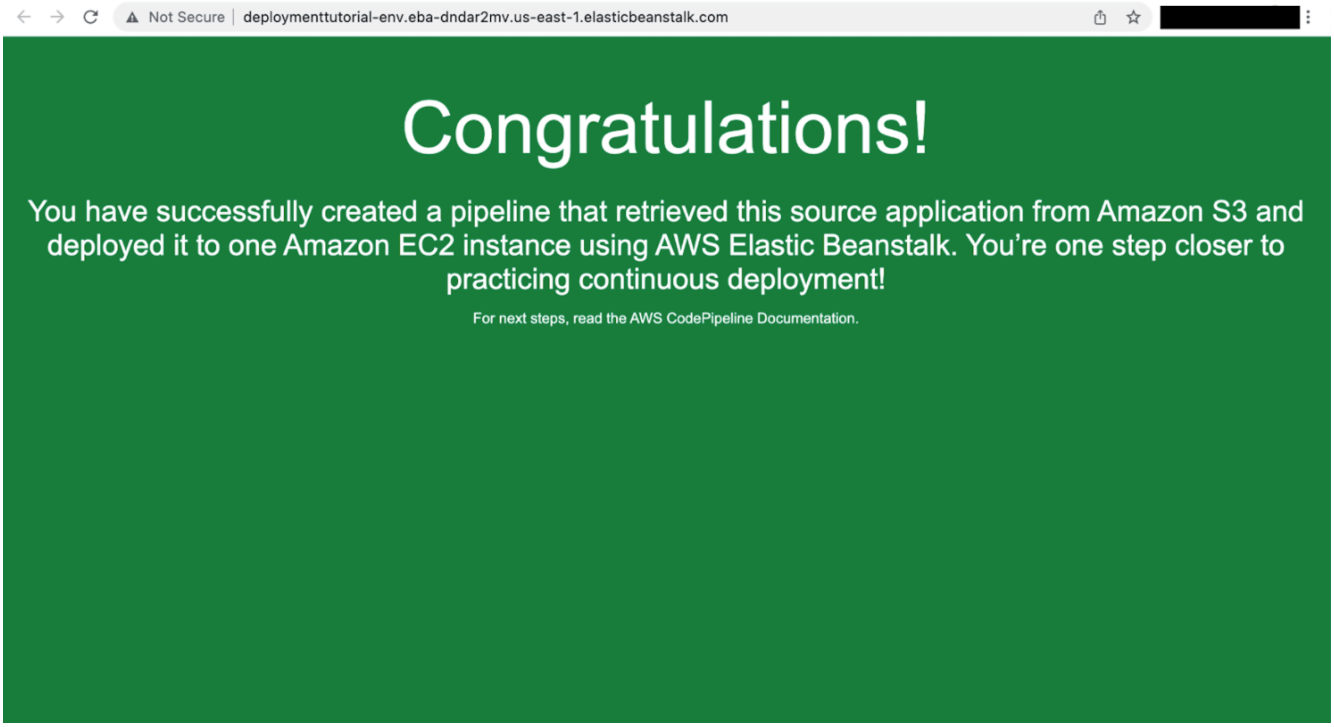
3. Select the URL

   Select the URL to view the sample website again.



4. View the page

   Confirm that the updated text appears on the webpage.

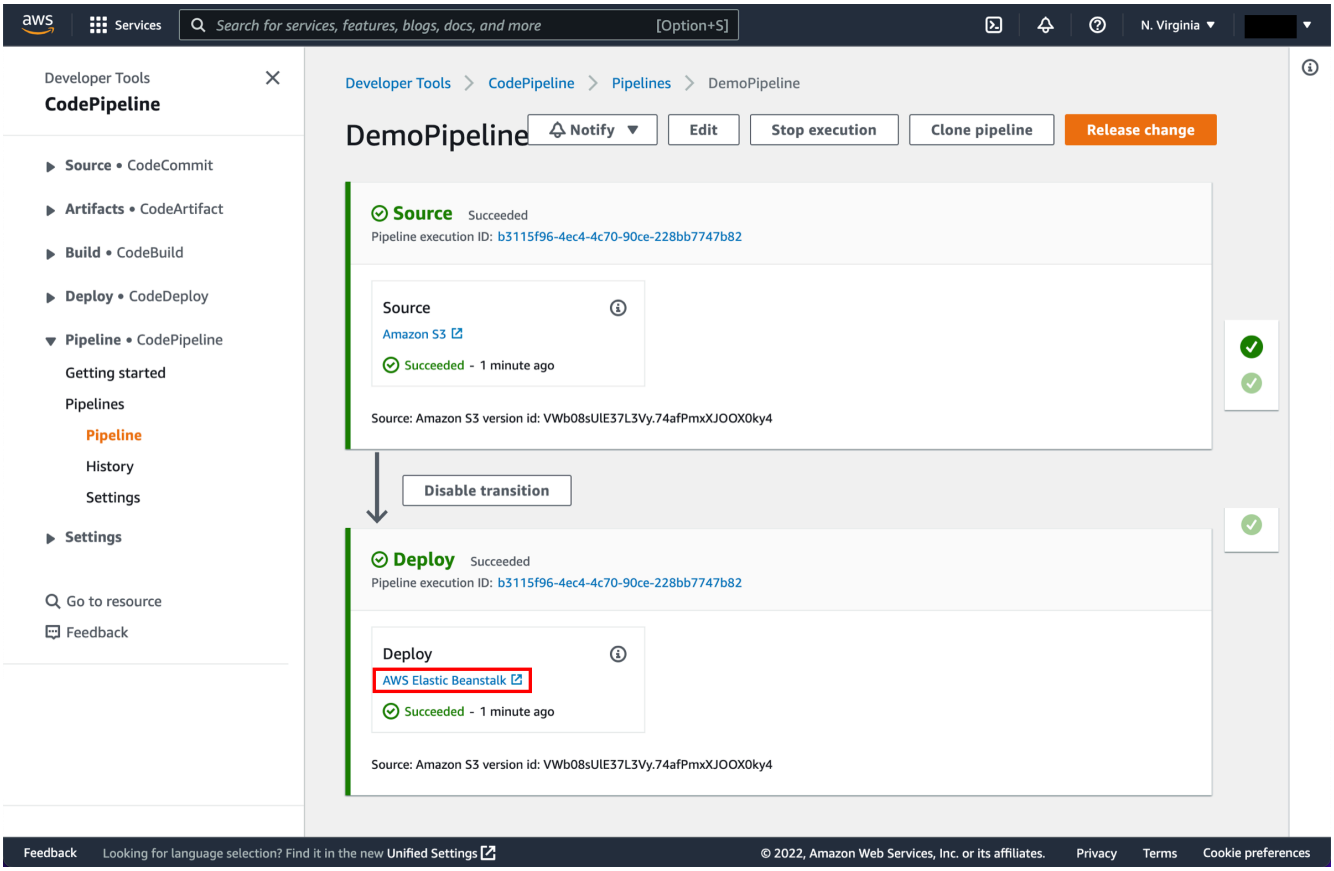


AWS CodeCommit

1. Choose Elastic Beanstalk

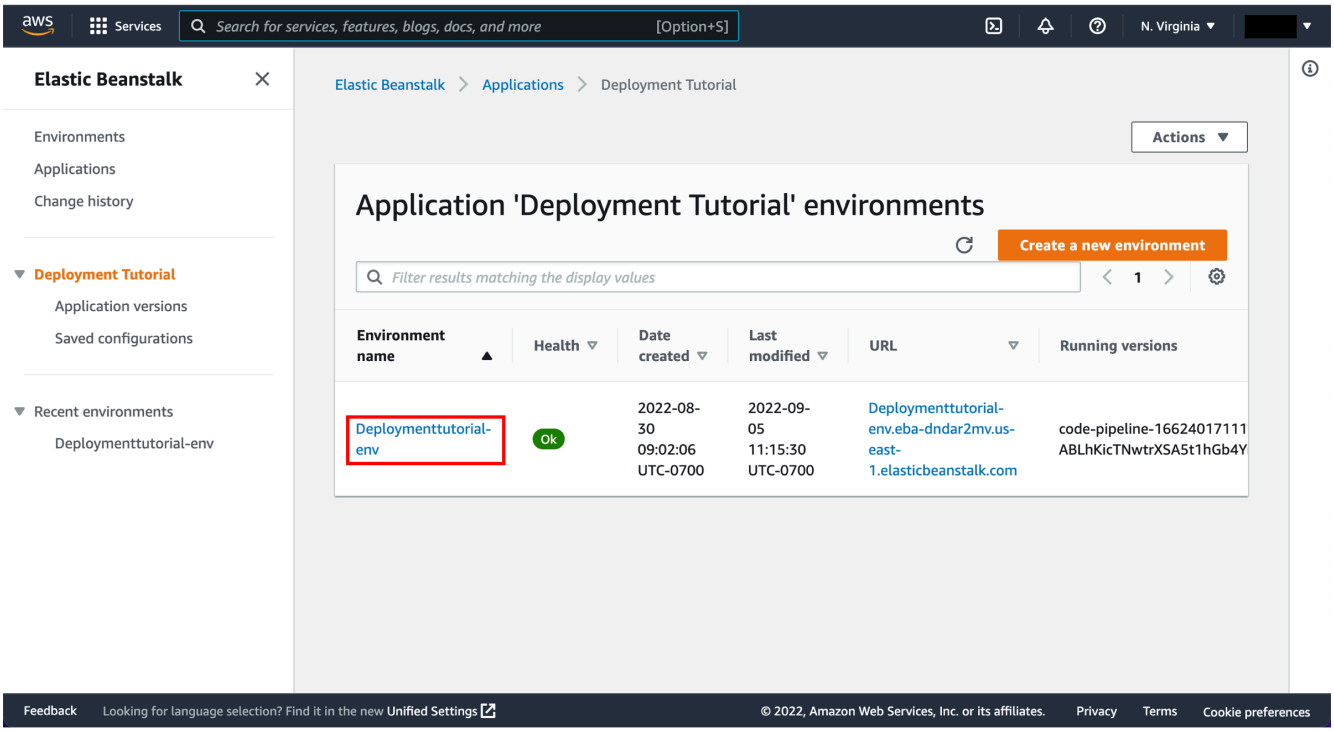
   Return to your pipeline in the CodePipeline console. In a few minutes, you should see the Source change to blue, indicating that the pipeline has detected the changes you made to your source repository. Once this occurs, it will automatically move the updated code to Elastic Beanstalk.

   After the pipeline status displays **Succeeded**, in the status area for the Beta stage, choose **AWS Elastic Beanstalk**.



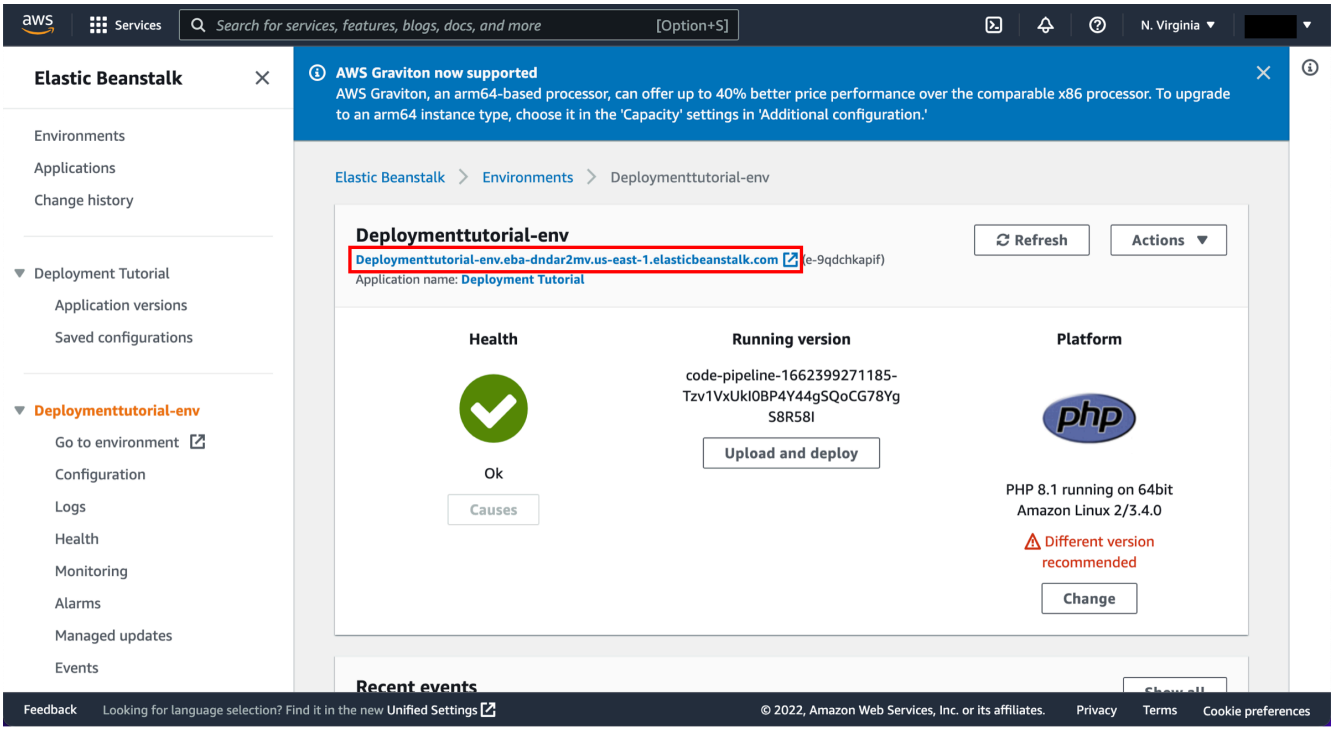
2. Select the environment

   The AWS Elastic Beanstalk console opens with the details of the deployment. Select the environment you created earlier, called **Deploymenttutorial-env**.



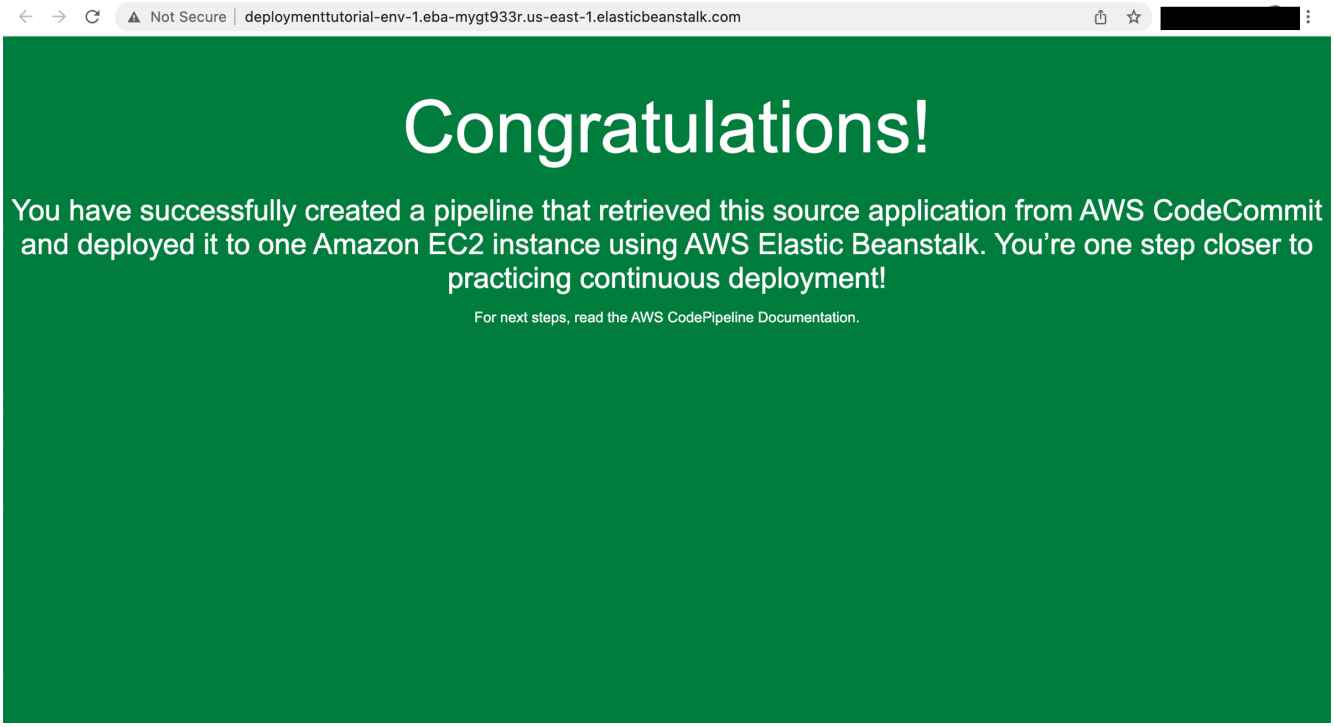
3. Select the URL

   Select the URL to view the sample website again.



4. View the page

   Confirm that the updated text appears on the webpage.

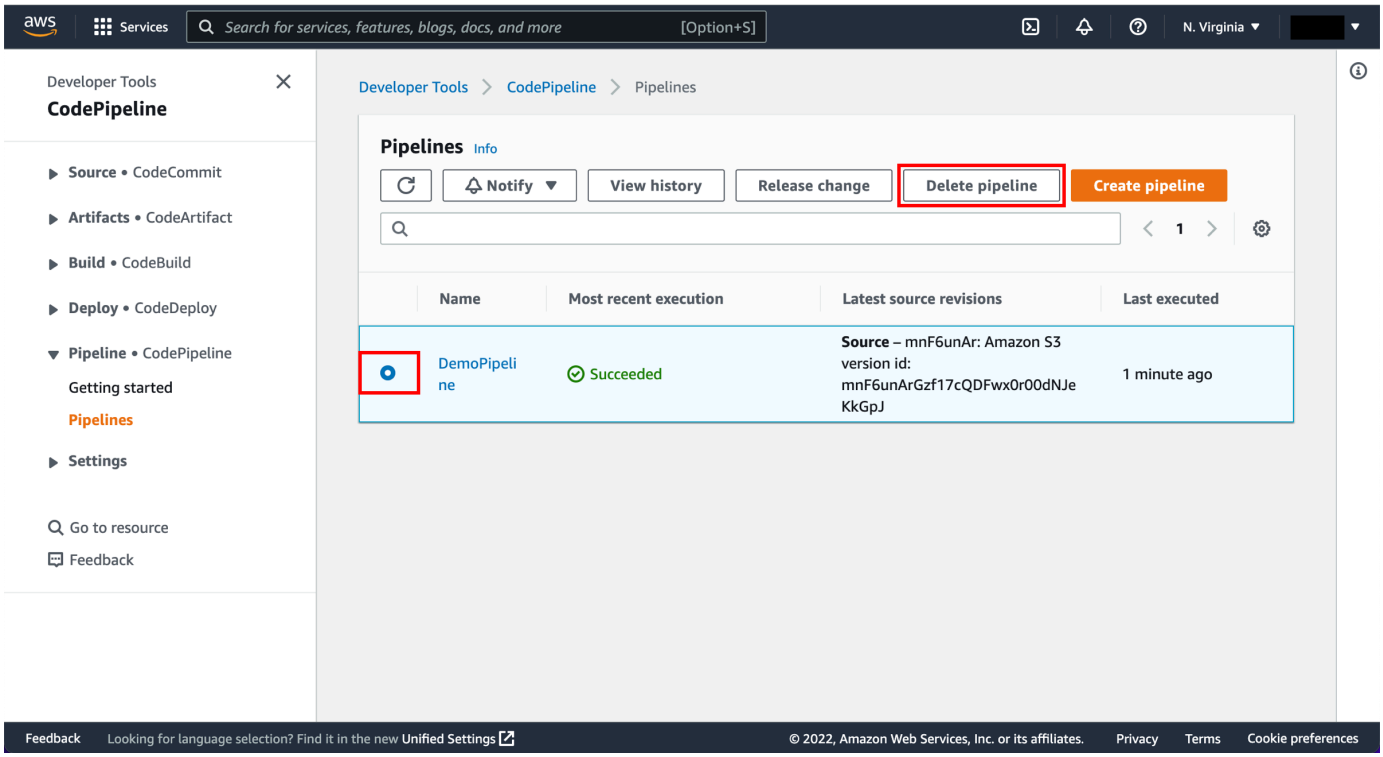


Clean up resources

To avoid future charges, you will delete all the resources you launched throughout this tutorial, which includes the pipeline, the Elastic Beanstalk application, and the source you set up to host the code.

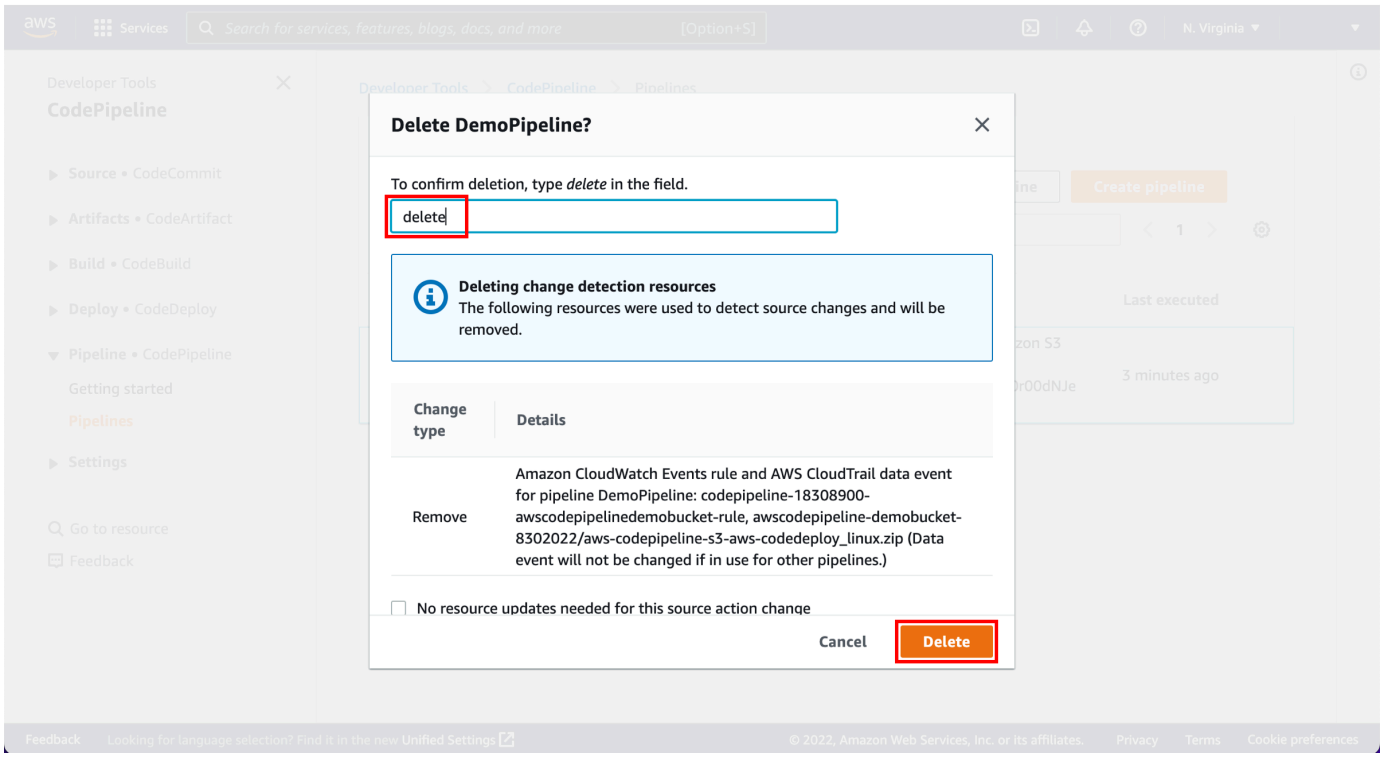
1. Delete the pipeline

   First, you will delete your pipeline. In the **Pipelines** view, select the pipeline radio button and select **Delete pipeline.**



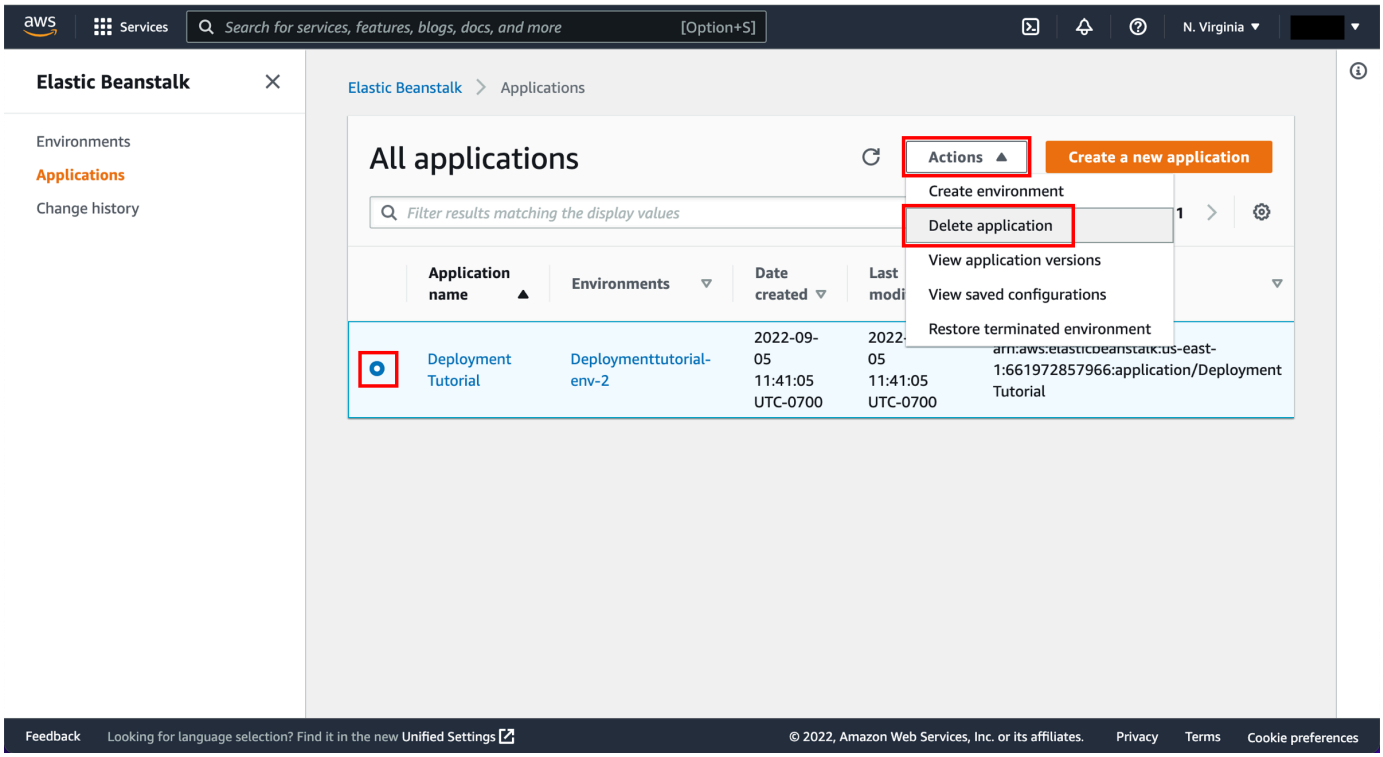
2. Confirm deletion

To confirm deletion, enter **delete** in the field and choose **Delete**.



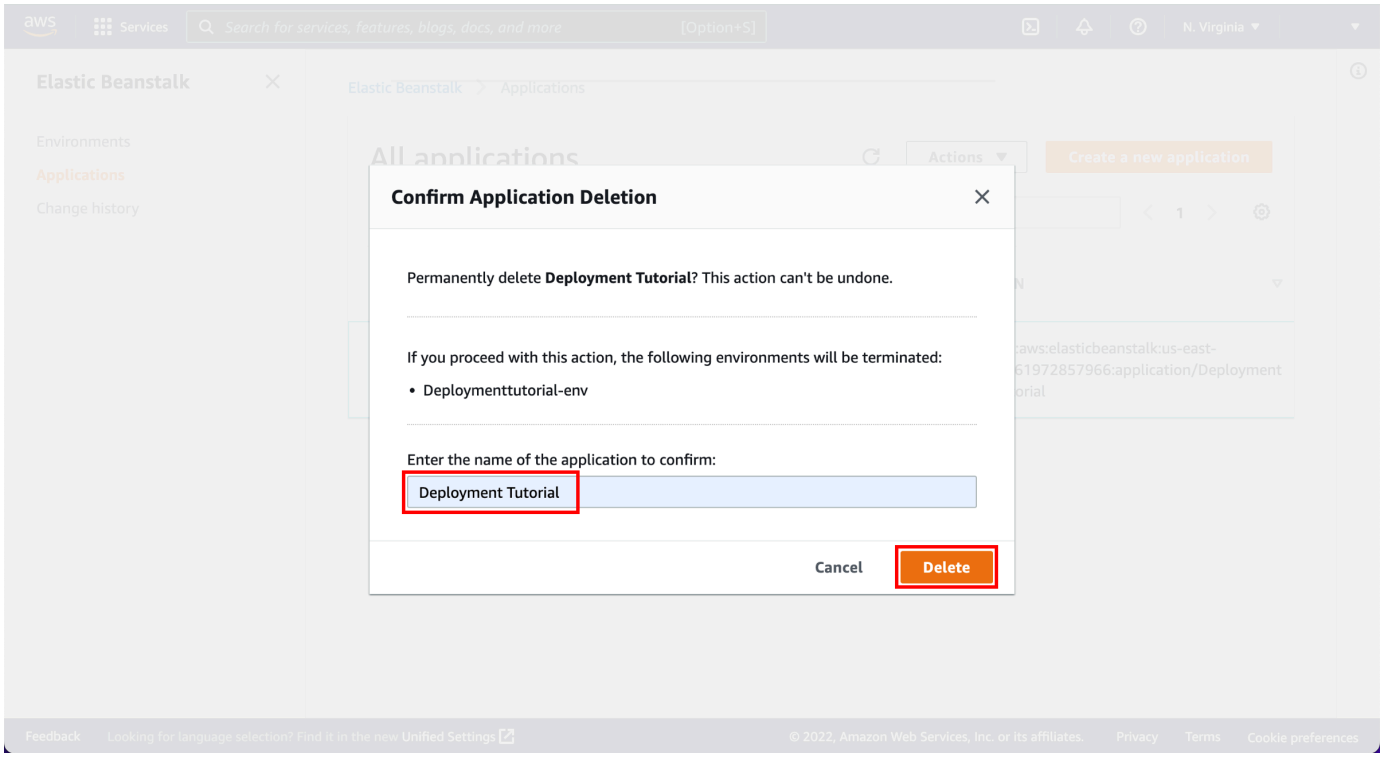
3. Delete the Beanstalk application

Second, delete your Elastic Beanstalk application. Visit the Elastic Beanstalk **Applications** page. Select the radio button for the **Deployment Tutorial.** Select **Actions** and **Delete application.**



4. Confirm deletion

   In the **Confirm Application Deletion** window, enter the name of the application to be deleted and choose **Delete**.

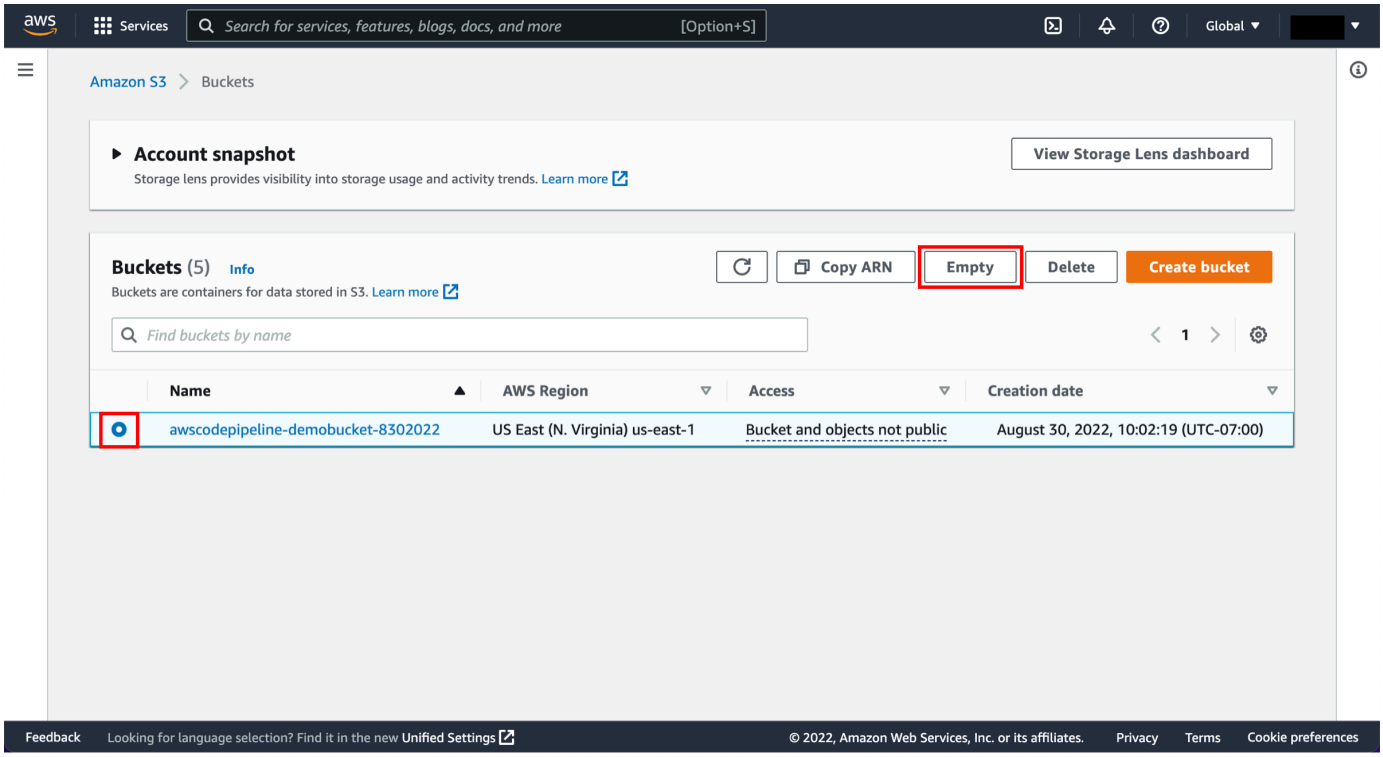


(Optional) Delete Amazon S3 resources

If you used Amazon S3 as your source, you can delete the resources to avoid future charges.

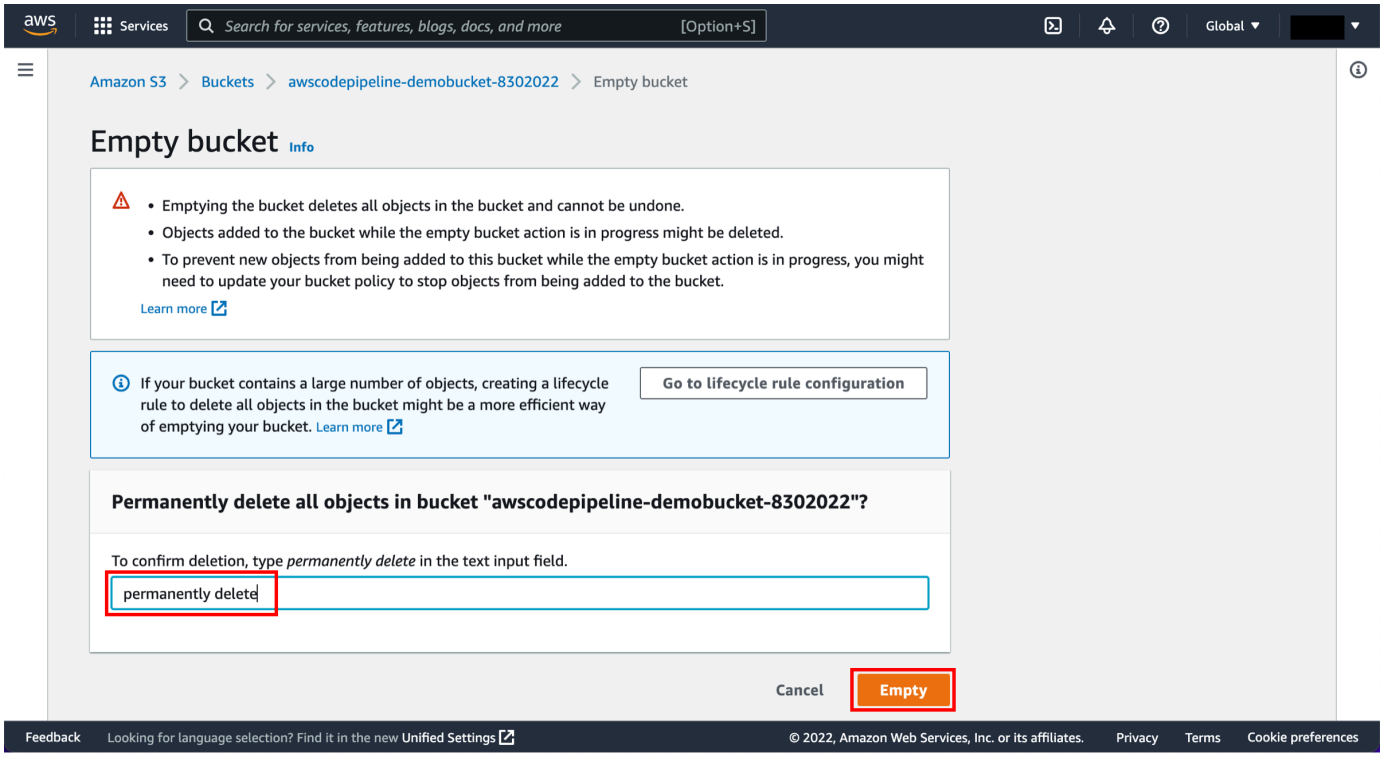
1. Empty the bucket contents

   Visit the S3 console. First, we will empty the S3 bucket. Select the radio button next to the **awscodepipeline** bucket and choose **Empty.**



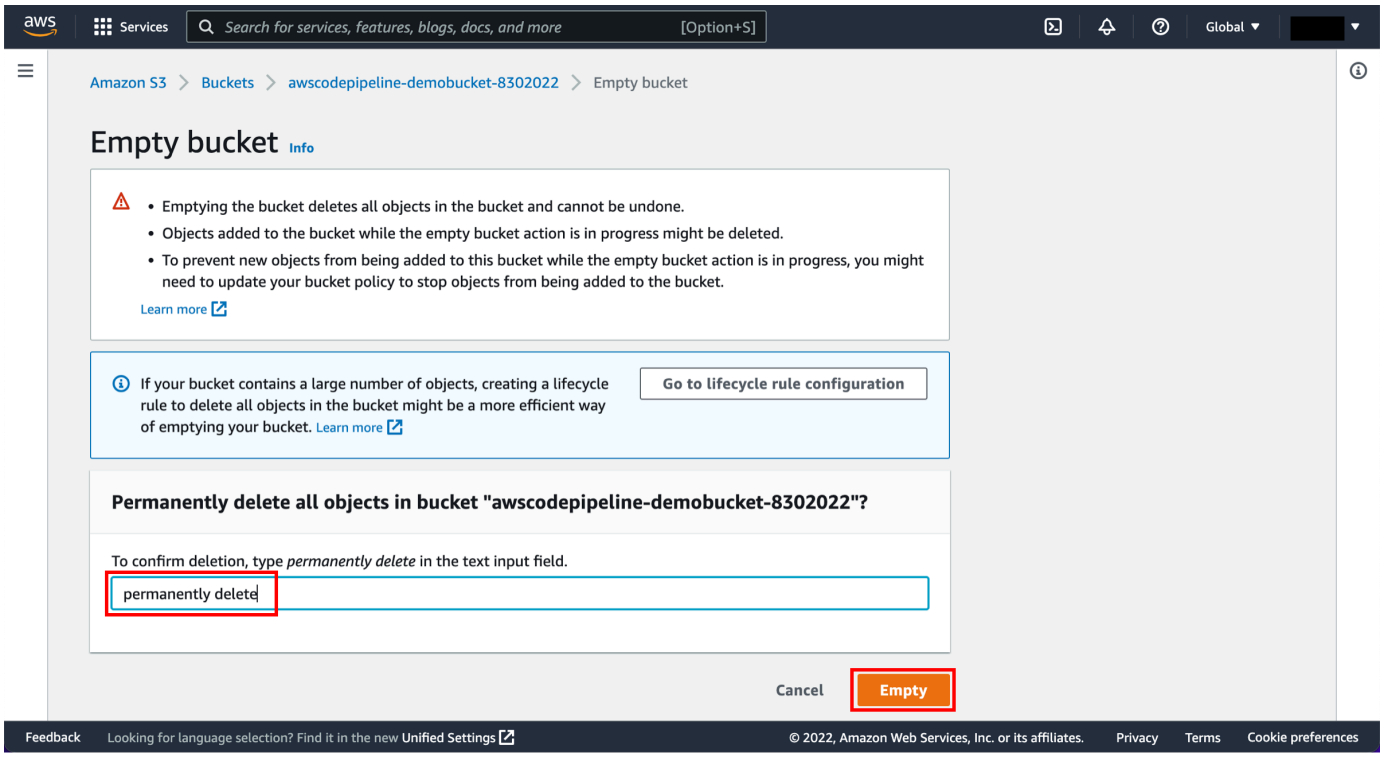
2. Confirm deletion

When a confirmation message appears, enter **permanently delete** in the text input field and choose **Empty**.



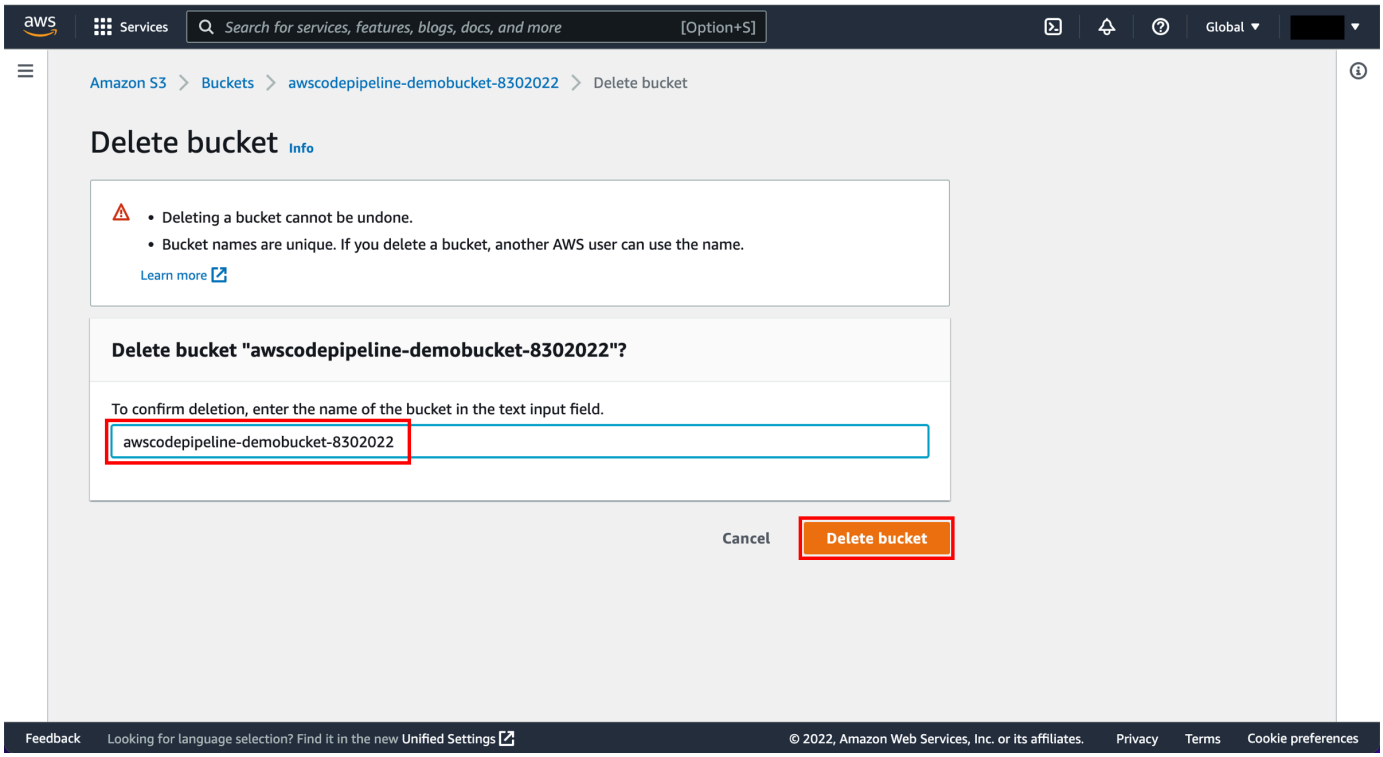
3. Delete the bucket

   Now we will delete the bucket. Select the radio button next to the **awscodepipeline** bucket and choose **Delete.**



4. Confirm deletion

   When a confirmation message appears, enter the bucket name and then choose **Delete bucket.**

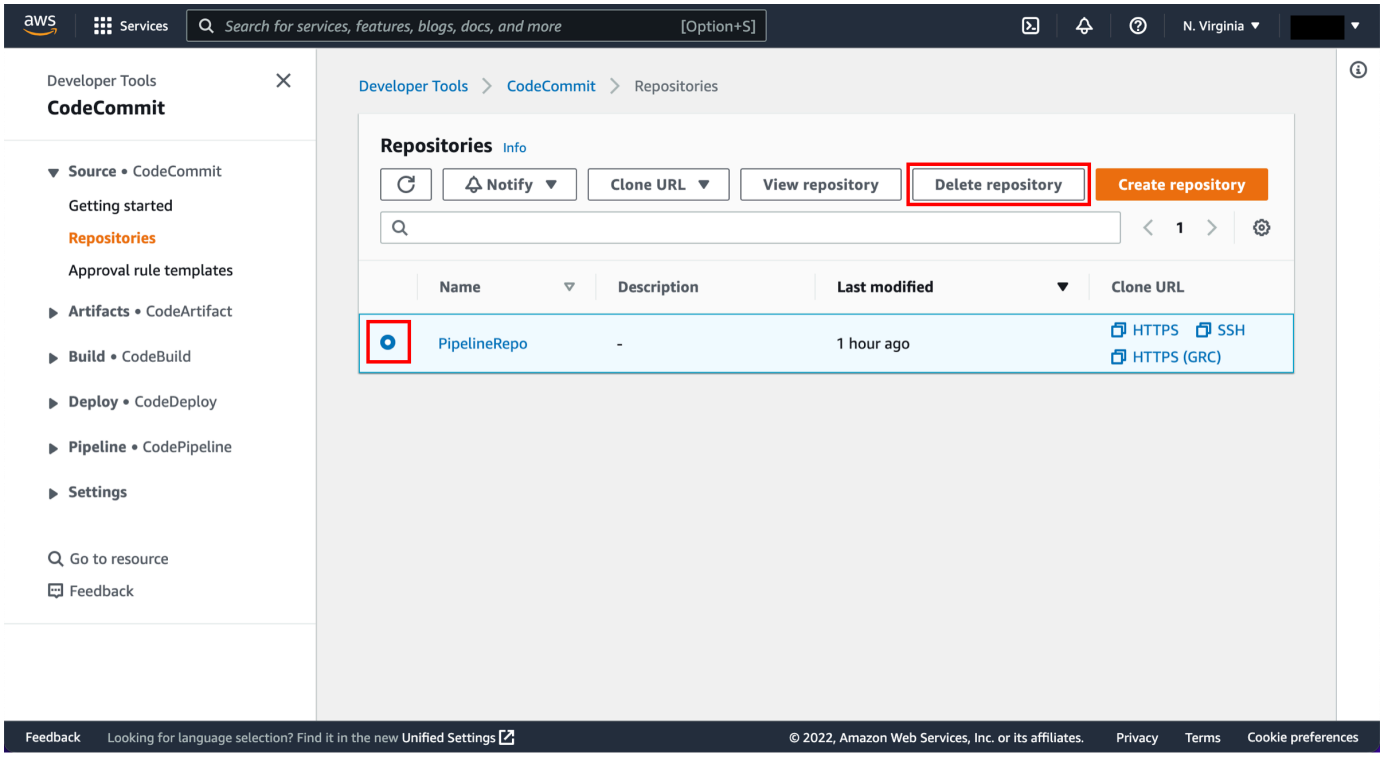


(Optional) Delete AWS CodeCommit resources

If you used AWS CodeCommit as your source, you can delete the resources to avoid future charges.

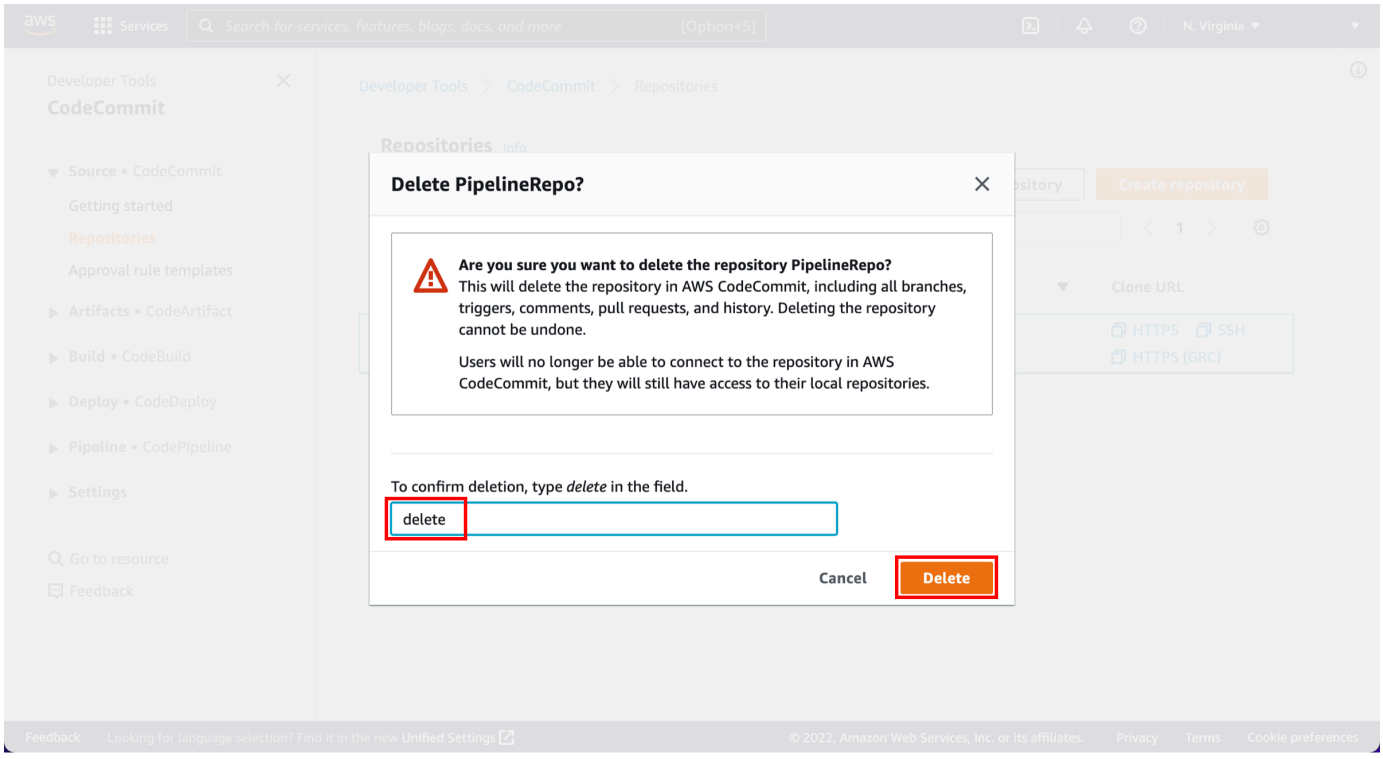
1. Delete the repository

   Open the AWS CodeCommit repository. Select the radio button next to the repository you created and choose **Delete repository.**



2. Confirm deletion

A confirmation window will pop up. Enter **delete** and choose **Delete**.



Conclusion

Congratulations! You have successfully created an automated software release pipeline using AWS CodePipeline. Using CodePipeline, you created a pipeline that uses GitHub, Amazon S3, or AWS CodeCommit as the source location for application code and then deploys the code to an Amazon EC2 instance managed by AWS Elastic Beanstalk. Your pipeline will automatically deploy your code every time there is a code change. You are one step closer to practicing continuous deployment!