PART VII

Developing Cloud Native Applications in AWS

CHAPTER 19

Developing Serverless Applications with Lambda

In this chapter, you will learn

- Lambda functions
- Lambda applications
- Lambda layers

This chapter will discuss how to develop serverless functions and applications using AWS Lambda.

AWS Lambda

AWS Lambda is a serverless compute service where you just provide your code to execute without having to provision any servers and pay only for the execution time. AWS Lambda manages the provision, scaling, and termination of servers automatically and charges only when your code is running. AWS Lambda supports Go, Node.js, Java, C#, Ruby, Python, and PowerShell to write your code for any application or backend services without the need for any underlying tasks, including code monitoring, logging, scaling, capacity provisioning, and server and operating system maintenance.

AWS Lambda can be automatically triggered based on events like data changes in an Amazon DynamoDB table, or changes to an Amazon S3 bucket, or AWS SDK API calls, or HTTP requests from the Amazon API Gateway, or data streaming data in Kinesis. You can also create your own serverless application or service composed of functions that can be triggered by events.

AWS Lambda Functions

The AWS Lambda function is an event-driven compute service that uses your code, chosen memory, timeout period, IAM role, and AWS service event to trigger the execution. The following are some of the key concepts that you need to understand before building your first serverless function and application:

- **Runtime** This allows functions in various languages to use the same execution environment. You can use the runtime provided by Lambda or build your own runtime that sits in between the Lambda service and your function code, relaying responses between the two and invocating events.
- **Event** Function uses the event, which is a JSON-format document to process, and then runtime converts it and sends it back to your function. The structure and contents of the event can be determined when invoking the function. For example, a custom event for timestamp data is as follows:

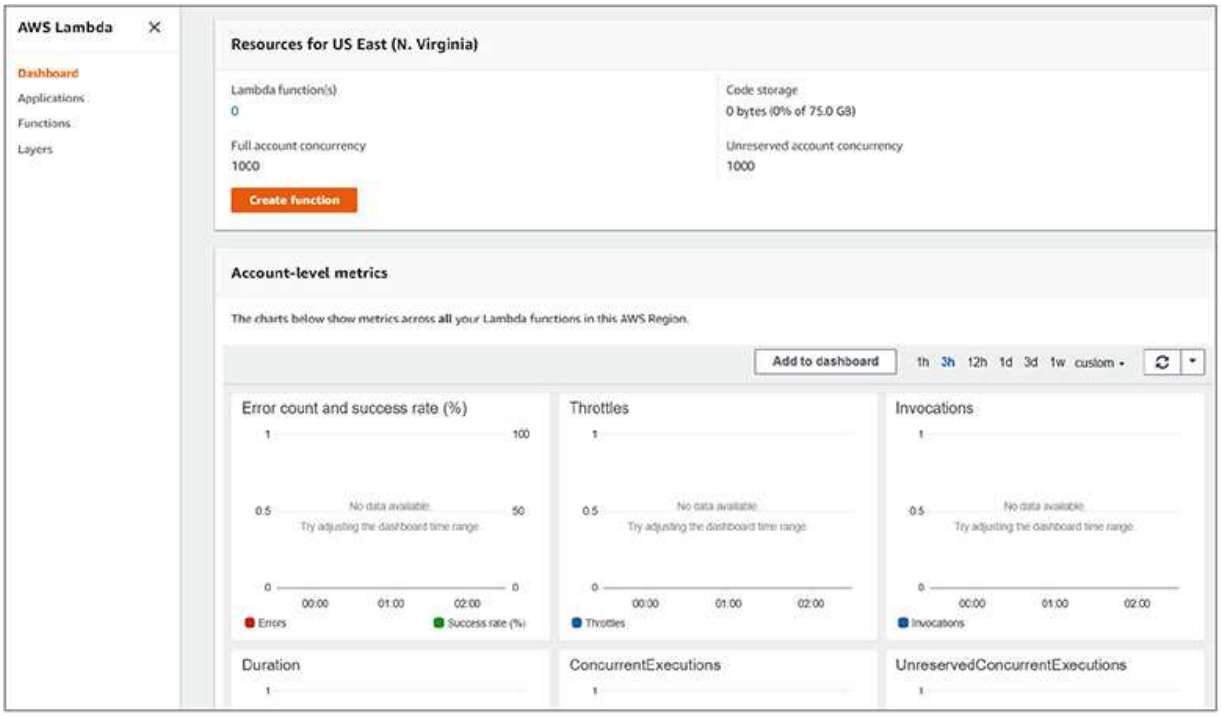
```
{
  "TimeFormat": 24,
  "TimeZone": EST,
  "Time": 15.35,
  "Year": 2020,
  "Month": 12,
  "Day": 25,
}
```

- **Concurrency** This is the number of requests that a function can serve at any given time. Lambda provisions an instance when your function is invoked to process the event. The instance can process another event request when the current function is finished; if not, then another instance is provisioned to process the concurrent request. You can configure a specific level of concurrency to limit this.
- **Trigger** This is a configuration that invokes your Lambda function, including any AWS service events, custom application events, or event source mapping, that reads from a stream or queue to invoke the function.

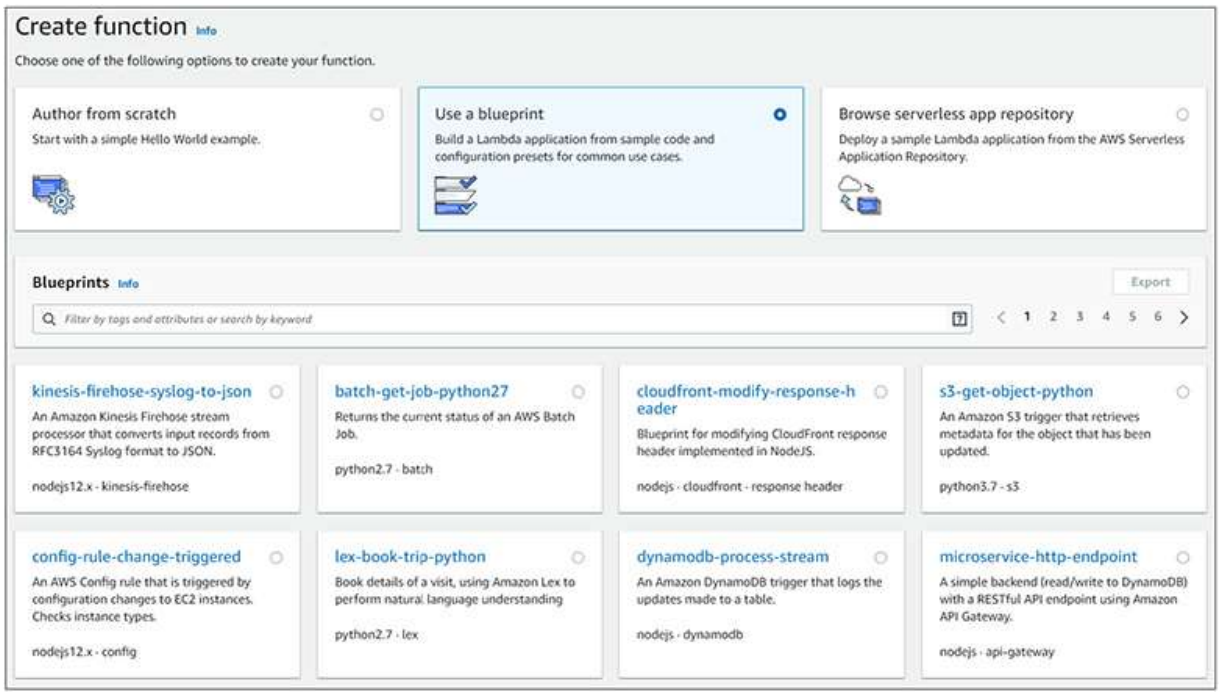
- **Versioning** You can leverage versioning to store new code and configuration of your Lambda function. It will be used along with aliases to perform rolling or blue/green deployments.
- **Scaling** Scaling is automatically handled by Lambda when your function receives a concurrent request while it's processing a request by launching another instance to handle the increased load.
- **High availability** When you create your Lambda function to connect to a Virtual Private Cloud (VPC) and specify subnets in more than one Availability Zones, Lambda runs your function in multiple Availability Zones to ensure high availability.
- **Reserved concurrency** You can reserve concurrency to handle additional requests, but you cannot exceed the specified number of concurrent invocations, which ensures that you have available concurrency when needed.
- **Retries** Lambda retries the execution automatically, with delays between each invocation triggered by AWS services and other clients.
- **Dead-letter queue** Lambda can be configured to send failed retry requests to a dead-letter queue, which can be an Amazon SQS queue or Amazon SNS topic that will be used for reprocessing or troubleshooting.

Let us create a function that logs the message pushed to the SNS topic.

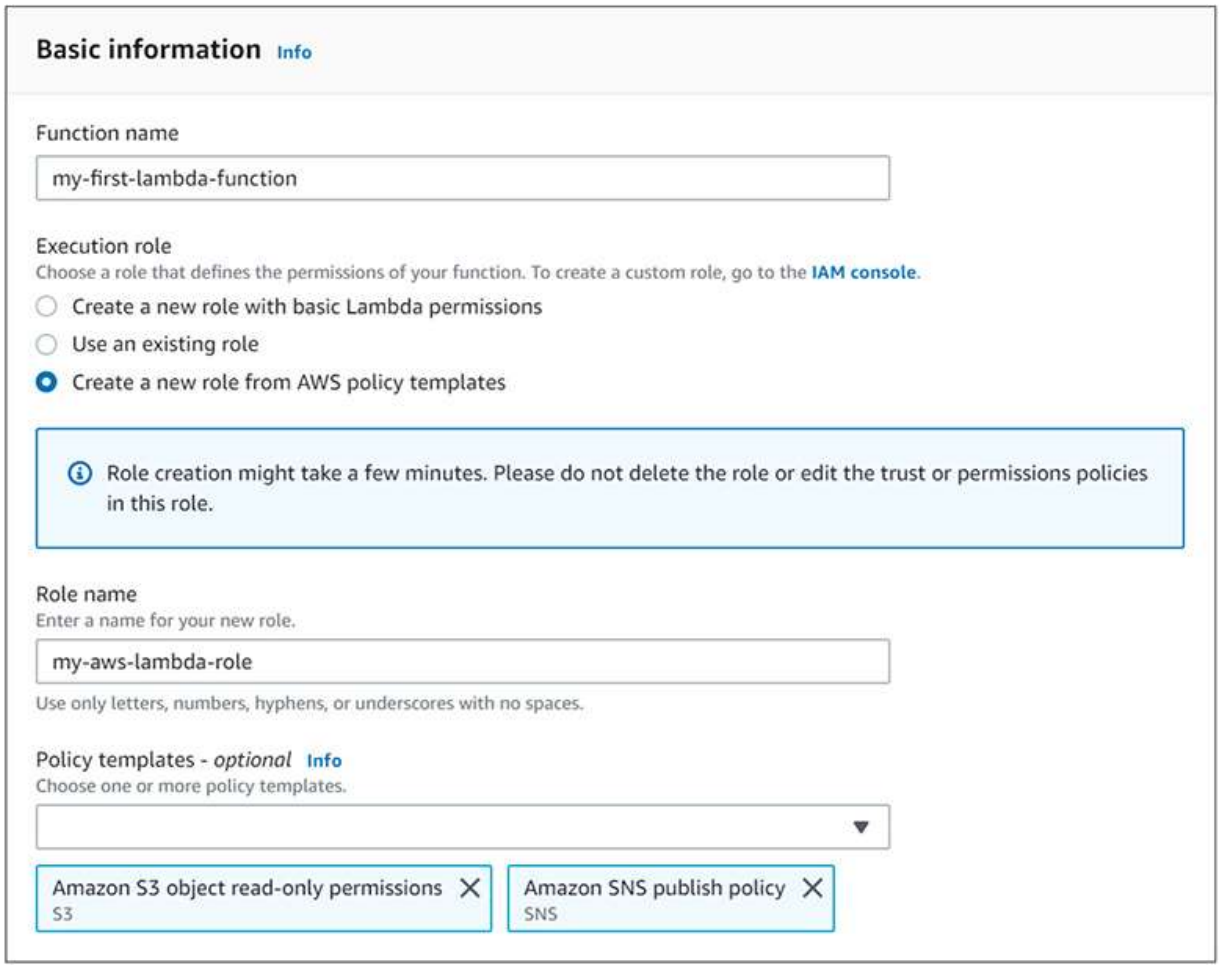
1. Log in to your AWS Management Console and select Services from top-left screen. Choose AWS Lambda. The AWS Lambda console page will appear, as shown here.



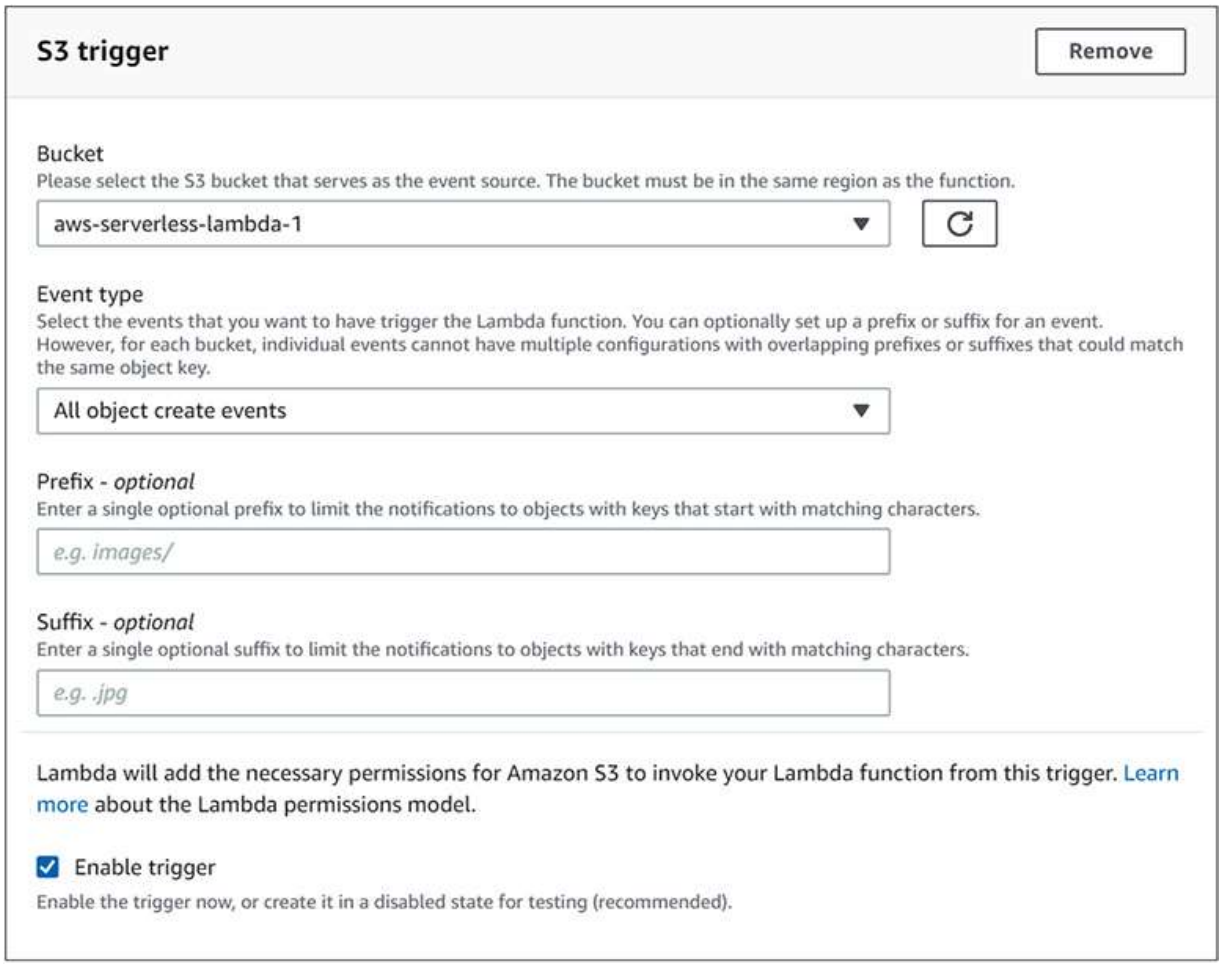
2. Click on the Create Function button that will take you to the AWS Lambda Create Function screen.



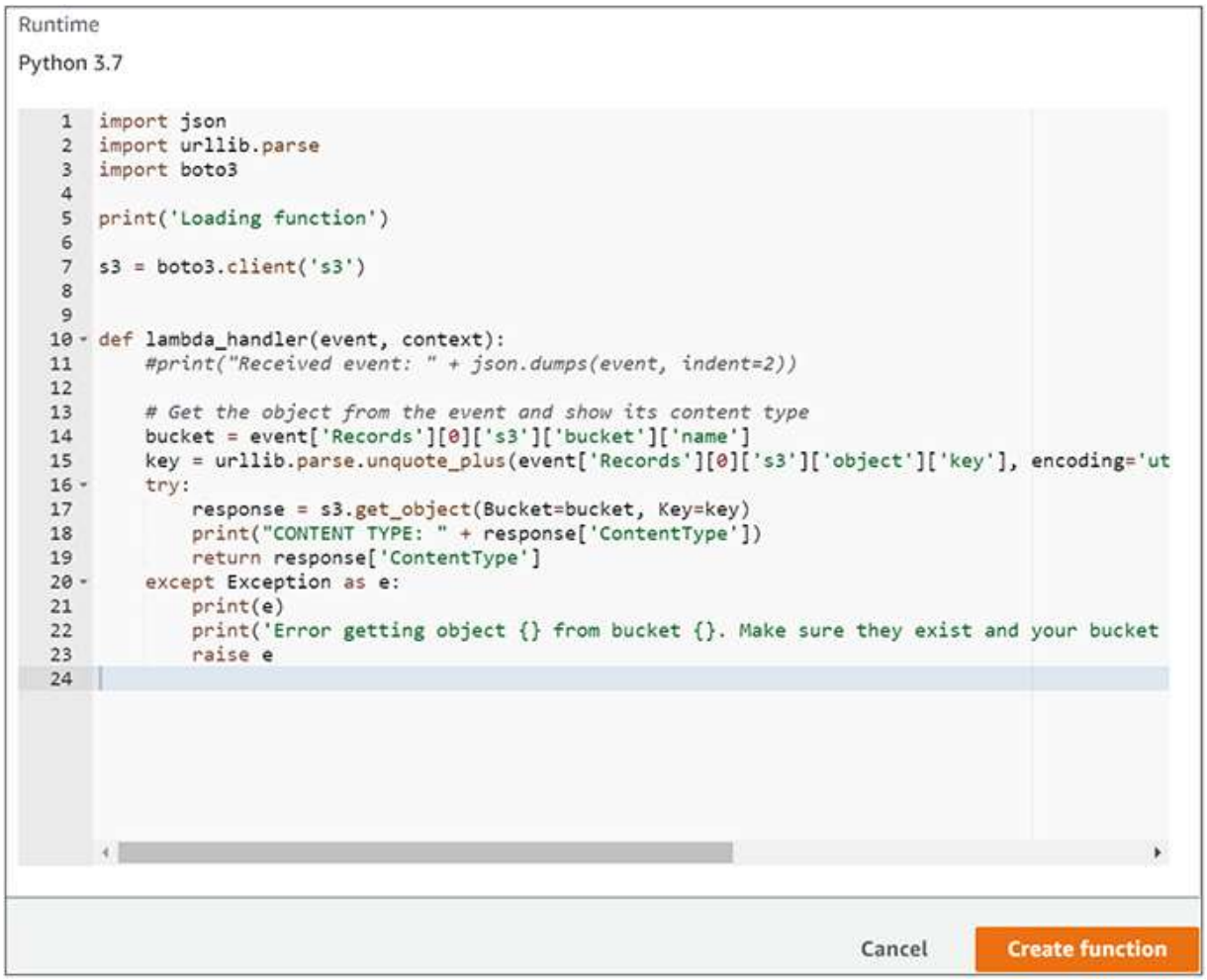
3. Now select the s3-get-object-python function from the blueprint that will take you to the Basic Information page. Here you need to enter a name for your function, choose AWS Policy Templates, and then provide a role name. Choose Policy Templates from the dropdown and choose the S3 and SNS policies.



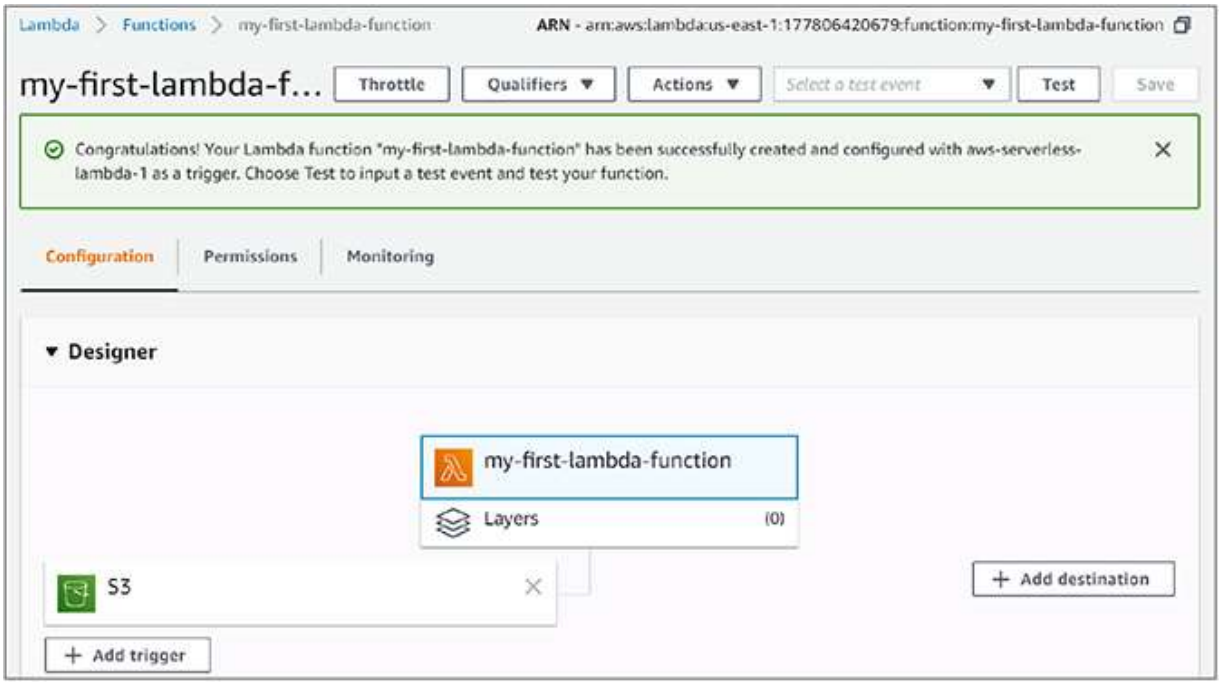
4. Now you need to configure the S3 trigger that can be run each time the defined event occurs. Choose your existing bucket from the dropdown menu, and select All Object Create Events. Provide the Prefix and Suffix as appropriate, and choose Enable Trigger.



5. The last part is Lambda function code, which is preconfigured by the blueprint and can be updated after the function is created. This function uses Python 3.7. Click on the Create Function button.



Your first AWS Lambda is successfully created, and you can explore testing different values and updating the code based on your need. This is the beginning of your exciting serverless journey. Explore many blueprint functions to quickly get started with creating and deploying AWS Lambda functions.



AWS Lambda Applications

An AWS Lambda application consists of Lambda functions, events, and triggers that work as a single package that you can deploy and manage as one resource. Lambda applications can be integrated with developer tools like the AWS SAM CLI. In addition, a collection of Lambda applications can be deployed easily with AWS CodePipeline for your projects. AWS CloudFormation, along with AWS SAM, provides a local testing platform for serverless application development by defining your application's resources and managing the application as a stack. This allows you to safely add and modify resources and roll back to the previous state of your application stack.

AWS Lambda Layers

A Lambda function can be configured to pull additional code in a ZIP archive format that contains a custom runtime, libraries, and content in the form of layers—up to five layers at a time. Custom layers, AWS, or third-party AWS customer published layers can be used like libraries in your function without including them in your deployment package. Resource-based policies can be used to grant layer usage permissions to specific AWS accounts or AWS organizations. The runtime uses libraries in a different location, under /opt,

where layers are extracted in the function execution environment depending on the language. AWS SAM can be used to manage layers and its configurations.

AWS Lambda Security

AWS Lambda follows the AWS shared responsibility model, including compliance and regulations for data protection. AWS recommends using multifactor authentication (MFA) and SSL/TLS to communicate, capturing all user activity logging with AWS CloudTrail. AWS strongly recommends not using any sensitive identifying information in function names and freeform tags, since the metadata might get picked up in diagnostic logs, and never include external URL credential information.

All Lambda communication is encrypted with TLS, and the Lambda API endpoint supports only HTTPS secure connections. The environment variables can be used to store secrets securely because they are encrypted at rest. Environment variable values can be encrypted on the client side from the Lambda console before sending them to Lambda, which prevents secrets from being displayed unencrypted in the Lambda console or in the function configuration that's returned by the Lambda API. You can use customer-managed key to encrypt data in Amazon CloudWatch logs and AWS X-Ray, where the data is encrypted by default using the AWS-managed keys. All files that you upload are encrypted by default in Lambda, including deployment packages and layer archives.

Chapter Review

This chapter began by explaining AWS Lambda, which is a serverless compute service where you execute your code without provisioning any servers and pay only for the execution time. It manages the provision, scaling, and termination of servers and supports Go, C#, Ruby, Python, Node.js, Java, and PowerShell to write your code for serverless functions and applications. AWS Lambda handles all the administration of underlying tasks, including code monitoring, logging, scaling, capacity provisioning, and server and operating system maintenance. The AWS Lambda function is an event-driven compute service that uses your code, chosen memory, timeout period, IAM role, and AWS service event to trigger the execution. You can

use the runtime provided by Lambda or build your own runtime that sits in between the Lambda service and your function code, relaying responses between the two and invocating events.

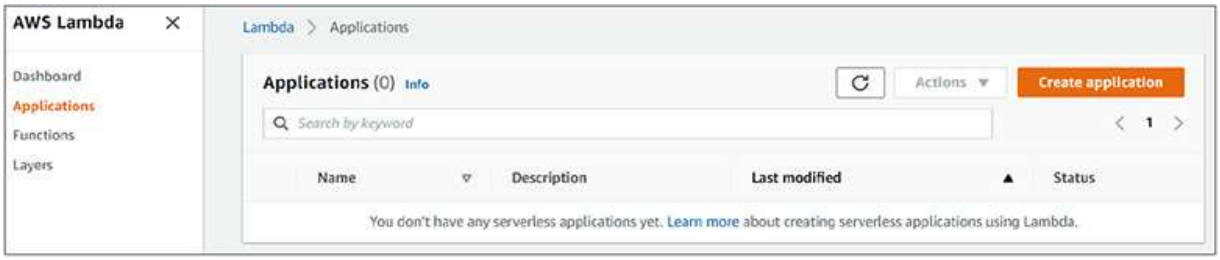
AWS Lambda applications can be created from the AWS Management Console using the AWS SAM CLI, AWS CodeBuild, or AWS CodePipeline. An AWS Lambda application is a collection of Lambda applications that can be deployed easily with AWS CodePipeline for your projects. AWS CloudFormation, along with AWS SAM, provides a local testing platform for serverless application development by defining your application's resources and managing the application as a stack. A Lambda function can be configured with custom layers, AWS, or third-party AWS customer published layers, which can be used like libraries in your function without including them in your deployment package. Resource-based policies can be used to grant layer usage permissions to specific AWS accounts or AWS organizations. The runtime uses libraries in a different location, under /opt, where layers are extracted in the function execution environment depending on the language.

Exercise

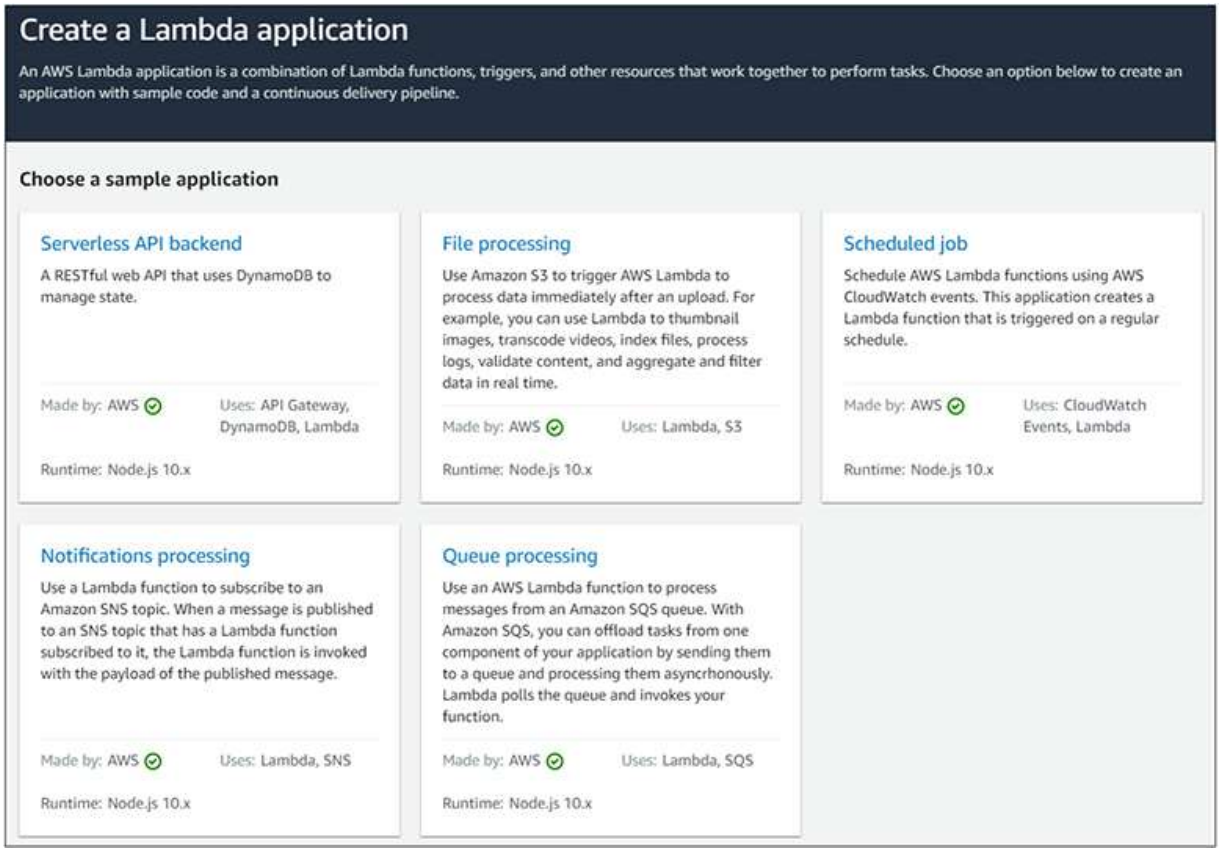
The following exercise will help you practice creating an AWS Lambda serverless application using development tools. You need to create an AWS account, as explained earlier, before performing the exercises. You can use the Free Tier when launching AWS resources, but make sure to terminate them at the end.

Exercise 19-1: Create a AWS Lambda Serverless Application Using the AWS Management Console

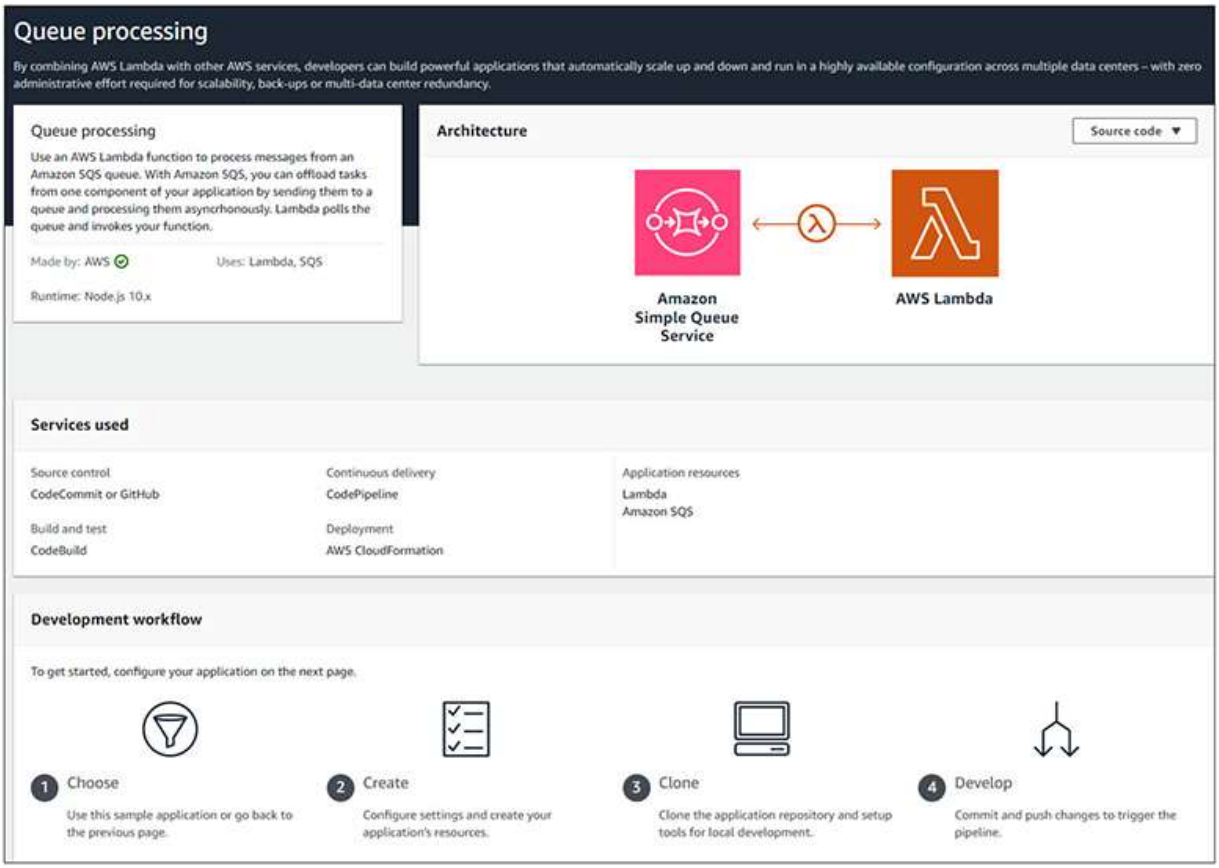
1. Use your AWS account e-mail address and password to sign in and then navigate to the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Applications and click on the Create Application button.



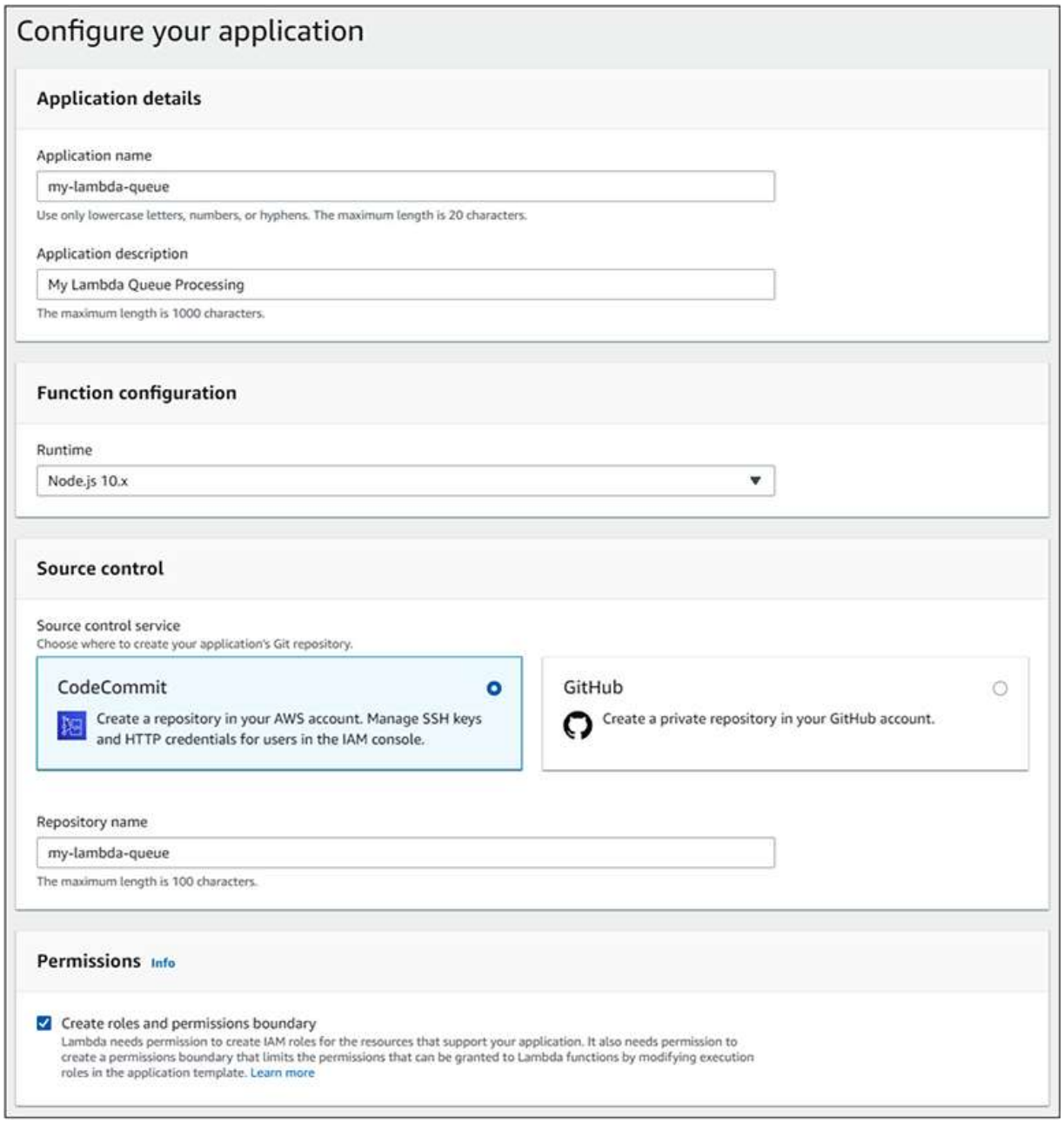
4. The Create A Lambda application page has a few sample applications. I encourage you to create and test a few sample AWS Lambda applications before your exam. In this exercise, I chose Queue Processing, which uses Lambda to process messages from your Amazon SQS queue.



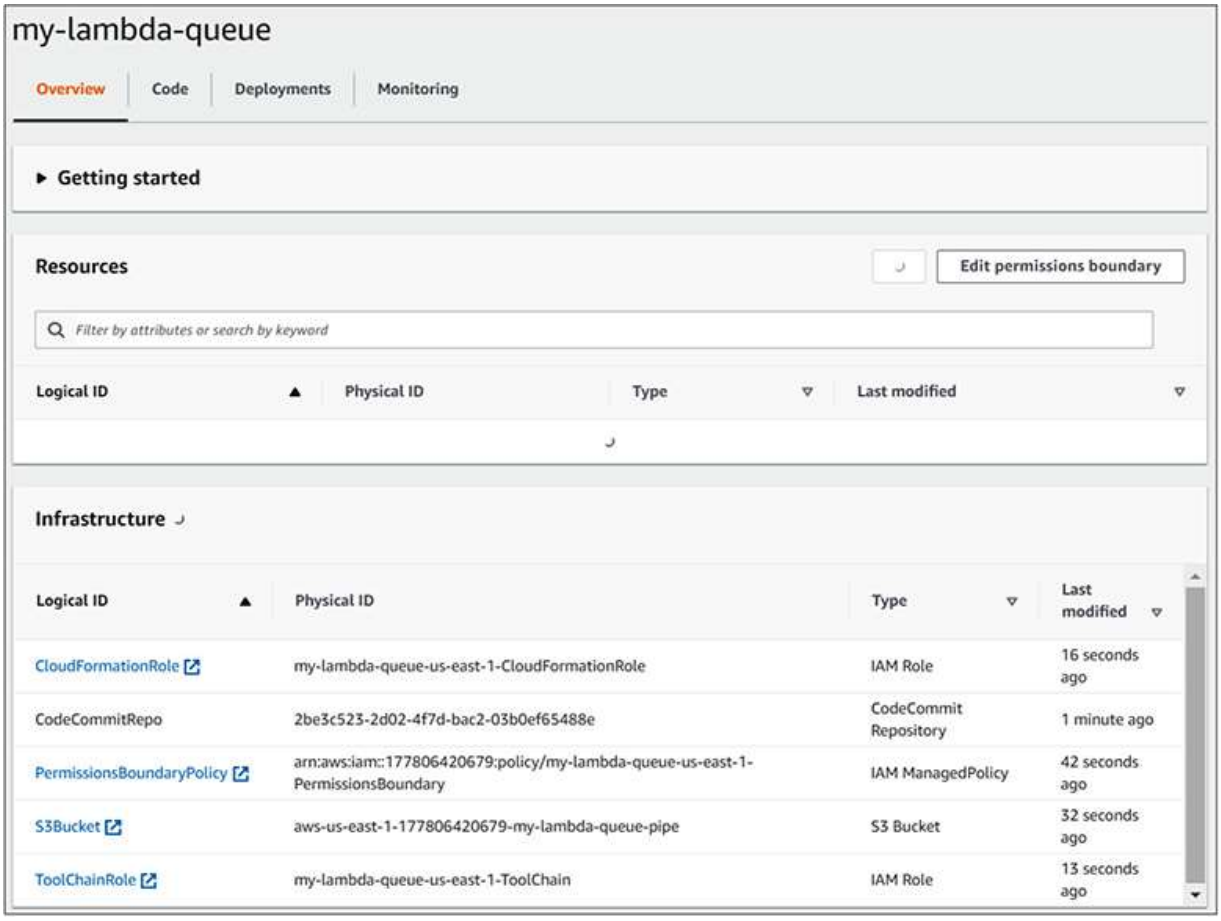
5. This sample AWS Lambda application uses Node.js and CodeCommit for source control. It uses CodeBuild for build and test and CodePipeline for continuous delivery. AWS CloudFormation is used in the background to deploy as an application template stack.



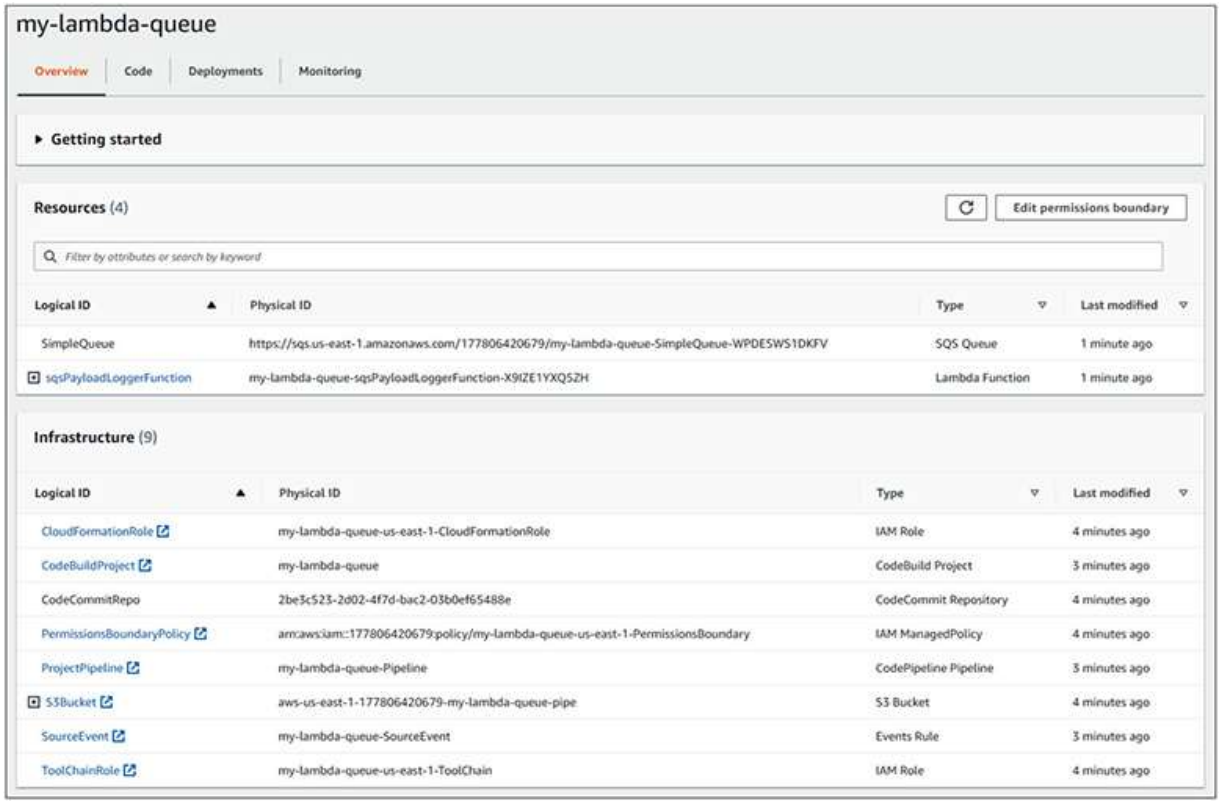
6. In the Configure Your Application screen, enter **my-lambda-queue** for the application name and provide a description. For Runtime, choose Node.js and choose CodeCommit for source control. For the Repository Name, type **my-lambda-queue**. Also select Permissions to create an appropriate role and execute this AWS serverless application.



7. When you click on the Create button, AWS Lambda starts provisioning all the resources you will need for this serverless application and shows you the progress.

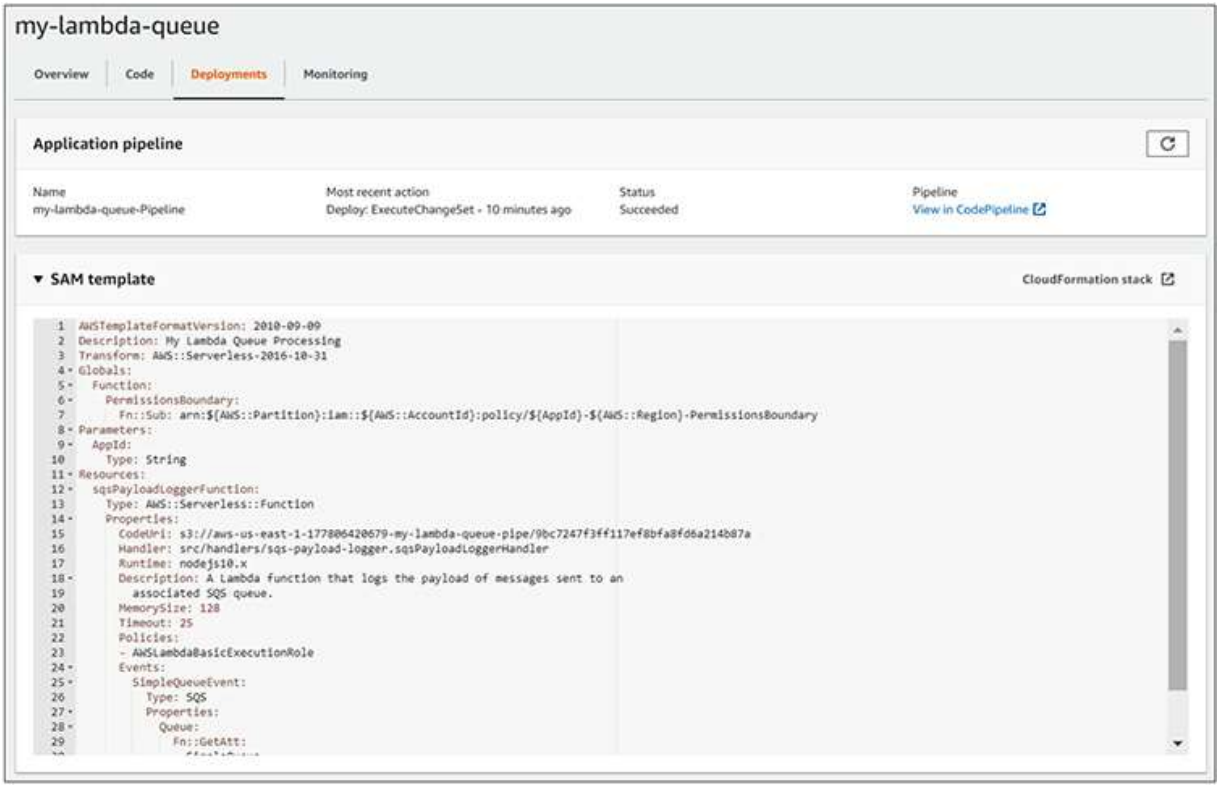


8. In a few minutes, all the resources will be provisioned and the Application Created message appears.

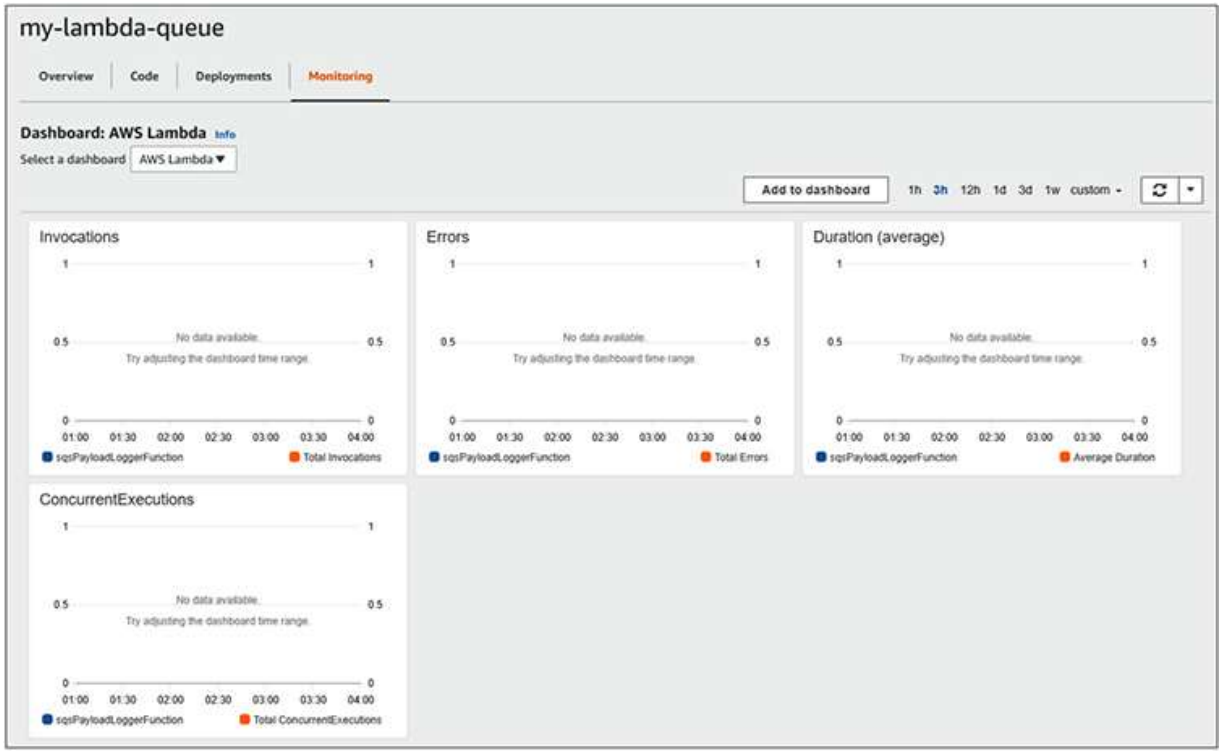


9. From the previous illustration, you can observe that four resources are created in addition to nine infrastructures for you, as shown at the bottom. On the Code tab, you can see the CodeCommit repository and option to clone the URL and SSH.

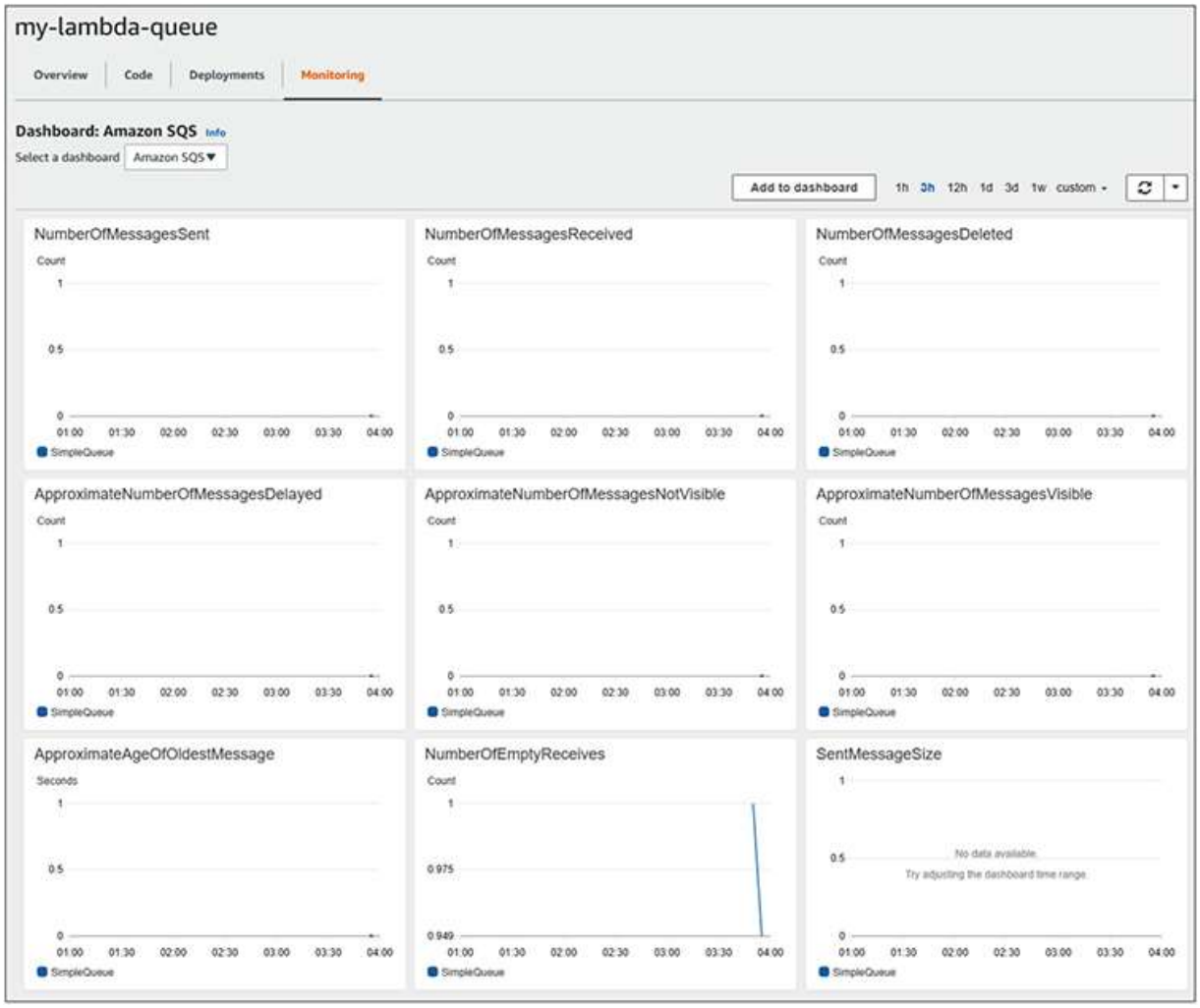
10. Now navigate to the Deployments tab, where you can find the CodePipeline application pipeline and its status, along with the SAM template that was used to create this serverless application.



11. Now navigate to the Monitoring tab where you can see a couple of Dashboards, one for AWS Lambda, which has invocations, errors, duration, and concurrent executions as different charts.



12. From the dropdown, choose Amazon SQS, which has charts for messages sent, received, deleted, visible, not visible, delayed, etc. You can create a queue and start seeing this serverless application in action, but do not forget to delete this once all the resources are completed.



Questions

The following questions will help you gauge your understanding of the contents in this chapter. Read all the answers carefully because there might be more than one correct answer. Choose the best response for each question.

1. You have batch jobs that run every day and use a dedicated server on-premises. Your manager asked you to explore a AWS service that can be leveraged to replace the batch server and run all your jobs without provisioning any server in your cloud environment. Which of the following services satisfies your requirement?

   A. Amazon S3

   B. AWS Lambda

   C. Amazon EC2

D. Amazon Athena

2. Your company asks you to stop and start your lower environment AWS EC2 instances and RDS databases during nonwork hours to save costs. Which AWS service can you leverage to create jobs that stop and start your instances and databases?

   A. Amazon SageMaker

   B. Amazon Kinesis

   C. AWS Amplify

   D. AWS Lambda

3. Which of the following languages are supported in AWS Lambda? (Choose all that apply.)

   A. Go

   B. Node.js

   C. Java

   D. C#

4. AWS Lambda provisioned an instance to execute your code, and before it finishes, you trigger the Lambda function again. What will happen in this scenario?

   A. Lambda will wait for the process to complete

   B. Lambda will fail

   C. Lambda will send a warning message

   D. Lambda will provision another instance to the handle additional request

5. What is the easiest way to develop, test locally, and deploy serverless applications in the AWS environment?

   A. Upload your code to Amazon S3

   B. Use the AWS Serverless Application Model (SAM)

   C. Provision an EC2 instance and develop your serverless application

   D. You can use the AWS Fargate service

6. Your serverless application is having issues, and you need to troubleshoot by tracing. Which AWS service will help you in this

scenario?

A. AWS X-Ray

B. AWS Batch

C. AWS Config

D. Amazon CodeCommit

7. You created a serverless application using Python. Which of the following runtimes can you choose? (Choose all that apply.)

   A. Python 3.8

   B. Python 3.7

   C. Python 3.6

   D. Python 2.7

8. You have an Amazon S3 bucket that stores sensitive information. You need to monitor any changes to the bucket and send an alert to your security team. How can you achieve this cost-effectively?

   A. Create an AWS Lambda that can be triggered as soon as any changes to this bucket occur

   B. Hire an AWS engineer to monitor this Amazon bucket

   C. Use the Amazon SQS queue to monitor the Amazon S3 bucket

   D. Use the Amazon CloudSearch service to monitor it

9. Your company receives data from multiple sources that needs to be formatted before being stored in an OLTP database. The frequency of incoming data can be very low to high, depending on the day of the week and time of the day. What is an efficient way to format the data cost-effectively?

   A. Create new table and store unformatted data

   B. Ask your SQL developer to write a query to format the data

   C. Provision an EC2 instance and process the data using it

   D. Use AWS Lambda to format the data before storing it in the database

10. How do you provision instances for your AWS Lambda functions?

    A. Run your code, and AWS Lambda takes care of provisioning and managing the instances

B. Provision the first instance, and AWS Lambda manages the rest of provisioning

C. It is serverless, so it does not need any instance to run your code

D. Select the auto-provision option

Answers

1. **B.** You can create AWS Lambda functions to run batch jobs.
2. **D.** AWS Lambda functions, along with Amazon CloudWatch, can be used to create, start, and stop jobs.
3. **A, B, C, D.** The supported languages are Go, Node.js, Java, and C# in addition to Ruby, Python, and PowerShell.
4. **D.** Lambda will provision another instance to handle the additional request concurrently.
5. **B.** The AWS SAM can be used to easily develop and test your serverless applications locally.
6. **A.** AWS X-Ray can be used to run tracing on your Lambda functions to troubleshoot.
7. **A, B, C, D.** Python 3.8, Python 3.7, Python 3.6, and Python 2.7 can be used as runtimes.
8. **A.** You can create an AWS Lambda that can be triggered as soon as any changes to this bucket occur.
9. **D.** You can use AWS Lambda to format the data before storing it in the database.
10. **A.** AWS Lambda takes care of all the provisioning and managing of instances.

Additional Resources

- **AWS Lambda** The recommended documentation for any AWS services, including Amazon Lambda, is the official AWS documentation, where you can get the most up-to-date information.

https://docs.aws.amazon.com/lambda/index.html and https://docs.aws.amazon.com/serverless-application-model/index.html

- **AWS Lambda Blog** This is the official blog for AWS Lambda, which has all the latest information in one place for useful functions. https://aws.amazon.com/blogs/compute/category/compute/aws-lambda/
- **AWS SAM Blog** This blog has all the latest information in one place for AWS SAM. https://aws.amazon.com/blogs/compute/tag/aws-sam/

CHAPTER 20

Deploying a Static Website on Amazon S3 Bucket

In this chapter, you will learn

- Deploy a static website using Amazon S3
- Deploy a static website using Amazon S3 and Amazon Route 53
- Deploy a static website using Amazon S3, Amazon Route 53, and Amazon CloudFront

In this chapter, we will discuss various ways of creating static websites using Amazon S3.

Amazon S3

Chapter 8 explored Amazon S3 and its different components in detail. It is an object storage service that can be used to store objects in buckets. It can also be used to deploy static websites, which have the following advantages:

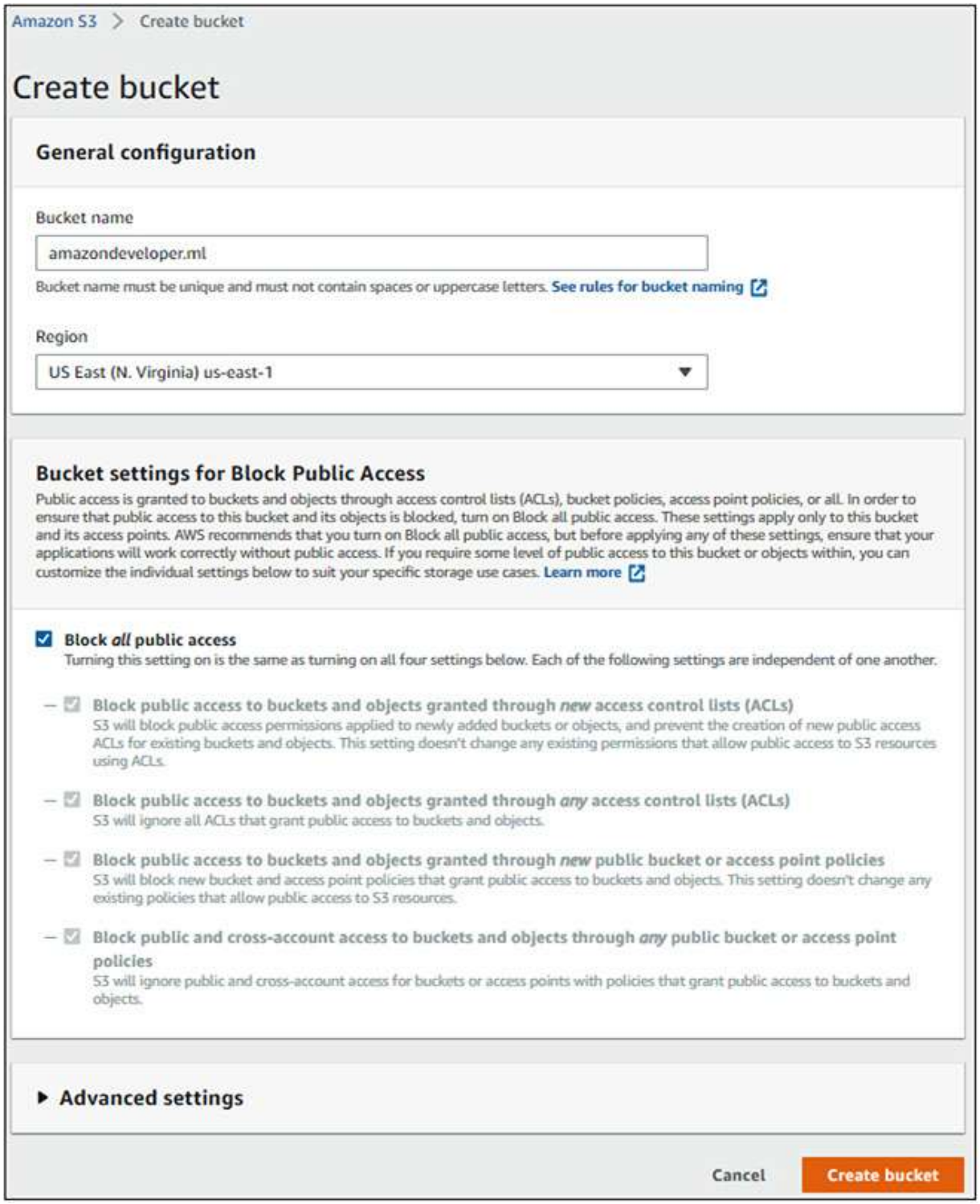
- It is easy to set up static websites in Amazon S3.
- It provides cost savings, as there are no web servers or databases to manage.
- You can host the static website using Amazon Route 53.
- You can deliver the contents locally using Amazon CloudFront.
- You can use Amazon Route 53 to manage your custom domain.

- You can deliver the contents securely using SSL from Amazon Certificate Manager.

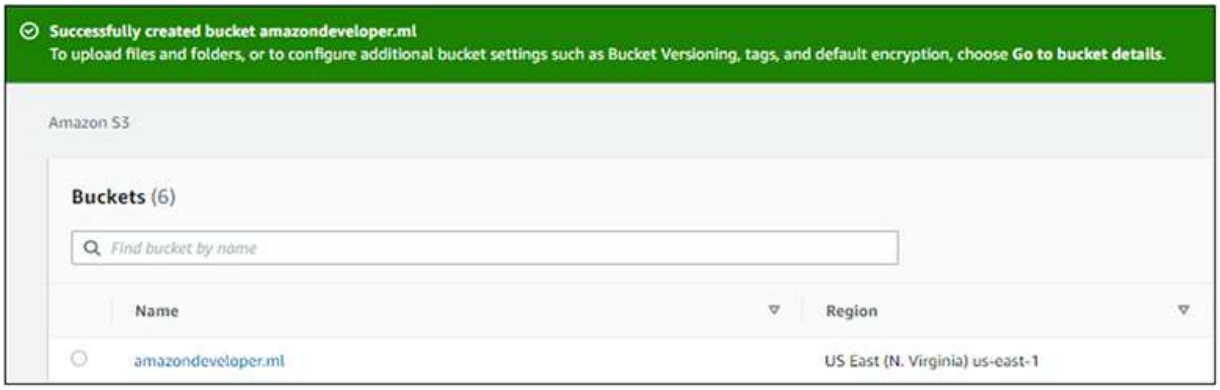
Deploy a Static Website Using Amazon S3

It is easy and simple to create a static website in Amazon S3. The following will take you on a step-by-step journey of creating a simple website.

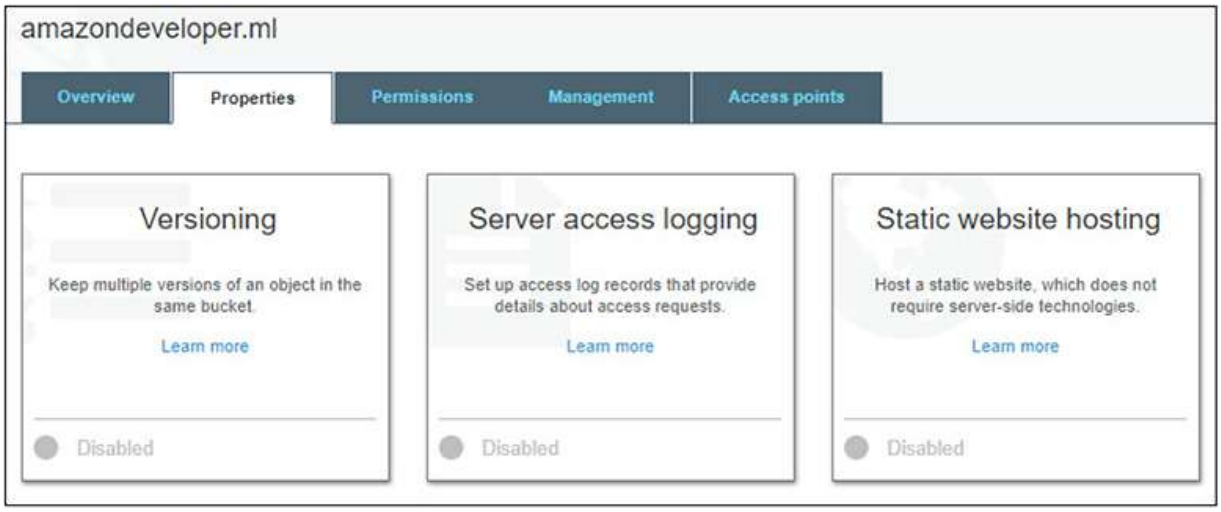
1. First log in to your AWS console, navigate to the Amazon S3 service page, and click Create Bucket.



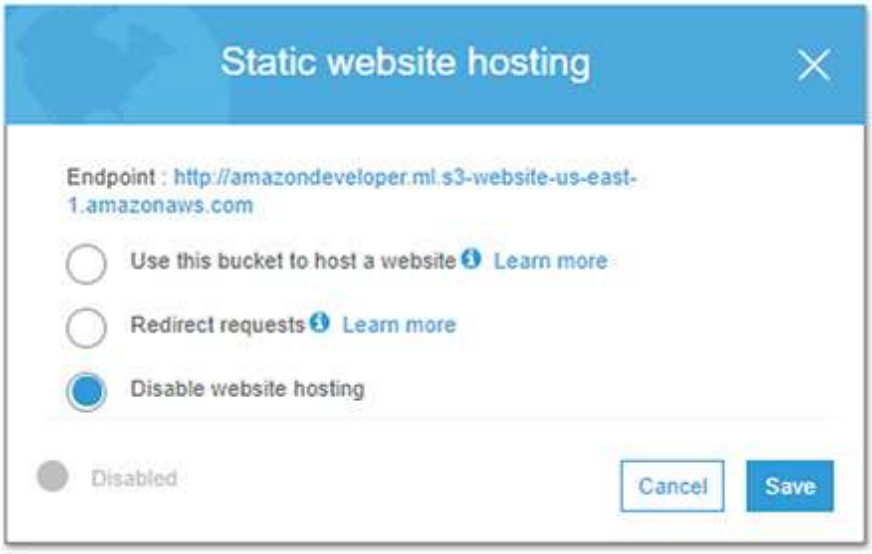
2. As shown next, a new Amazon S3 bucket called amazondeveloper.ml was created successfully.



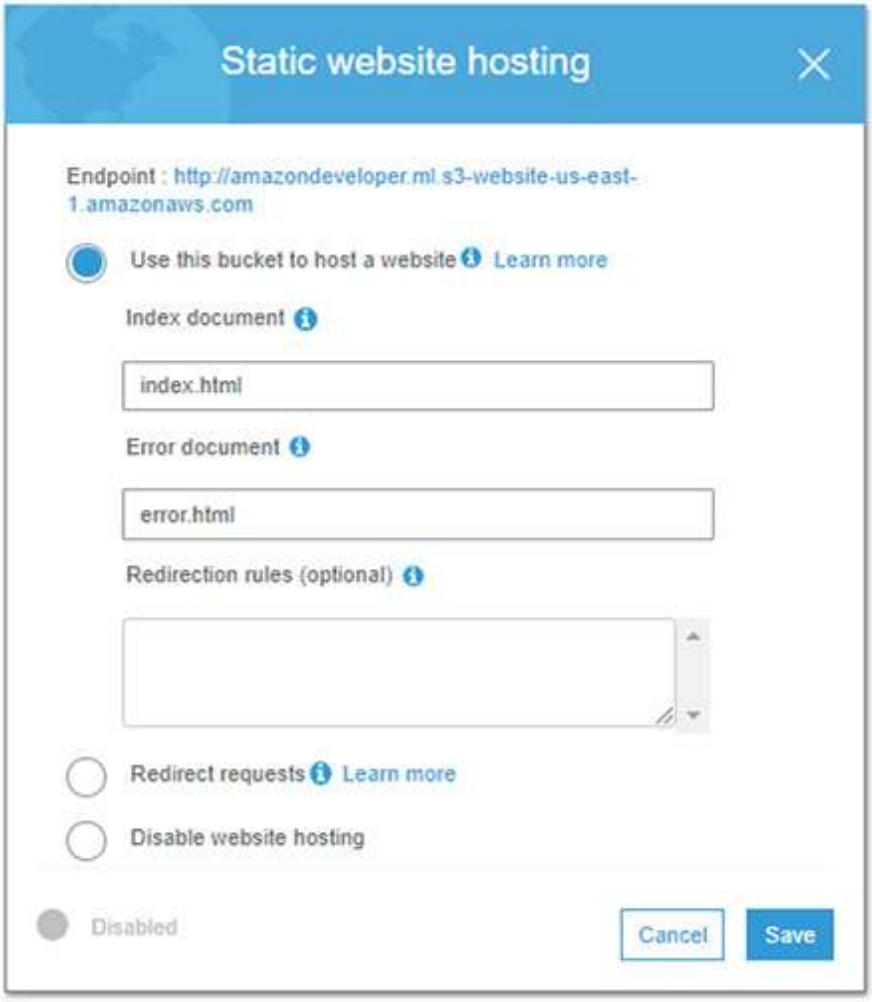
3. Click on the bucket name and select the Properties tab.



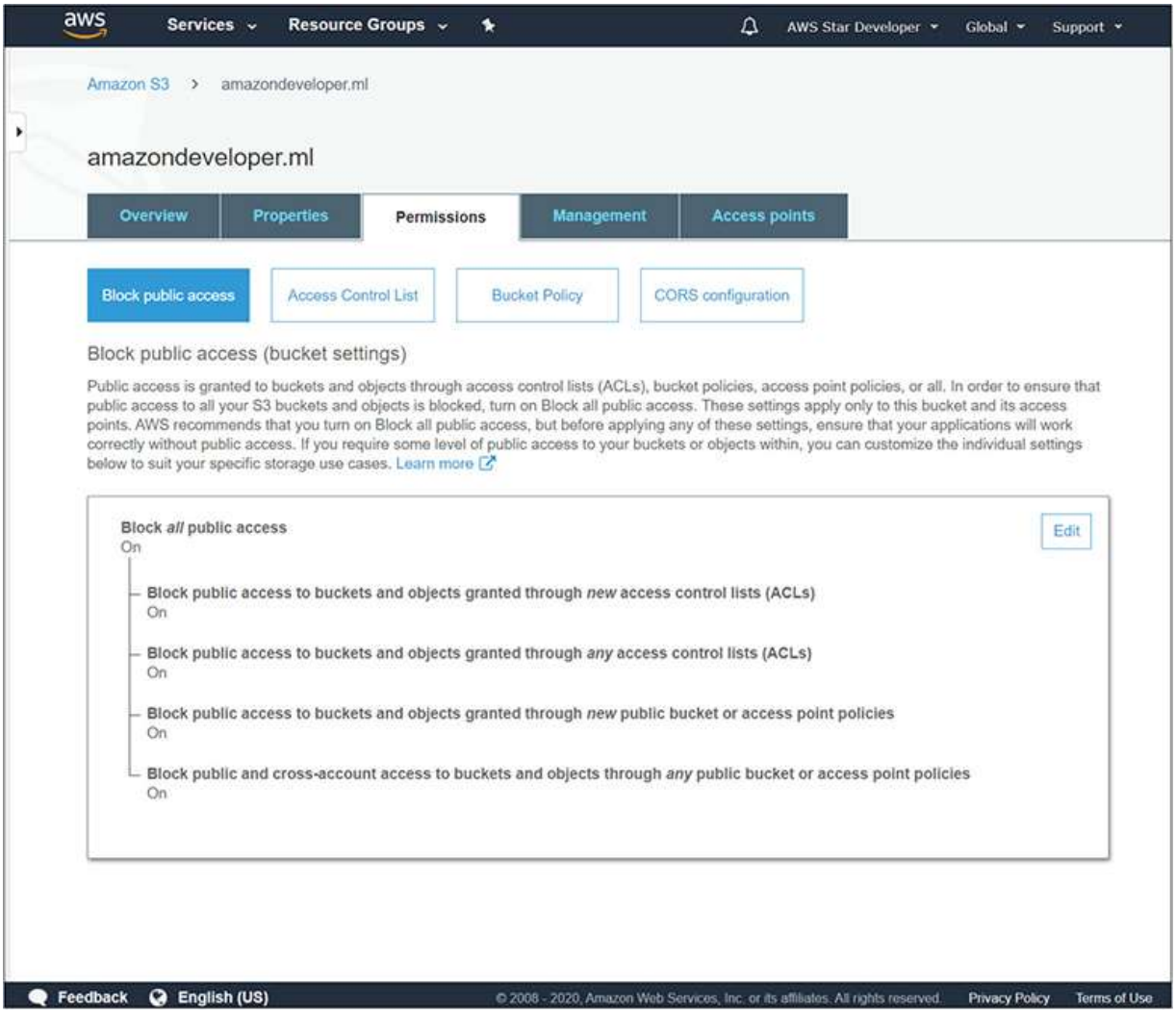
4. In order to make this Amazon S3 bucket a static website, select the Static Website Hosting box.



5. Select the option Use The Bucket To Host A Website and enter your index and error page.



6. Now static website hosting is enabled. Click on the Permissions tab to give read-only access to your objects.

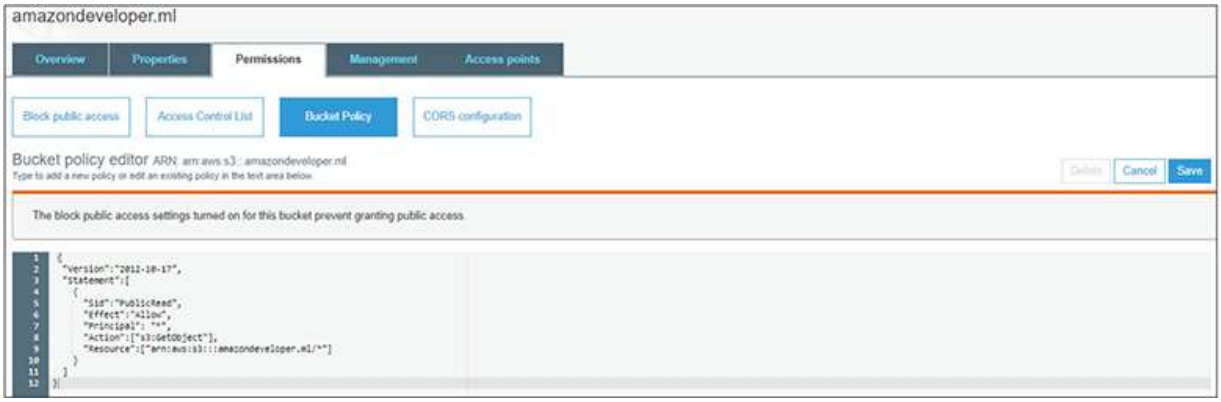


7. You can find the following Amazon read-only policy from the Amazon documentation (https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies.html#example-bucket-policies-use-case-2). You need to update the bucket name to the new Amazon S3 bucket name that we created in step 2.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"PublicRead",
      "Effect":"Allow",
      "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::amazondeveloper.ml/*"]
    }
  ]
}
```

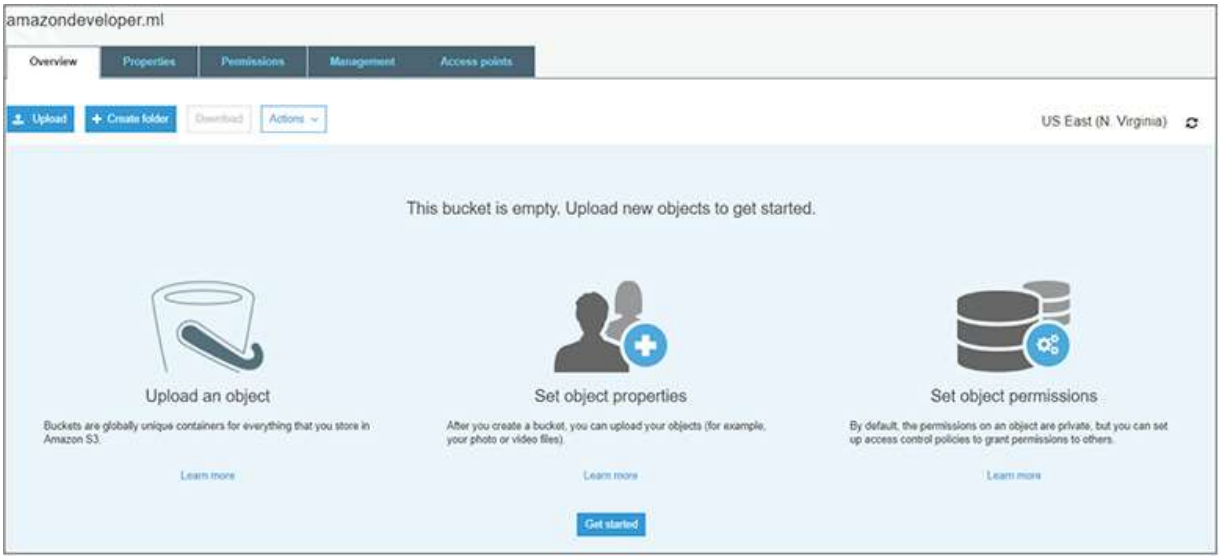
8. Enter the policy and click Save.

NOTE See the warning from AWS about granting public access to an Amazon S3 bucket, and never grant anonymous access unless it is absolutely necessary, such as when hosting a static website on Amazon S3.



9. You will see a warning in the Permissions tab regarding the bucket policy, since this bucket has public access.

10. Now click the Overview tab to upload your HTML files and images.



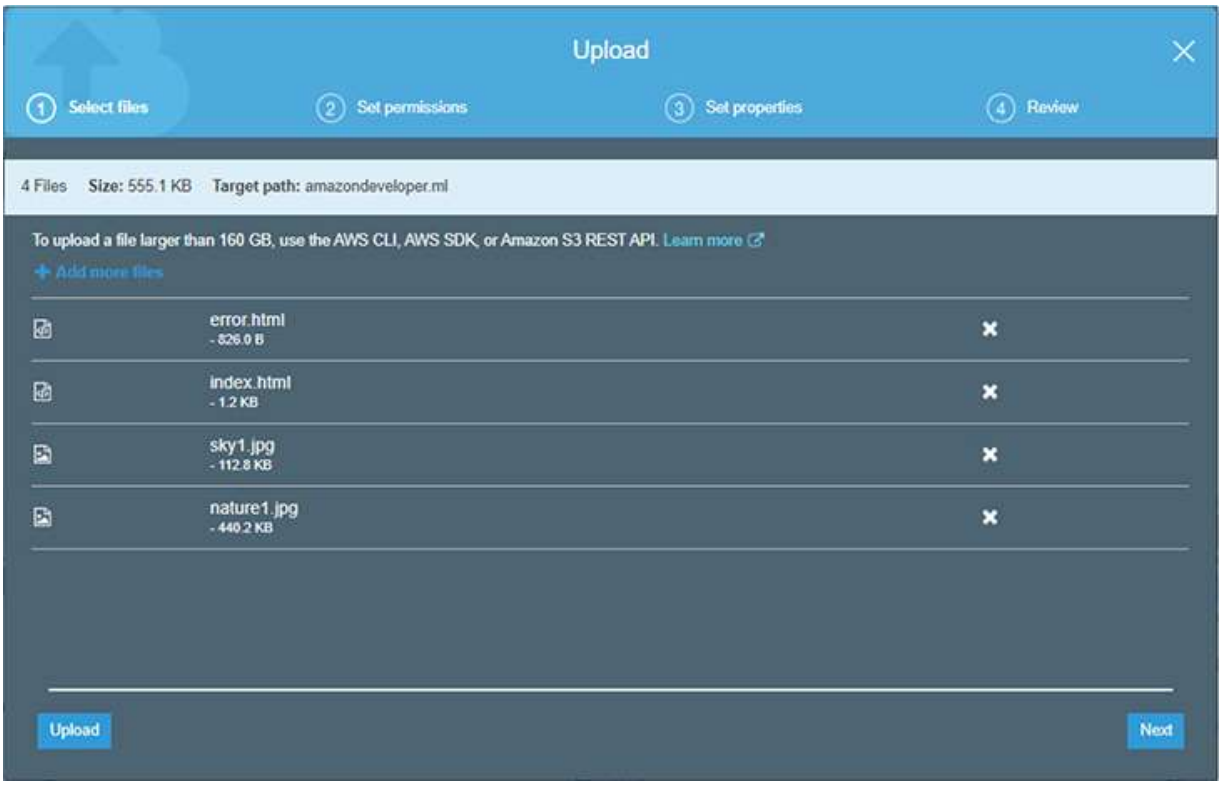
11. Create two simple HTML files in your local machine. You can use the following sample code to create your index.html file.

```
<!DOCTYPE html>
<html language="english">
   <head>
      <meta charset="utf-8">
      <title>
         A Simple Static Website Hosted on Amazon S3
      </title>
   </head>
   <body background="nature1.jpg">
      <br />
      <h1 align="center-left">
         <pre>
            <font face="Georgia" size="16" color="red"><span style="font-
size:200%">SHOPPING UNLIMITED</span></font>   </pre>
      </h1>
      <br />
      <h2 align="center-left">
      <pre>
      <MARQUEE DIRECTION="down" BEHAVIOR="alternate" scrolldelay="200"
scrollamount="3" STYLE="width:1400px; height:450px">
         <MARQUEE BEHAVIOR="alternate"><font face="Georgia" color="#FFFF00"
size="12">A Simple Static Website Hosted on Amazon S3</font></MARQUEE>
      </MARQUEE>
     <h3 align="center"">
         <pre>
      <font face="Arial Bold" size="7" color="#FFFF00"><a href="#" style="text-
decoration:none;font-size:100%;font-family:verdana;color:orange;text-
align:bottom">PRODUCTS</a>      <a href="#" style="text-decoration:none;font-
size:100%;font-family:verdana;color:orange">CONTACT US</a>           <a href="#"
style="text-decoration:none;font-size:100%;font-family:verdana;color:orange">AB
OUT US</a>
               </font></pre>
      </h3>
    </body>
</html>
```

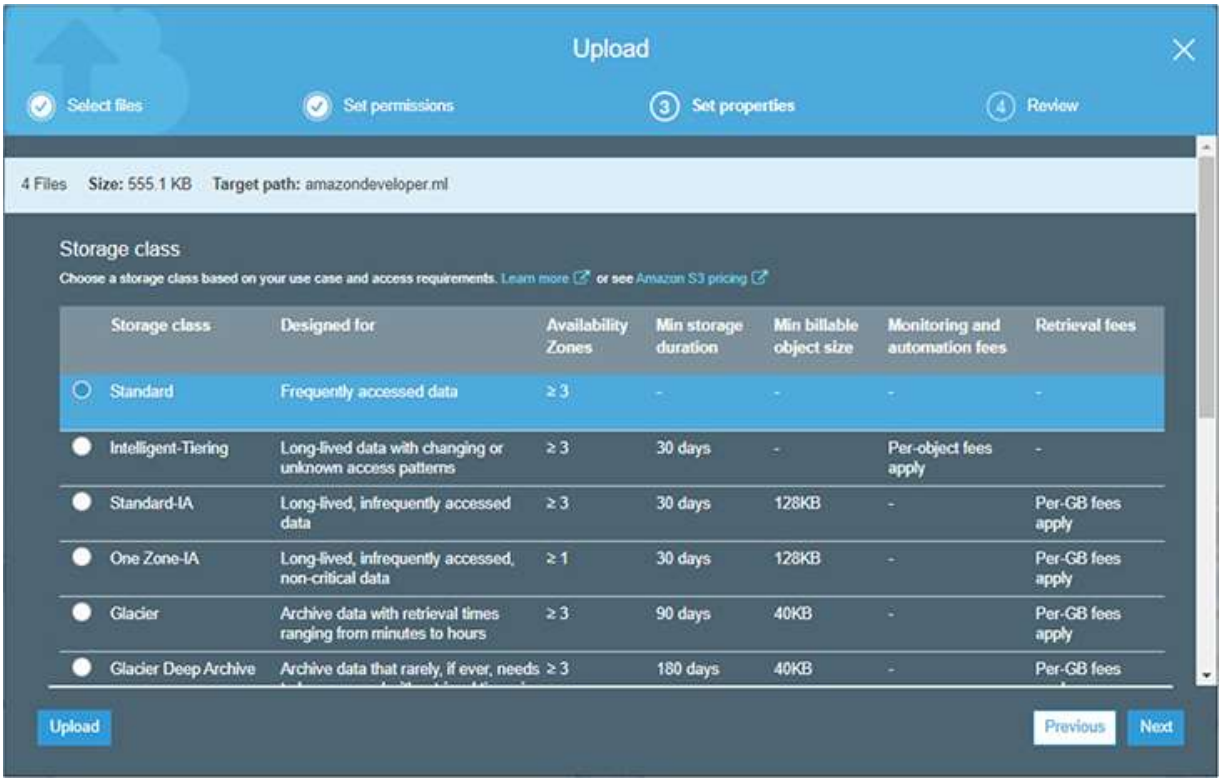
12. You can create a sample error.html page using the following sample code:

```
<!DOCTYPE html>
<html language="english">
   <head>
      <meta charset="utf-8">
      <title>
         A Simple Static Website Hosted on Amazon S3
      </title>
   </head>
   <body background="sky1.jpg">
      <br />
      <h1 align="center-left">
         <pre>
            <font face="Georgia" size="16" color="red"><span style="font-
size:200%">SHOPPING UNLIMITED</span></font>   </pre>
      </h1>
      <br />
      <h2 align="center">
      <pre>
      <MARQUEE DIRECTION="down" ONMOUSEOVER="this.stop()" ONMOUSEOUT="this.
start()"
BEHAVIOR="alternate" scrolldelay="100" STYLE="width:1400px; height:450px">
         <MARQUEE BEHAVIOR="alternate"><font face="Georgia" color="#FFFF00"
size="12">OOPS! Sorry For The Trouble!! We are working on it!!!</font></MARQUEE>
      </MARQUEE>
     </body>
</html>
```

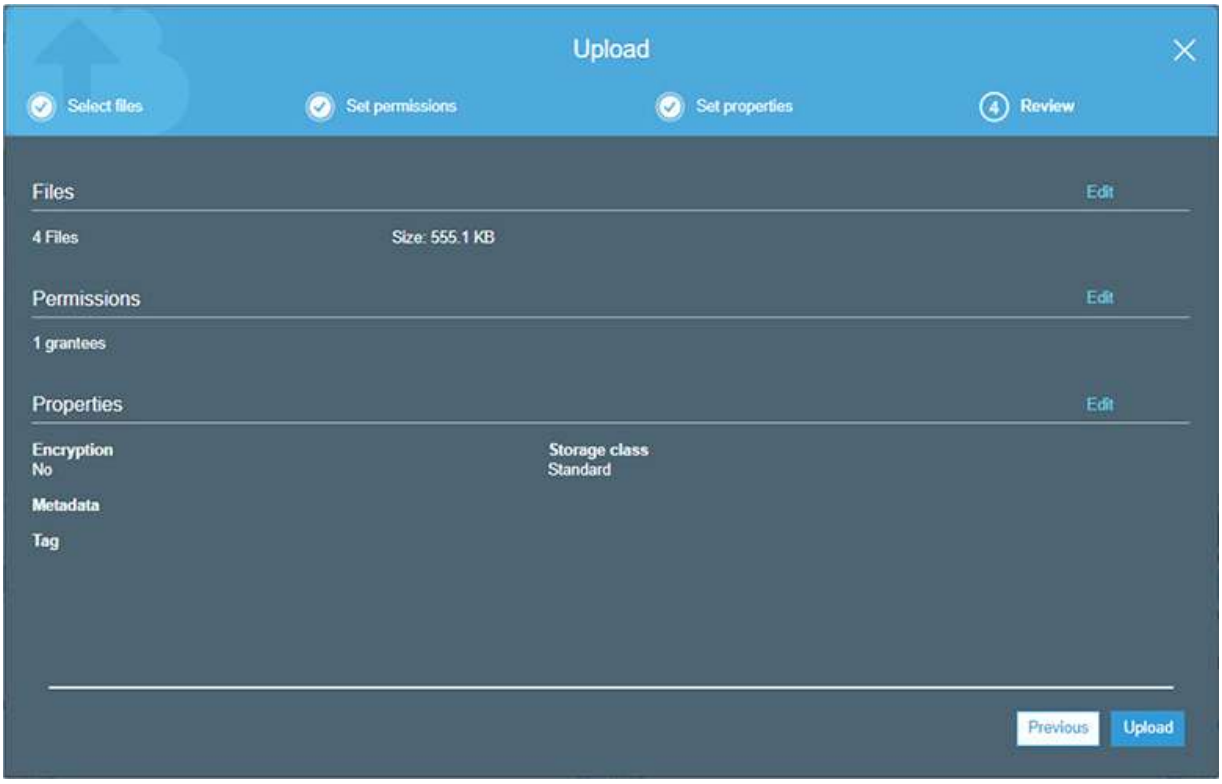
13. Just replace nature1.jpg and sky1.jpg with pictures of your choice and upload them.



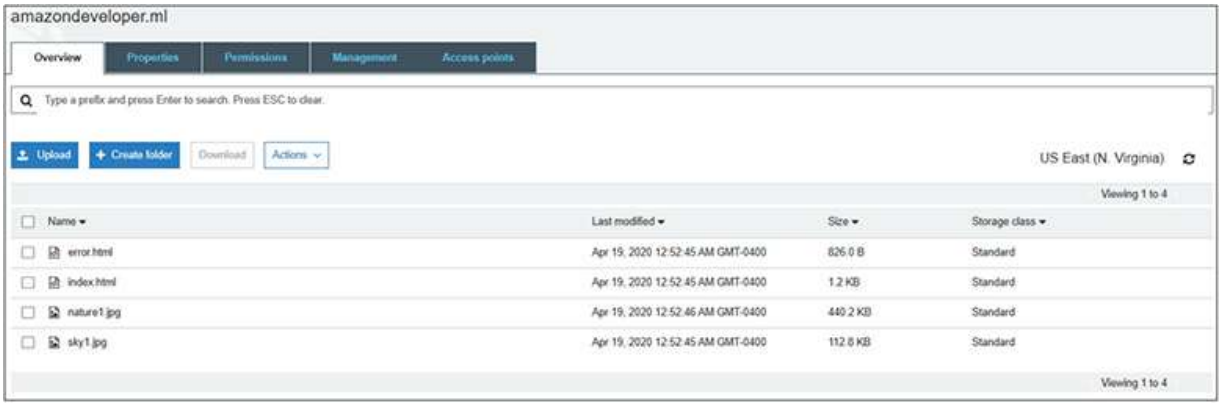
14. Choose the Standard storage class, since you will be frequently accessing this static site, as shown next. Note that here you have options to select Intelligent Tiering, Standard-IA, One Zone-IA, Glacier, Glacier Deep Archive, and Reduced Redundancy based on your requirements. Also, you have the option to encrypt the object using either the Amazon S3 master key or AWS KMS master key.



15. Here you can review all the options that you chose so far and update if necessary by clicking on the Previous button, shown next. If everything looks good, click the Upload button to upload the files.



16. Once the upload is successful, it will take you to the bucket, where you can view all the uploaded objects. Now the building of a simple static website is complete using Amazon S3.



17. You can access your new static website using your bucket name URL—for example, amazondeveloper.ml.s3-website-us-east-1.amazonaws.com—without provisioning any web server or database.

18. You can also access your website using the index.html URL—for example, amazondeveloper.ml.s3-website-us-east-1.amazonaws.com/index.html.

19. Similarly, you can access your website error page using the error.html URL—for example, amazondeveloper.ml.s3-website-us-east-1.amazonaws.com/error.html.

20. As a developer, you know very well that we need to test our implementation. So, let us enter the incorrect URL and see whether it displays the error page. You can add some word like “cloudy” at the end of the URL to test this—for example, amazondeveloper.ml.s3-website-us-east-1.amazonaws.com/cloudy.

Deploy a Static Website Using Amazon S3 and Amazon Route 53

You can create the Amazon S3 static website using a custom URL and host it in Amazon Route 53. You can get a free domain from the freenom site that can be used for building your static website.

1. Go to https://www.freenom.com and look for any domain name of your choice—for example, I checked amazondeveloper.ml.

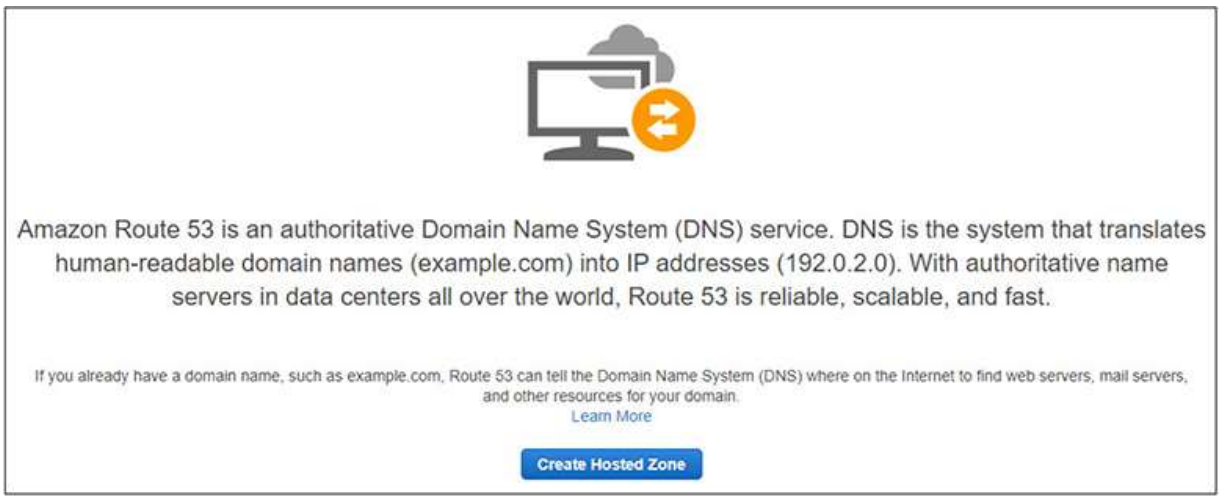
2. If the domain is available, you will see an option to select it.

3. When you click on Checkout after logging in, it will take you to the next page. Click Continue.

4. Next you will see the Review And Checkout page, where you can complete the order to get your new domain (which you will have to pay for after 12 months).

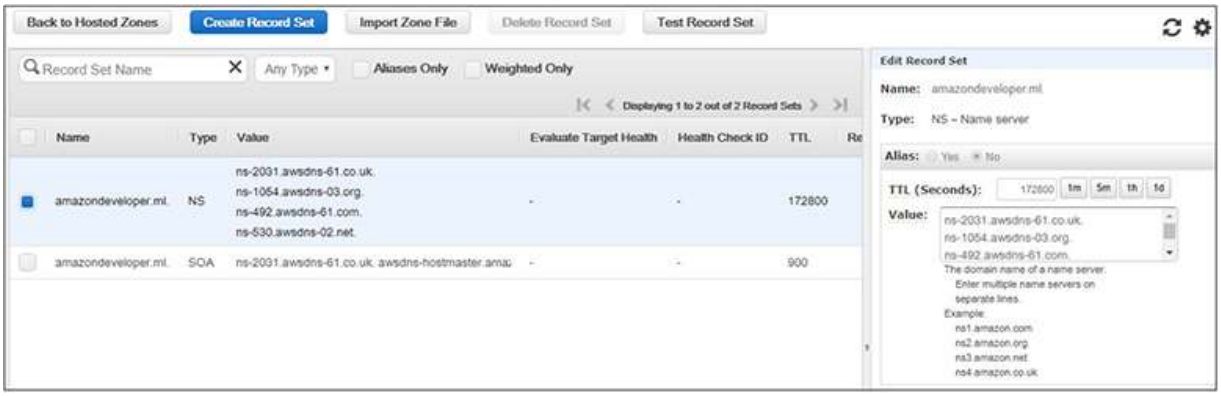
5. Now, you can go back to your AWS console and navigate to Amazon Route 53. Here you can see options for DNS management, traffic management, availability monitoring, and domain registration. Choose DNS Management to host your new domain.

6. The DNS management dashboard will appear, where you can create a hosted zone.

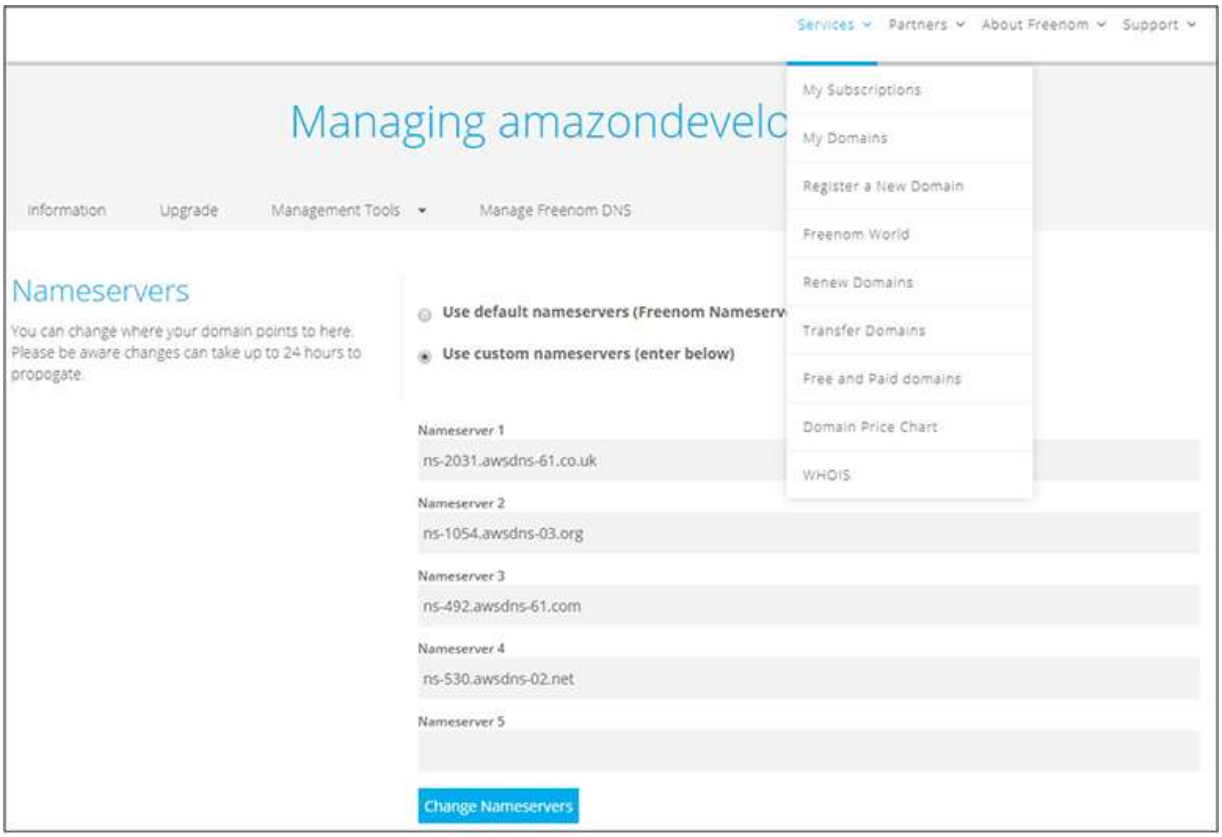


7. Enter your domain name—for example, **amazondeveloper.ml**—in the Domain Name field, select Public Hosted Zone for the Type, and click the Create button.

8. Note the namespace values that you need to update in the https://www.freenom.com website.



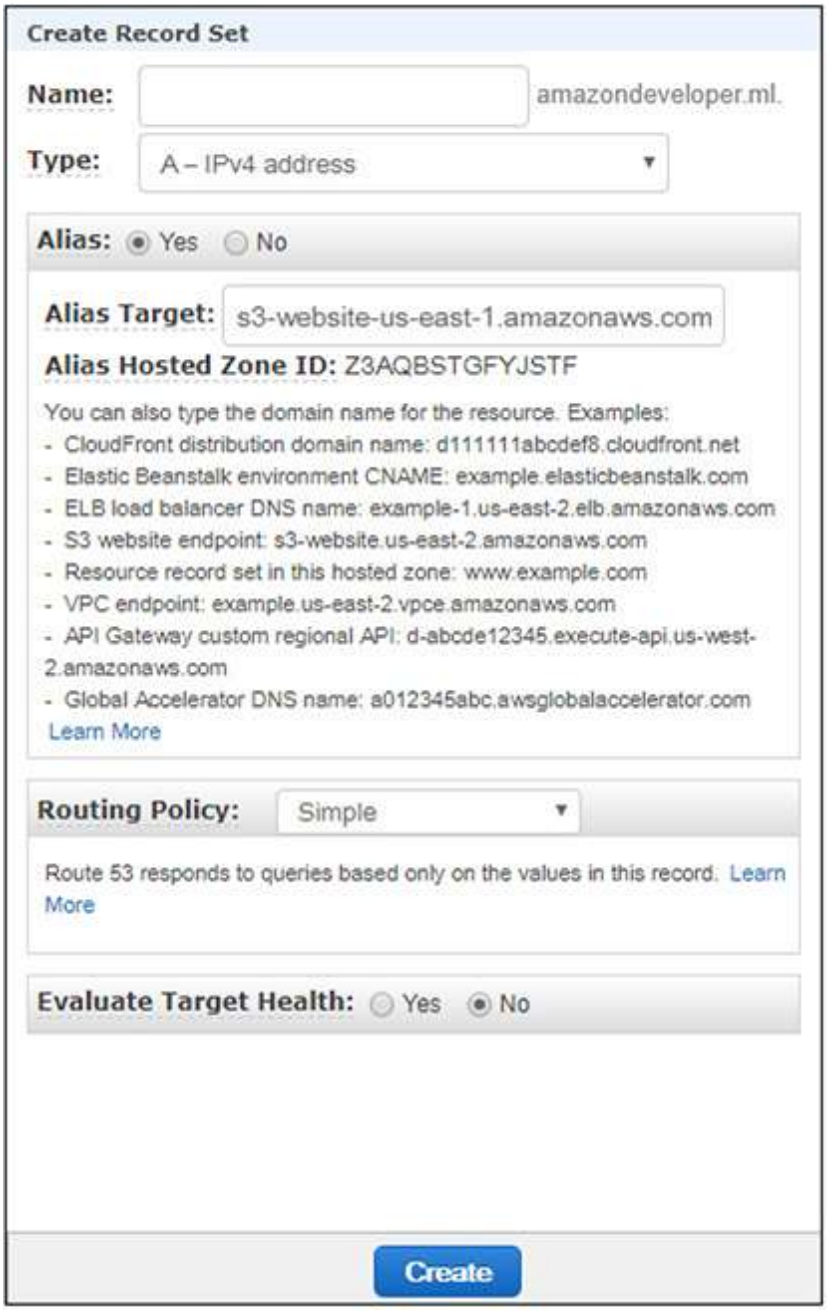
9. Now, go to https://wwwfreenom.com and navigate to Services | My Domains. Enter the namespace values from the previous step.



10. Navigate back to the AWS console and go to Amazon Route 53 DNS Management, and create Record Set, shown next. Select the Type as A and leave the name blank. For Alias, select Yes, and select the Amazon S3 bucket that we created in the previous exercise from the dropdown for Alias Target. Select Simple for the Routing Policy and leave

Evaluate Target Health set to No. After reviewing all the options, click the Create button.

NOTE Set Evaluate Target Health to Yes for production implementations and for a failover routing policy.



11. Now you have successfully hosted your Amazon S3 static website using your custom domain in Amazon Route 53. You can type your custom URL in your browser and press ENTER to see your new site.

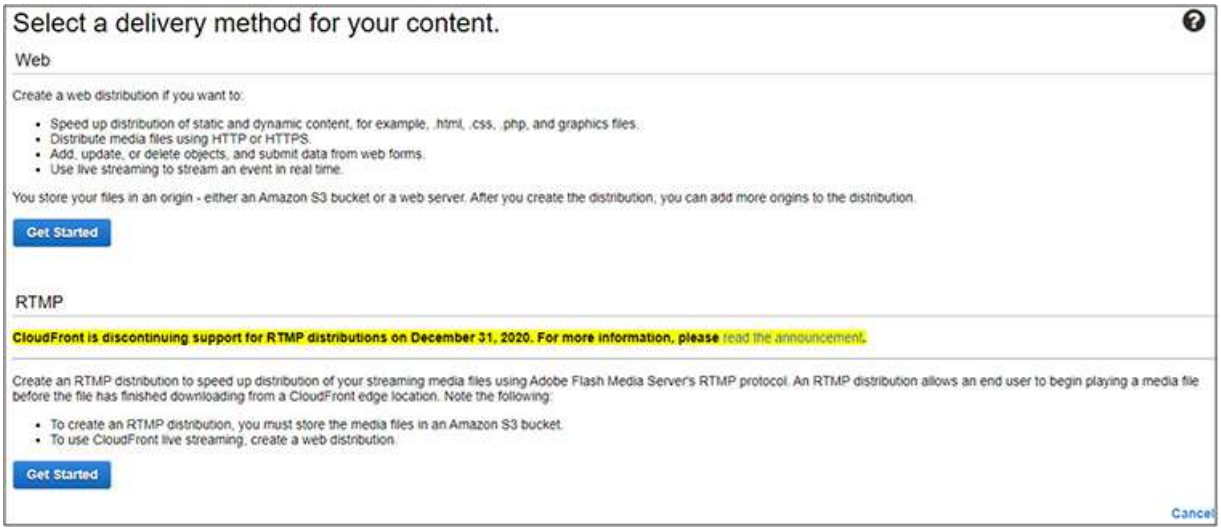
12. As you did before, test your implementation by adding any word at the end. The error.html page should appear and display the custom error message.

13. You also should be able to reach your website index.html page by entering **/index.html** at the end of your custom URL.

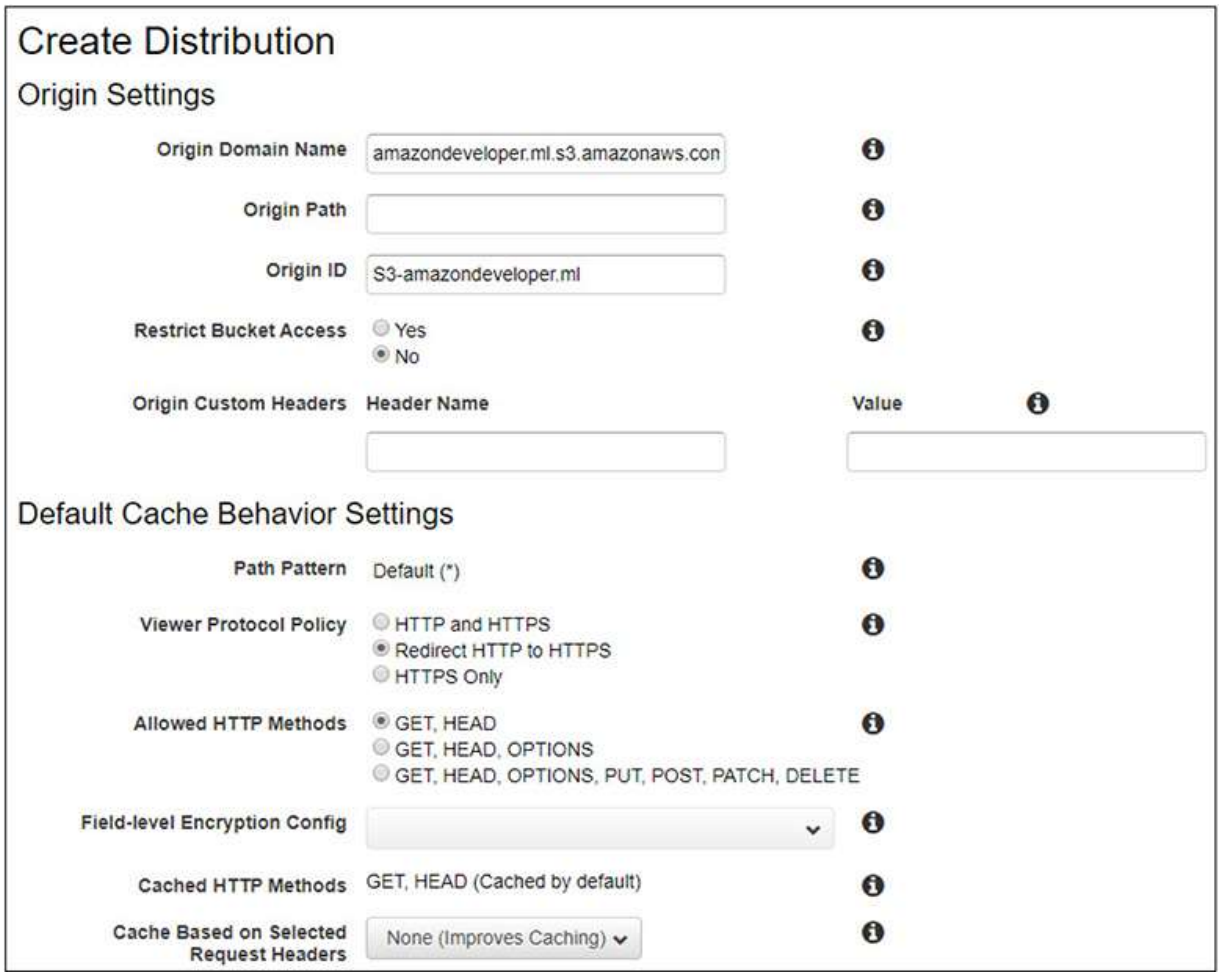
Deploy a Static Website Using Amazon S3, Amazon Route 53, and Amazon CloudFront

Your website has become very popular, and you have customers worldwide now. You can add Amazon CloudFront to deliver globally with low latency and high performance from the nearby points of presence (PoPs). Also, you can secure access to your site by using SSL.

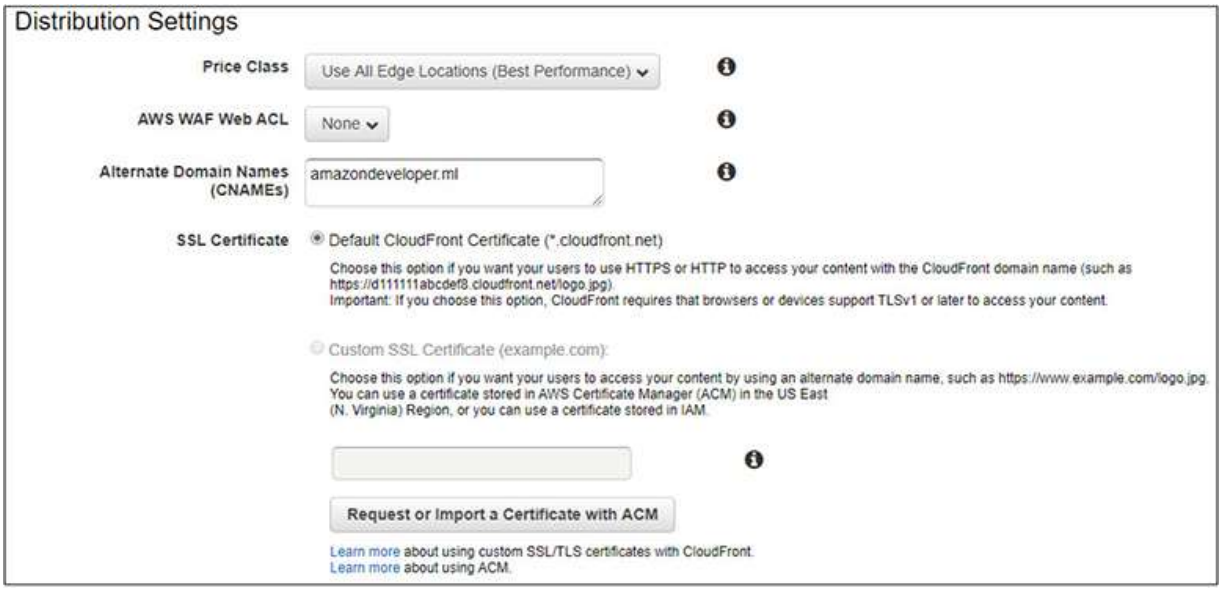
1. Log in to your AWS console and navigate to the Amazon CloudFront service. Click Get Started under the Web delivery method.



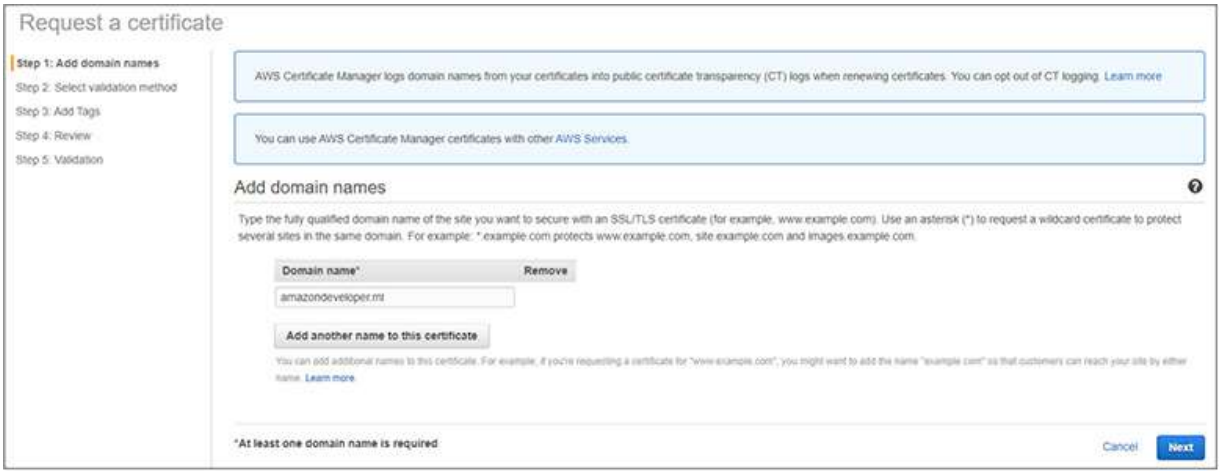
2. Enter your Amazon S3 bucket name in the Origin Domain Name field, and enter your domain name for the Origin ID—for example, amazondeveloper.ml—shown next. Restrict Bucket Access should be set to No for this exercise. Select Redirect HTTP to HTTPS for the Viewer Protocol policy, and allow only ready access by selecting GET in the Allowed HTTP Methods area.



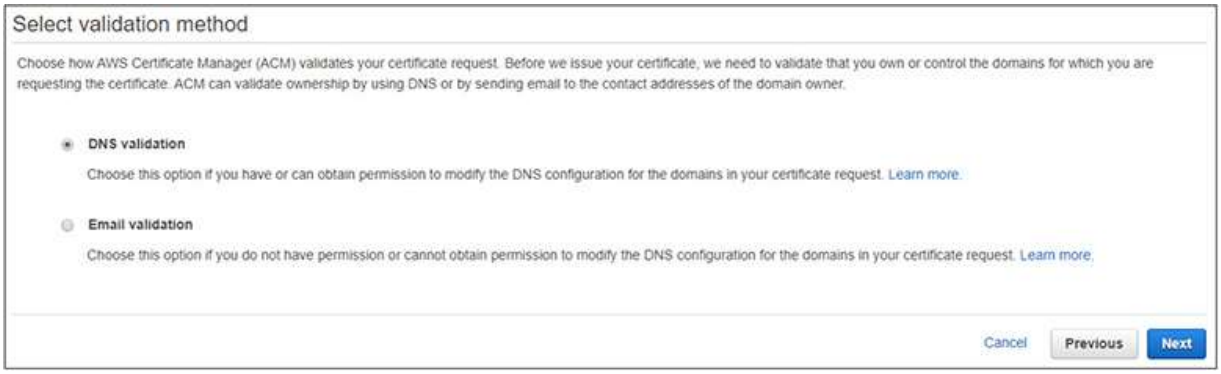
3. In the distribution settings, select Price Class for Use All Edge Locations. The CNAME should be your custom domain URL, and to create an SSL certificate, click the Request Or Import A Certificate With ACM button.



4. The Amazon Certificate Manager (ACM) page appears. Enter your domain name and click Next.

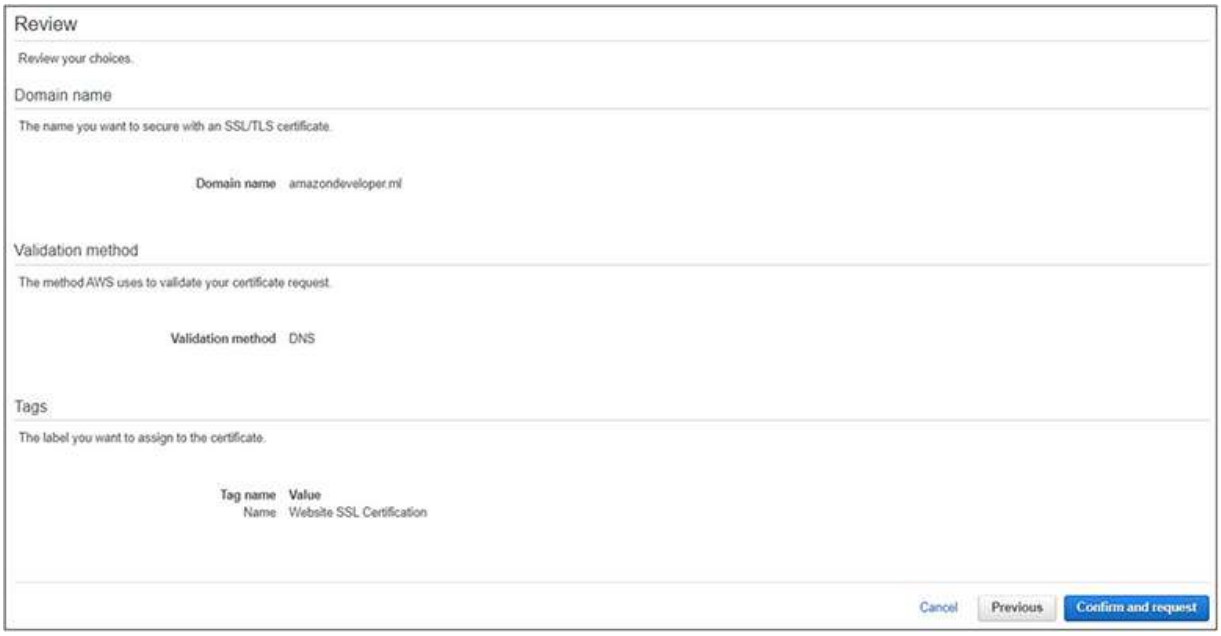


5. Select the DNS Validation method and click Next.

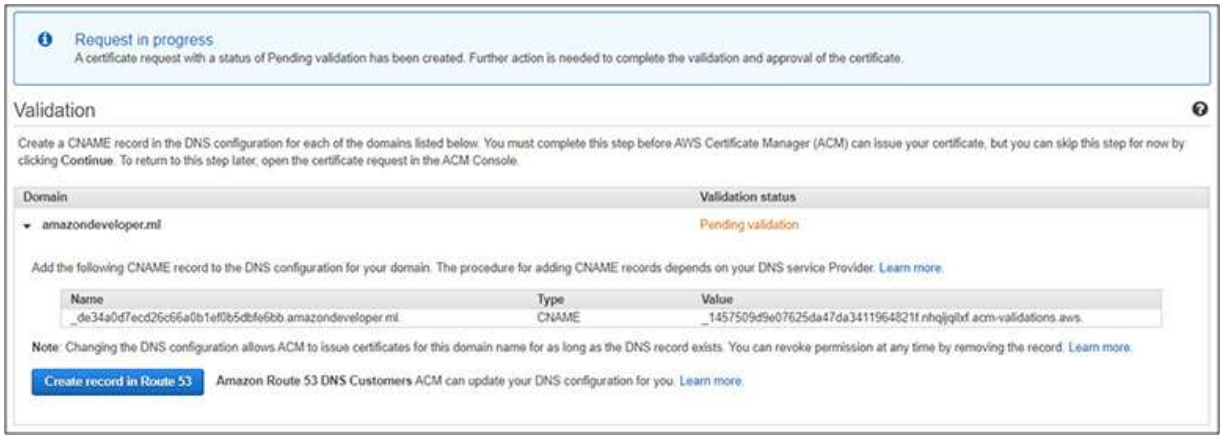


6. It is always a best practice to add tags, so add a name with a value like Website SSL Certification and click on the Review button.

7. In the Review page, make sure the values are correct. Click Confirm And Request.

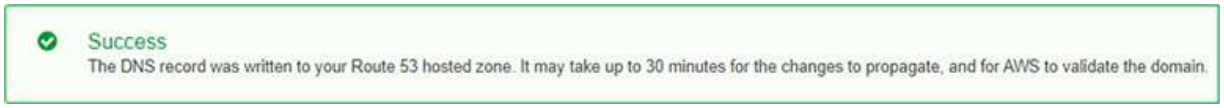


8. While the validation is in progress, you have an option to create a record in Amazon Route 53. Click on the Create Record In Route 53 button.



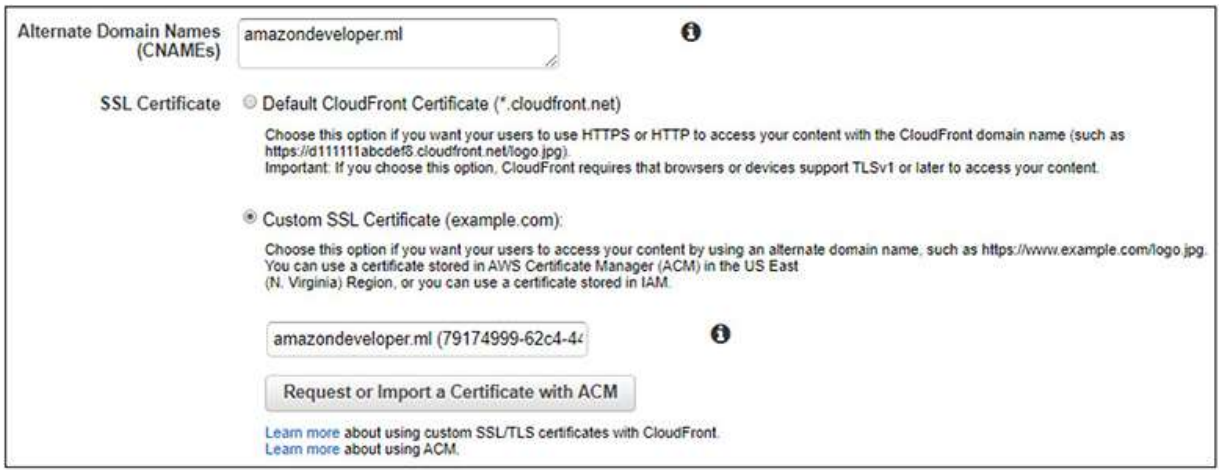
9. Here the values will be auto-populated for you to create the DNS record. Click the Create button to create the DNS record.

10. A success message appears. Since the validation status is in a pending state, you need to wait for couple of minutes.



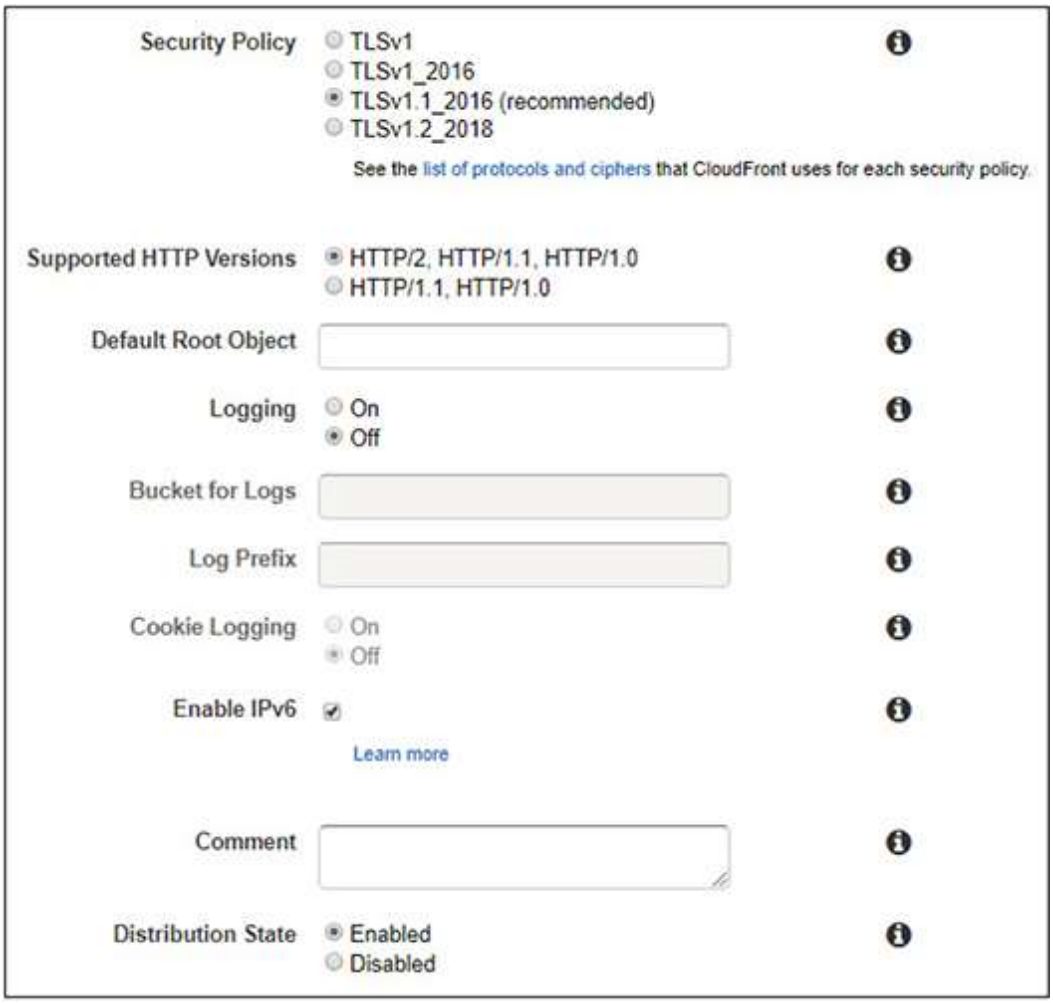
11. The SSL certificate status will be changed to Issued.

12. In the CloudFront page, select Custom SSL Certificate and select your ACM-issued SSL certificate.



13. The Security Policy should be the recommended TLS. Select HTTP/2, HTTP/1.1, and HTTP/1.0 as supported HTTP versions, shown next. You can leave the logging off and select the Distributed state as Enabled. Click the Create Distribution button to create your Amazon CloudFront distribution.

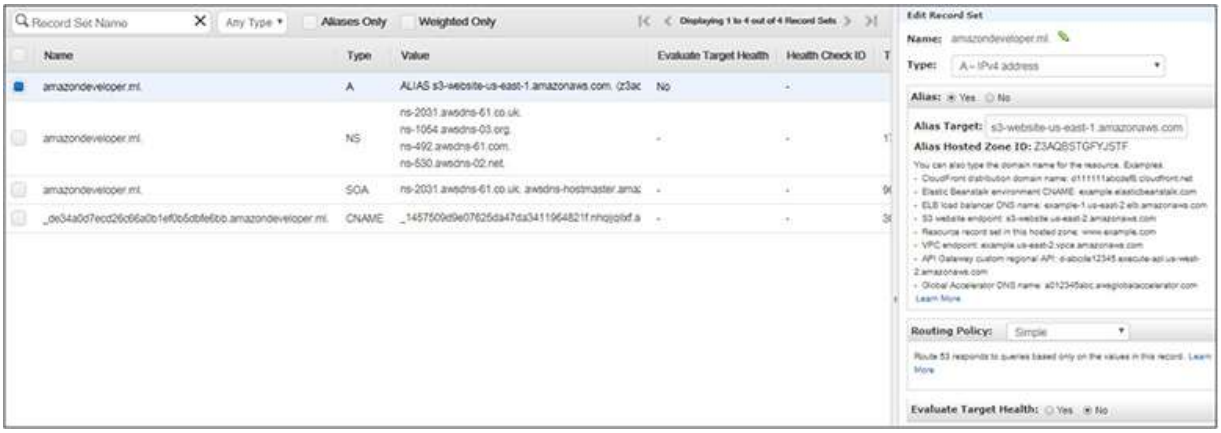
NOTE You need to enable logging for your production implementation and to troubleshoot any issues.



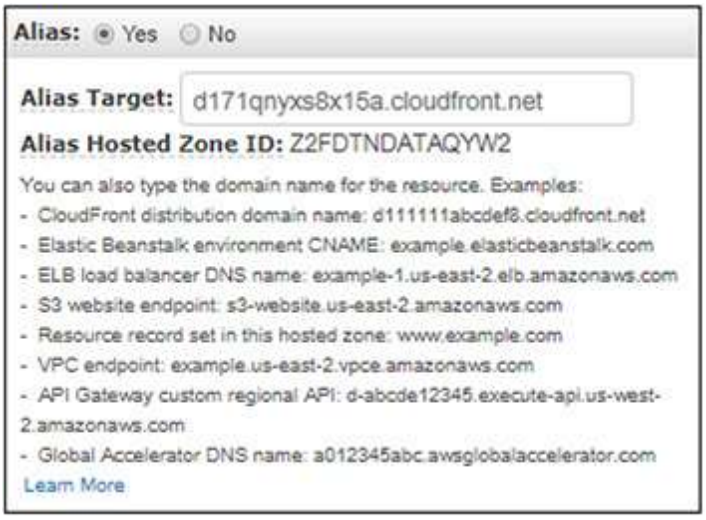
14. You will be redirected to the CloudFront Distributions page with your web distribution set to Enabled.

15. You may need to wait for a couple of minutes to see the status change from In Progress to Deployed.

16. Navigate to your Route 53 service page and update the A record, which is pointing to your Amazon S3 bucket in the Alias Target.



17. Update the Alias Target to your new CloudFront Web distribution from the dropdown and click Save Record Set.



18. You can type your custom domain name—for example, amazondeveloper.ml—in your browser and press ENTER. Your Amazon

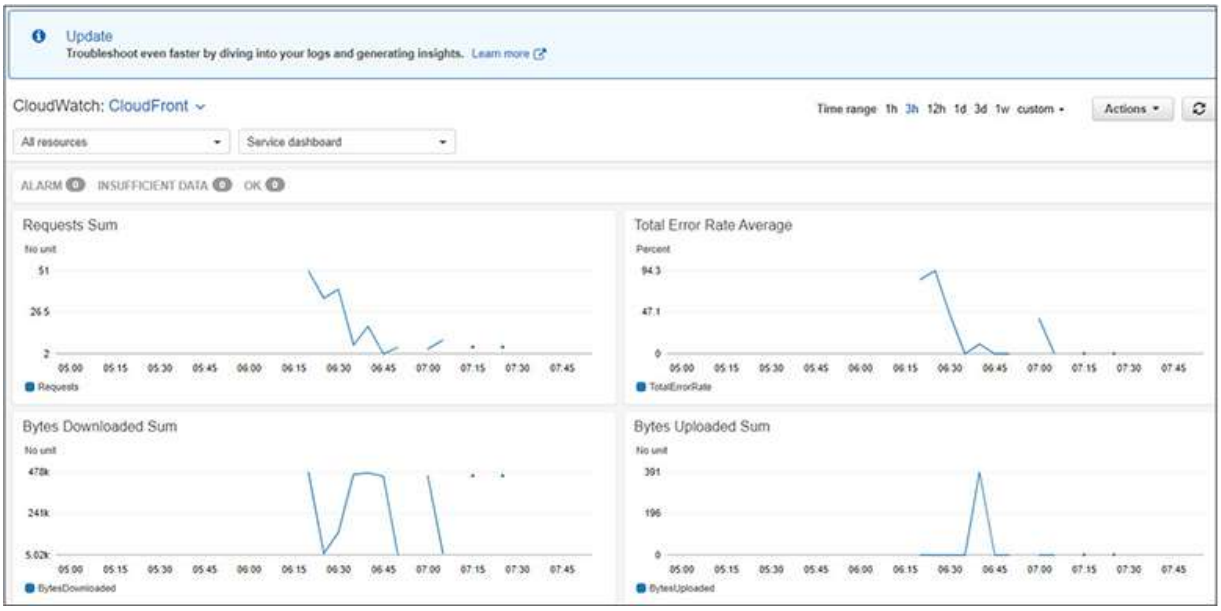
S3 static website is now delivered through the Amazon CloudFront location nearest to you.

19. You can also reach your website using your index.html. For example, you can type **amazondeveloper.ml/index.html** and press ENTER to see the index page.

20. Similarly, you can directly reach your error page by appending **error.html** to your custom domain name.

21. As a best practice, let us test our implementation by inserting a random word like “cloudy” after the domain name. It should take you to the error.html page and display your custom error.

22. You can navigate to Amazon CloudWatch to see the performance of your CloudFront distribution.



Chapter Review

This chapter began by explaining the advantages of using Amazon S3 to create a static website and then provided the detailed steps necessary to accomplish this. The chapter also provided the steps to create a custom domain and host it in Amazon Route 53 and then explained the steps to create a web distribution using Amazon CloudFront and the steps to create an SSL certificate using ACM. This experience will help you on the exam as well as in the real world.

Exercise

The following exercises will help you practice using Amazon S3 to deploy a website. You need to create an AWS account, as explained earlier, to perform these exercises. You can use the Free Tier when launching AWS resources, but make sure to terminate them at the end.

Exercise 20-1: Clean Up the Resources Created in This Chapter Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in and then navigate to the AWS CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, select the web distribution that you created in this chapter and choose to disable it. You will be prompted to confirm this action.
4. When the distribution is disabled, choose to delete it. You will be again prompted to confirm this action.
5. Now navigate to the Amazon Route 53 console at https://console.aws.amazon.com/route53/.
6. Go to Record Sets and delete the record set that you created in this chapter.
7. Go to Hosted Zone, select your domain, and delete it.
8. Then navigate to the Amazon S3 console at https://console.aws.amazon.com/s3/.
9. From the bucket list, select the bucket that you created in this chapter and delete all the objects before deleting the bucket.
10. Verify all the resources you created in this chapter are deleted to prevent any charges from incurring.

Questions

The following questions will help you gauge your understanding of the contents in this chapter. Read all the answers carefully because there might

be more than one correct answer. Choose the best responses for each question.

1. You need to create a website in very short time. The website will have only static contents, including video and audio files. Which of the following options can you use to create a website quickly?

   A. Use AWS CloudFormation to create an EC2 instance and use RDS to deploy your website

   B. You can use Amazon S3 to upload HTML files, video files, and audio files and to host your static website

   C. Provision EC2 instances in each availability zone and use Elastic Load Balancer to route website requests

   D. Provision Amazon RDS with a read replica to host your website

2. You created and hosted a static website in Amazon S3, but you cannot access your website. What do you need to configure to make your static website hosted in Amazon S3 work? (Choose three.)

   A. You need to create and upload the index.html file

   B. You need to enable website hosting in Properties

   C. You need to add a bucket policy to make your bucket publically readable

   D. You need to upload video files to your Amazon S3 bucket

3. You created a static website hosted in Amazon S3, and your team want to use an existing URL instead of using the Amazon S3 URL. Which of the following AWS services can be used to achieve this?

   A. Amazon Athena

   B. AWS Lambda

   C. AWS Fargate

   D. Amazon Route 53

4. Your team hosts a static website using Amazon S3, and they want to display a user-friendly custom message when an error occurs. How can you achieve this?

   A. Delete the Amazon S3 bucket and create it again using the S3 Glacier storage class

B. Encrypt your Amazon S3 bucket using Amazon S3 managed keys

C. Upload a custom error.html page and enter the name in the Error Document box

D. Create another Amazon S3 bucket to display the error message

5. Your customers are getting errors when accessing your static website, which is hosted on Amazon S3. How can you access the logs to perform analysis to troubleshoot and resolve this issue?

A. Amazon S3 does not support logging

B. Choose Enable Logging in Amazon S3 Properties Server Access Logging and provide the target bucket

C. You need to delete the Amazon S3 bucket and create it with client-side encryption

D. You need to set up a lifecycle policy to enable logging in Amazon S3

6. A company stores static web content in Amazon S3, and the security team requires the contents to be encrypted at rest. Amazon S3 provides which of the following options to encrypt the bucket objects? (Choose three.)

A. Use server-side encryption with Amazon S3 managed keys (SSE-S3)

B. Use server-side encryption with customer master keys (CMKs) stored in the AWS Key Management Service (SSE-KMS)

C. Use server-side encryption with customer-provided keys (SSE-C)

D. Use server-side encryption with an EC2 key pair (SSE-EC2)

7. Your team is planning to encrypt all static website objects being uploaded to Amazon S3, but they do not want to manage the encryption key. Which of the following options is available in Amazon S3 to achieve this?

A. Amazon S3 Managed Keys (SSE-S3)

B. AWS Key Management Service (SSE-KMS)

C. Customer-provided keys (SSE-C)

D. Client-provided keys (SSE-K)

8. Your security team wants to restrict access to your Amazon S3 bucket, which you have used for static website hosting. How can you restrict

access to your bucket and still allow users to access your static website?

A. Update your Amazon S3 bucket policy to deny access

B. Create an IAM role that denies Amazon S3 access and attach it to this bucket

C. You need to configure Amazon S3 to use SSL for more security

D. Create a CloudFront web distribution with option Yes selected to restrict bucket access and use the origin access identity

9. You hosted a static website using Amazon S3 and used Amazon Route 53 to host your domain. You configured Amazon CloudFront to deliver the content. After reviewing the logs, you discovered that your website is being delivered from Amazon S3 and not from Amazon CloudFront. How can you fix this?

A. Update the A record in Amazon Route 53 and choose Amazon CloudFront distribution as the Alias Target

B. Delete the Amazon CloudFront web distribution and create an RTMP distribution

C. Add a deny bucket policy to your Amazon S3 bucket

D. You need to use the Amazon CloudFront URL

10. A global company has deployed their training website, which has thousands of large videos, using Amazon S3. Employees who are accessing content from a different country are complaining about video loading delays. How can you ensure that all employees have a seamless experience, regardless of which part of the world they log in from?

A. Create a separate Amazon S3 bucket for training videos

B. Use Amazon CloudFront to deliver the content

C. Enable server-side encryption for the Amazon S3 bucket

D. Upload the videos to Amazon DynamoDB

Answers

1. **B.** You can use Amazon S3 to upload HTML files, video files, and audio files and to host your static website.

2. **A, B, C.** You need to create and upload the index.html file and enable website hosting, and you need to add a bucket policy to make your bucket publicly readable.
3. **D.** You can use Amazon Route 53 to host your custom domain.
4. **C.** You can upload the custom error.html code and enter the name in the Error Document box.
5. **B.** Choose Enable Logging in Amazon S3 Properties Server Access Logging and provide the target bucket.
6. **A, B, C.** You can use SSE-S3, SSE-KMS, and SSE-C to encrypt your Amazon S3 bucket objects.
7. **A.** SSE-S3 is correct in this context.
8. **D.** You can create a CloudFront web distribution with Yes selected to restrict bucket access and use the origin access identity.
9. **A.** You need to update the A record in Amazon Route 53 and choose Amazon CloudFront distribution for the Alias Target.
10. **B.** You can use Amazon CloudFront to deliver the content.

Additional Resources

- **Amazon S3** The recommended documentation for any AWS service, including Amazon S3, is the official AWS documentation, where you can always get the most up-to-date information.
  `https://docs.aws.amazon.com/s3/index.html`
- **Amazon S3 Blog** This is the official blog for Amazon S3, which has all the latest information about different use cases in one place.
  `https://aws.amazon.com/blogs/aws/tag/amazon-s3/`
- **Amazon S3 Defense in Depth** This blog shows you how to achieve defense in depth by applying multiple security controls to prevent data leakage.
  `https://twitter.com/Werner/status/971521075396751361` and `https://aws.amazon.com/blogs/security/how-to-use-bucket-policies-and-apply-defense-in-depth-to-help-secure-your-amazon-s3-data/`

CHAPTER 21

Deploying a Web Application Using AWS Elastic Beanstalk

In this chapter, you will learn

- Deploy a sample application in AWS Elastic Beanstalk
- Create, migrate, and deploy a custom application into AWS Elastic Beanstalk

This chapter will provide the steps to deploy a simple web application into the AWS Elastic Beanstalk service.

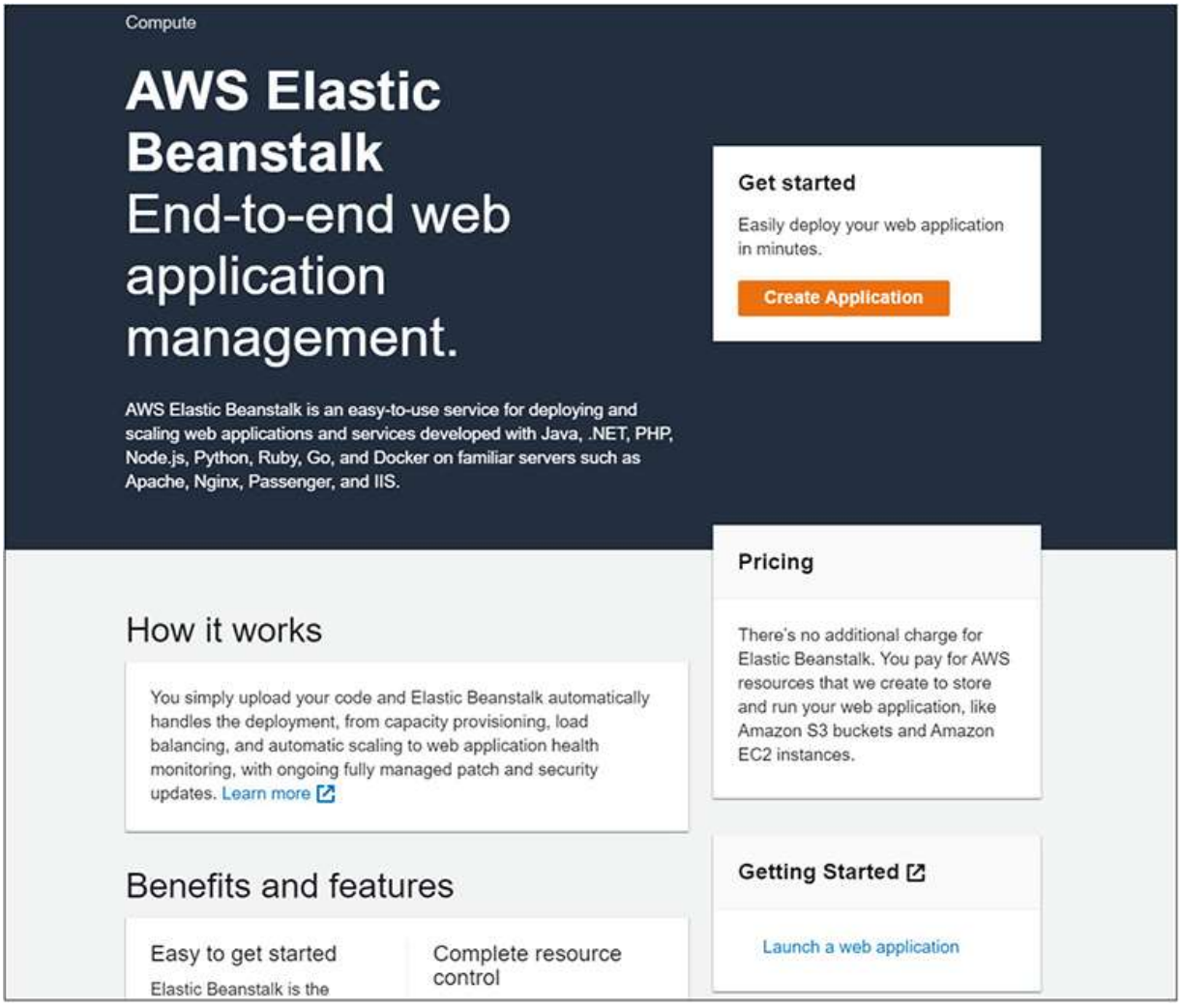
AWS Elastic Beanstalk

As a developer, you always want to spend more time developing applications for your customers instead of installing, configuring, and managing the infrastructure to run those applications. AWS Elastic Beanstalk enables you to deploy your application developed in Java, .NET, PHP, Python, Node.js, Ruby, and Go by just uploading the application source bundle, such as the .war file for Java. AWS Elastic Beanstalk automatically takes care of provisioning EC2 instances, the load balancer, database, scaling, and monitoring application health. You can use the Elastic Beanstalk CLI or AWS CLI or the AWS Elastic Beanstalk console to interact with AWS Elastic Beanstalk. It is free to use AWS Elastic Beanstalk, and you are charged only for the underlying AWS resources.

Deploy an Application in AWS Elastic Beanstalk

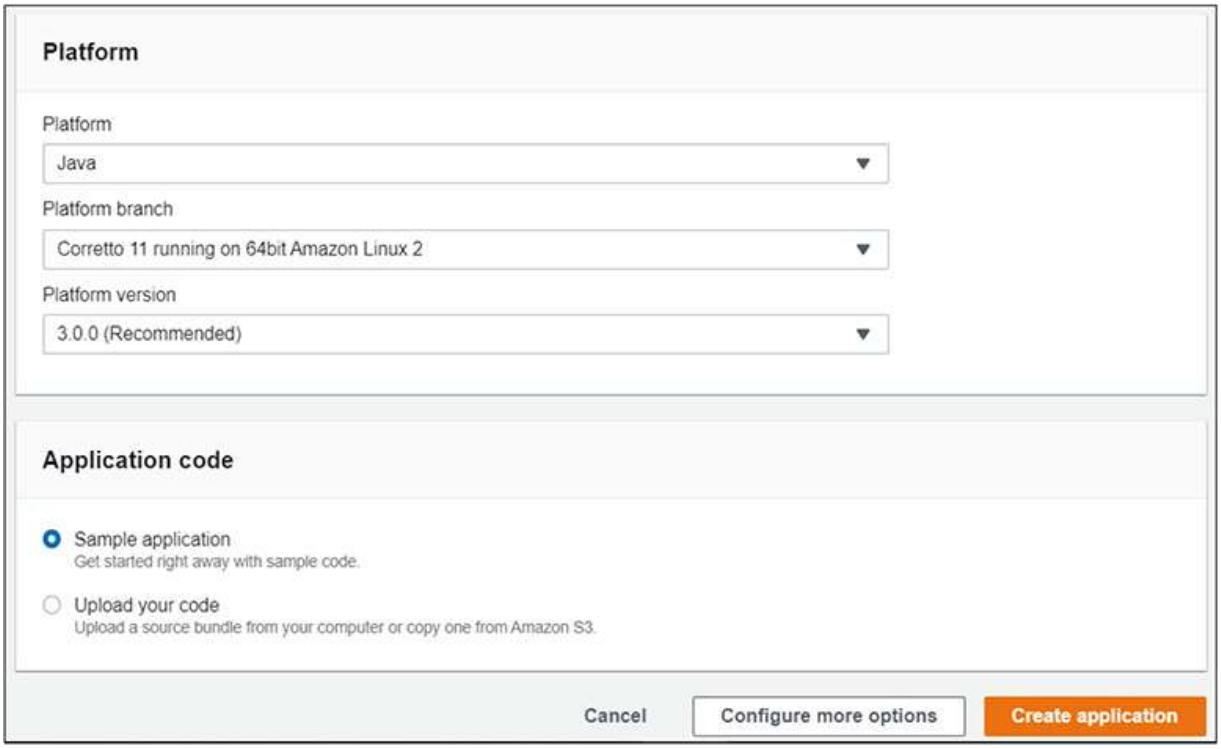
Let us deploy a sample application and explore the steps involved in using AWS Elastic Beanstalk.

1. Log in to your AWS console and navigate to AWS Elastic Beanstalk. You will see the list of environments and applications listed on this page if you provisioned them previously; if not, you will see the welcome page, shown next. Click on the Create Application button to get started.

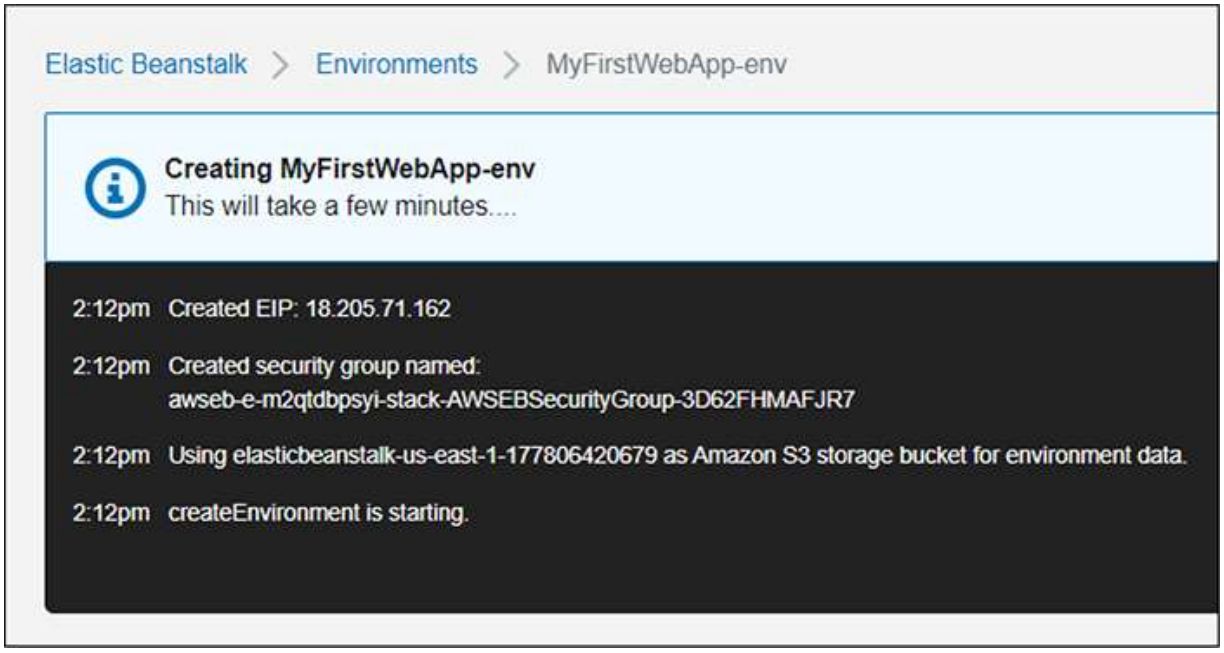


2. You will be provided with two options: Web Server Environment, which is the primary environment, and Worker Environment, which includes the Auto Scaling group and one or more Amazon EC2 instances

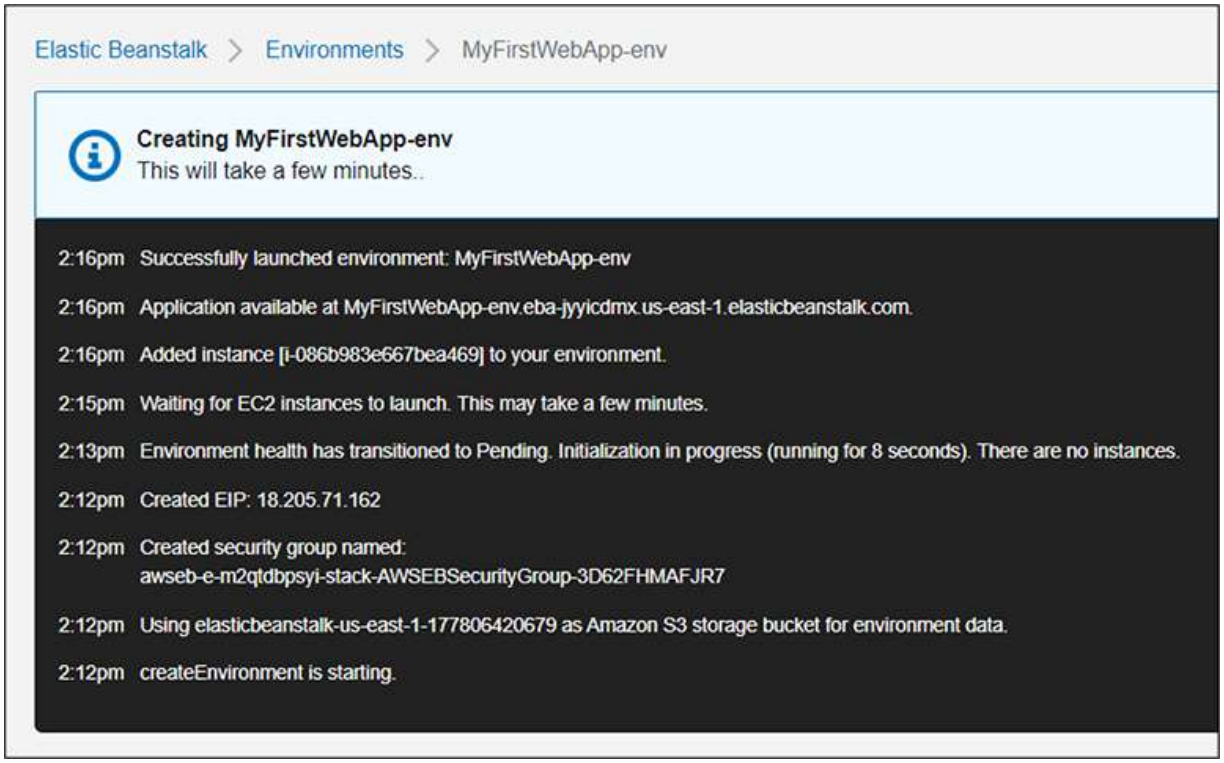
to process additional requests using the SQS queue. Select Web Environment and enter **my-first-web-app** for the application name and **sample** for the tag value. Select Java for the platform and leave the other values at their default settings. Click the Create Application button.



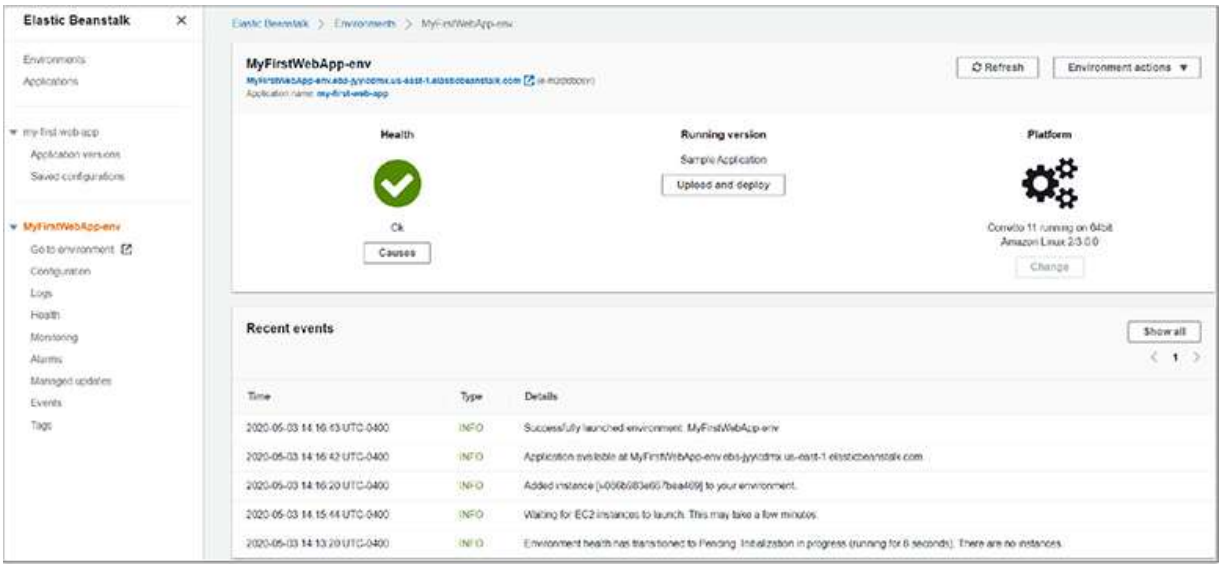
3. The environment provisioning begins by creating a security group and elastic IP. You can see that it uses an Amazon S3 bucket for environment data.



4. After few minutes, Elastic Beanstalk provisions EC2 instances and a load balancer if required. Once ready, you will see the message such as "Successfully launched."



5. The environment page lists the configuration, logs, health, monitoring, alarms, managed updates, events, and tags of your new application. The application name—my-first-web-app—and its URL—MyFirstWebApp-env.eba-jyyicdmx.us-east-1.elasticbeanstalk.com—are displayed at the top of the screen. You can see the health of the application as Ok and the Upload And Deploy button, which can be used to deploy an updated version of your application. All the recent events are displayed at the bottom of the screen. You can access the newly deployed web application using the URL displayed—for example, MyFirstWebApp-env.eba-jyyicdmx.us-east-1.elasticbeanstalk.com.



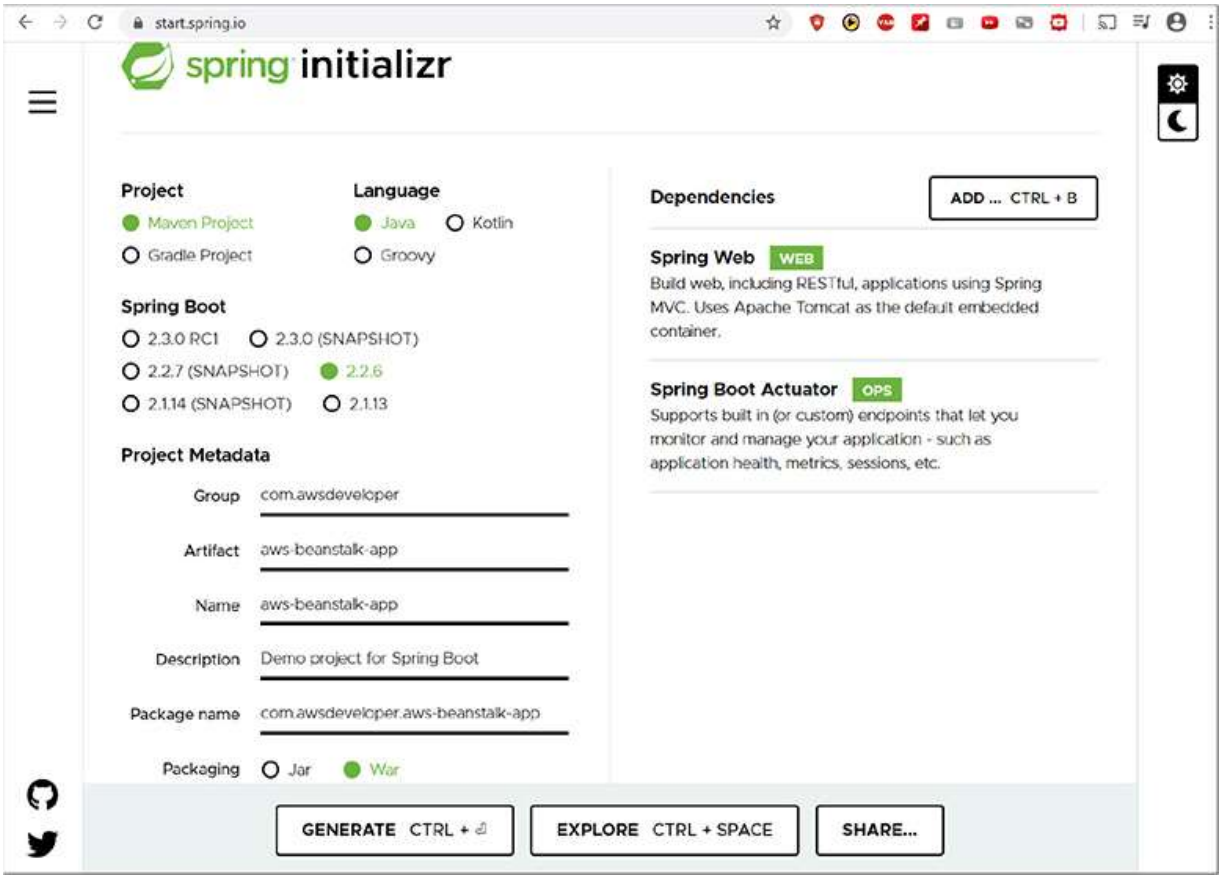
6. You will see the Congratulations message for the sample web application.

You have successfully deployed your first web application using AWS Elastic Beanstalk. Next we will migrate a sample application from your local machine to AWS Elastic Beanstalk.

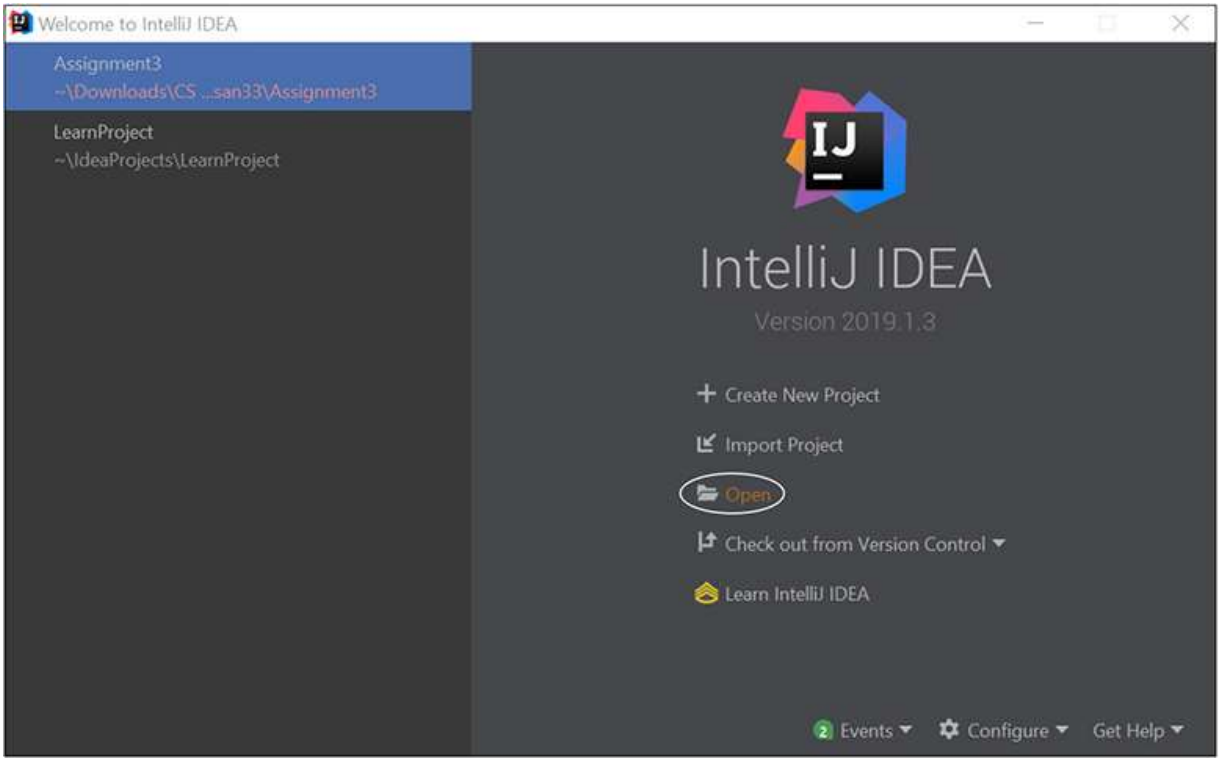
Migrate and Deploy an Application to AWS Elastic Beanstalk

This example uses start.spring.io to get the sample spring boot application code, but you can use any sample application code.

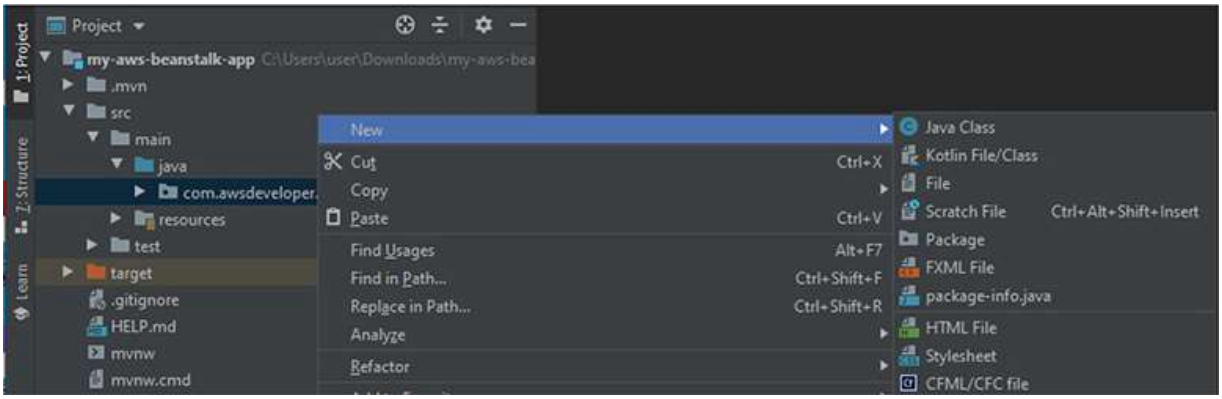
1. Select the Maven Project and Java as the language, with the dependencies Spring and Spring Boot Actuator to see the health using a custom endpoint. For Group Name enter **com.awsdeveloper** and for Maven Artifact enter **aws-beanstalk-app**. Select war packaging and then click on Generate, which will generate the WAR file in ZIP format.



2. Go to the download location of the ZIP file in your local machine and unzip it. This example used IntelliJ IDEA, but you can use any IDE of your choice. Choose Open and select the location of the unzipped WAR folder.



3. Explore the src, main, and java locations and see the standard Java files. Create another class by right-clicking com and selecting a new class for testing.

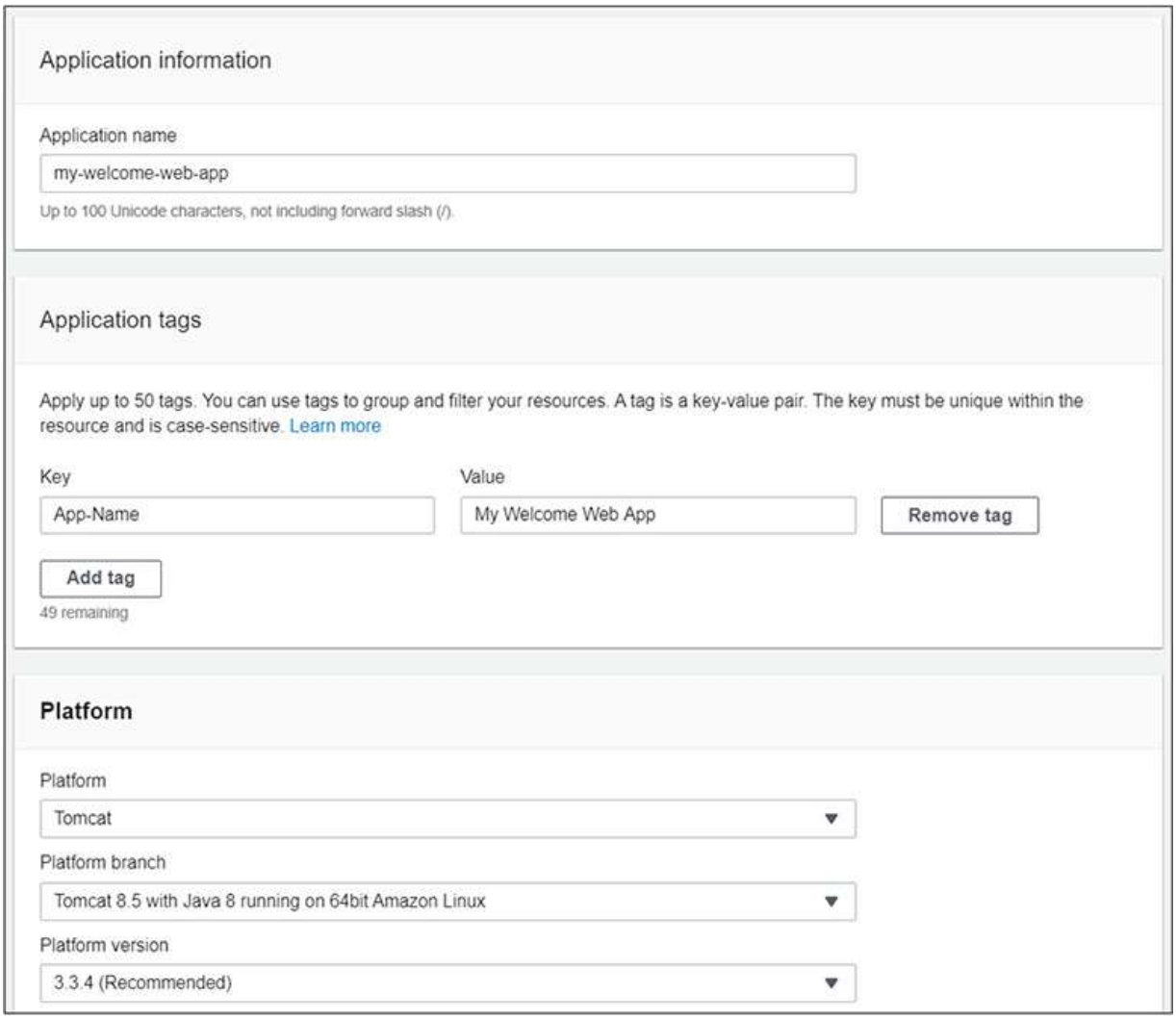


4. Create a new class called **HelloWorld.java** and enter the following code:

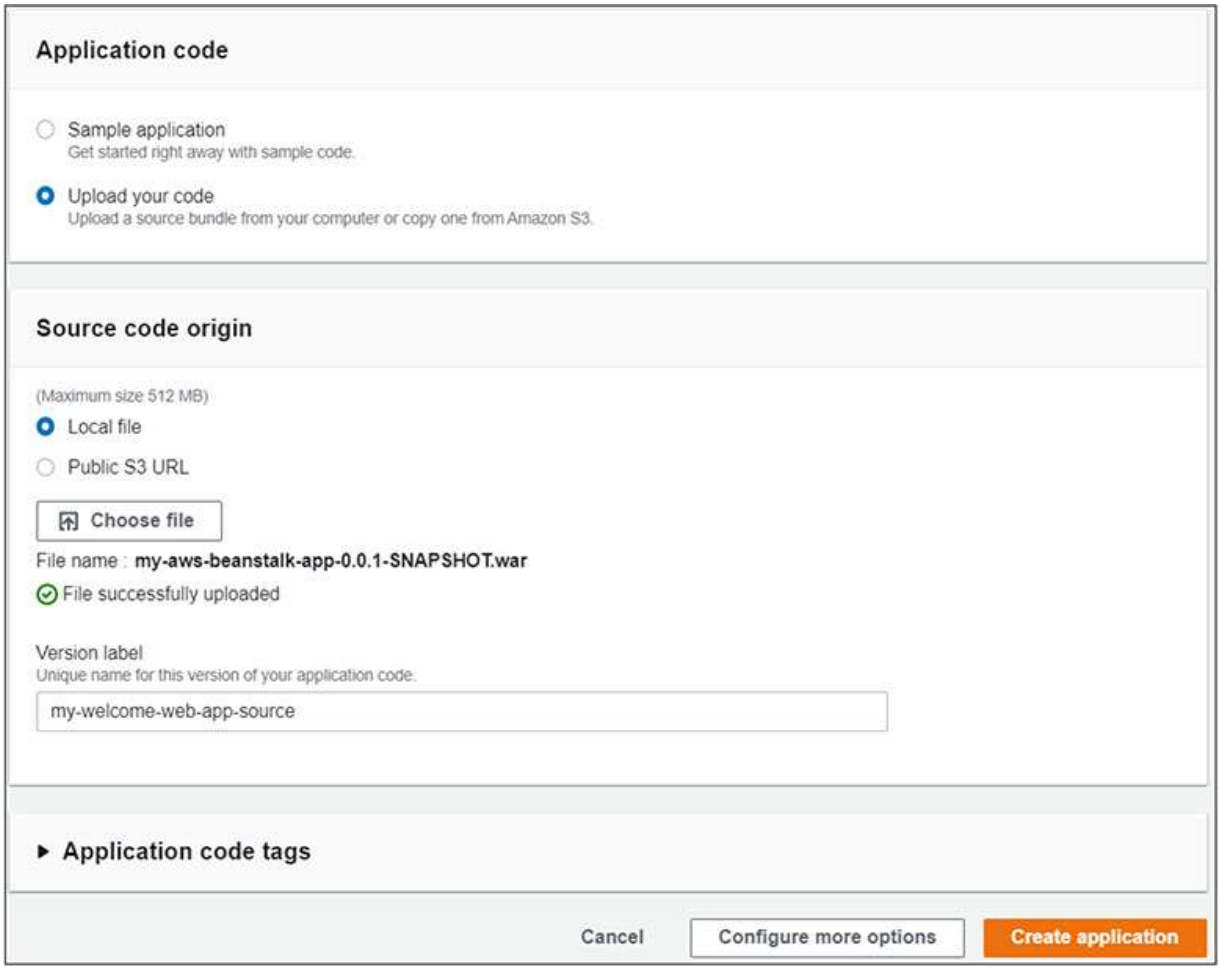
```
package com.awsdeveloper.myawsbeanstalkapp;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/welcome")
public class HelloWorld {
    @GetMapping
    public String welcome() {
        return "Welcome to My AWS Elastic Beanstalk Application !!!";
    }
}
```

Then click the Maven on right side and select Package to create an updated WAR file. The WAR file location will be displayed at the bottom along with a build successful message.

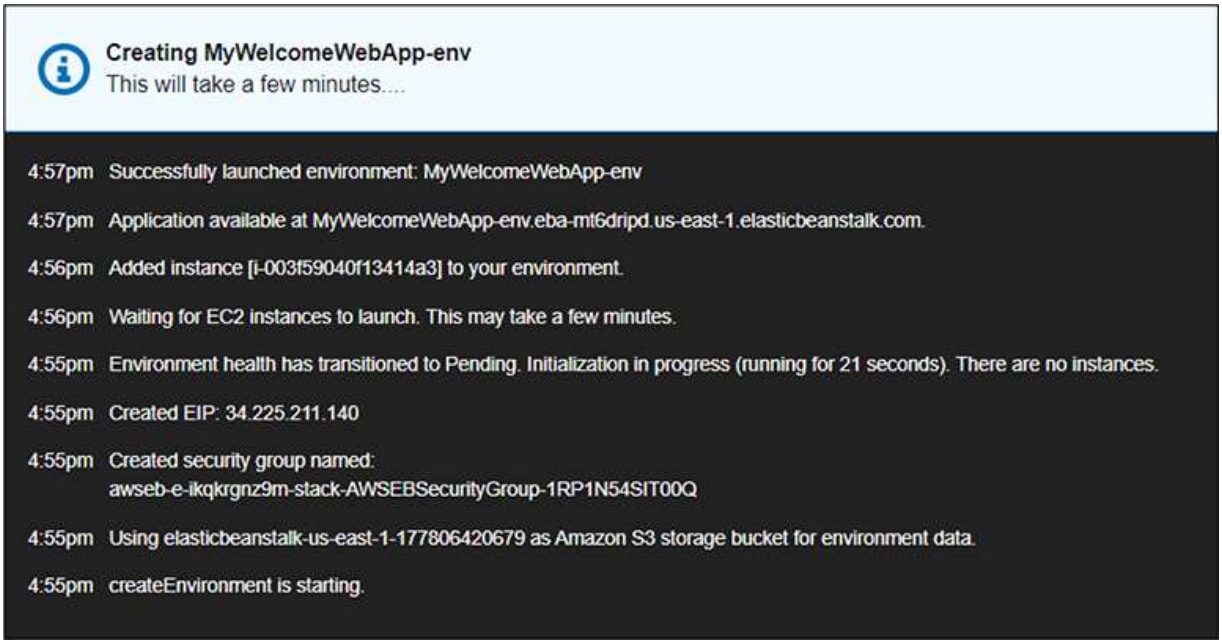
5. Log in to your AWS console, navigate to the AWS Elastic Beanstalk service, and click on the Create Application button.
6. Enter **my-welcome-web-app** for the application name and add tags, which is a best practice to follow when creating any AWS resource. Select Tomcat for the platform and leave the other options at the default settings.



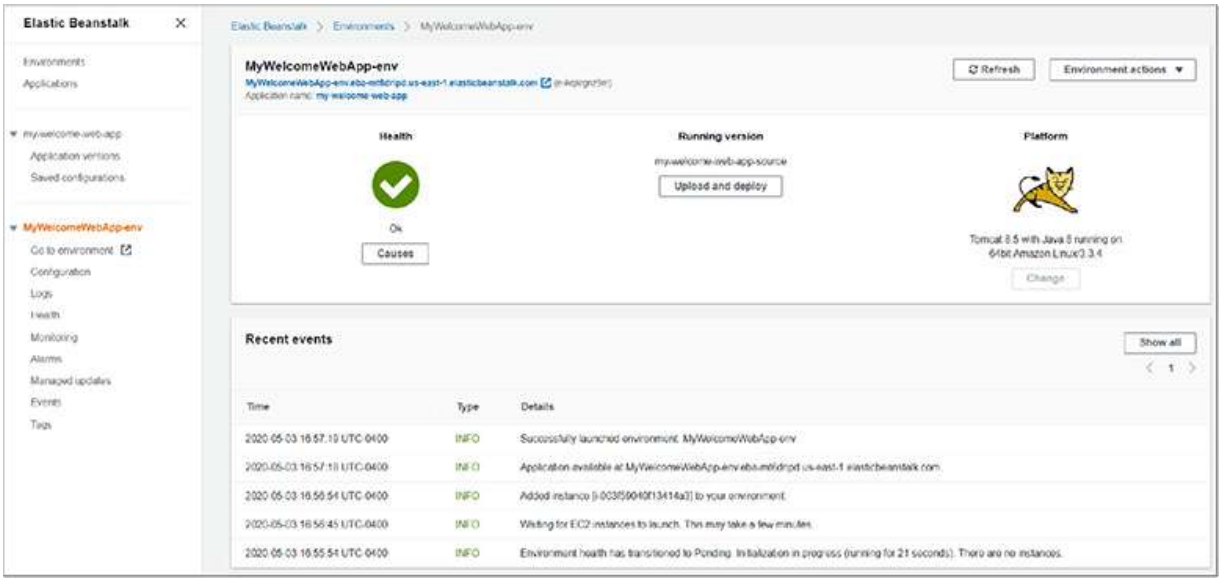
7. For the Application Code field, select the Upload Your Code option, which will display the source code origin. You can either use Amazon S3 bucket or upload your local file. Once you upload the WAR file that was generated in the previous step, you will see a successful upload message. Click the Create Application button to get started with AWS Elastic Beanstalk.



Within a few minutes, you will start seeing the resources like Elastic IP, EC2 instance, security group, and load balancer, if required. The application URL will also be displayed, along with the new application environment name.



Once the provisioning is complete, it will display the environment page with health and platform details. You can deploy an updated version of your application using the Upload And Deploy button. You can view all the recent events at the bottom of the page. Also you have the option to explore configuration, logs, health, monitoring, alarms, managed updates, and events and tags.



8. Enter your new application URL—for example, **mywelcome.eba-mt6dripd.us-east-1.elasticbeanstalk.com/welcome**—in your browser. The welcome message that we set up in our HelloWorld.java will be displayed here.

You can also see your application's health status using actuator in the URL—for example, mywelcome.eba-mt6dripd.us-east-1.elasticbeanstalk.com/welcome.

You have successfully created a Java Spring boot application and built it using Maven before packaging it as a WAR file. You then migrated the application to AWS Elastic Beanstalk using the upload option and deployed it successfully.

Chapter Review

This chapter began by explaining how developers can use AWS Elastic Beanstalk to easily deploy their application instead of spending time provisioning and managing its infrastructure. First, the chapter provided the steps to successfully create and deploy a sample Java application in AWS Elastic Beanstalk. It then explained the process to create a Java Spring boot application locally and upload it to AWS Elastic Beanstalk to build the web application.

Exercise

The following exercises will help you practice deleting the environments that you created in this chapter. There is no charge to use AWS Elastic Beanstalk, but you will need to pay for the underlying resources. You need to create an AWS account, as explained earlier, to perform these exercises. You can use

the Free Tier when launching AWS resources, but make sure to terminate them at the end.

Exercise 21-1: Delete the AWS Elastic Beanstalk Environment Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in and then navigate to the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Environments.
4. Select the environments that you created in this chapter.
5. Choose Actions, navigate to the bottom, and select Terminate.
6. Wait for a couple of minutes and then verify the environments are terminated.
7. Now select Applications and delete the applications that you created in this chapter.
8. Navigate to EC2 and verify the instances are terminated and the EIPs are released.

Questions

The following questions will help you gauge your understanding of the contents in this chapter. Read all the answers carefully because there might be more than one correct answer. Choose the best responses for each question.

1. Your developers are complaining that they are spending more time troubleshooting the application infrastructure then developing applications. Which AWS service can help your developers concentrate only on application development?

   A. Amazon EC2

   B. Amazon S3

   C. AWS Elastic Beanstalk

D. Amazon DynamoDB

2. Your company security policy mandates that your application should not be accessible publicly. How can you make your AWS Elastic Beanstalk application private?

   A. You can use Amazon VPC, security group rules, network ACLs, and custom route tables to make it private

   B. You cannot make AWS Elastic Beanstalk applications private, so you need to use Amazon EC2

   C. By default, the AWS Elastic Beanstalk applications are private

   D. You can use AWS IAM to deny public access

3. Which of the following languages are supported in AWS Elastic Beanstalk? (Choose all that apply.)

   A. Java

   B. Node.js

   C. .NET

   D. Python

4. How do you provision EC2 instances, RDS databases, and a load balancer for your AWS Elastic Beanstalk application?

   A. Use the AWS CloudFormation template

   B. Use Terraform to deploy the infrastructure

   C. Use Ansible to provision those resources

   D. AWS Elastic Beanstalk automatically provisions the infrastructure

5. Your AWS Elastic Beanstalk application needs a database to store data. Which of the following databases are supported in AWS Elastic Beanstalk? (Choose two.)

   A. Amazon SageMaker

   B. Amazon RDS

   C. Amazon DynamoDB

   D. Amazon Cognito

6. Your company deployed applications in AWS Elastic Beanstalk and asked you to manage the platform updates. How can you manage the

underlying platform to make sure it runs on the latest version?

A. Opt in to automatically update the platform using AWS Elastic Beanstalk

B. Set up a monitoring alarm to receive a notification when the new updates are available

C. The AWS Elastic Beanstalk underlying platform cannot be updated

D. You can use AWS CloudFormation to update the application platform

7. Your company is planning to migrate and deploy many applications in AWS Elastic Beanstalk. They asked you to find the cost of deploying 1,000 applications. Which is of the following is correct regarding the cost of AWS Elastic Beanstalk?

A. It uses a pay-as-you-go pricing model

B. You can use a reserved pricing model

C. There is no cost; you pay only for the underlying AWS resources

D. You can use a spot-pricing model

8. You have an application running in production that was deployed using AWS Elastic Beanstalk, and you added a new feature to your application in your local development environment. How can you deploy your updated application in AWS Elastic Beanstalk?

A. You can use the Upload And Deploy button on the environment overview page

B. You need to delete the existing application and deploy the updated application in AWS Elastic Beanstalk

C. You cannot update your application in AWS Elastic Beanstalk

D. You need to deploy the updated application in another region and use a load balancer

9. A company deployed many applications in AWS Elastic Beanstalk. One of the applications needs to be scaled using auto-scaling with minimum and maximum instances and a load balancer. How can you achieve this requirement?

A. You need to delete and create the application again with a load balanced type

B. You need to use Amazon EC2 instances to auto-scale your application

C. You need to deploy another AWS Elastic Beanstalk and use Elastic Load Balancer

D. You can edit the capacity configuration and update the environment type to load balanced by providing the required number of instances

10. Your company has deployed critical business application in AWS Elastic Beanstalk and tasked you with monitoring the application. How can you monitor this application and set up an alarm to alert you?

A. You need to set up a cron job to monitor the application

B. You need to set up Amazon CloudTrail for monitoring and to provide the alarm

C. You can add an alarm from the AWS Elastic Beanstalk Environments page

D. You cannot set up an alarm for the AWS Elastic Beanstalk application

Answers

1. **C.** AWS Elastic Beanstalk can be used to save your developers from carrying out infrastructure provisioning.
2. **A.** You can use Amazon VPC, security group rules, network ACLs, and custom route tables to make it private.
3. **A, B, C, D.** Java, Node.js, .NET, Python, PHP, Ruby, and Go are supported languages.
4. **D.** AWS Elastic Beanstalk automatically provisions the infrastructure for your application.
5. **B, C.** Amazon RDS and Amazon DynamoDB are supported.
6. **A.** You can opt in to automatically update the platform using AWS Elastic Beanstalk.
7. **C.** There is no cost to use AWS Elastic Beanstalk; you pay only for the underlying AWS resources.

8. **A.** You can use the Upload And Deploy option on the environment overview page to deploy the updated application.
9. **D.** You can edit the capacity configuration and update the environment type to load balanced by providing the required number of instances.
10. **C.** You can add an alarm from the AWS Elastic Beanstalk Environments page.

Additional Resources

- **AWS Elastic Beanstalk** The recommended documentation for any AWS service, including AWS Elastic Beanstalk, is the official AWS documentation, where you can always get the most up-to-date information.
  `https://docs.aws.amazon.com/elastic-beanstalk/index.html`
- **AWS Elastic Beanstalk Blog** This is the official blog of AWS Elastic Beanstalk, which has all the latest information about different use cases in one place.
  `https://aws.amazon.com/blogs/devops/tag/elastic-beanstalk/`

CHAPTER 22

Migrating Your Application and Database to AWS

In this chapter, you will learn

- Application migration
- Database migration

This chapter provides experience in performing real-time migration of an application and database from on-premises to AWS Cloud.

AWS Migration

Many organizations are migrating their on-premises workload, such as their business applications and databases, to the cloud. This chapter provides you with hands-on experience that will help you in the real world when performing migration, and includes information for the AWS certification. Enterprises follow many cloud migration strategies, including the 6R (Rehosting, Re-platforming, Repurchasing, Refactoring, Retain, and Retire) migration strategy. The first step in any migration strategy is the discovery of existing resources. The resources might be spread across many datacenters across the country, or across the globe, based on your business. Once you complete the discovery phase by finding all the resources and their dependencies, you need to split them into at least three categories, such as easy, medium, and complex. The easy resources might be simple applications and databases with less dependency, such as your development and proof-of-concept (POC) environments. The medium resources are applications and

databases with fewer dependencies, such as your preproduction and user acceptance testing environments. The complex resources are often your business-critical applications and databases, which have many dependencies and a large impact on your business. Based on these classifications, you need plan your cloud migration strategy. Enterprises migrate the easy workloads first and gain experience with positive reinforcement of cloud migration. Next you can either migrate your medium or complex workload based on your experience and requirements.

The 6Rs of cloud migration and their advantages are as follows:

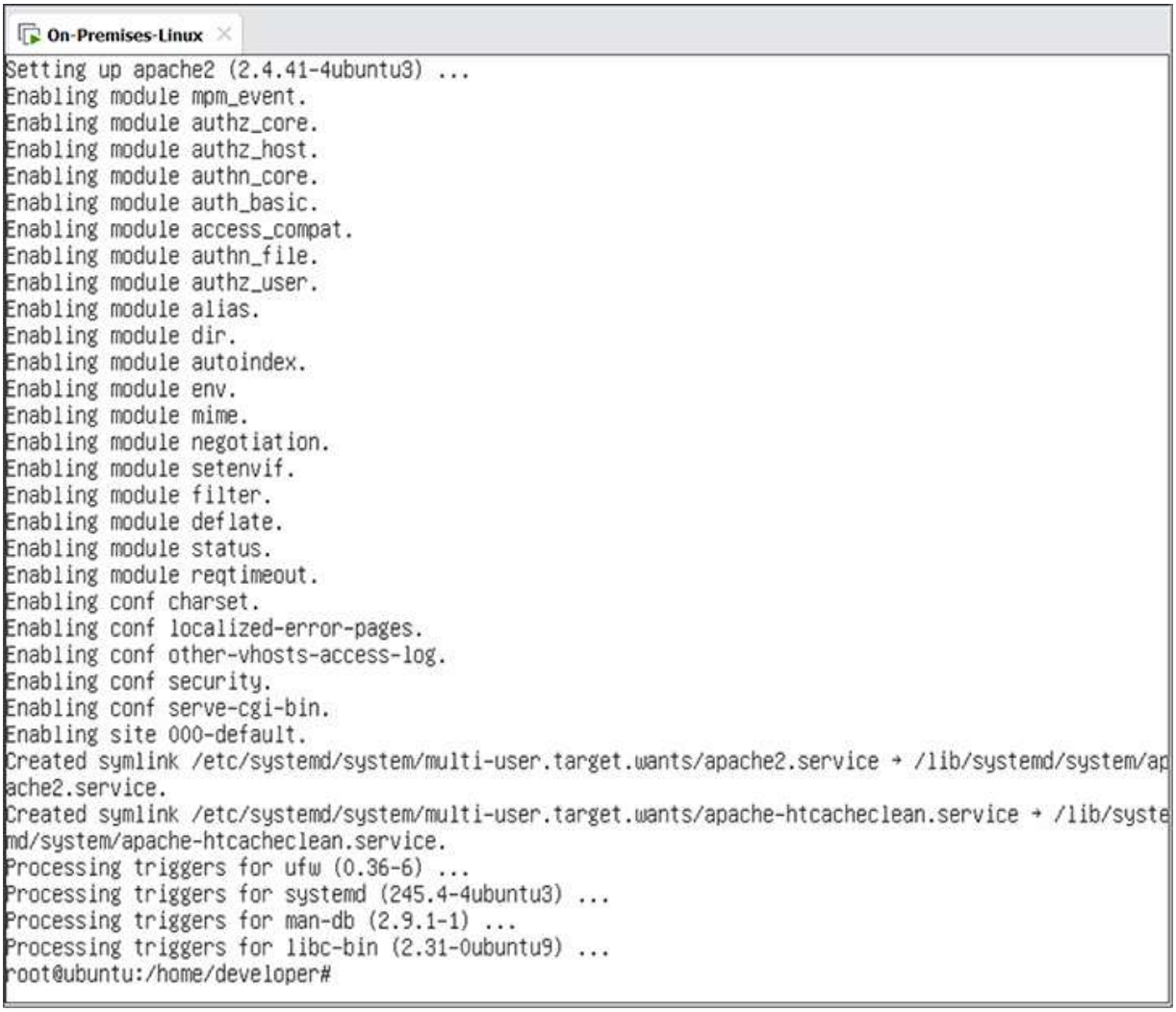
- **Rehosting** This is also called "lift and shift" because here you just take the image copy of your on-premises server and migrate it to AWS Cloud using AWS VM Import. It is the easiest migration option, since you are not changing the hosted application or database and there is either no change or few changes to this configuration. This option will provide the benefit of the cloud in terms of availability, scalability, and cost, but it does not offer the full potential of the native cloud and tools.
- **Re-platforming** This is also called "lift, tinker, and shift" because you either migrate your application to Amazon Elastic Beanstalk or migrate your database to Amazon Relational Database Service (Amazon RDS). It will also provide the benefit of the cloud, but migrating large monolithic complex applications or databases requires more work.
- **Repurchasing** This option involves purchasing and migrating to Software as a Service (SaaS) platforms such as Salesforce for customer relationship management (CRM) and Workday for human resource (HR) applications.
- **Refactoring** This is also called "re-architecting" because it involves changing your big monolithic applications into small microservices or using a serverless architecture to improve business agility. It is the most expensive and complex migration strategy, but you get the full benefit of the cloud.
- **Retain** This is also called "revisit" because you do nothing and revisit the issue at a later point in time. There are applications and databases that do not need to be migrated to the cloud because of a specific business requirement or a compliance and regulations requirement.

- **Retire** You will be surprised to see how many resources are underutilized or not at all utilized in your datacenter, so you can shut down and move these out of your datacenter.

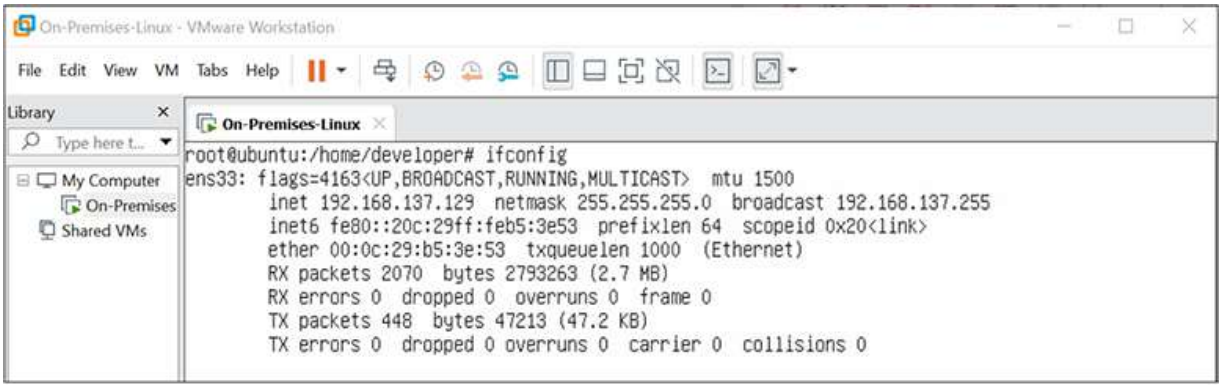
Application Migration

This section will take you through the step-by-step journey of "lift and shift" that you can practice using your AWS free account from your laptop or desktop. First you need to install VMware Workstation on your laptop or desktop and then create a virtual machine with any operating system. I created a Ubuntu Linux virtual machine and installed an Apache web server.

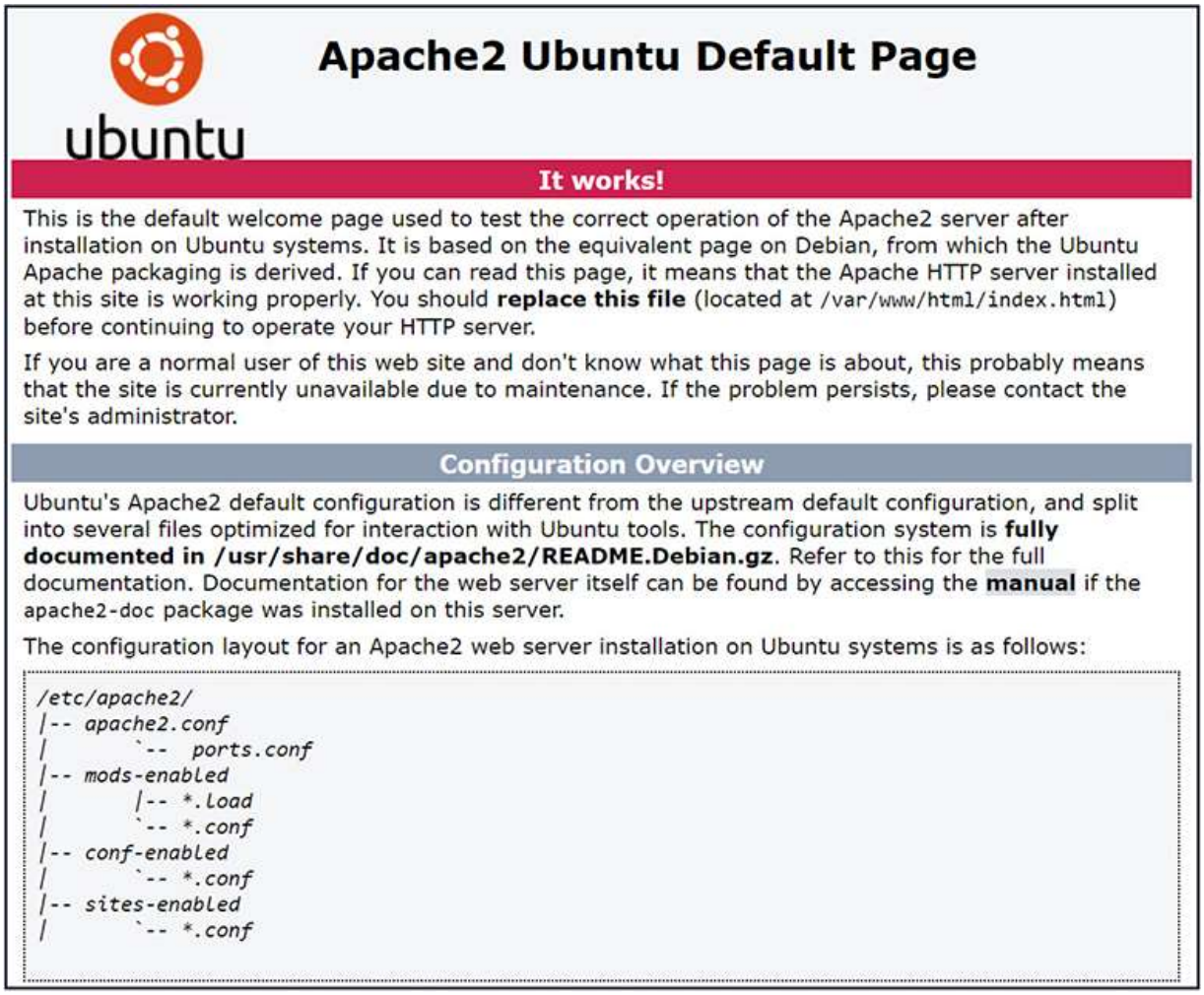
Log in to your Linux virtual machine and type **sudo apt install apache2** for Ubuntu or **sudo yum install apache2** for RedHat Linux. The Apache web server will be installed.



You can find the IPv4 IP address of your virtual machine using the ifconfig command. You need to note the IP address after the inet value to use in the next step.



Open your browser and type the IPv4 IP address you noted in the previous step and press ENTER. You can see the default Apache page.

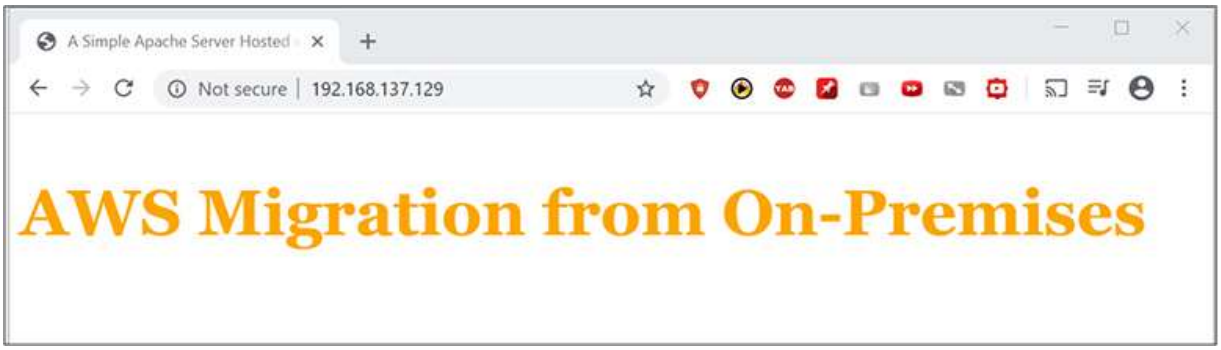


Open the /var/www/html/index.html file using Vim, and replace its content with the following sample HTML to display a custom page for the migration experiment.

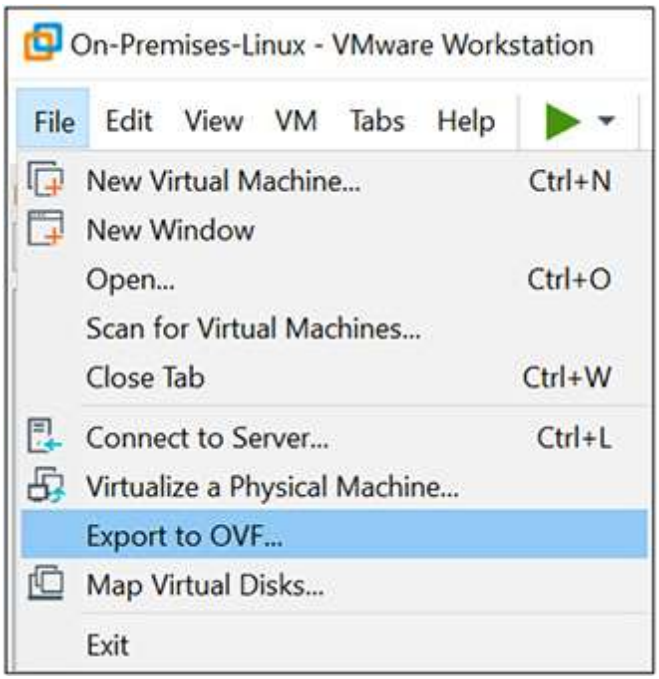
```
<!DOCTYPE html>
<html language="english">
   <head>
      <meta charset="utf-8">
      <title>
         A Simple Apache Server Hosted on On-Premises
      </title>
   </head>
```

```
   <body>
      <h1 align="left">
        <font face="Georgia" size="12" color="orange"><span style="font-
size:100%">AWS Migration from On-Premises</span></font>
      </h1>
     </body>
</html>
```

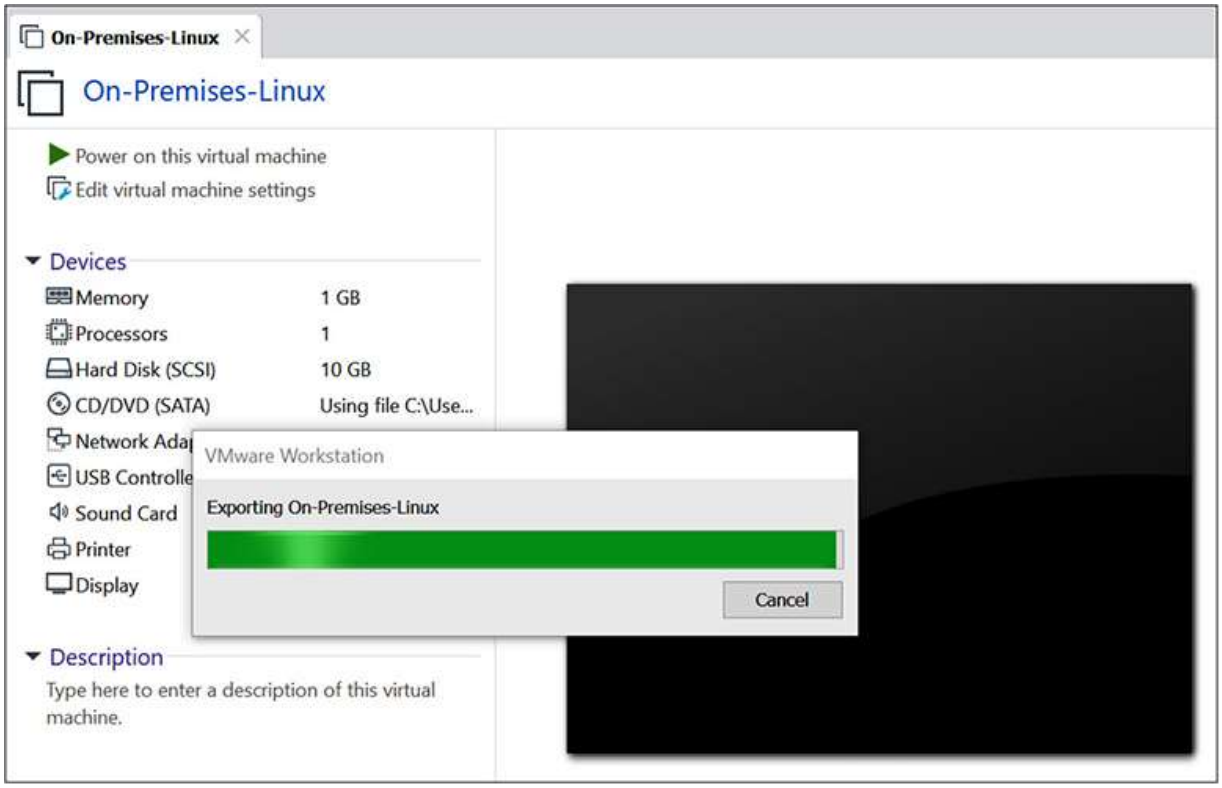
After saving the updated index.html file, you can either refresh the browser or type the IP address in another browser and press ENTER. You will see the custom page with "AWS Migration from On-Premises" in orange.



Now we are ready for the lift and shift migration. Select Export To OVF from the File menu of your VMware workstation to export the VM image for the migration.



Three files are created, and we need the file with the .vmdk extension for our migration to AWS Cloud. This will take few minutes, based on the size of your virtual machine, and it exports the image in .vmdk format.



Log in to your AWS Management Console, navigate to the IAM service, and select Create Role. You need to create a vmimport role with the following trust policy, which provides the STS assume role access:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "vmie.amazonaws.com",
          "ec2.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:Externalid": "vmimport"
        }
      }
    }
  ]
}
```

Once the role is created, attach the following policy to your newly created vmimport role, which grants Amazon S3 get access and Amazon EC2 copy image access:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::'on-premises-migration'",
            "arn:aws:s3:::'on-premises-migration'/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:ModifySnapshotAttribute",
            "ec2:CopySnapshot",
            "ec2:RegisterImage",
            "ec2:Describe*"
         ],
         "Resource":"*"
      }
   ]
}
```

Once the policy is successfully attached to your vmimport role, navigate to the Amazon S3 console and create a bucket—for example, on-premises-migration—to upload the .vmdk file that was exported from the VMware workstation. Now choose the bucket and upload your .vmdk file.

Create the following .json file as containers.json in your local machine:

```
[
  {
    "Description": "My On-Premises Server",
    "Format": "vmdk",
    "UserBucket": {
        "S3Bucket": "on-premises-migration",
        "S3Key": "On-Premises-Linux-disk1.vmdk"
    }
}]
```

Execute the following command from your local machine using the AWS Command Line Interface (AWS CLI) by providing an appropriate file location for the .json file: $ aws ec2 import-image

```
--description "My On-Premises server VM"
--disk-containers "file://C:\Users\user\Downloads\containers.json"
```

As soon as you enter the ec2 import-image command, you can see the status as “active” and the status message as “pending.”

```
C:\Users\user>aws ec2 import-image --description "My On-Premises server VM" --disk-containers "file://C:\Users\user\Downloads\containers.json"
{
    "Description": "My On-Premises server VM",
    "ImportTaskId": "import-ami-053b01f85bea53c8e",
    "Progress": "2",
    "SnapshotDetails": [
        {
            "DiskImageSize": 0.0,
            "Format": "VMDK",
            "UserBucket": {
                "S3Bucket": "on-premises-migration",
                "S3Key": "On-Premises-Linux-disk1.vmdk"
            }
        }
    ],
    "Status": "active",
    "StatusMessage": "pending"
}

C:\Users\user>
```

After a few minutes, the status message will change to “validating” when AWS starts validating your image. You can see the status of your import-image using the aws ec2 describe-import-image-tasks command.

```
            }
        }
    ],
    "Status": "active",
    "StatusMessage": "pending"
}

C:\Users\user>aws ec2 describe-import-image-tasks
{
    "ImportImageTasks": [
        {
            "Description": "My On-Premises server VM",
            "ImportTaskId": "import-ami-053b01f85bea53c8e",
            "Progress": "19",
            "SnapshotDetails": [],
            "Status": "active",
            "StatusMessage": "converting"
        },
        {
            "Description": "My On-Premises server VM",
            "ImportTaskId": "import-ami-0fdd07b2eacef0f05",
            "Progress": "2",
            "SnapshotDetails": [],
            "Status": "active",
            "StatusMessage": "validating"
        }
    ]
}

C:\Users\user>
```

After a few minutes, the status message changes to “converting.” The image validation is complete, and AWS starts converting your image.

```
C:\Users\user>aws ec2 describe-import-image-tasks
{
    "ImportImageTasks": [
        {
            "Description": "My On-Premises server VM",
            "ImportTaskId": "import-ami-053b01f85bea53c8e",
            "Progress": "19",
            "SnapshotDetails": [
                {
                    "Description": "My On-Premises Server",
                    "DiskImageSize": 1276295168.0,
                    "Format": "VMDK",
                    "Status": "active",
                    "UserBucket": {
                        "S3Bucket": "on-premises-migration",
                        "S3Key": "On-Premises-Linux-disk1.vmdk"
                    }
                }
            ],
            "Status": "active",
            "StatusMessage": "converting"
        },
        {
            "Description": "My On-Premises server VM",
            "ImportTaskId": "import-ami-0fdd07b2eacef0f05",
            "Progress": "19",
            "SnapshotDetails": [],
            "Status": "active",
            "StatusMessage": "converting"
        }
    ]
}

C:\Users\user>
```

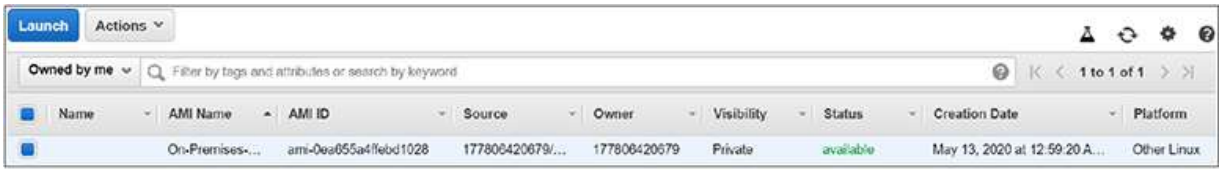
The import status message will change to “updating” once the image conversion is complete.

```
C:\Users\user>aws ec2 describe-import-image-tasks
{
    "ImportImageTasks": [
        {
            "Description": "My On-Premises server VM",
            "ImportTaskId": "import-ami-053b01f85bea53c8e",
            "Progress": "20",
            "SnapshotDetails": [
                {
                    "Description": "My On-Premises Server",
                    "DiskImageSize": 1276295168.0,
                    "Format": "VMDK",
                    "Status": "completed",
                    "UserBucket": {
                        "S3Bucket": "on-premises-migration",
                        "S3Key": "On-Premises-Linux-disk1.vmdk"
                    }
                }
            ],
            "Status": "active",
            "StatusMessage": "updating"
        },
        {
            "Description": "My On-Premises server VM",
            "ImportTaskId": "import-ami-0fdd07b2eacef0f05",
            "Progress": "22",
            "SnapshotDetails": [
                {
                    "Description": "My On-Premises Server",
                    "DiskImageSize": 1276295168.0,
                    "Format": "VMDK",
                    "Status": "completed",
                    "UserBucket": {
                        "S3Bucket": "on-premises-migration",
                        "S3Key": "On-Premises-Linux-disk1.vmdk"
                    }
                }
            ],
            "Status": "active",
            "StatusMessage": "updating"
        }
    ]
}
```

The import task will end when the status is changed to “completed.”

```
C:\Users\user>aws ec2 describe-import-image-tasks
{
    "ImportImageTasks": [
        {
            "Description": "My On-Premises server VM",
            "ImportTaskId": "import-ami-053b01f85bea53c8e",
            "Progress": "27",
            "SnapshotDetails": [
                {
                    "Description": "My On-Premises Server",
                    "DiskImageSize": 1276295168.0,
                    "Format": "VMDK",
                    "Status": "completed",
                    "UserBucket": {
                        "S3Bucket": "on-premises-migration",
                        "S3Key": "On-Premises-Linux-disk1.vmdk"
                    }
                }
            ],
            "Status": "active",
            "StatusMessage": "updating"
        },
        {
            "Description": "My On-Premises server VM",
            "ImportTaskId": "import-ami-0fdd07b2eacef0f05",
            "SnapshotDetails": [
                {
                    "Description": "My On-Premises Server",
                    "DiskImageSize": 1276295168.0,
                    "Format": "VMDK",
                    "Status": "completed",
                    "UserBucket": {
                        "S3Bucket": "on-premises-migration",
                        "S3Key": "On-Premises-Linux-disk1.vmdk"
                    }
                }
```

Navigate to the Amazon EC2 service page in the AWS Management Console and click on AMIs on the left pane below Images. You will see the new image that you imported. You need to select Owned By Me below the Launch button if the image is not displayed.



You have successfully migrated your application server to AWS, and you can create the server using the Launch button. In few seconds, your EC2 instance will be provisioned. You can view the EC2 instance from the EC2 instance page by clicking on Instances on the left navigation pane. Note the IPv4 public IP to launch the application and test the migration.

Enter the public IP in your browser, and you will see the custom message "AWS Migration from On-Premises" in orange. Open port 80 for the HTTP protocol to allow the ingress traffic in the EC2 security group if you get an access error.

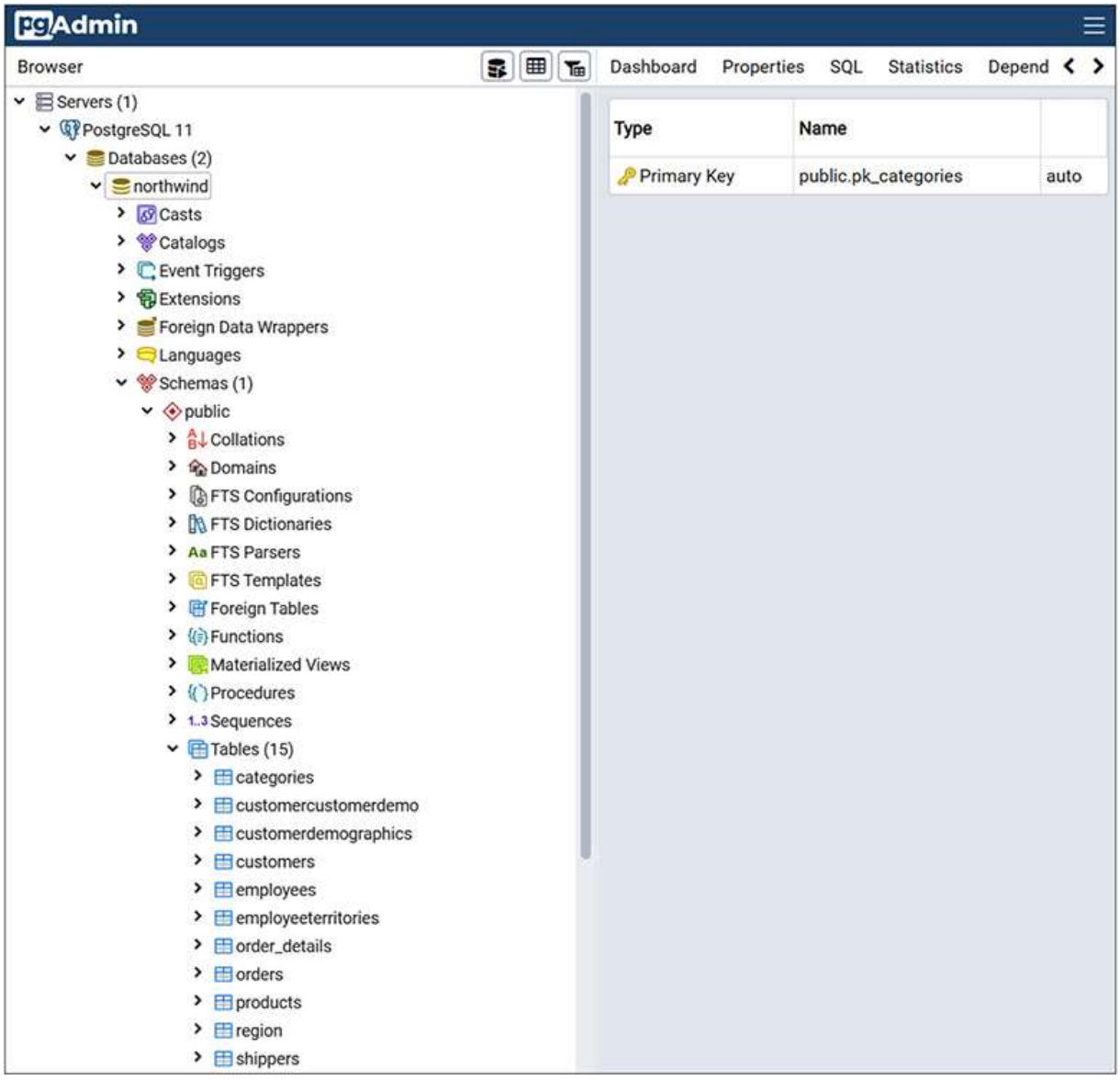
You have migrated the server using the lift and shift strategy and tested it successfully. In the real world, you need to automate the process to migrate hundreds or thousands of application servers to AWS Cloud.

AWS Server Migration Service

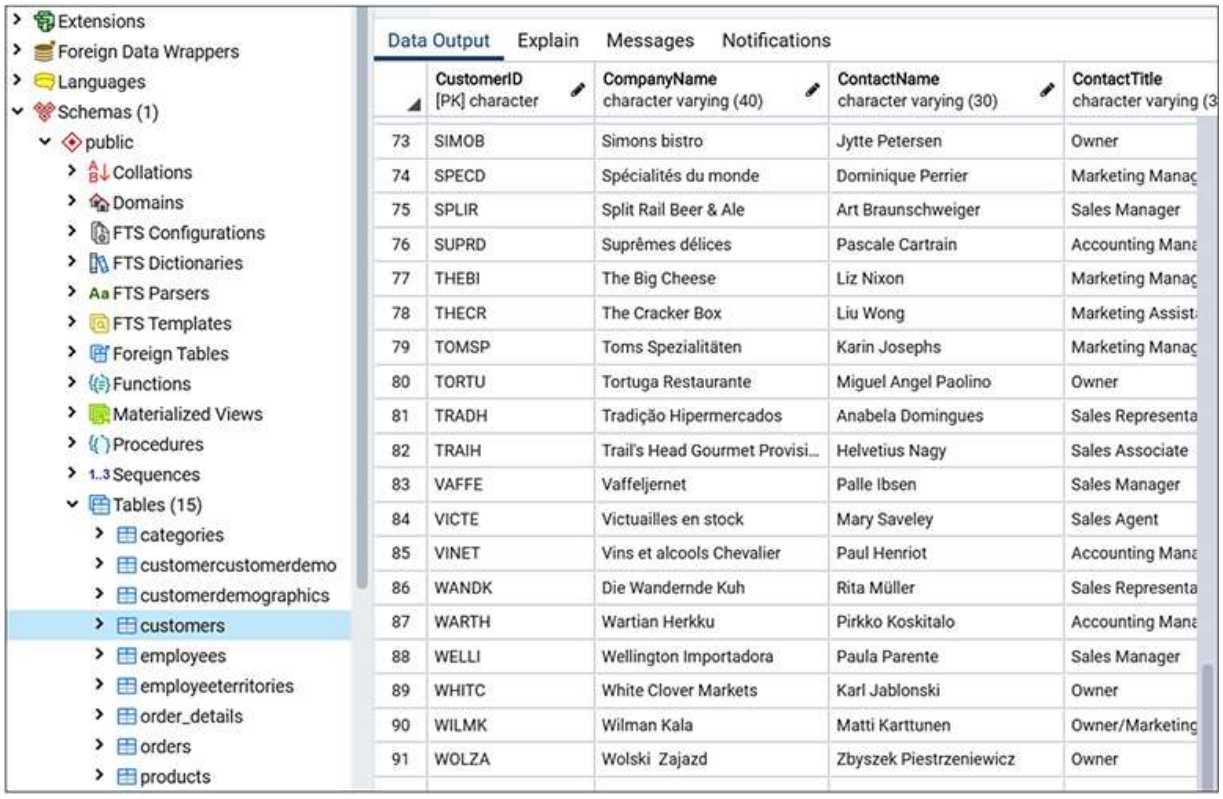
You can also use AWS Server Migration Service or third-party migration tools to automate, schedule, and migrate your thousands of on-premises workloads to AWS Cloud. AWS Server Migration Service can be used to migrate instances from VMware-vSphere and Windows-Hyper-V to AWS. It provides automated, live replication that you can see from the AWS Management Console. AWS recommends using AWS Server Migration Service instead of using EC2 VM Import for migration, and it allows replication of your on-premises servers to AWS for up to 90 days for each server.

Database Migration

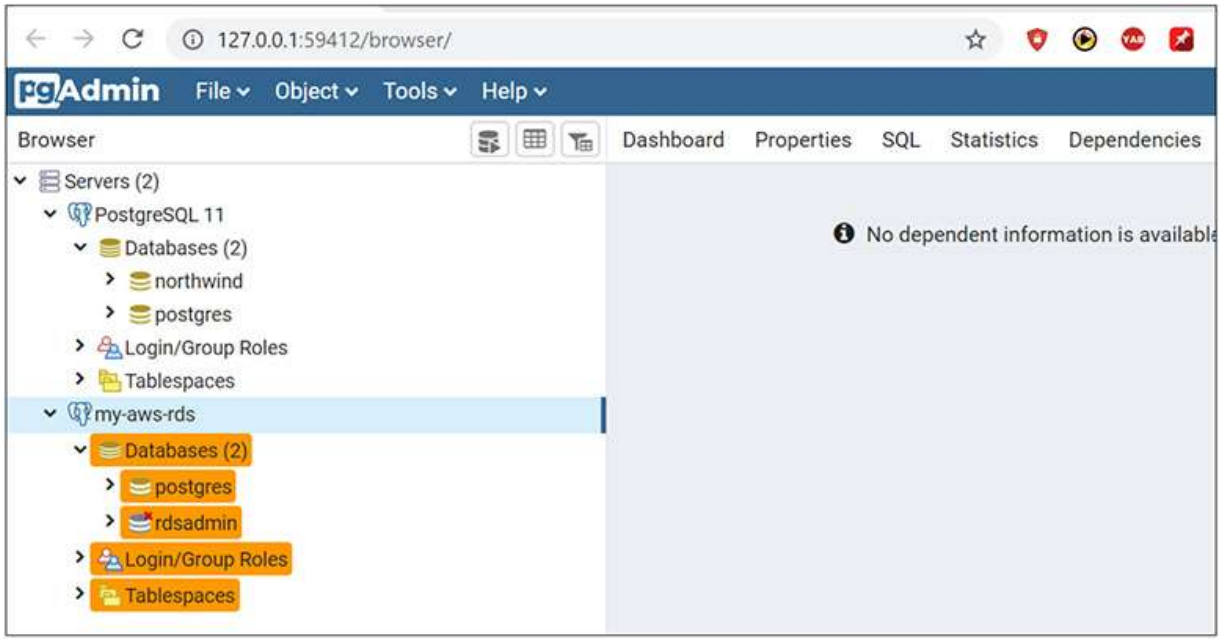
This section will demonstrate the “lift, tinker, and shift” strategy to migrate a PostgreSQL database from on-premises to AWS RDS. I have PostgreSQL database installed on my local machine, which has a database called northwind with 15 tables.



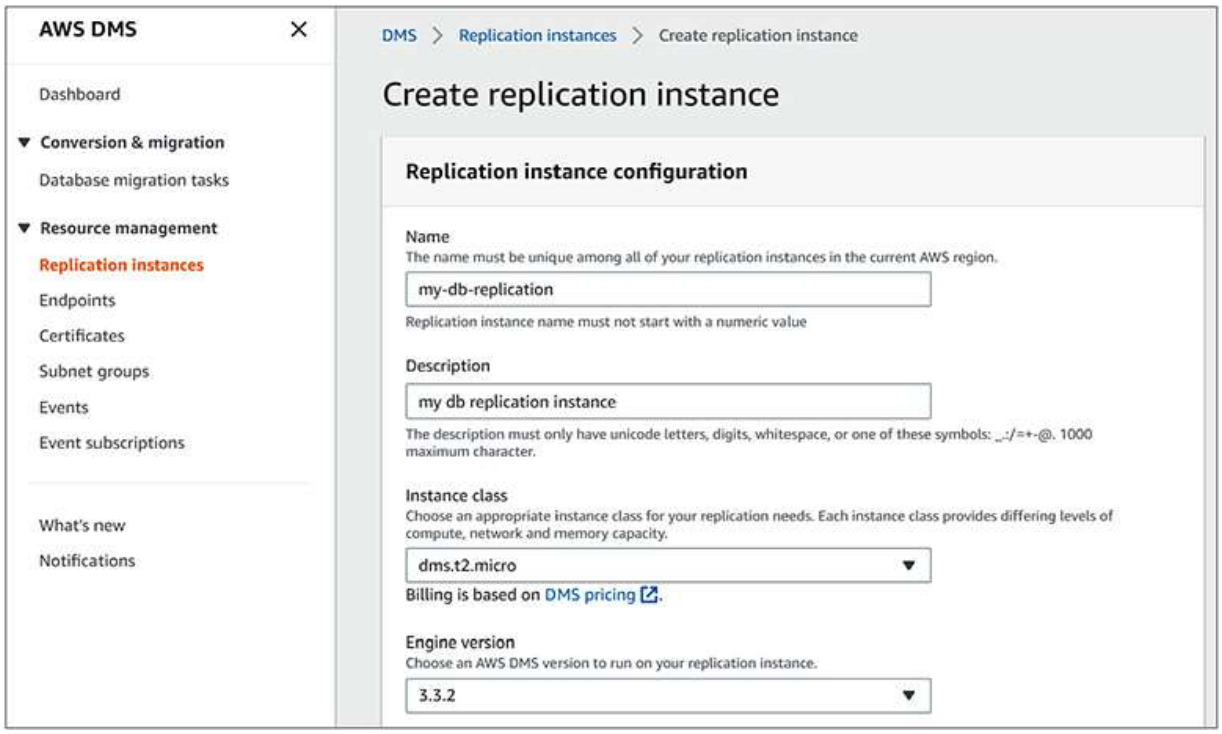
When I run a select query on the customer table in the northwind database, it displays 91 rows. I am planning to use AWS Database Migration Service to migrate the northwind PostgreSQL database into the Amazon RDS PostgreSQL database that I created using the AWS Management Console.



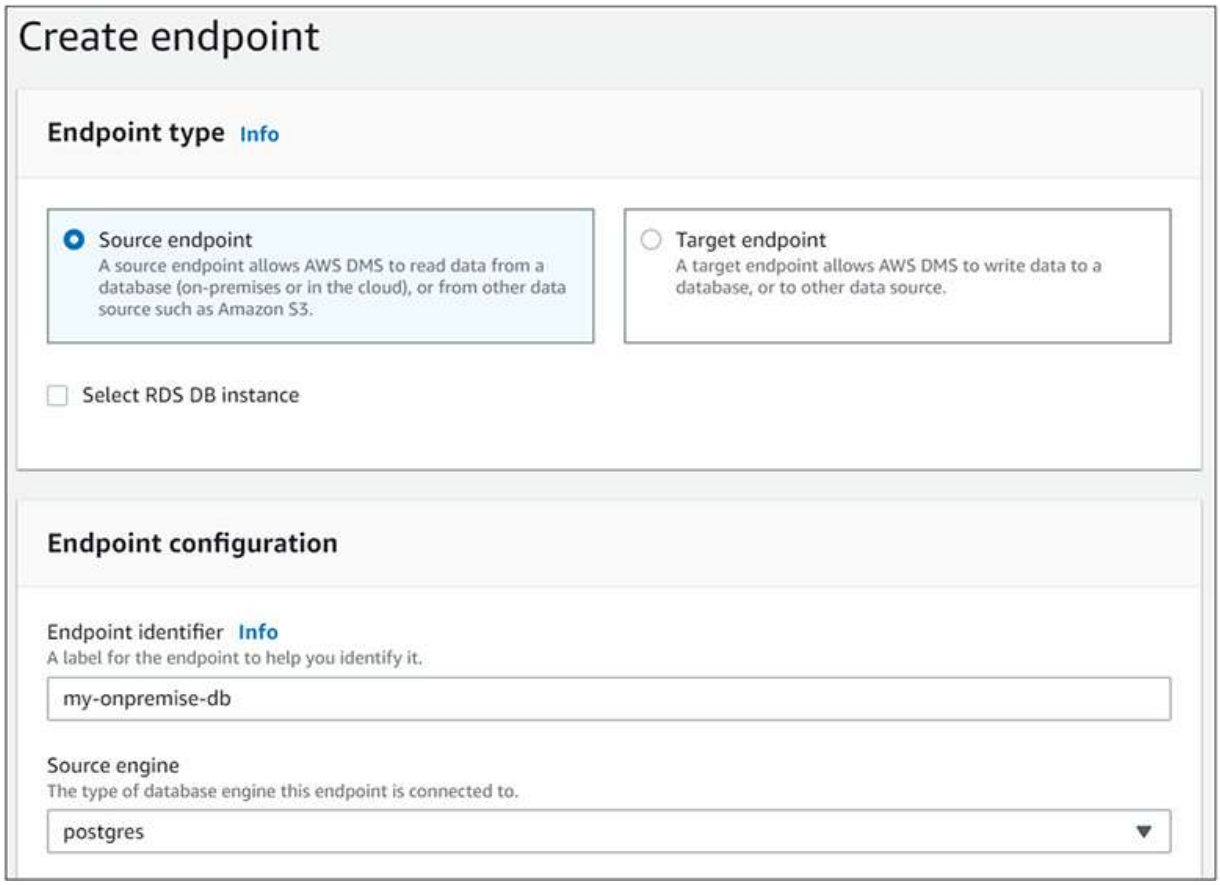
The Amazon RDS database connection is highlighted in orange in the pgAdmin tool browser. As you can see, it has only the default postgres database along with the rdsadmin database.



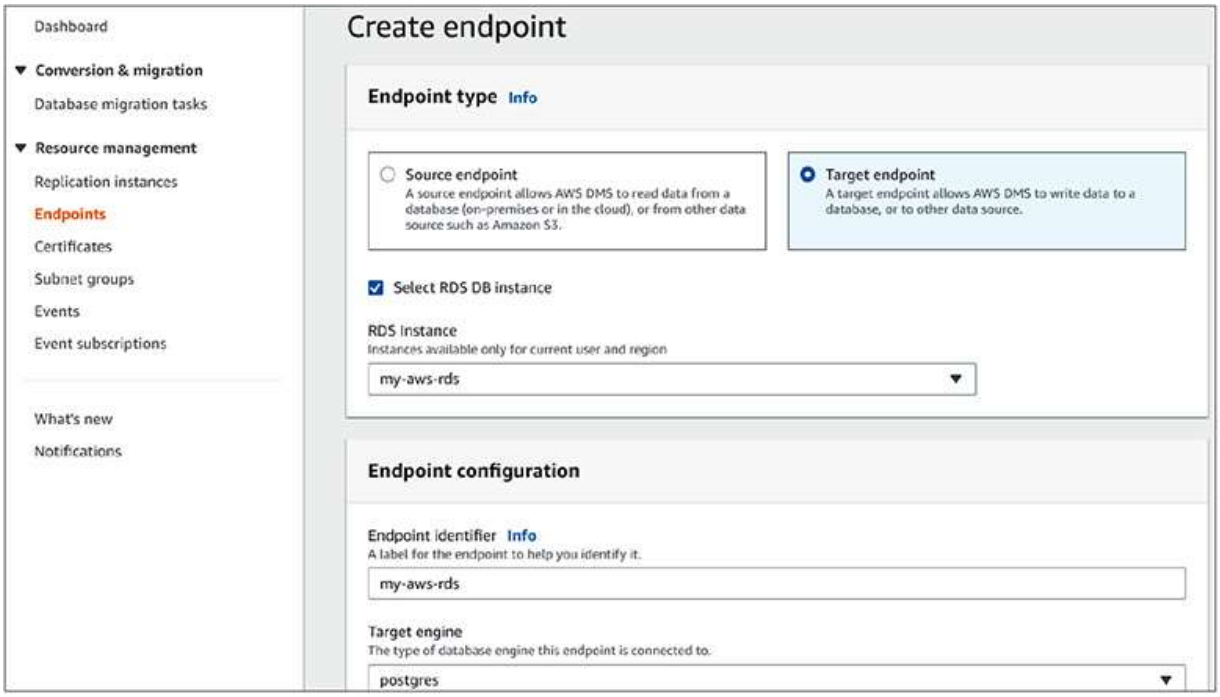
Log in to your AWS Management Console and navigate to the AWS Database Management Service. Select Replication Instances from the left pane, give it a name—for example, **my-db-replication**—and select the appropriate instance class based on your database size. You can leave the remaining values at the default settings and click the Create button.



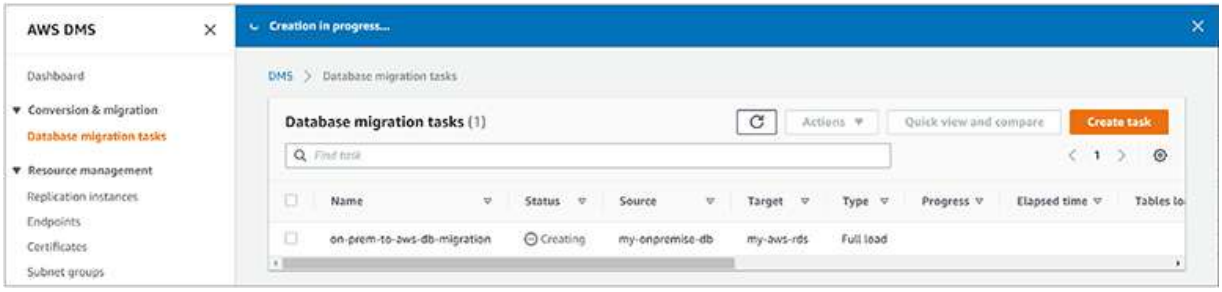
Next, create endpoints for both source and target databases. Select Endpoints from the left navigation pane. Select Source Endpoint and provide the endpoint identifier—for example, **my-onpremise-db**. Select Postgres for the source engine, and then enter the server name, port, username, and password along with the database name (in this case, northwind).



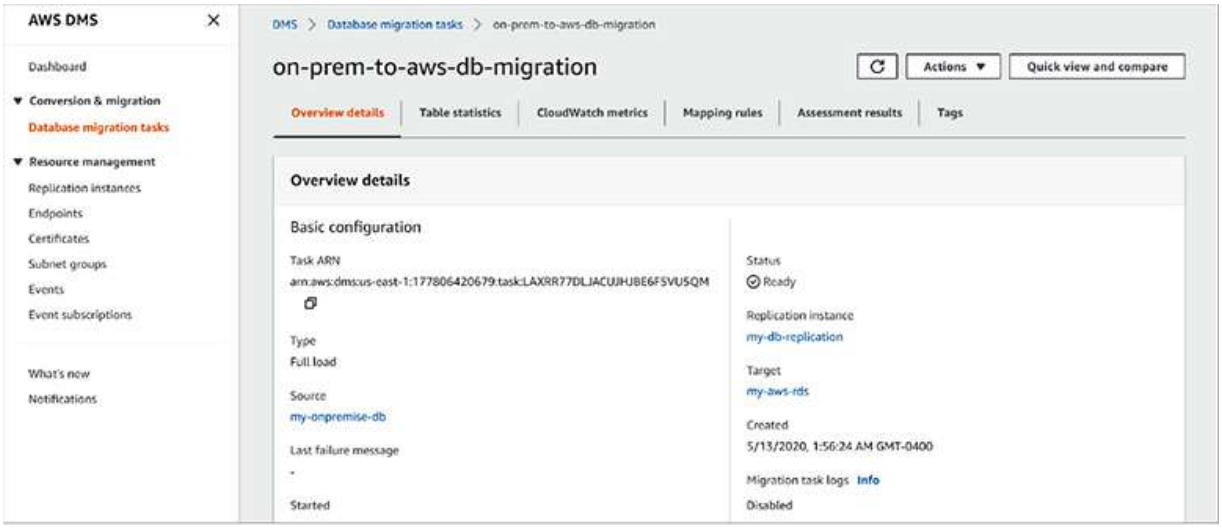
Again, select Endpoints from the left navigation pane and select Target Endpoint for the endpoint time. Select the Select RDS DB Instance checkbox. Select the RDS instance from the dropdown, and enter the name—such as, **my-aws-rds**—for the endpoint identifier. Select Postgres for the target engine from the dropdown and leave other options at their default settings.



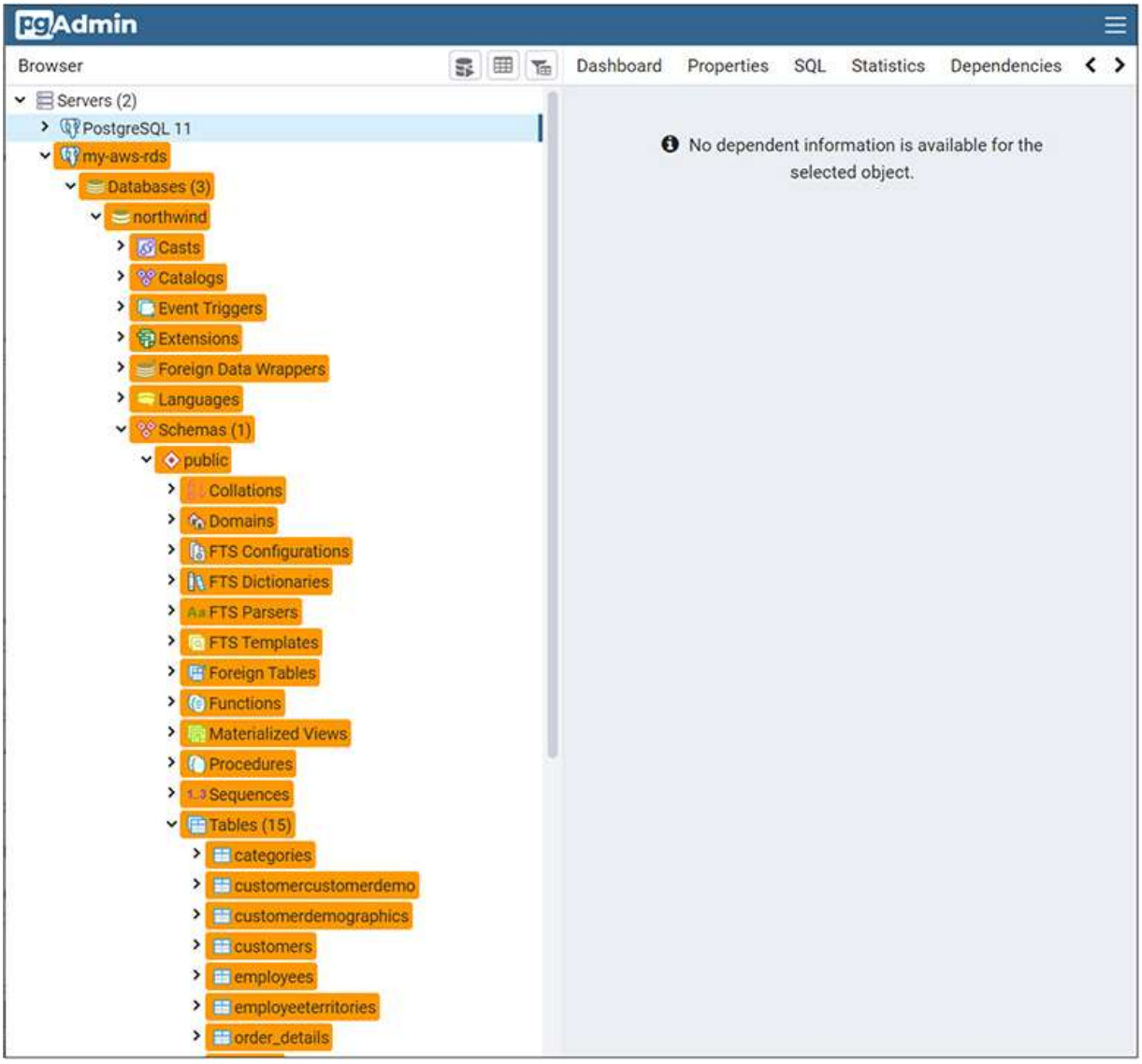
Now, you need to create the database migration tasks from the left navigation pane. For the task identifier, enter **on-prem-to-aws-db-migration** and choose Replication Instance from the dropdown. Select the previously created source database endpoint and target database endpoint from the dropdown. For migration type select Migrate Existing Data, and you can select Replication if so desired. Leave the other options at their default settings, and click the Create Task button.



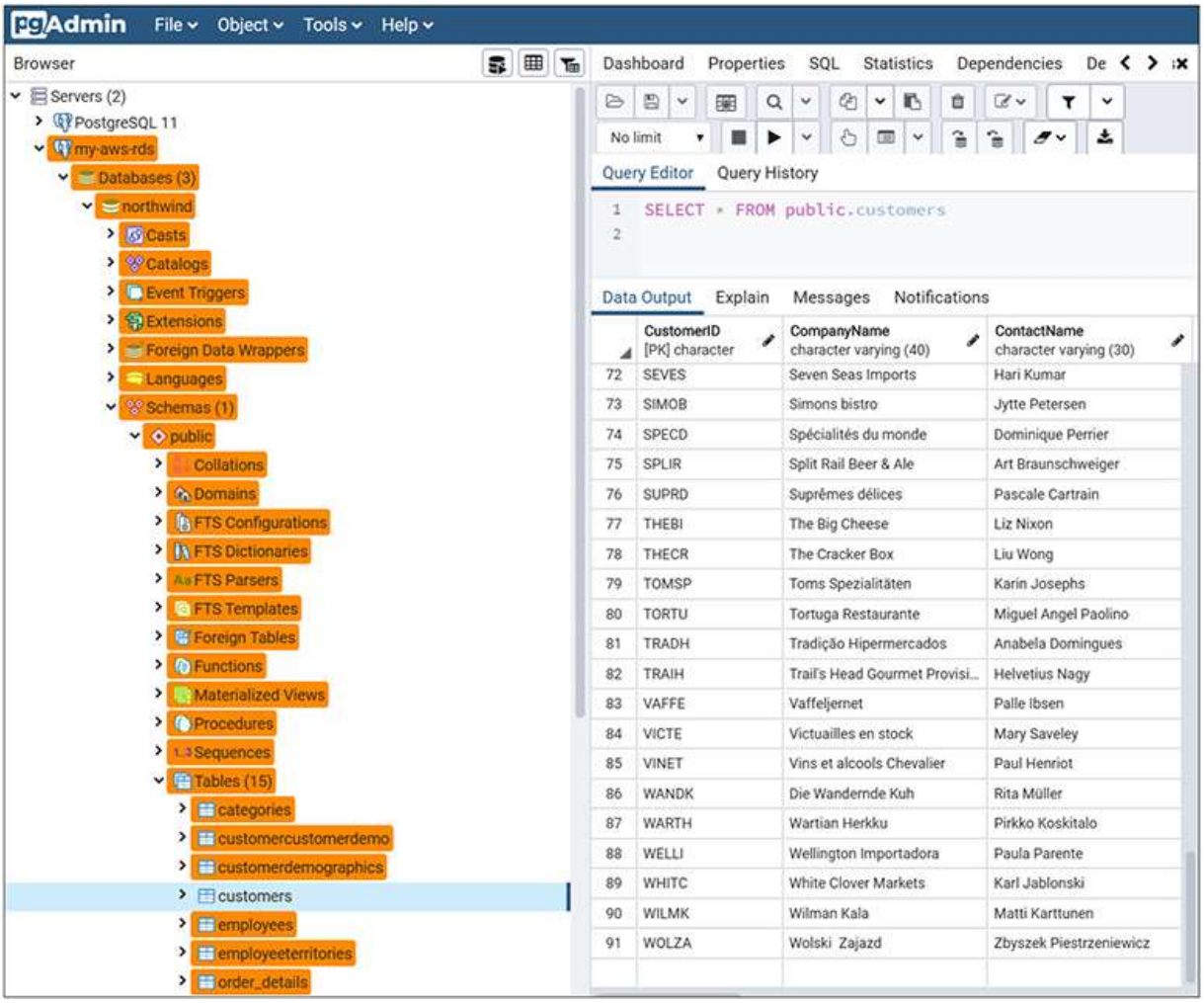
If you selected Start Task On Create in the previous step, then the task automatically starts, and you can see the overview, statistics, and metrics. After few minutes, based on your database size, the task will be completed successfully.



Now you can refresh the databases of your Amazon RDS DB connection and see the migrated northwind database. You can also see 15 tables are migrated along with the data.



Now let us run the same select query on the customer table. As you can see, all the data migrated successfully.



The practical experience of migrating an application and database will help you both on the certification exam and in the real world.

Chapter Review

This chapter began by explaining the 6Rs migration strategy that most enterprises follow. Then it explained step by step how to migrate a Linux web server from on-premises to AWS Cloud using the lift and shift migration strategy. And finally it provided the steps to migrate a PostgreSQL database from on-premises to Amazon RDS by using the lift, tinker, and shift migration strategy.

Exercises

The following exercises will help you delete all the resources that you created in this chapter. You need to create an AWS account, as explained earlier, to perform these exercises. You can use the Free Tier when launching AWS resources, but make sure to terminate them at the end.

Exercise 22-1: Delete the Amazon EC2 Instance Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in and then navigate to the AWS Elastic Compute Cloud Service (Amazon EC2) console at https://console.aws.amazon.com/ec2/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Instance.
4. In the center panel, select the instance. Then choose Actions and Instance State.
5. Select Terminate to terminate the instance.

Exercise 22-2: Delete Amazon RDS Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in and then navigate to the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Databases.
4. In the center panel, select the database. Then choose Actions and then Delete to delete the database.

Exercise 22-3: Delete the Amazon S3 Bucket Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in and then navigate to the AWS Simple Storage Service (Amazon S3) console at

https://console.aws.amazon.com/s3/.

2. From the navigation pane, choose Buckets.
3. In the center panel, select the bucket you want to delete.
4. First delete all the objects in the bucket, including the .vmdk image, then choose Delete Bucket.

Questions

The following questions will help you gauge your understanding of the contents in this chapter. Read all the answers carefully because there might be more than one correct answer. Choose the best responses for each question.

1. Your company has decided to migrate to AWS Cloud. What is the first thing they need to do to migrate all the resources, including applications and databases?

   A. Hire a cloud expert and expect him to migrate everything

   B. First migrate all the critical applications

   C. First migrate all the mission-critical databases

   D. Discover all your resources, like applications, databases, and dependencies

2. Your company plans to migrate their on-premises workload to AWS Cloud in two years. What kind of workload do you suggest migrating to AWS Cloud first?

   A. You suggest migating complex production applications that bring revenue to your company

   B. You suggest migrating production databases that have a high impact to your business first, followed by applications

   C. You suggest migrating all the resources from on-premises to the cloud at the same time

   D. You suggest migrating development and POC environments that have a minimum impact to your business to the cloud

3. Which of the following are part of the 6Rs of cloud migration? (Choose three.)

A. Rehosting

B. Re-platforming

C. Relocation

D. Refactoring

4. What service can you leverage to migrate your on-premises applications to AWS Cloud?

A. AWS Server Migration Service

B. AWS Ground Station

C. AWS CloudHSM

D. AWS Control Tower

5. Your company decided to quickly migrate the application and databases. Which of the following 6R migration strategies you would suggest to do the lift and shift?

A. Re-platforming

B. Rehosting

C. Refactoring

D. Retain

6. A company wants to take advantages of the cloud by using the Platform as a Service (PaaS) service from AWS but does not have time or resources to develop cloud-native applications. What migration strategy you would suggest?

A. Retire

B. Repurchasing

C. Rehosting

D. Re-platforming

7. Your company is planning to split all the big monolithic applications into microservices and serverless applications and leverage cloud-native development. What migration strategy you would suggest?

A. Repurchasing

B. Re-platforming

C. Refactoring

D. Rehosting

8. Your company has many legacy applications that are not supported in cloud environments and have stringent compliance regulations to keep the data on-premises. What migration strategy is suitable for this scenario?

   A. Retain

   B. Retire

   C. Rehost

   D. Repurchase

9. A company wants to migrate its customer relationship management (CRM), enterprise resource planning (ERP), and human resources (HR) software to the cloud instead of managing them on-premises. Which migration you would suggest for this scenario?

   A. Rehosting

   B. Refactoring

   C. Re-platforming

   D. Repurchasing

10. A company has discovered many redundant applications and databases during its cloud migration discovery phase. What kind of migration strategy you would suggest?

    A. Retire

    B. Repurchasing

    C. Rehosting

    D. Retain

Answers

1. **D.** You need to first discover all your resources, like applications, databases, and dependencies.
2. **D.** You need to migrate minimum-impact development and POC environments to the cloud and gain confidence.

3. **A, B, D.** Rehosting, re-platforming, and refactoring. (The other three are repurchasing, retain, and retire.)
4. **A.** You can use AWS Server Migration Service to migrate your on-premises applications to the cloud.
5. **B.** Rehosting, which is also known as lift and shift.
6. **D.** You can use re-platforming to migrate the application to AWS Elastic Beanstalk and the database to Amazon RDS.
7. **C.** The refactoring migration strategy allows you to build cloud-native and serverless applications.
8. **A.** When you have compliance regulations to keep some workload in on-premises, you can use the retain strategy.
9. **D.** The repurchasing strategy allows you to purchase CRM, ERP, or HR cloud-based solutions from the AWS marketplace.
10. **A.** You need to use the retire strategy to shut down and remove those servers from your datacenter.

Additional Resources

- **AWS Migration Whitepaper** This whitepaper explains all the details you need to know to migrate your workload from on-premises to AWS Cloud.
  `https://d1.awsstatic.com/whitepapers/Migration/aws-migration-whitepaper.pdf`
- **AWS Enterprise Migration Strategy Blog** These blogs provide all the best practices and steps to plan and migrate your enterprise workload.
  `https://aws.amazon.com/blogs/enterprise-strategy/category/enterprise-strategy/migration-enterprise-strategy/`
- **AWS Migration Best Practices** This whitepaper explains all the best practices and strategies that you can follow for your cloud migration.
  `https://d1.awsstatic.com/Migration/migrating-to-aws-ebook.pdf`

PART VIII

Building, Deploying, and Debugging Cloud Applications

CHAPTER 23

Hosting Secure Repositories Using AWS CodeCommit

In this chapter, you will learn

- AWS CodeCommit

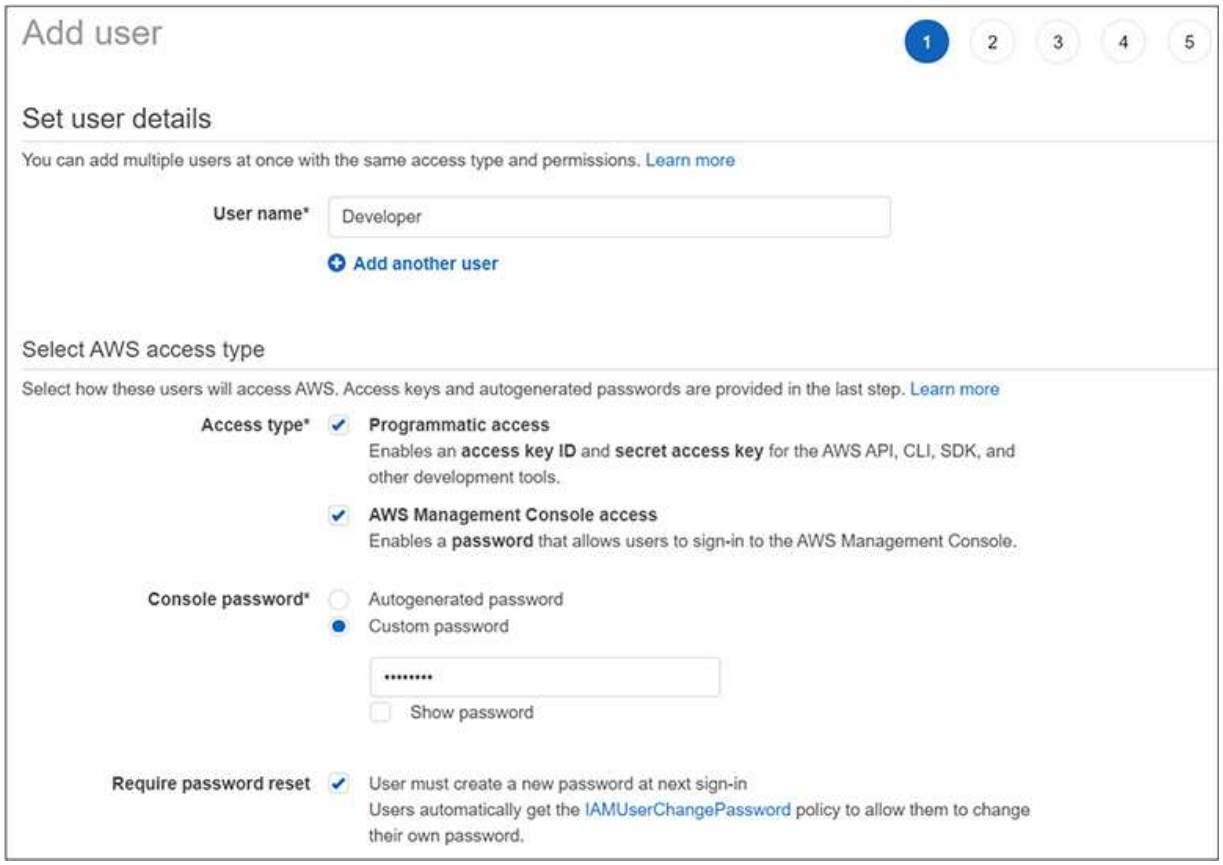
This chapter will teach you how to create repository in AWS CodeCommit and pull, update, and commit changes from your local machine and AWS Cloud9.

AWS CodeCommit

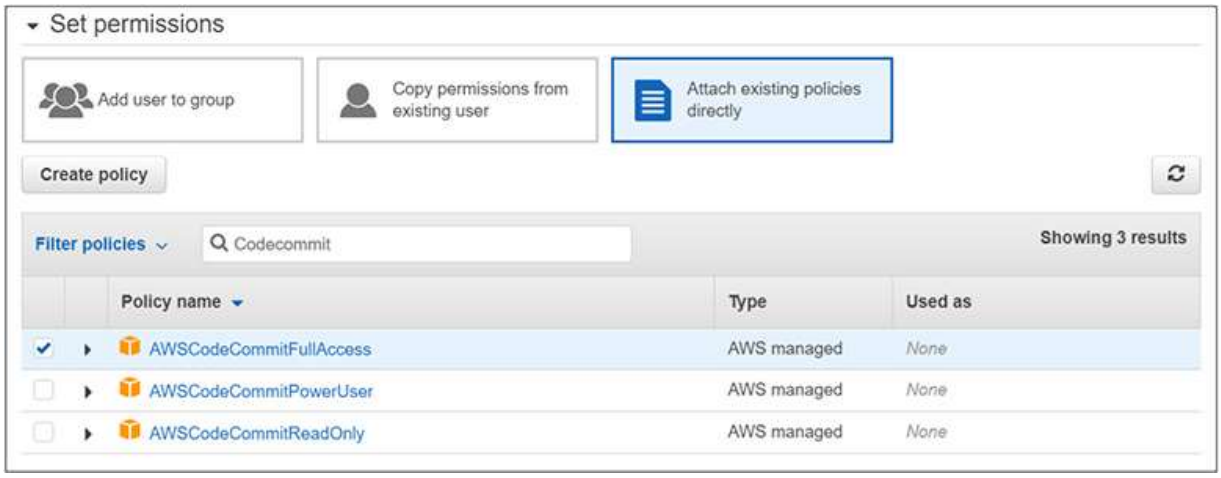
As a developer, you often use a source code repository such as GitHub, GitLab, or other tools. AWS CodeCommit provides a private, secure, scalable, and managed source code repository that is closely integrated with other AWS services and third-party services. This section provides the step-by-step procedures to create a repository; pull code; create branches; and commit, compare, and merge your source code.

You need to access AWS CodeCommit from your local machine using either an HTTPS or SSH connection, so you need the relevant credentials. AWS recommends using a separate user instead of using your root credentials. So, first create a new user with required access, as shown in the following illustration. Log in to the AWS Management Console and navigate to the IAM service. Select Users from the pane on the left and click on the Add User button. Enter a username—for example, **Developer**—and choose

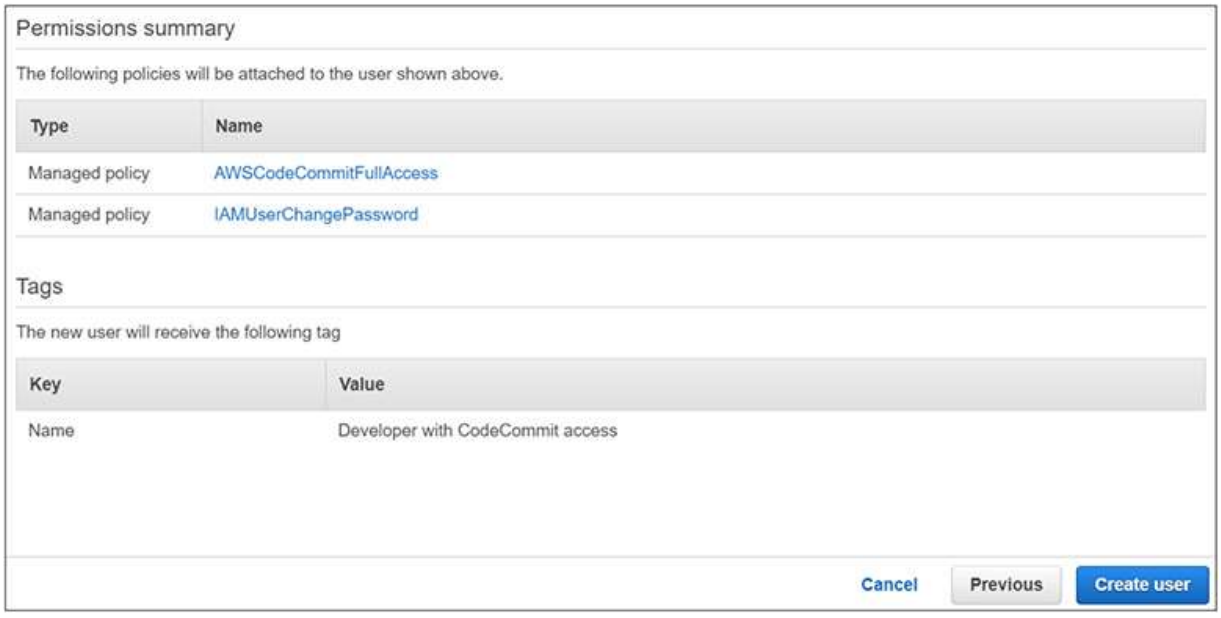
the access type for both programmatic access and console access. Enter a password and click Next: Permissions.



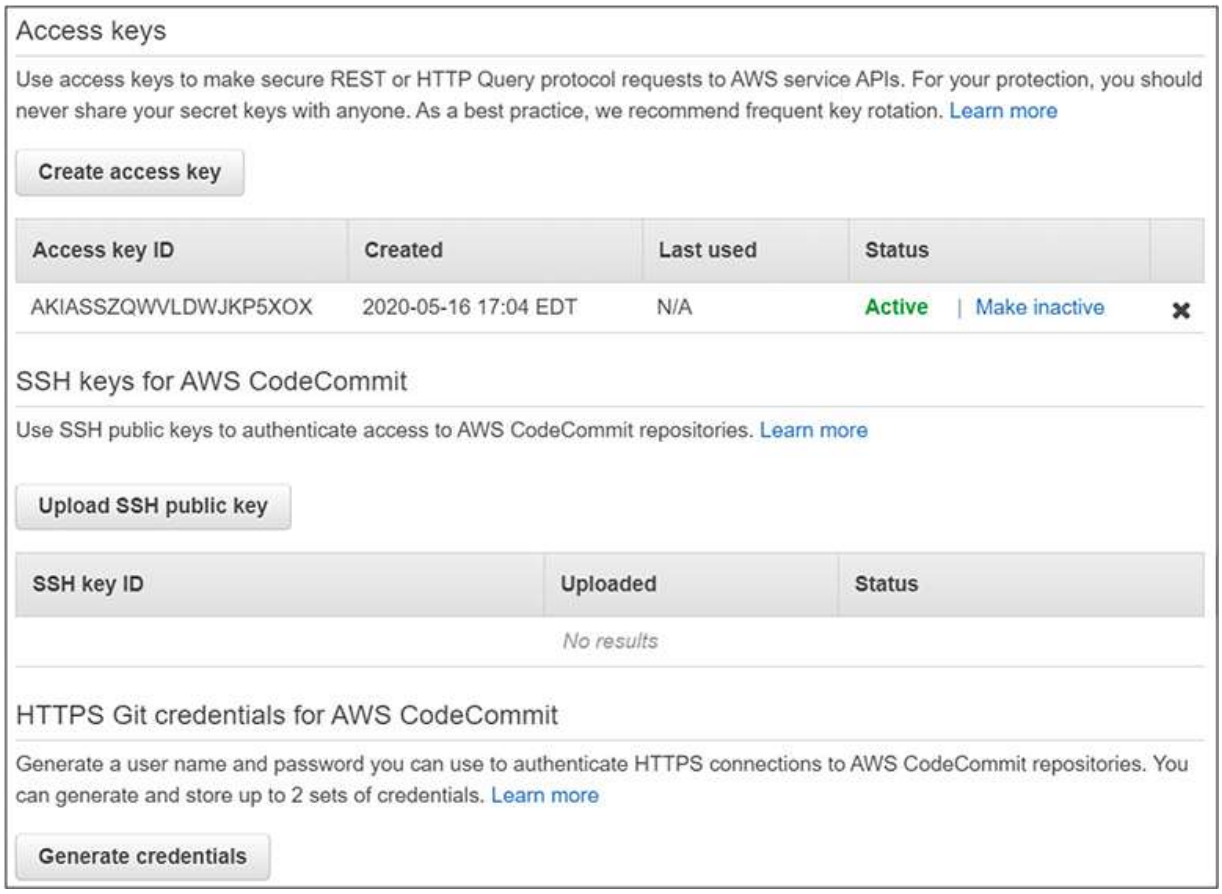
Here you can either create a custom policy or attach an existing policy directly. Search for CodeCommit in the filter policies, and choose AWSCodeCommitFullAccess. If you have a user with similar permissions, then you can select Copy Or Add This User To A group. Once you select the required permissions, click the Next: Tags button.



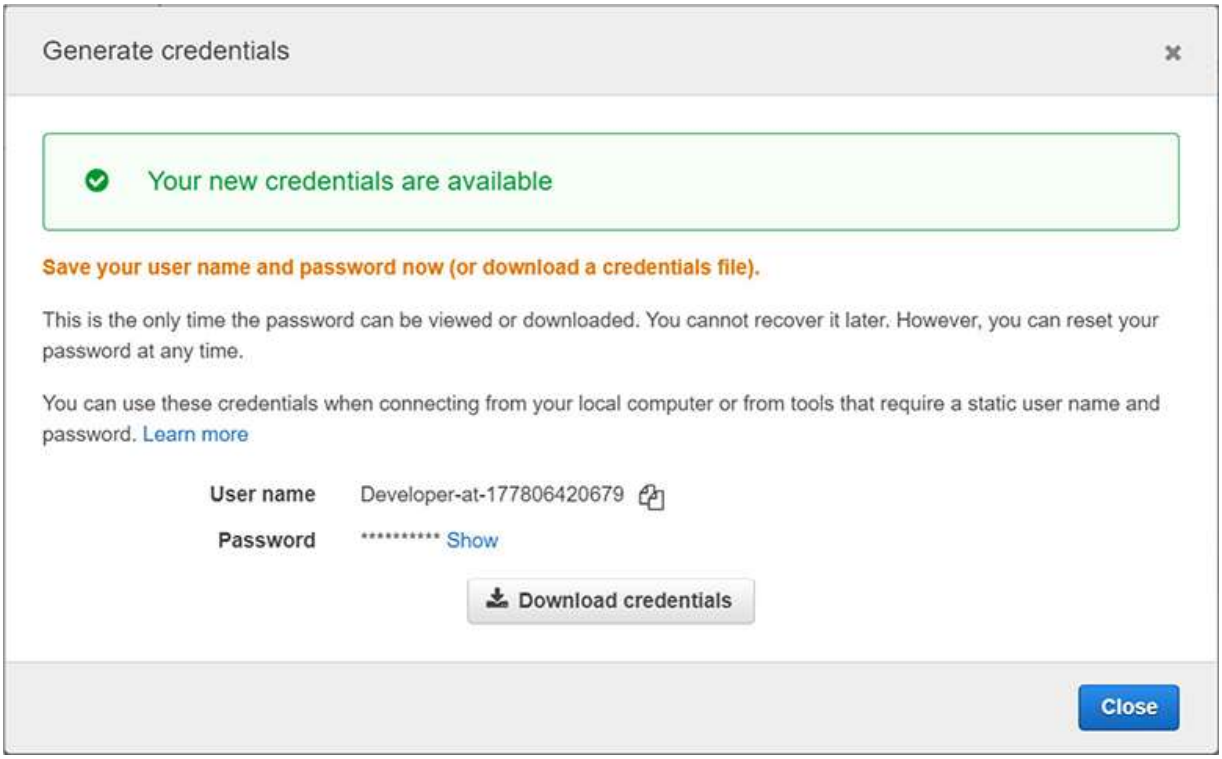
In accordance with the standard least-privilege security best practice, grant only the required permissions and click Create User.



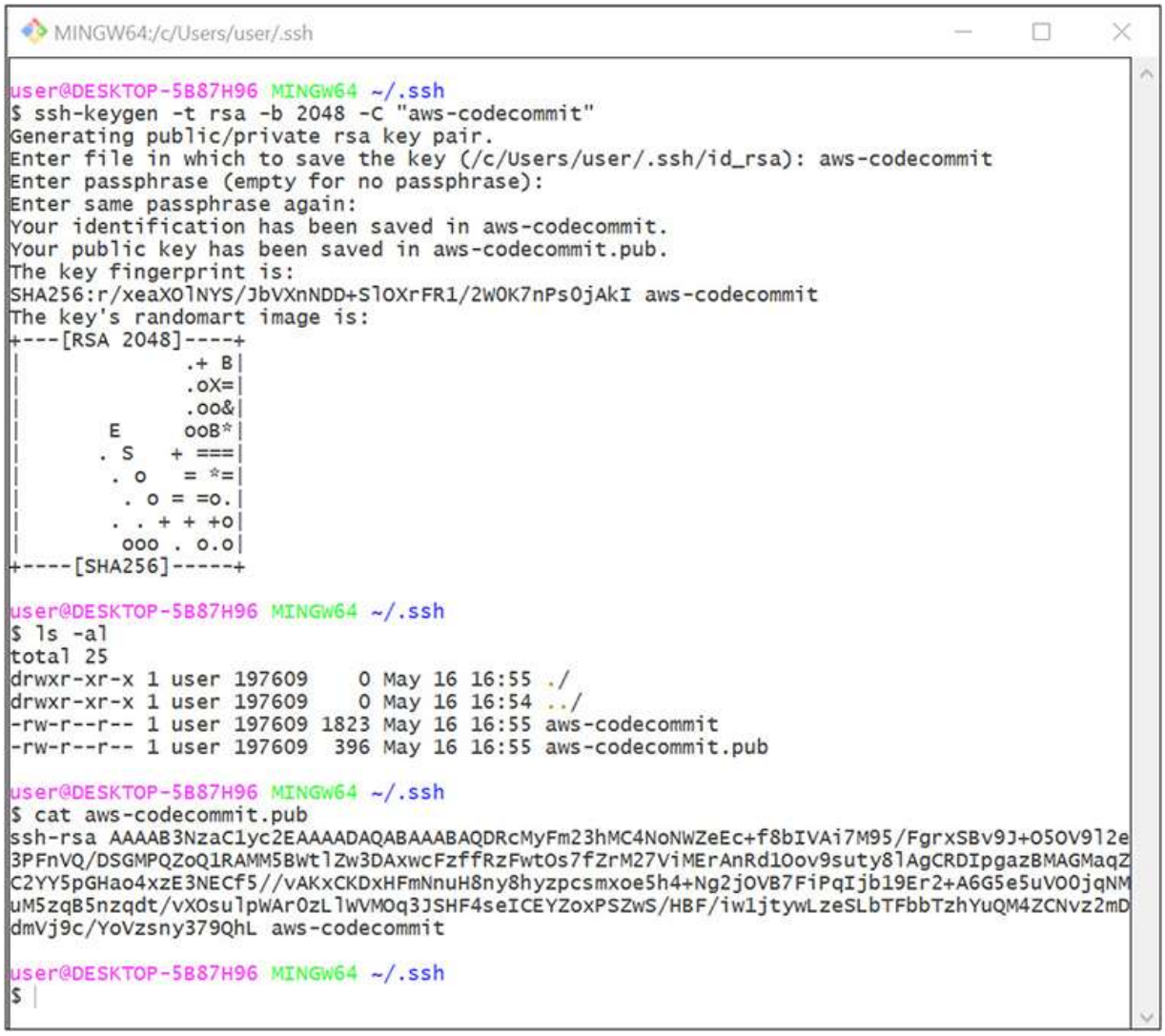
Once the user is created, click on the username, and scroll down. Upload your SSH public key and download the HTTPS Git credentials for AWS CodeCommit. You just need either one normally, but this example is using both HTTPS and SSH to access AWS CodeCommit.



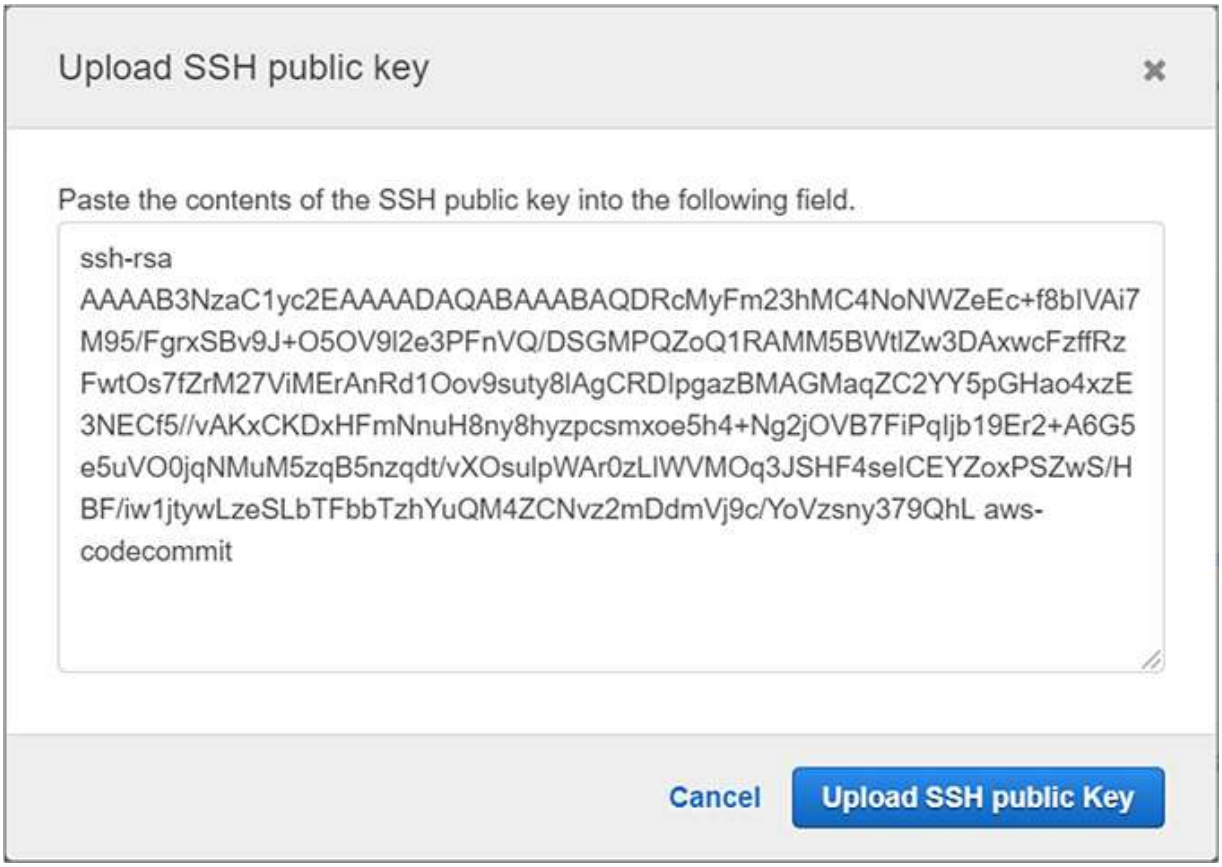
Click on Generate Credentials for HTTPS, and you will be prompted with the following screen. Here either copy the username and password or download the credentials.



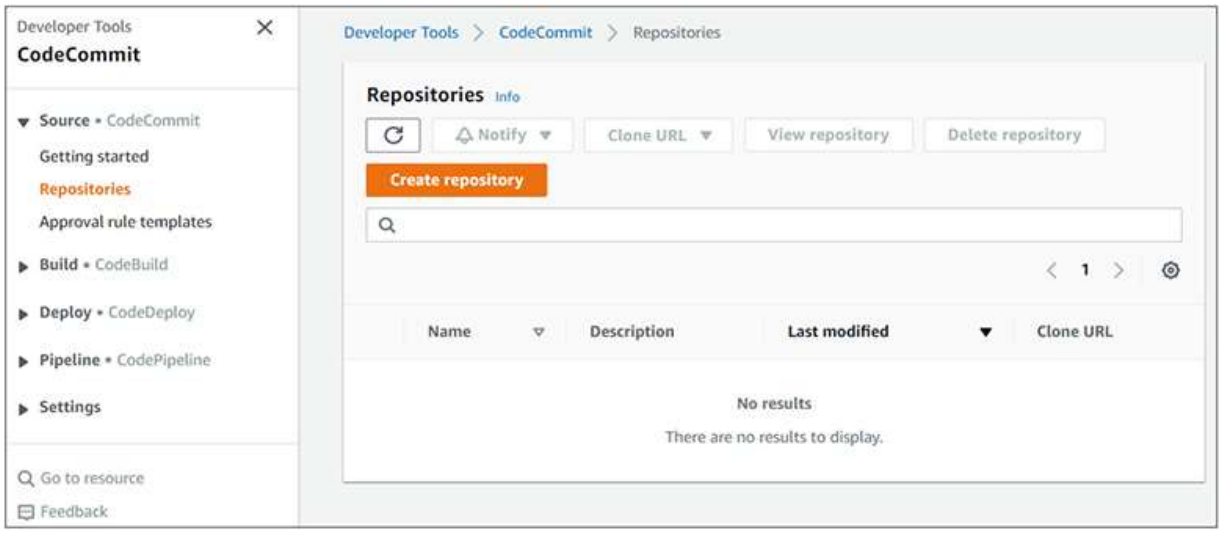
You can use Git Bash to simulate bash in Windows or use Terminal in Mac. Create and navigate to the .ssh folder and type **ssh-keygen -t rsa -b 2048 -C "aws-codecommit"** and leave the defaults for other prompts. Now open your .pub file and copy the contents.



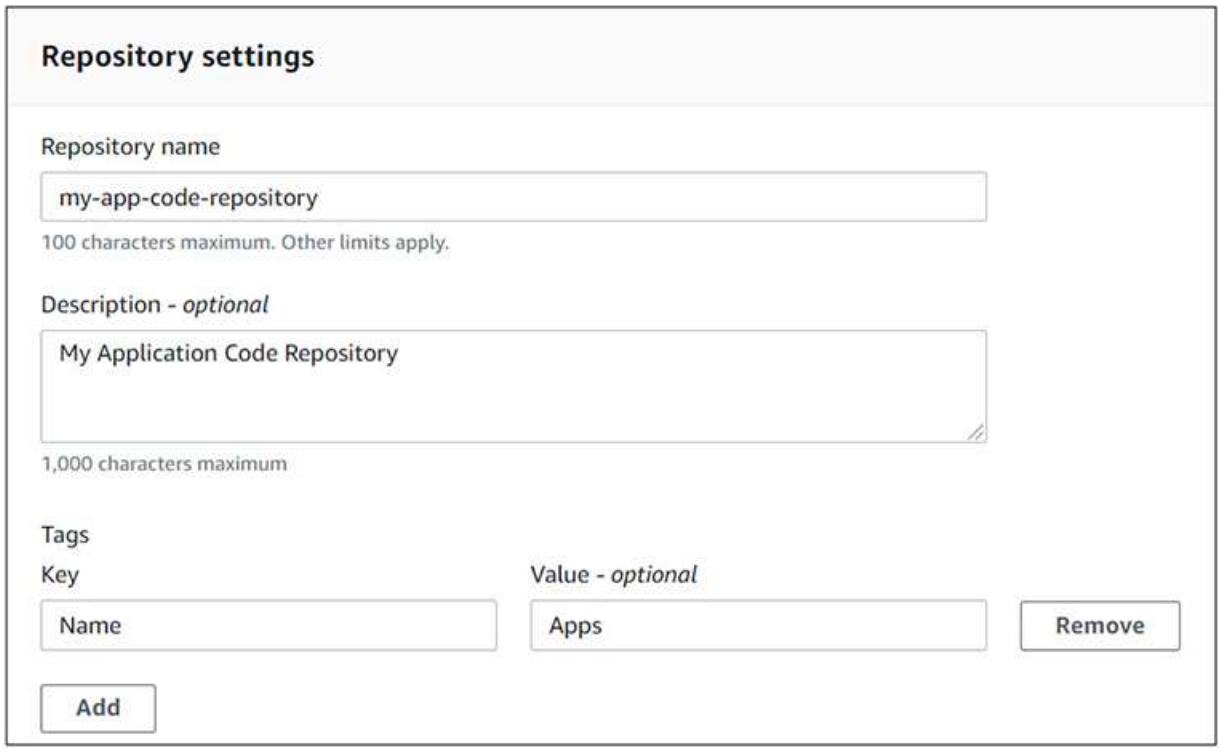
Switch to your AWS Management Console and navigate to the IAM service. Select the Developer user and click Upload SSH Public Key. Paste your .pub content and click on Upload SSH Public Key. You can delegate repository access through the IAM role to other AWS accounts using AWS Security Token Service (STS), where an IAM user can assume the role when running commands.



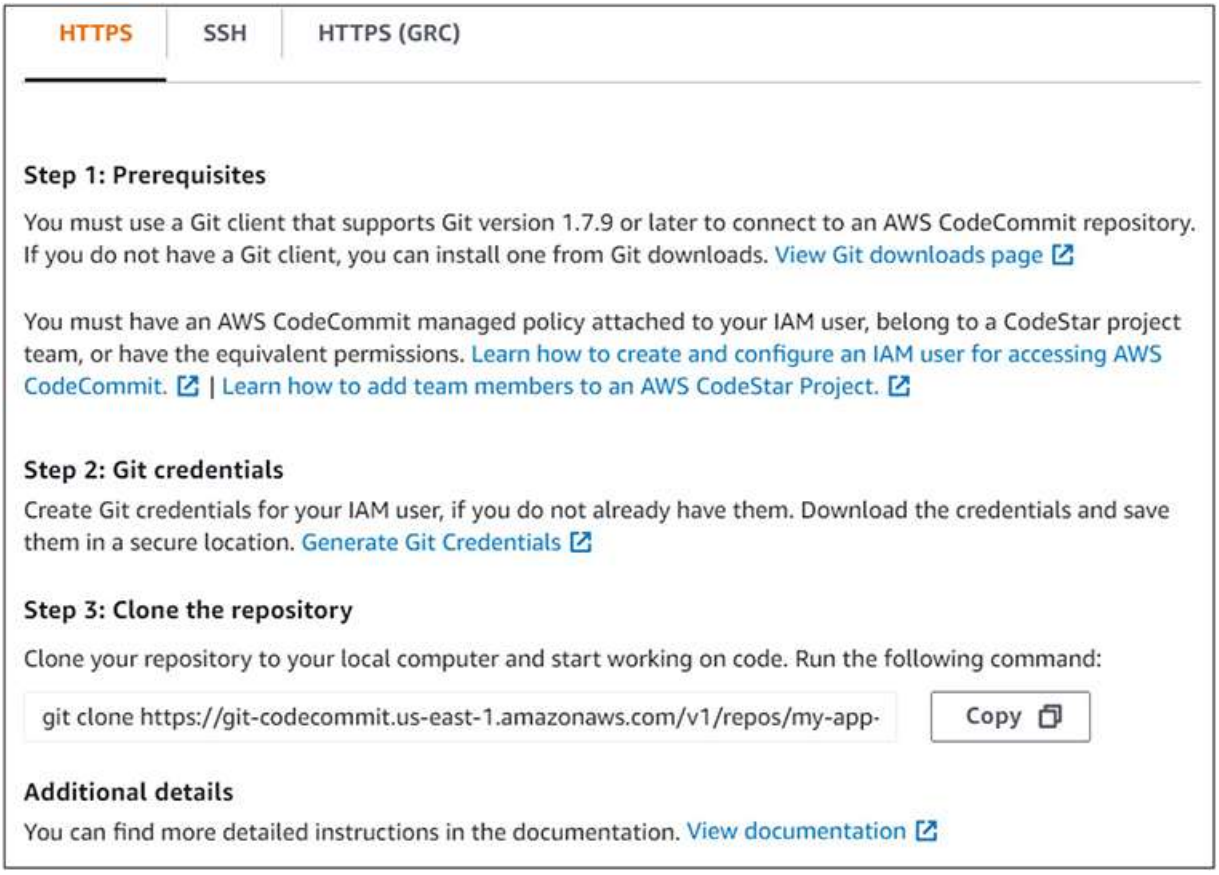
Now log in to the AWS Management Console as the Developer user and navigate to the AWS CodeCommit service. Select Repositories on the left pane, and you will see the list of repositories that you created already.



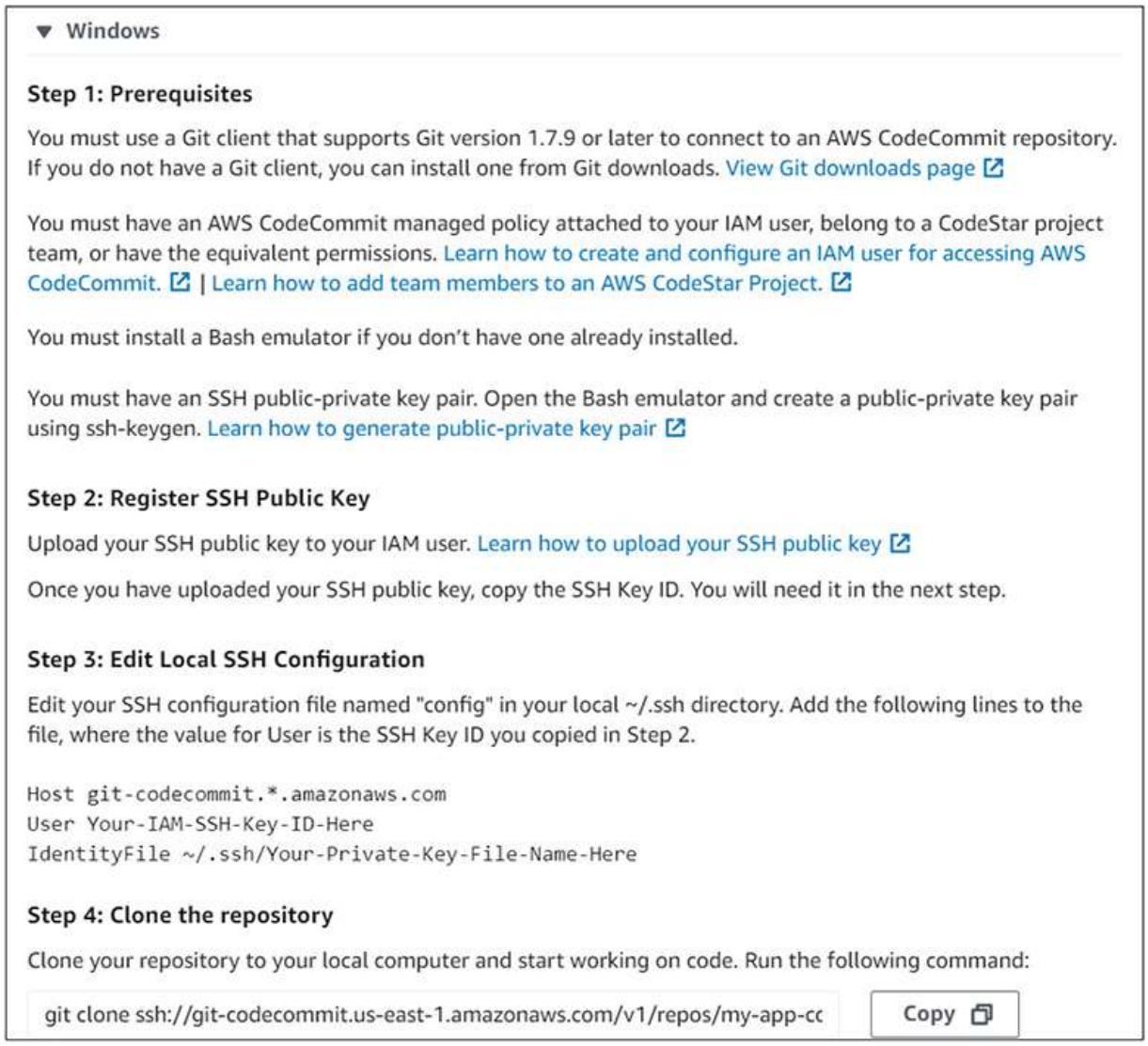
Enter **my-app-code-repository** for the name and provide a description. As a best practice add tags and click the Create button.



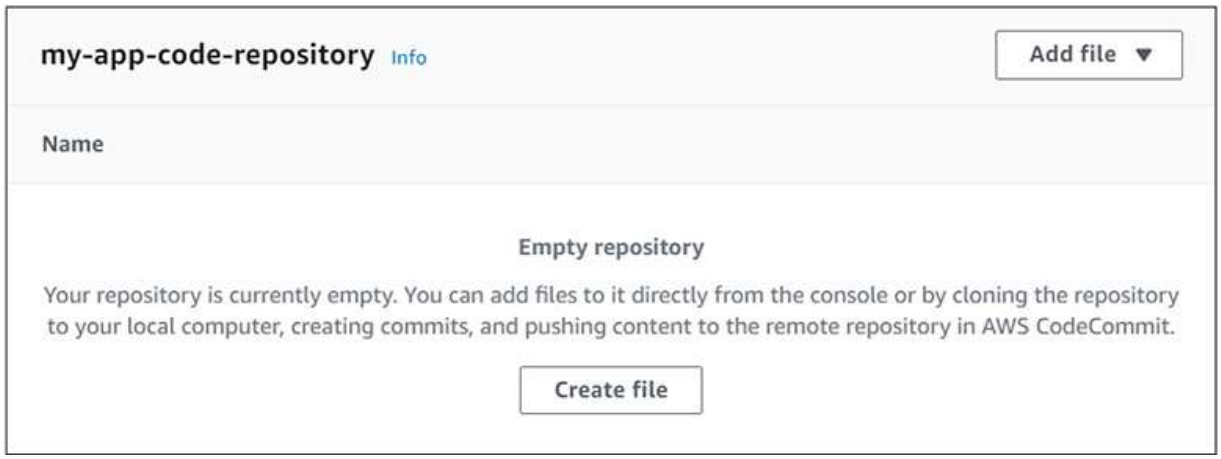
Note the connection details for HTTPS.



Note the SSH connection details.

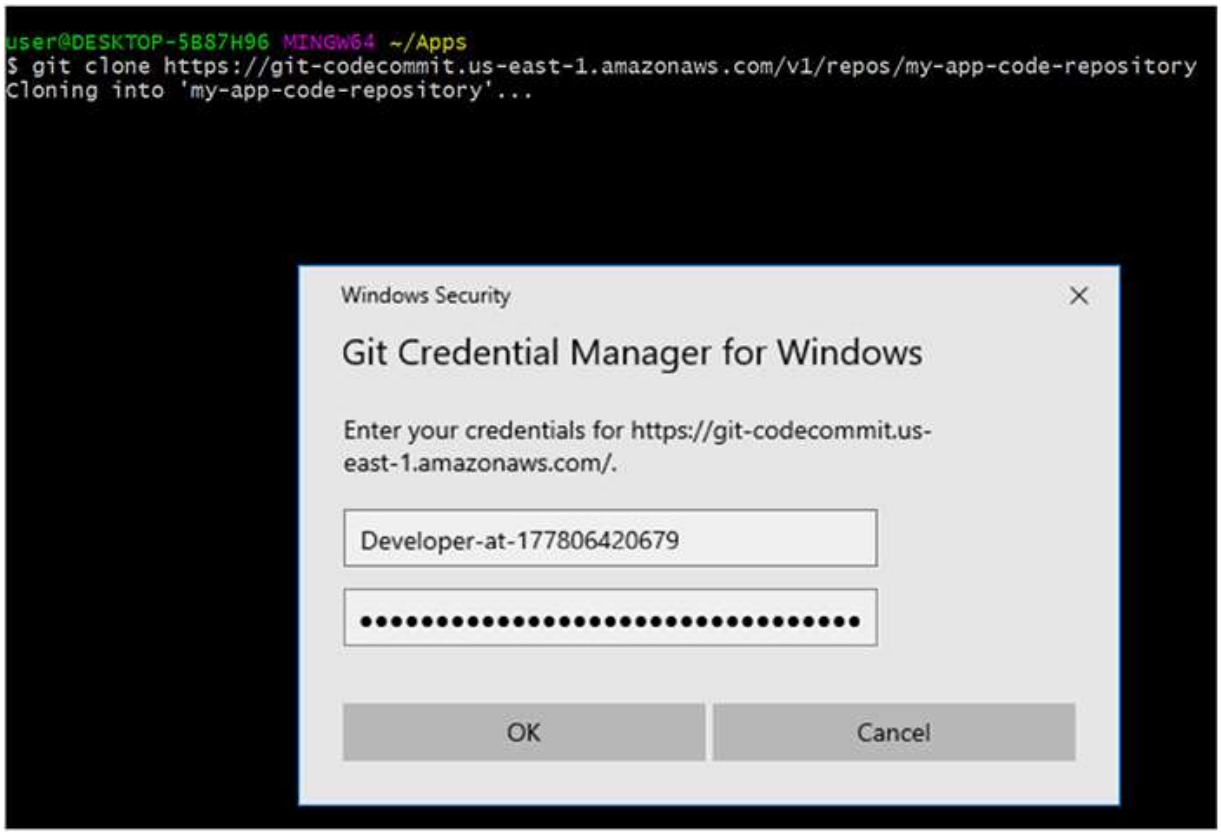


Click on the Create File button to create your first example source code.



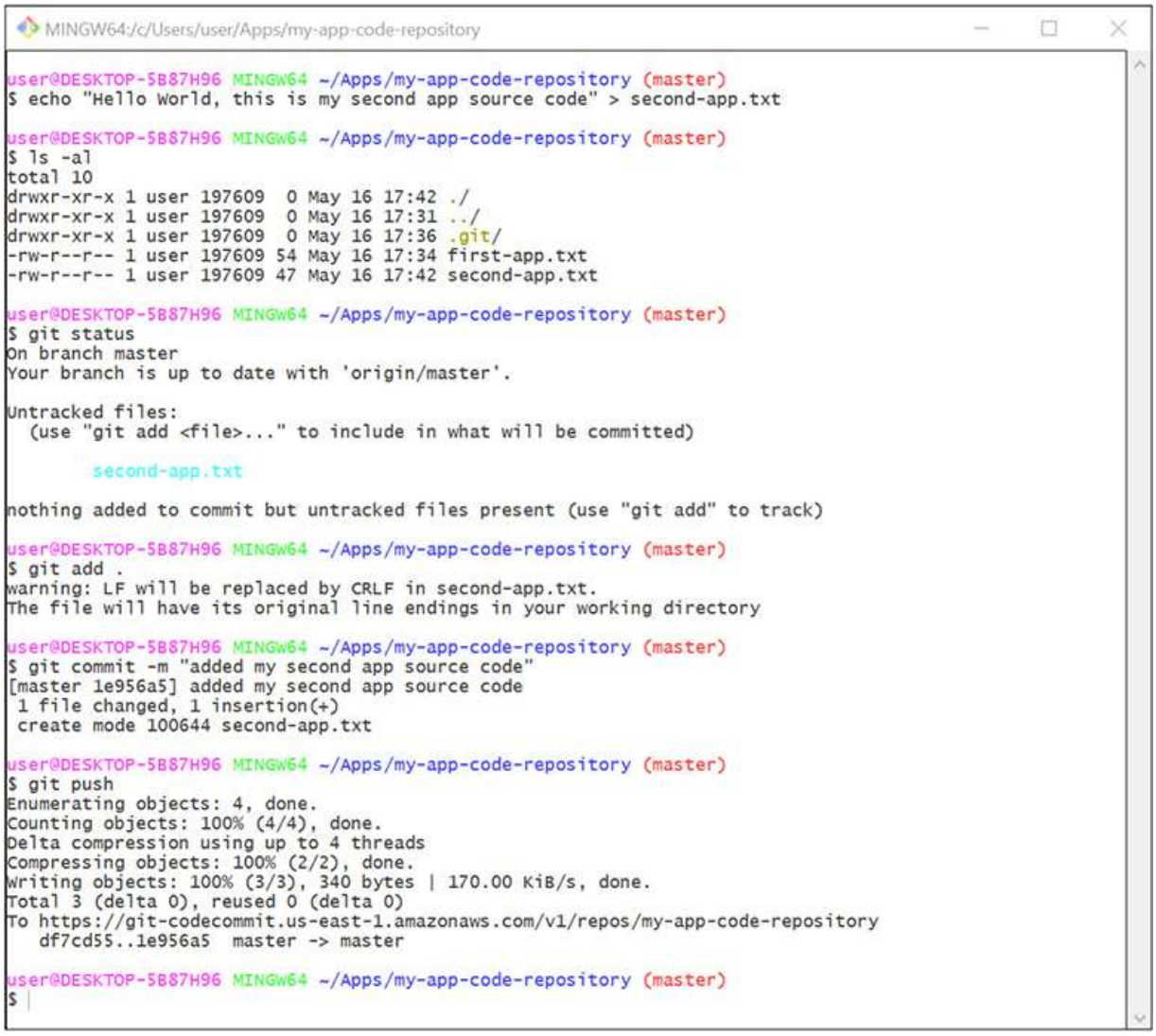
Type **welcome to your secure, private source code repository** and click Commit Changes.

Now we are ready to clone our new repository using the HTTPS connection. From your local machine, type **git clone https://git-commit.us-east-1.amazoneaws.com/v1/repos/my-app-code-repository** and press ENTER. Enter your HTTPS username and password and click OK.

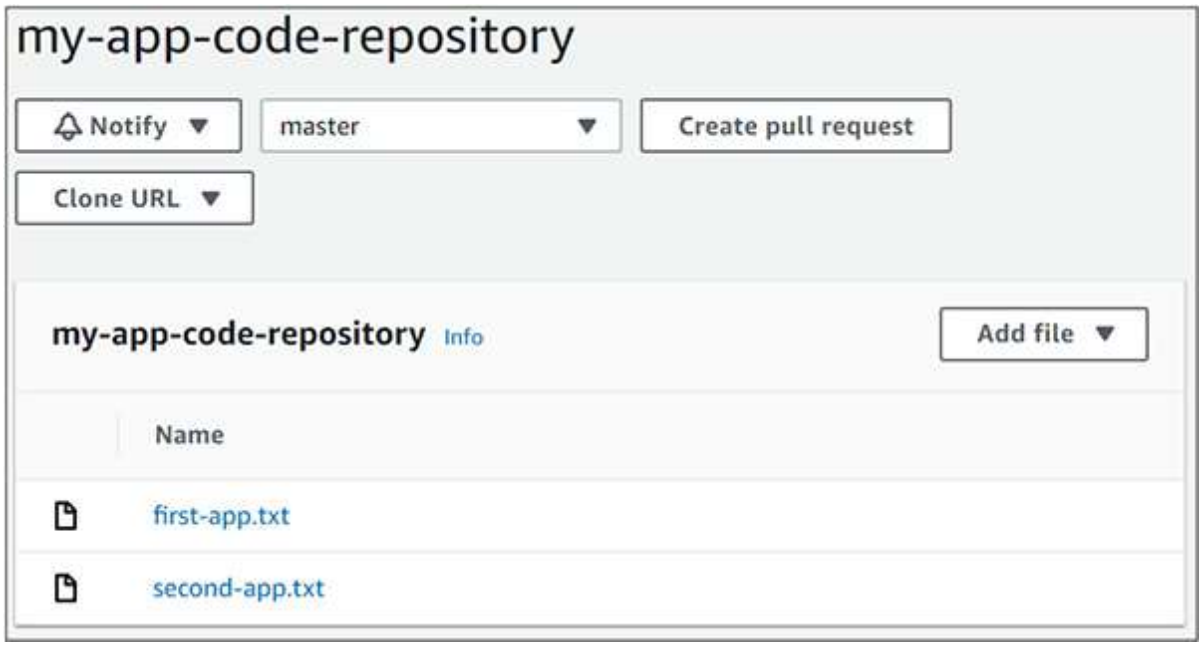


Navigate to your repository folder—for example, cd my-app-code-repository—and you can see the source file that we created earlier.

Create a new file called **second-app.txt** and add it to git using the git add command. Then commit the file using the git commit -m “comment” and push it to the remote AWS CodeCommit repository using the git push command.



Navigate to the AWS CodeCommit service in the AWS Management Console and select my-app-code-repository. You can see the new source code for second-app.txt that you pushed from your local machine.

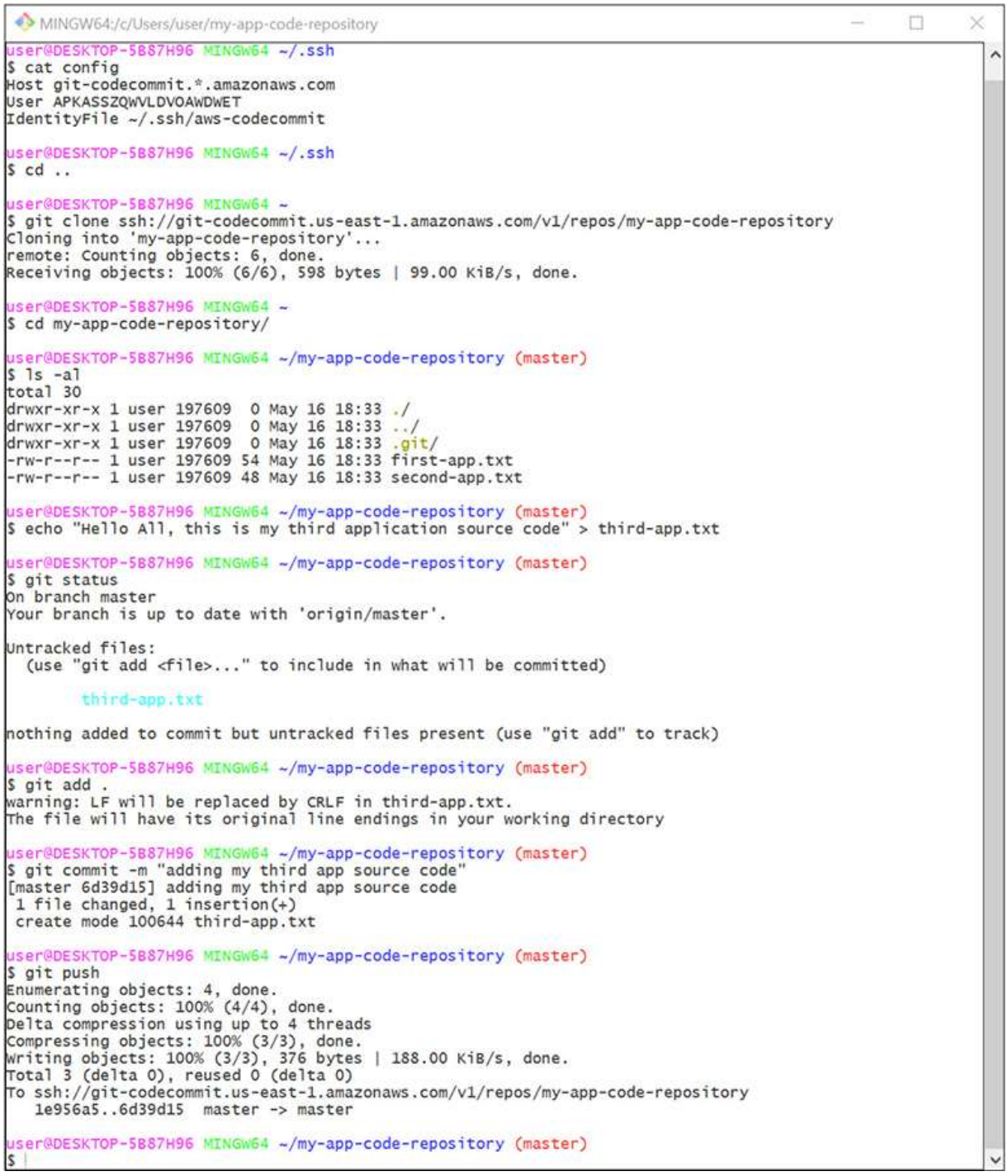


Click on the second-app.txt, and you can see the code.

Now, let us clone the AWS CodeCommit repository using SSH, where you do not need to enter your credentials each time.

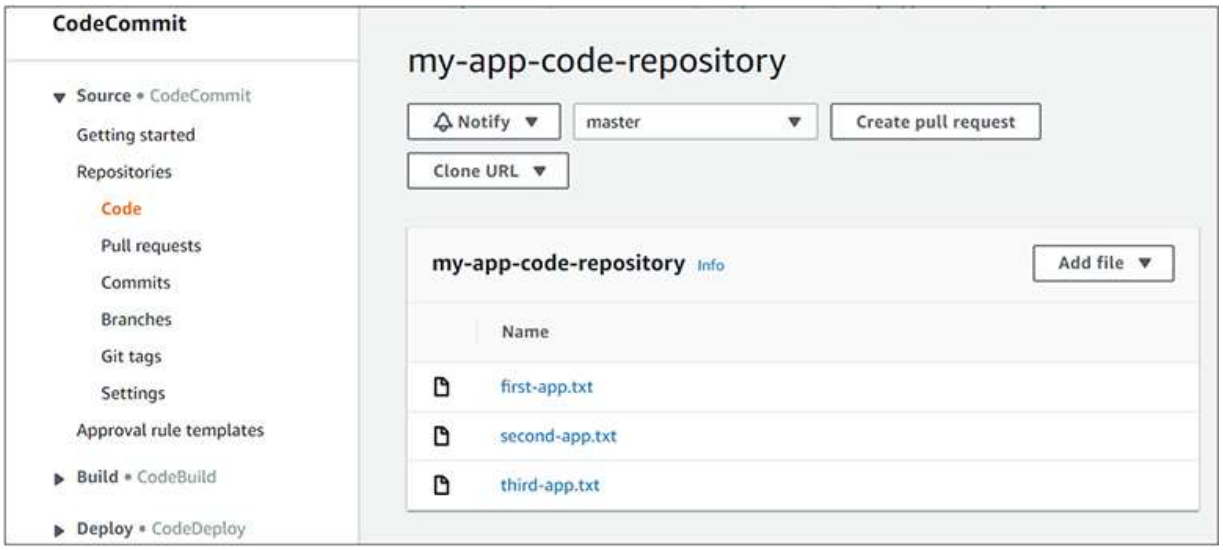
```
Host git-codecommit.*.amazonaws.com
User ABCDEFGHIJKLMNOP (Your SSH username)
IdentityFile ~/.ssh/aws-codecommit (Your SSH keyname)
```

Navigate to the .ssh folder and create the config file, as shown here.

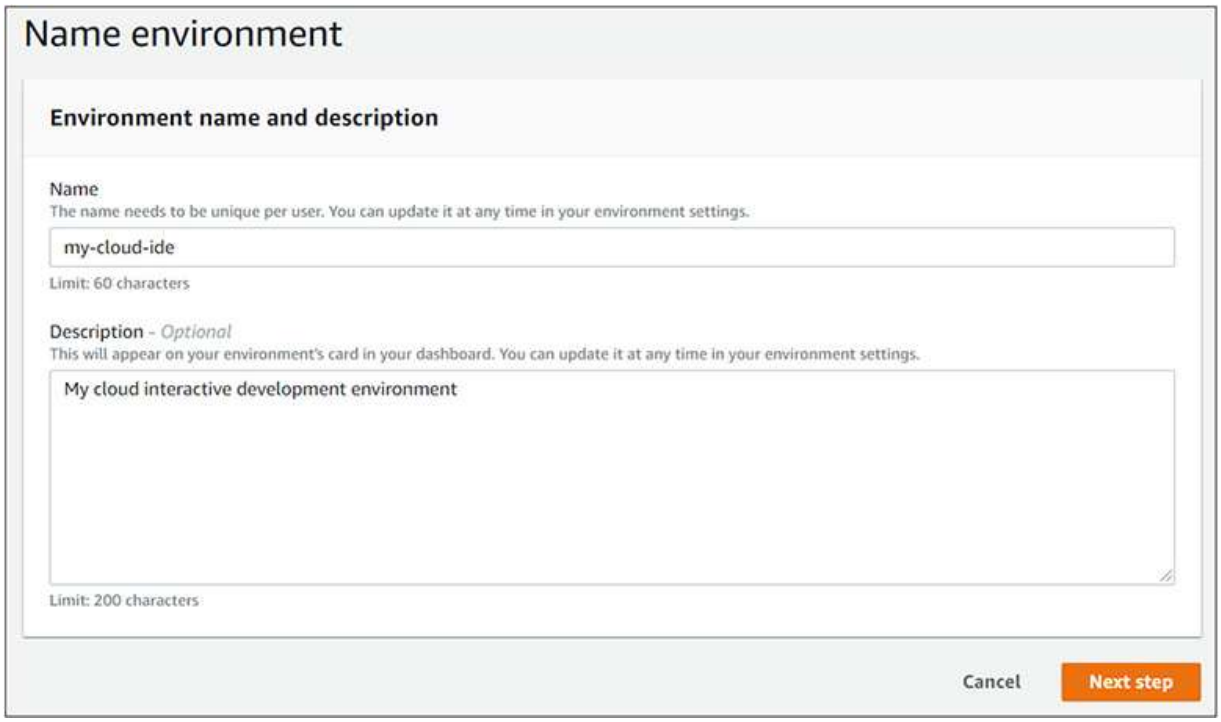


Create a new source code file called **third-app.txt** and add it using the git add command. Then you need to commit it using the git commit -m "comment" and push the new code to your AWS CodeCommit remote repository using the git push command.

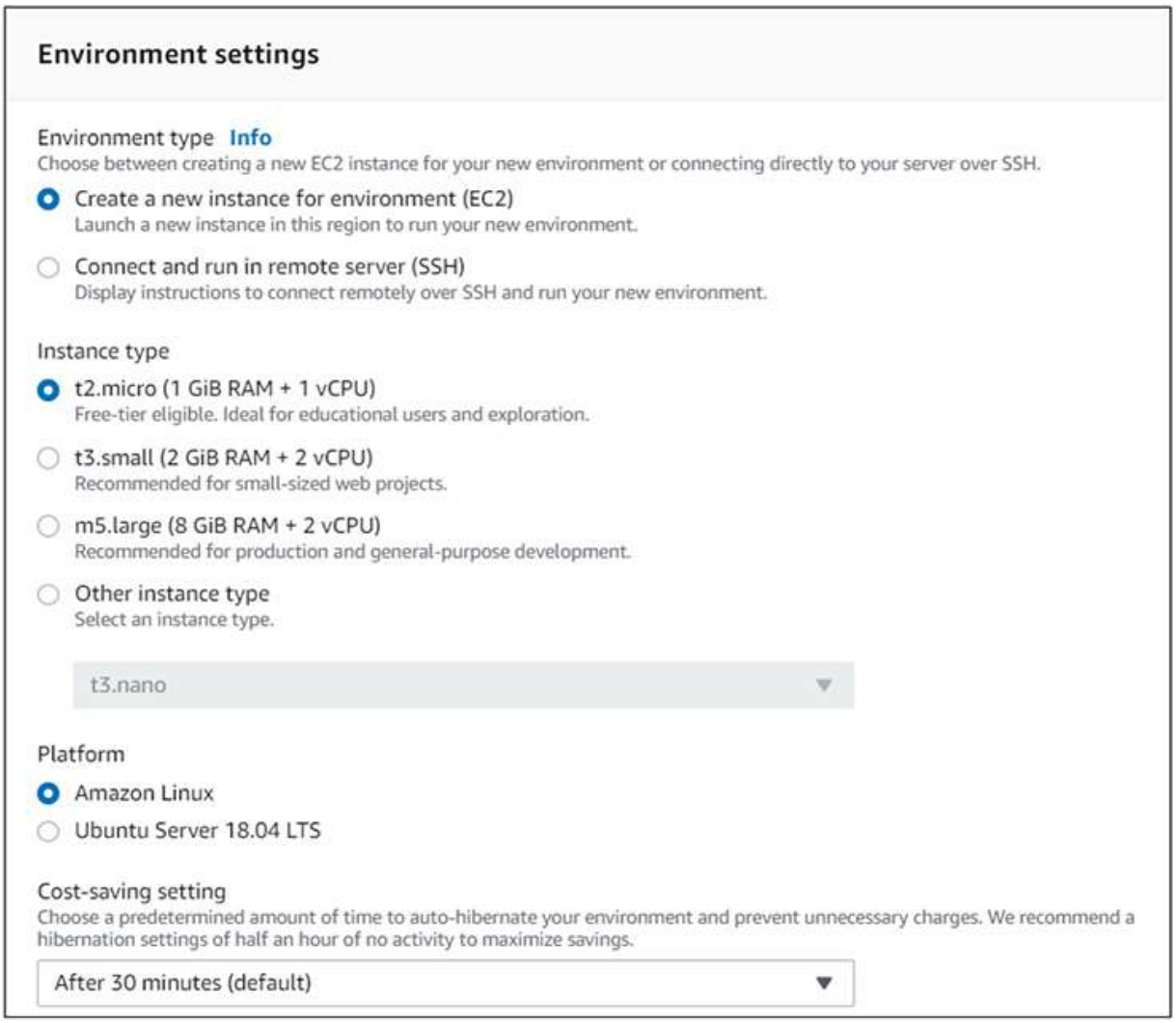
Navigate to the AWS CodeCommit service in the AWS Management Console and select Code from the left pane. You can see third-app.txt is added to the repository.



Now we will use AWS Cloud9, which is an integrated development environment (IDE), to work on your repository using a browser instead of using your local machine. Navigate to AWS Cloud9 from the AWS Management Console and click Create Environment. For the name, enter **my-cloud-ide** and click Next Step.



Leave all the default values selected and click Next Step.



Review all the details and click Create Environment.

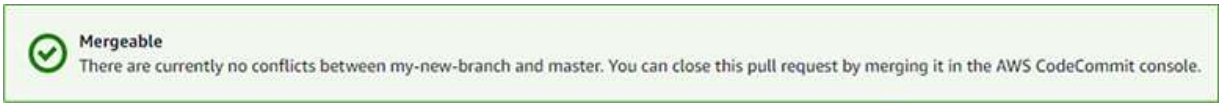
The AWS Cloud9 IDE environment creation is in progress.

AWS Cloud9 IDE is ready, and you are logged in as the Developer user by default. You can clone the repository using the git clone HTTPS command. Create a new branch using the git branch my-new-branch command and navigate to the new branch. Create a new source code my-cloud9-app.txt and add it using the git add command. You can commit the code using the git commit -m “comment” and use git push to push the new source code.

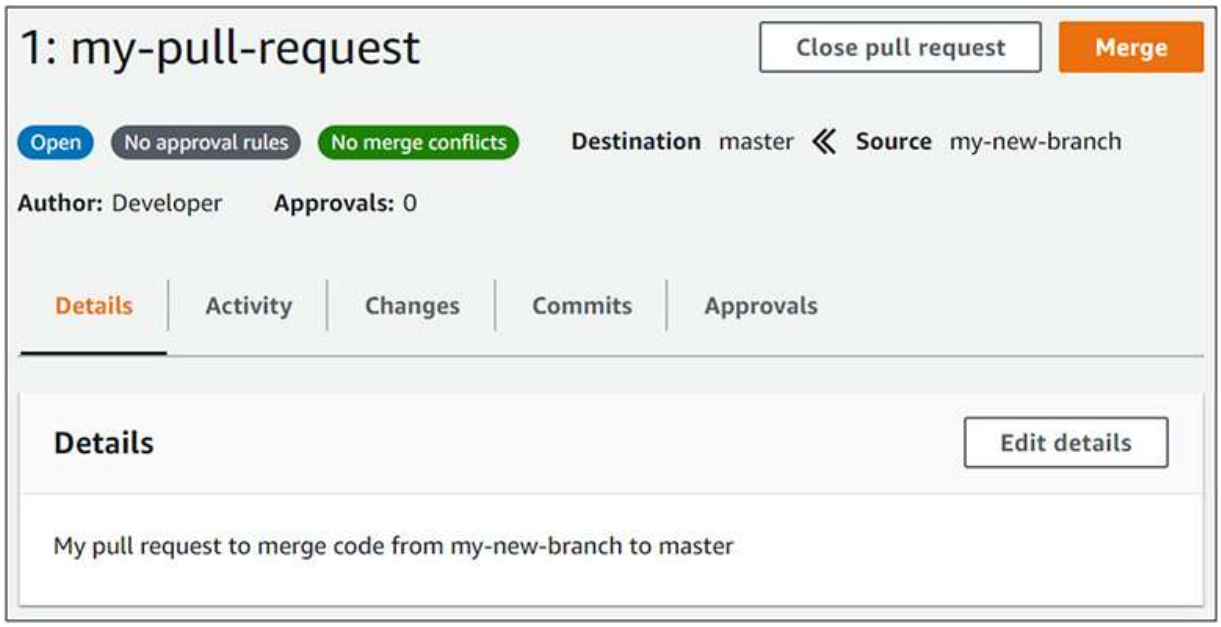
```
git - "ip-172-31-80-30"  ×  Immediate  ×  ⊕
Developer:~/environment $ git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/my-app-code-repository
Cloning into 'my-app-code-repository'...
remote: Counting objects: 9, done.
Unpacking objects: 100% (9/9), done.
Developer:~/environment $ cd my-app-code-repository/
Developer:~/environment/my-app-code-repository (master) $ git branch my-new-branch
Developer:~/environment/my-app-code-repository (master) $ git checkout my-new-branch
Switched to branch 'my-new-branch'
Developer:~/environment/my-app-code-repository (my-new-branch) $ echo "AWS Cloud9 welcomes you all" > my-cloud9-app.txt

Developer:~/environment/my-app-code-repository (my-new-branch) $ git add .
Developer:~/environment/my-app-code-repository (my-new-branch) $ git commit -m "adding my cloud9 app"
[my-new-branch fe3cd76] adding my cloud9 app
 Committer: EC2 Default User <ec2-user@ip-172-31-80-30.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)
 create mode 100644 my-cloud9-app.txt
Developer:~/environment/my-app-code-repository (my-new-branch) $ git push -u origin my-new-branch
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 396 bytes | 396.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/my-app-code-repository
 * [new branch]      my-new-branch -> my-new-branch
Branch my-new-branch set up to track remote branch my-new-branch from origin.
Developer:~/environment/my-app-code-repository (my-new-branch) $
```

Since we created the new source code in a branch, you need to create a pull request on the AWS CodeCommit page. Select the destination as master and the source as my-new-branch. Click the Compare button before creating the pull request.

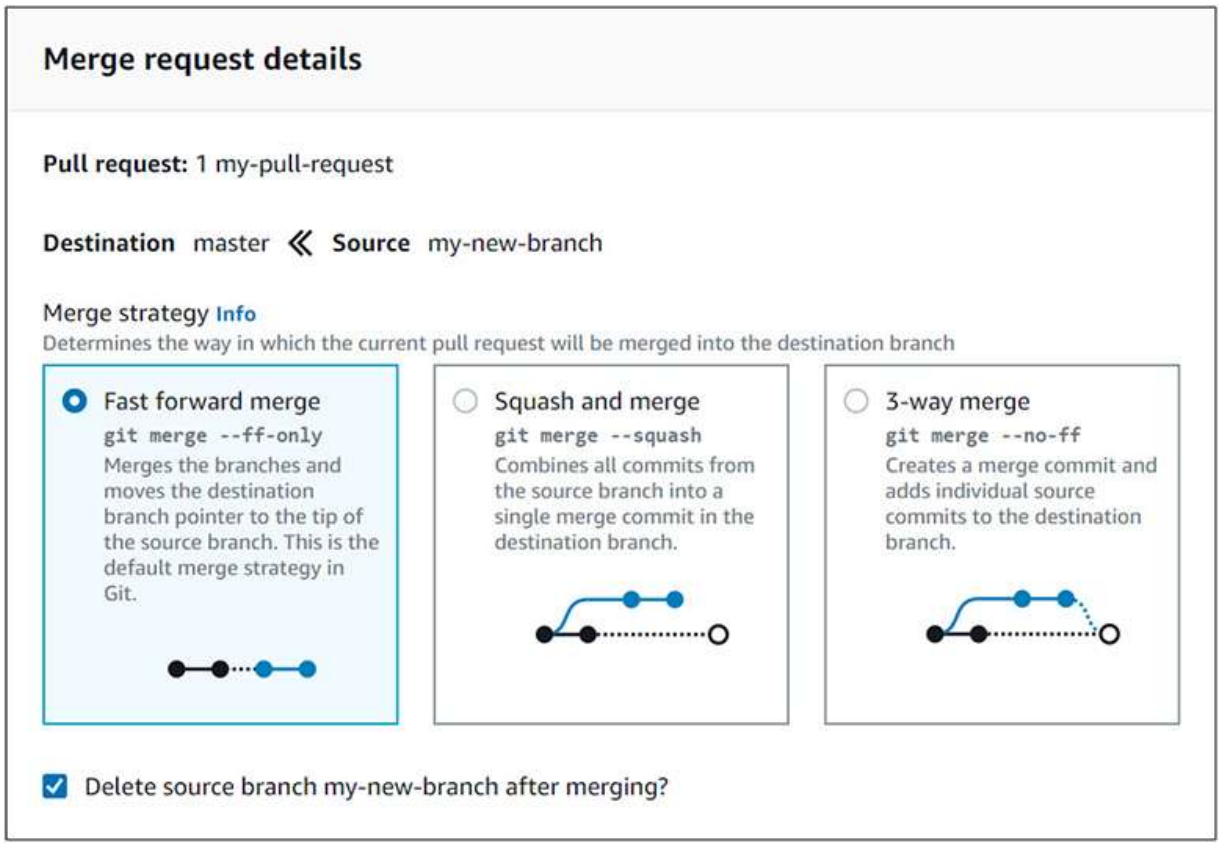
In the next screen, you will see the status as Mergeable. After reviewing the changes, click the Create Pull Request button.



You will see a success message with no merge conflicts. After reviewing the details, click the Merge button.



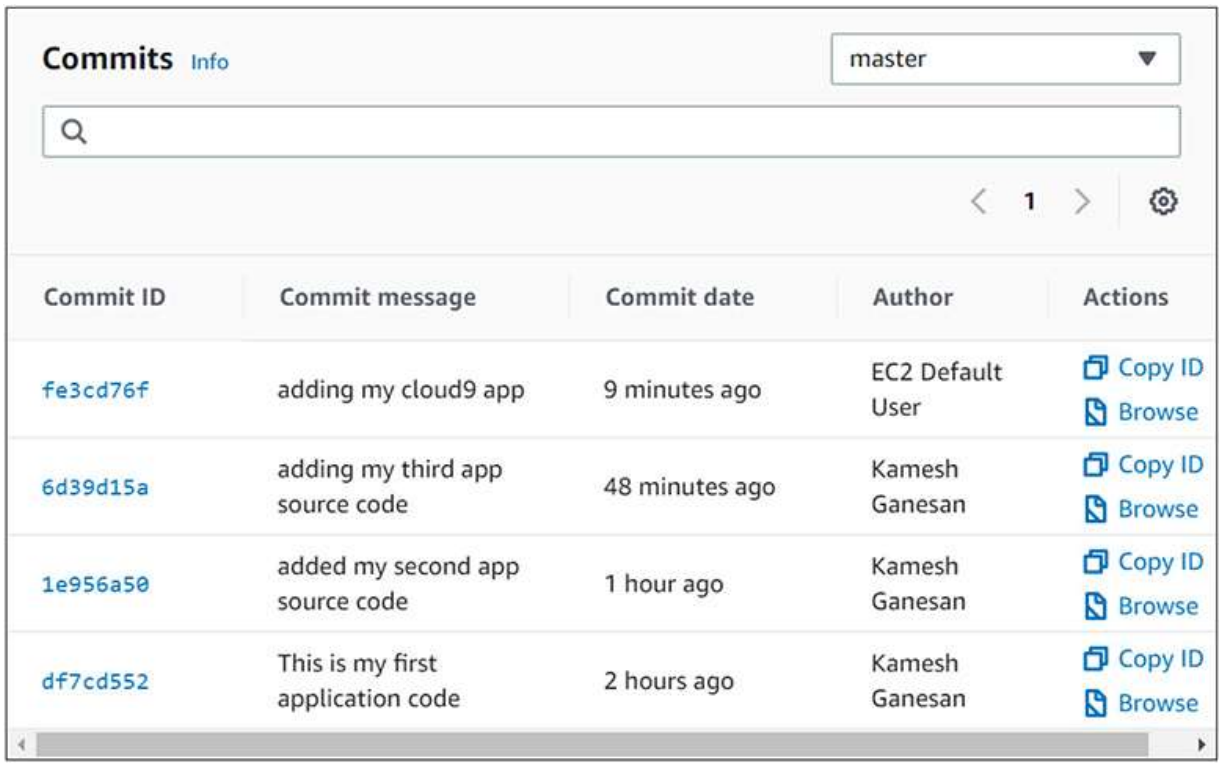
Here you have three merge options to select from. Choose Fast Forward Merge, which is the default git merge strategy, and click on the Merge Pull Request button. You can use other merge strategies, such as squash and merge or three-way merge, based on your requirements.



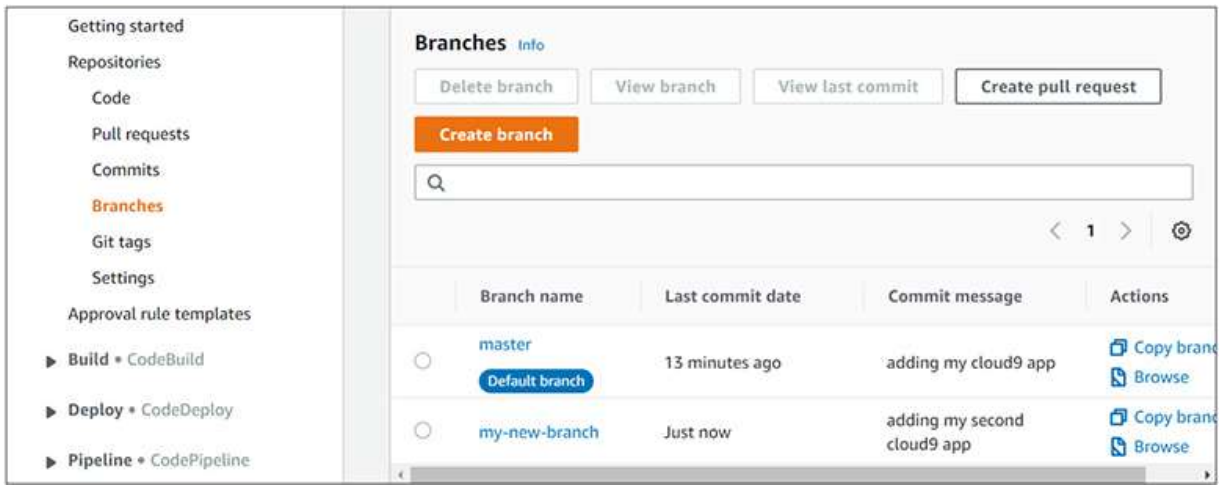
You can see that my-new-branch has been merged into the master with a success message.

When you click on Code on the left pane, you can see the new source code my-cloud9-app that we created using the AWS Cloud9 IDE.

You can see all the commits by selecting Commits from the left pane. You can explore all the commits from here, and you can list commits from the master or from any of the branches.



You can explore all your branches by choosing Branches from left navigation pane.



You can configure notification rules for a repository when comments are added to commits or pull requests. You also can set up a notification when a pull request is created or merged, in addition to the creation, deletion, or updating of branches. An Amazon CloudWatch Events rule and Amazon SNS topic can be configured for notifications. The existing Git repository can be migrated to the AWS CodeCommit repository. First, you need to create a repository in AWS CodeCommit. Then you need to clone the existing repository in your local machine and push it to AWS CodeCommit.

Chapter Review

This chapter began by explaining what AWS CodeCommit is. You created an IAM user with just AWS CodeCommit access. Then you went through the process of creating a repository; new source code; a new branch; and pull requests, as well as adding, committing, pushing, and comparing the source code using AWS CodeCommit from your local machine, as well as from the interactive development environment AWS Cloud9.

You gained the experience of creating a source code repository using AWS CodeCommit service and cloned it to your local machine as well as to AWS Cloud9 IDE. You also created sample source code from different places and pushed it to your AWS CodeCommit remote repository. This experience will help you in AWS Developer certification as well as in real world when you start working on AWS cloud environment.

Exercises

The following exercises will help you practice using AWS CodeCommit. You need to create an AWS account, as explained earlier, before performing these exercises. You can use the Free Tier when launching AWS resources, but make sure to terminate them at the end.

Exercise 23-1: Delete the AWS CodeCommit Repository Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CodeCommit console at https://console.aws.amazon.com/codecommit/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Repositories.
4. Choose Settings.
5. Navigate to the Delete Repository on the General tab. Choose Delete Repository.
6. Enter delete in the popup window and choose to delete.

Exercise 23-2: Delete the AWS Cloud9 Environment Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS Cloud9 console at https://console.aws.amazon.com/cloud9/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Environments.
4. Choose the environment you want to delete and click on the Delete button.
5. Enter delete in the popup window and choose to delete. The Amazon EC2 instance will be terminated by AWS Cloud9.

Questions

The following questions will help you gauge your understanding of the contents in this chapter. Read all the answers carefully because there might be more than one correct answer. Choose the best responses for each question.

1. Your company security team directed you to find a solution for storing source code securely, which should be a managed solution for everyone in your team who is spread across the globe. They should be able to use their existing git tools. Which AWS service you will suggest?

   A. AWS CodeBuild

   B. AWS CodeDeploy

   C. AWS CodePipeline

   D. AWS CodeCommit

2. Your company security policy mandates that the source code should not be cloned to your local machine. Your company is using AWS CodeCommit to store and share the source code repository securely across the team. Which AWS service you would recommend to clone and work on your source code securely?

   A. Amazon Athena

   B. AWS Cloud9

   C. AWS Outposts

   D. AWS Config

3. Your company is developing a new product, and many teams are working on different features of this product. As a developer, how you will make sure each team can work on their feature set and merge the source code once ready in AWS CodeCommit?

   A. Create a new repository for each team and combine when ready

   B. Create a different branch for each team and merge to the master when ready

   C. Ask each team to work on their local machine and upload to AWS CodeCommit when ready

   D. AWS CodeCommit does not support merging the source code

4. A team member cloned the application source code repository to a local machine and made all the necessary changes. The team member is getting an error while pushing the code to AWS CodeCommit. What is the correct sequence of git command execution?

   A. Git add – Git commit – Git push

   B. Git add – Git push – Git commit

   C. Git commit – Git Push – Git add

   D. Git push – Git add – Git commit

5. Your development team should be able to push changes to only their branches but not allowed to push changes or create a merge pull request to the master branch in AWS CodeCommit. How can you achieve this in AWS?

   A. Create a policy with allow push, merge, and add to the master and attach it to the developer IAM group

   B. Enable server-side encryption for the master branch

   C. Enable client-side encryption for the master branch

   D. Create a policy with deny push, merge; add to the master; and attach it to the developer IAM group

6. Your company has many AWS accounts and created the AWS CodeCommit repository in the development AWS account. How can you allow cross-account access to the development account AWS CodeCommit repository?

   A. Create an IAM role to delegate access and use AWS STS to assume the role in another AWS accounts

   B. Create a VPN connection between all your AWS accounts to enable cross-account access

   C. Create VPC peering to connect all your AWS accounts to enable cross-account access

   D. Create an IAM user in each account and grant access directly from the development AWS account

7. What three merge strategies are available in AWS CodeCommit? (Choose three.)

A. Two-way merge strategy

B. Squash merge strategy

C. Three-way merge strategy

D. Fast-forward merge strategy

8. A DevOps team configured an AWS CodeCommit repository, and they want to receive a notification whenever a developer commits or merges changes to the master. Which of the following steps will alert the DevOps team of this?

   A. Set up an Amazon CloudWatch events rule to be triggered and add the Amazon SNS topic as the notification target

   B. Set up Amazon CloudTrail for repository changes and add the Amazon SNS topic as the notification target

   C. Set up AWS Lambda to be triggered every five minutes and add the Amazon SNS topic as the notification target

   D. Set up AWS Batch to be triggered and add the Amazon SNS topic as the notification target

9. What two protocols are supported in AWS CodeCommit to clone a repository? (Choose two.)

   A. TCP

   B. HTTPS

   C. HTTP

   D. SSH

10. You have an existing Git repository that you want to migrate to AWS CodeCommit. What migration steps do you need to perform? (Choose three.)

    A. Create an AWS CodeCommit repository

    B. Clone the existing Git repository in your local machine

    C. Delete the AWS CodeCommit repository

    D. Push the changes to your AWS CodeCommit repository

Answers

1. **D.** You can use AWS CodeCommit to collaborate on the source code with other team members.
2. **B.** AWS Cloud9 provides a secure integrated development environment in AWS Cloud.
3. **B.** You can create a different branch for each team and merge them to the master when ready.
4. **A.** You need add the files first using git add, then commit the changes using git commit, and finally push the changes to the remote repository using git push.
5. **D.** You can create a policy with deny access to push, merge, and add to the master and attach it to the developer IAM group.
6. **A.** You need to create an IAM role to delegate access and use AWS STS to assume the role in another AWS account.
7. **B, C, D.** The three merge strategies in AWS CodeCommit are fast-forward, squash, and three-way.
8. **A.** Set up an Amazon CloudWatch events rule to be triggered and add the Amazon SNS topic as the notification target.
9. **B, D.** HTTPS and SSH are the two protocols used in AWS CodeCommit.
10. **A, B, D.** First, create an AWS CodeCommit repository and then clone the existing Git repository in your local machine. Finally, push the changes to your AWS CodeCommit repository.

Additional Resources

- **AWS CodeCommit** AWS official documentation is the only place where you will get up-to-date information about any AWS service, including AWS CodeCommit.
  `https://docs.aws.amazon.com/codecommit/index.html`
- **AWS CodeCommit Blog** AWS blogs are another great source of information where you get step-by-step implementations of AWS CodeCommit along with other AWS services.
  `https://aws.amazon.com/blogs/devops/tag/aws-codecommit/`

CHAPTER 24

Building an Application Using AWS CodeBuild

In this chapter, you will learn

- AWS CodeBuild
- Working with CodeBuild
- Test reporting

This chapter will show how to build a Docker image and store it in the Amazon Elastic Container Registry (Amazon ECR).

AWS CodeBuild

As a developer you might have code-building tools such as Gradle, Apache Ant, Apache Maven, Bamboo, CruiseControl, or Hudson that automates the process of building an application from your source code by compiling and packaging the code into an executable application. A typical build automation comprises scripting and automating different tasks that you as a developer perform in your daily life.

AWS CodeBuild service, which is a managed build service in AWS Cloud, compiles the source code and produces artifacts that can be used to deploy your application. AWS CodeBuild scales on demand without the need to provision or manage the build servers.

Build Projects

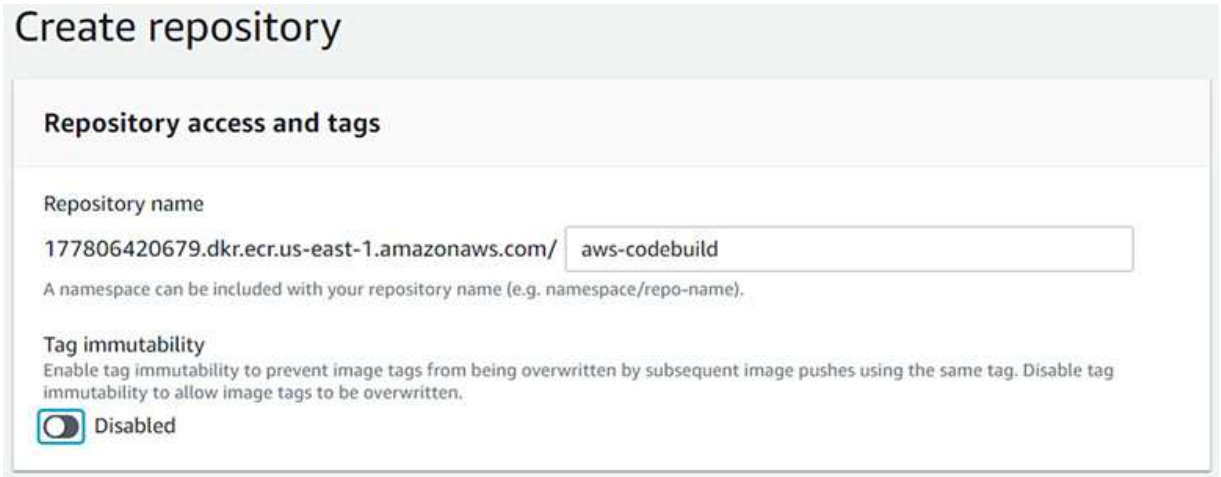
A build project contains all the required details that need to be provided to AWS CodeBuild as input, such as source code location (AWS CodeCommit, Amazon S3, GitHub, Bitbucket), the build environment to use, how to run a build, and where to store the build output.

Build Environment

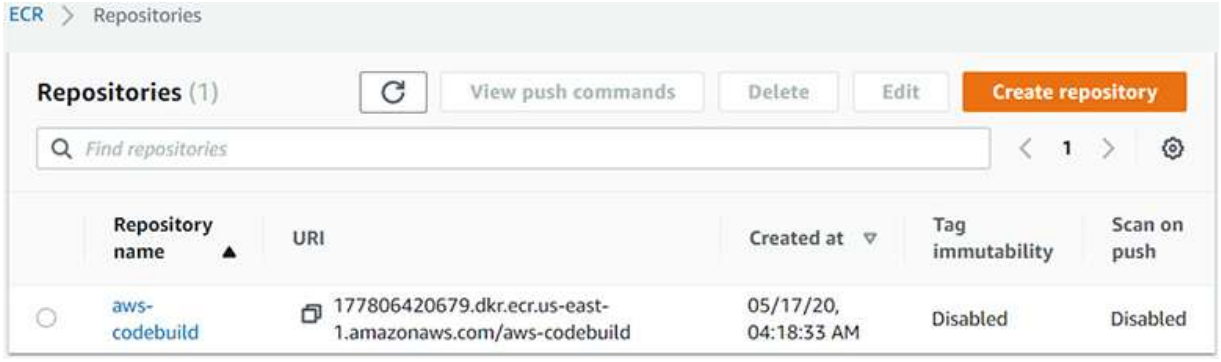
A build environment consists of the runtime of your programming language, tools, and operating system that AWS CodeBuild uses to run any build. The nohup command is used to run the background tasks, and you can forcibly stop the running background tasks by using the disown command. AWS CodeBuild provides several environment variables, such as AWS_REGION, CODEBUILD_BUILD_IMAGE, and CODEBUILD_LOG_PATH, and you can also provide custom environment variables. The printenv command lists all the available environment variables in your build environment.

Working with AWS CodeBuild

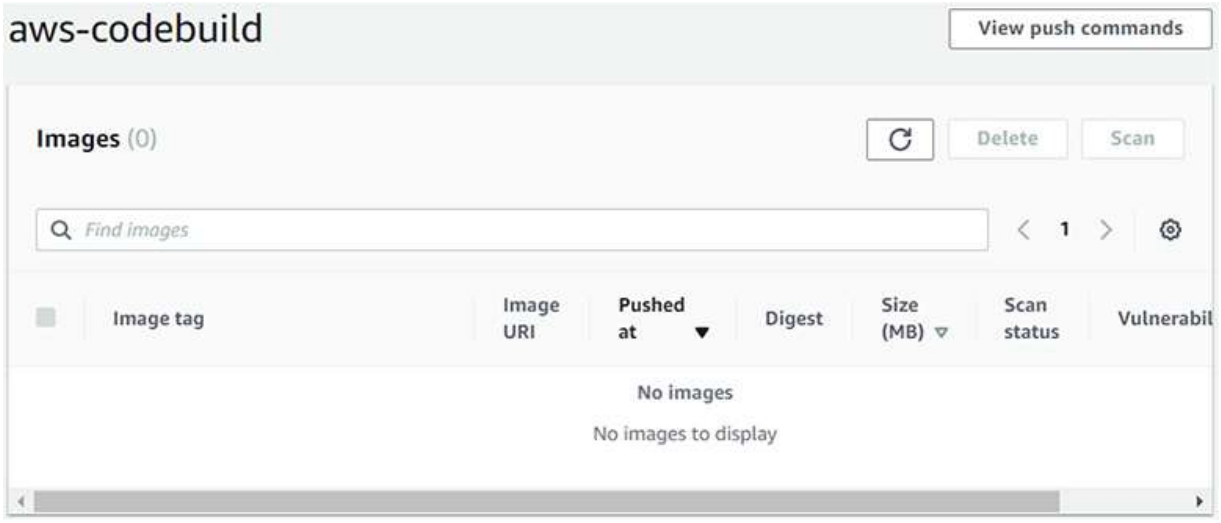
We are going to create a container image registry in Amazon ECR and build all the required specification configuration source code files using AWS CodeCommit and build the Docker application code using AWS CodeBuild. First, log in to your AWS management console, navigate to the Amazon ECR service, and choose Create Repository. Enter the repository name as **aws-codebuild** and click on the Create Repository button.



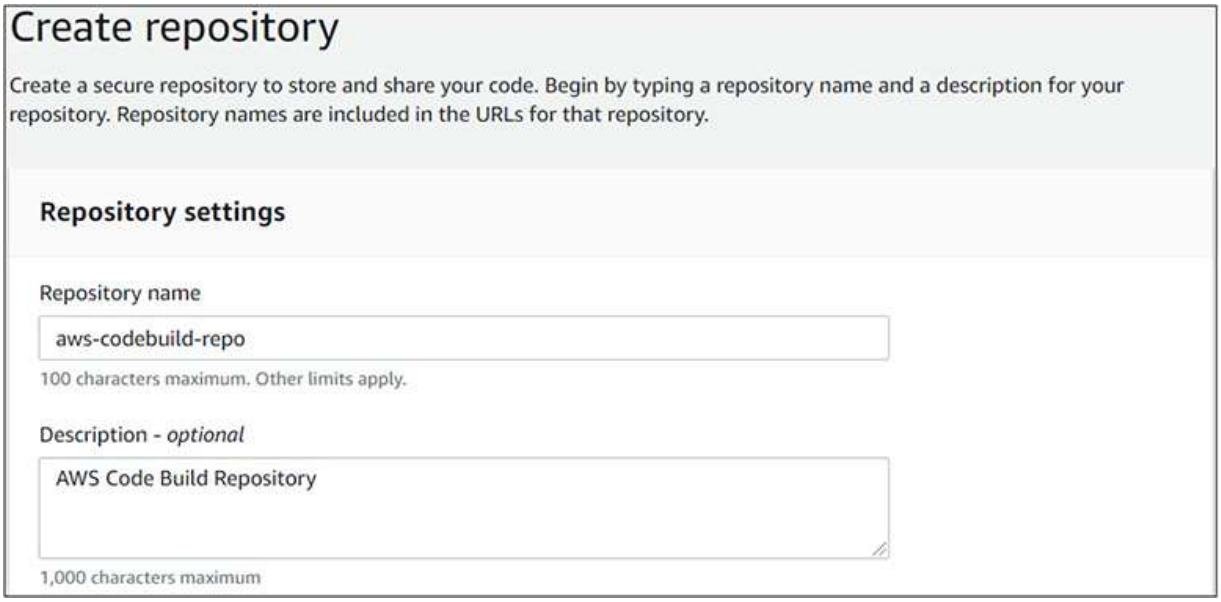
You will be directed to the ECR Repositories page, where you can see the repository name and ECR repository URI. Note down this URI, which we need during the source code configuration in later steps.



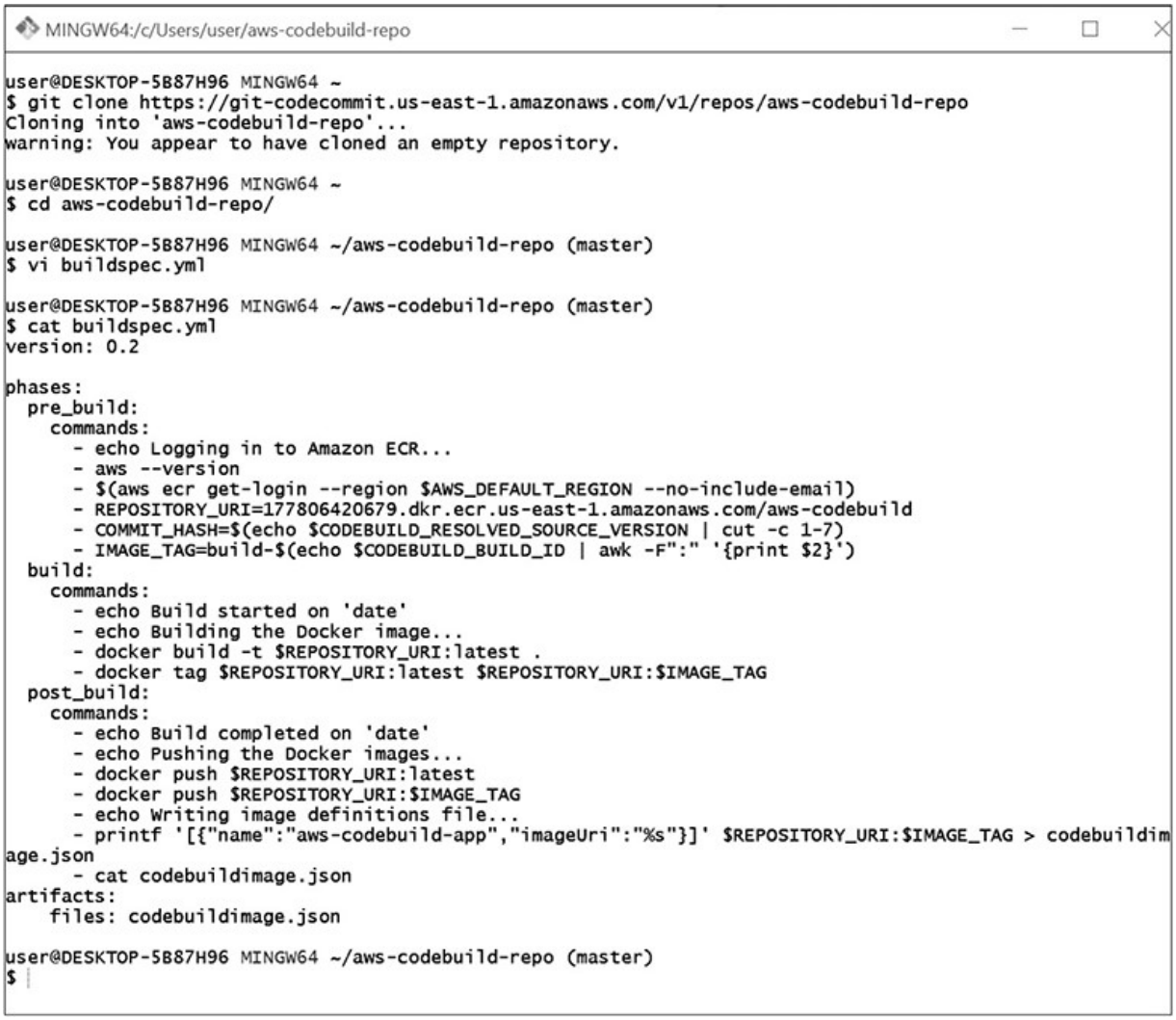
Since it is a new ECR repository, it will not contain any Docker images.



Now, navigate to AWS CodeCommit and choose Create Repository. Enter the repository name as **aws-codebuild-repo**, and then enter a description before clicking on the Create button.



Now, from your local machine, clone the empty repository that you just created in AWS CodeCommit using the git clone command, as shown here.



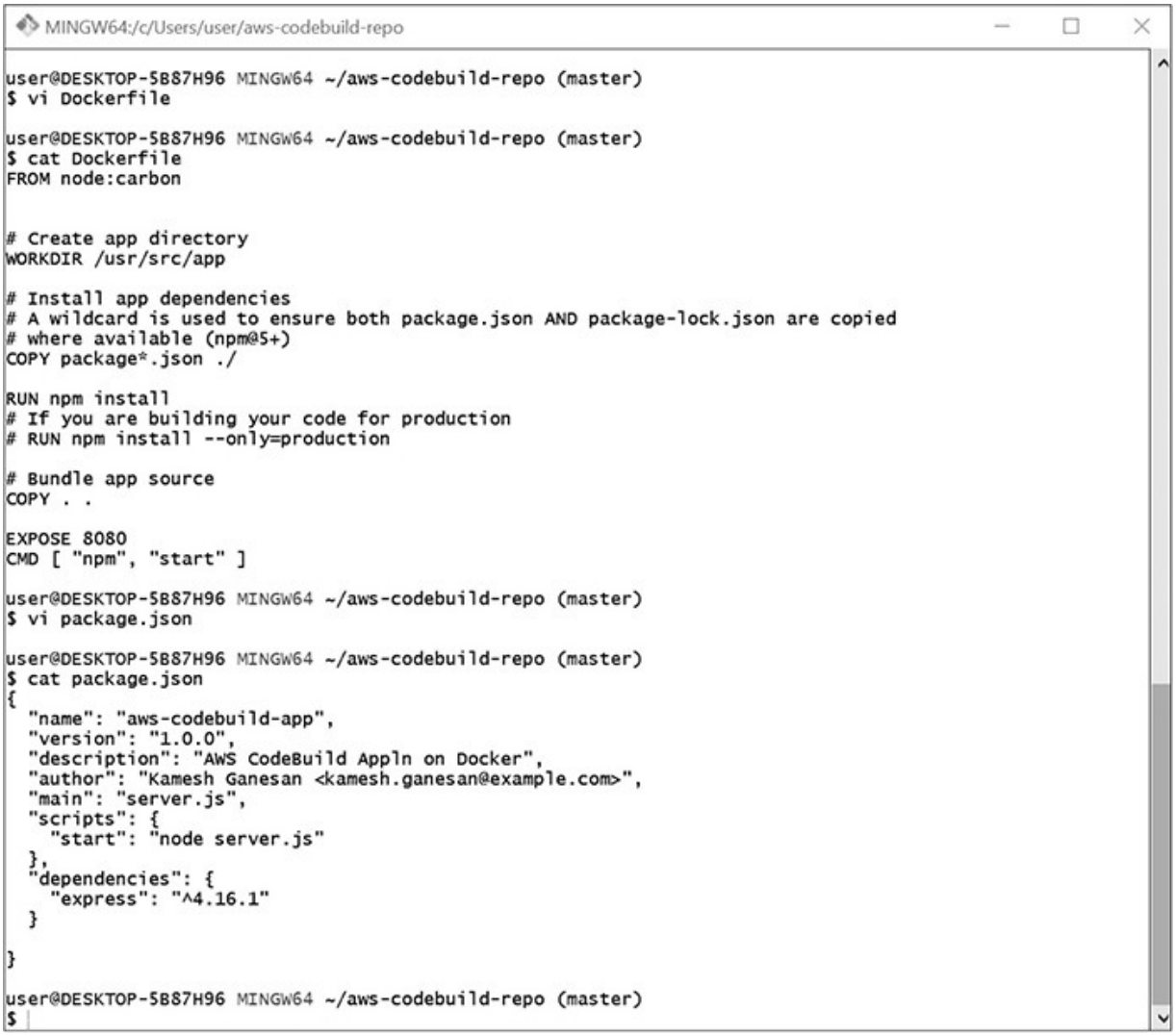
Your source code must have a buildspec file, which contains the build specifications that could include a set of build commands in YAML format and the related settings that AWS CodeBuild uses to run a build. To declare a buildspec, navigate to aws-codebuild-repo and create a buildspec YAML file:

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
      - REPOSITORY_URI=177806420679.dkr.ecr.us-east-1.amazonaws.com/aws-codebuild
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=build-$(echo $CODEBUILD_BUILD_ID | awk -F":" '{print $2}')
```

```
  build:
    commands:
      - echo Build started on 'date'
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on 'date'
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"aws-codebuild-app-v1.0","imageUri":"%s"}]' $REPOSITORY_
URI:$IMAGE_TAG > codebuildimage.json
      - cat codebuildimage.json
artifacts:
    files: codebuildimage.json
```

Next create another file called **Dockerfile** with source code:

```
FROM node:carbon
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 8080
CMD [ "npm", "start" ]
```



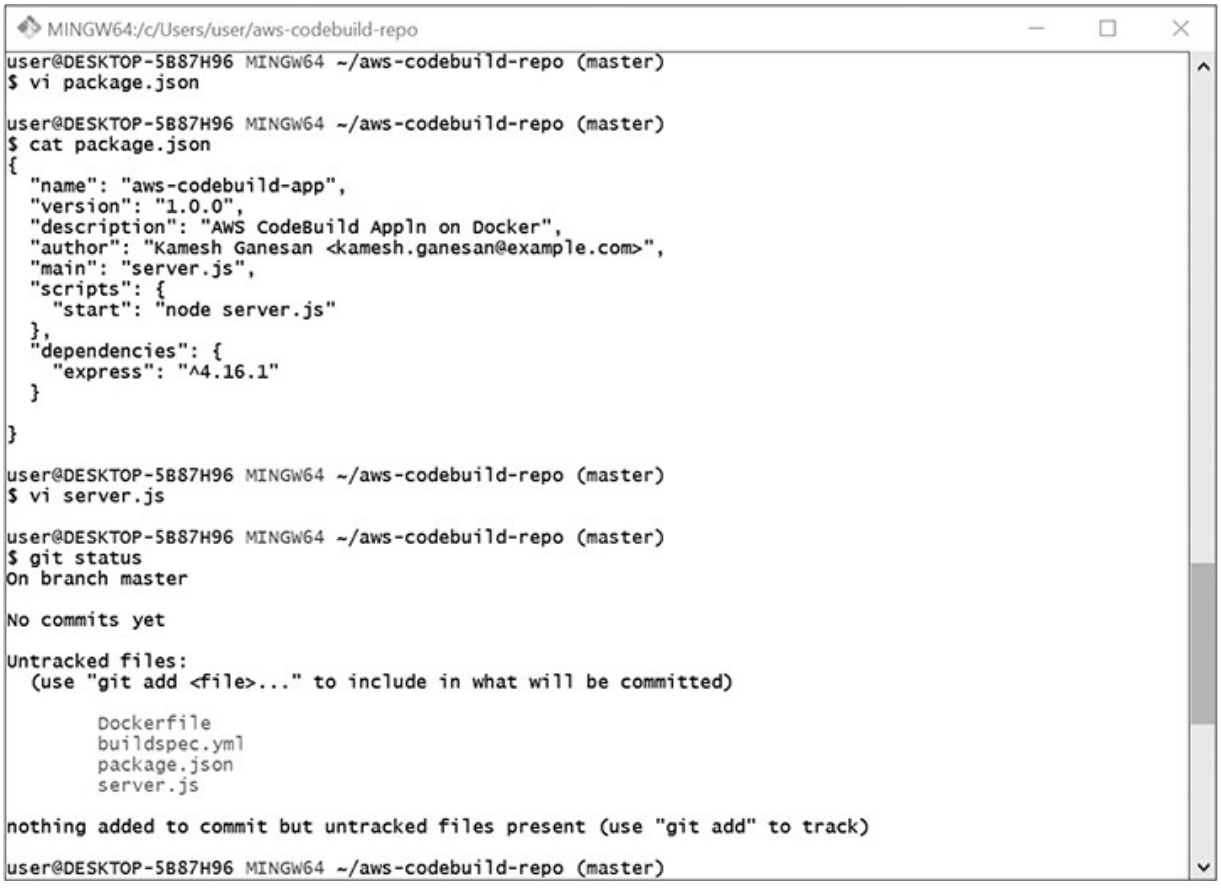
Then create a .json file called **package.json** using the source code:

```
{
  "name": "aws-codebuild-app",
  "version": "1.0.0",
  "description": "AWS CodeBuild Appln on Docker",
  "author": "Kamesh Ganesan <kamesh.ganesan@example.com>",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
     },
  "dependencies": {
    "express": "^4.16.1"
     }
  }
```

Finally, create a JavaScript file called **server.js** with the following source:

```
const express = require('express');
// Constants used in this application
const PORT = 8080;
const HOST = '0.0.0.0';
// AWS CodeBuild Application
const app = express();
app.get('/', (req, res) => {
  res.send('<h1 style="color:green;">AWS CodeBuild Application v1.0</h1> \n');
});
app.listen(PORT, HOST);
console.log(`Running on http://${HOST}:${PORT}`);
```

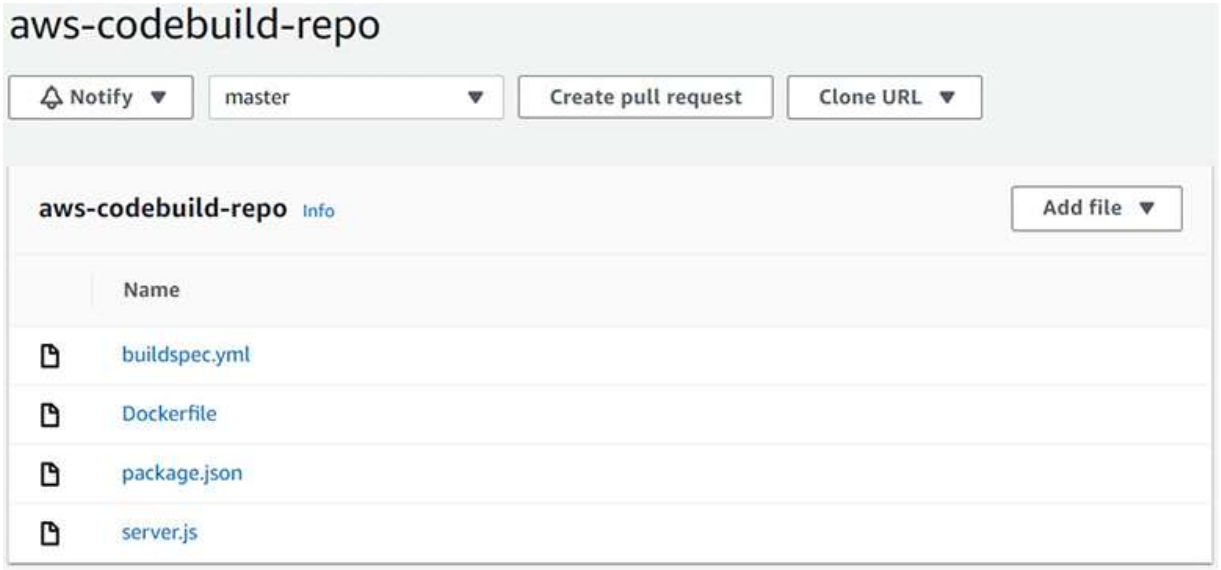
Enter the git status to see the four files that you created.



Add the files using the git add command and commit the changes using git commit -m “comment.” When everything is successful, push the code to the remote repository using the git push command.

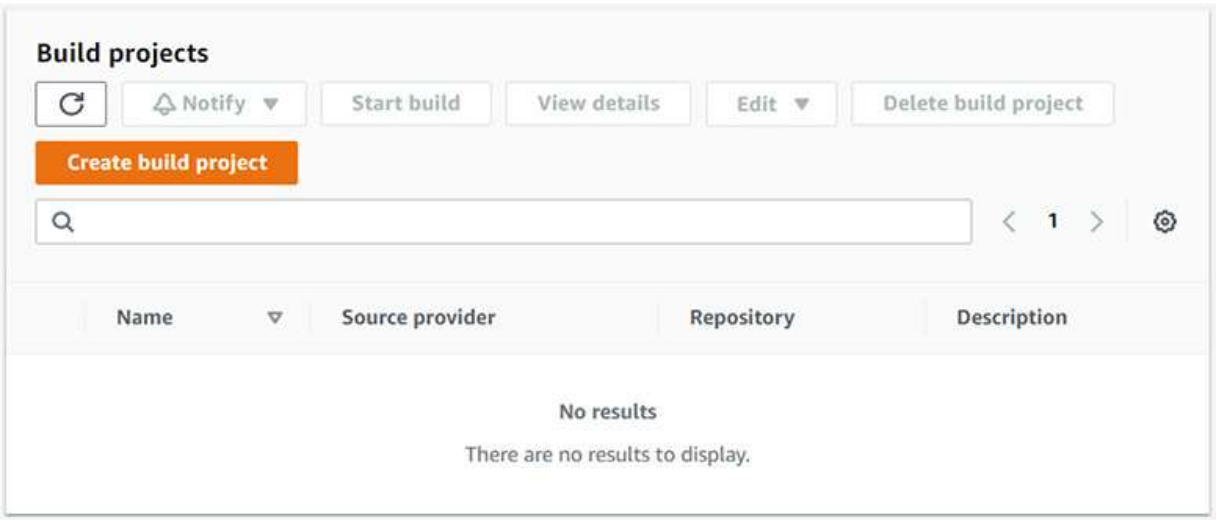
```
MINGW64:/c/Users/user/aws-codebuild-repo

user@DESKTOP-5B87H96 MINGW64 ~/aws-codebuild-repo (master)
$ git add .
warning: LF will be replaced by CRLF in Dockerfile.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in buildspec.yml.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in package.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in server.js.
The file will have its original line endings in your working directory

user@DESKTOP-5B87H96 MINGW64 ~/aws-codebuild-repo (master)
$ git commit -m "docker image build using aws-codebuild"
[master (root-commit) 03bdf0f] docker image build using aws-codebuild
 4 files changed, 78 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 buildspec.yml
 create mode 100644 package.json
 create mode 100644 server.js

user@DESKTOP-5B87H96 MINGW64 ~/aws-codebuild-repo (master)
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.52 KiB | 517.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/aws-codebuild-repo
 * [new branch]      master -> master

user@DESKTOP-5B87H96 MINGW64 ~/aws-codebuild-repo (master)
$
```

Next navigate to the AWS CodeCommit service using the AWS Management Console. Choose Repositories and then Code from the left pane. You can see all the source code has been pushed successfully.

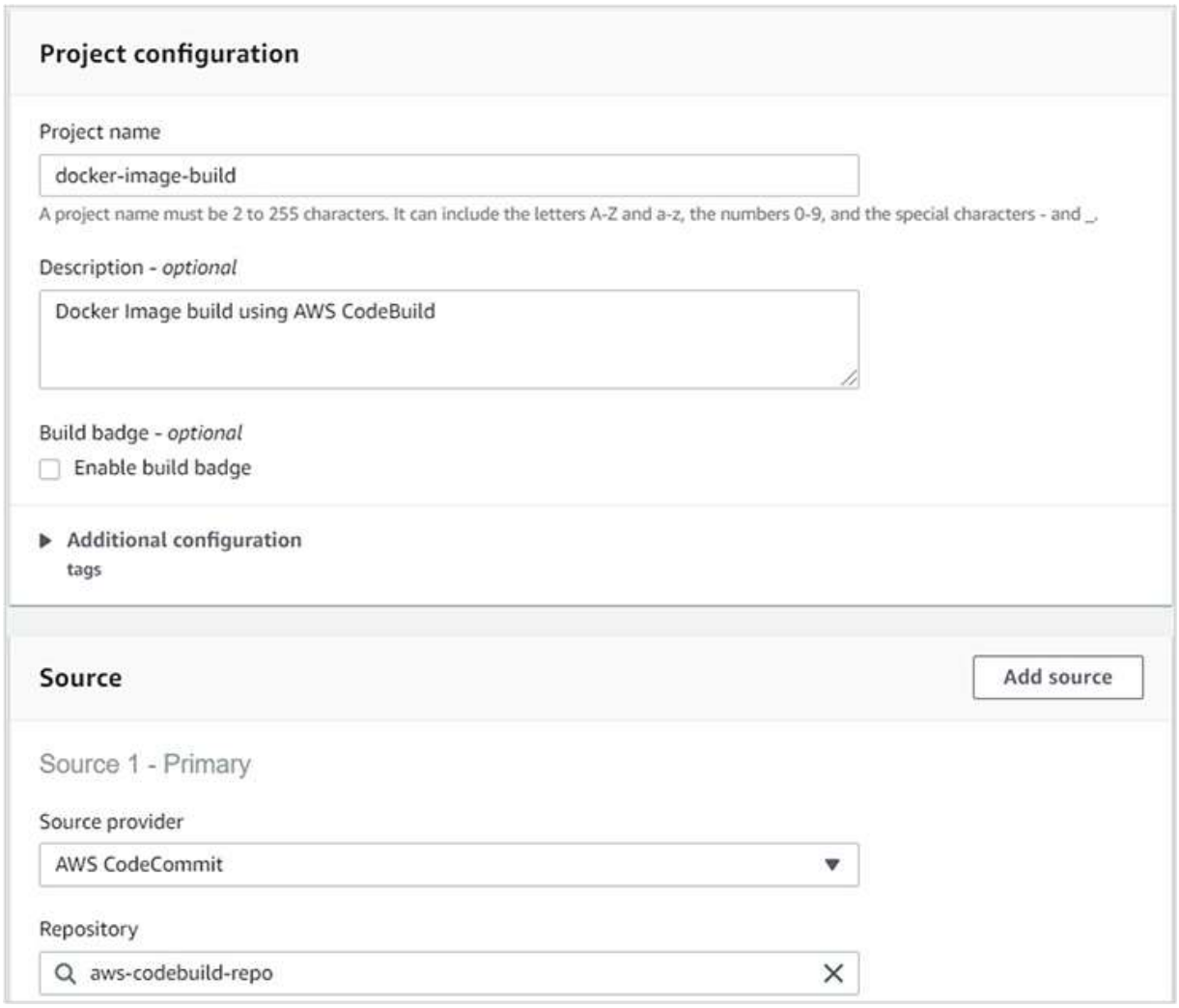


All the prerequisites are ready for us to build our Docker code. Navigate to the AWS CodeBuild service and choose Build Projects. Click the Create

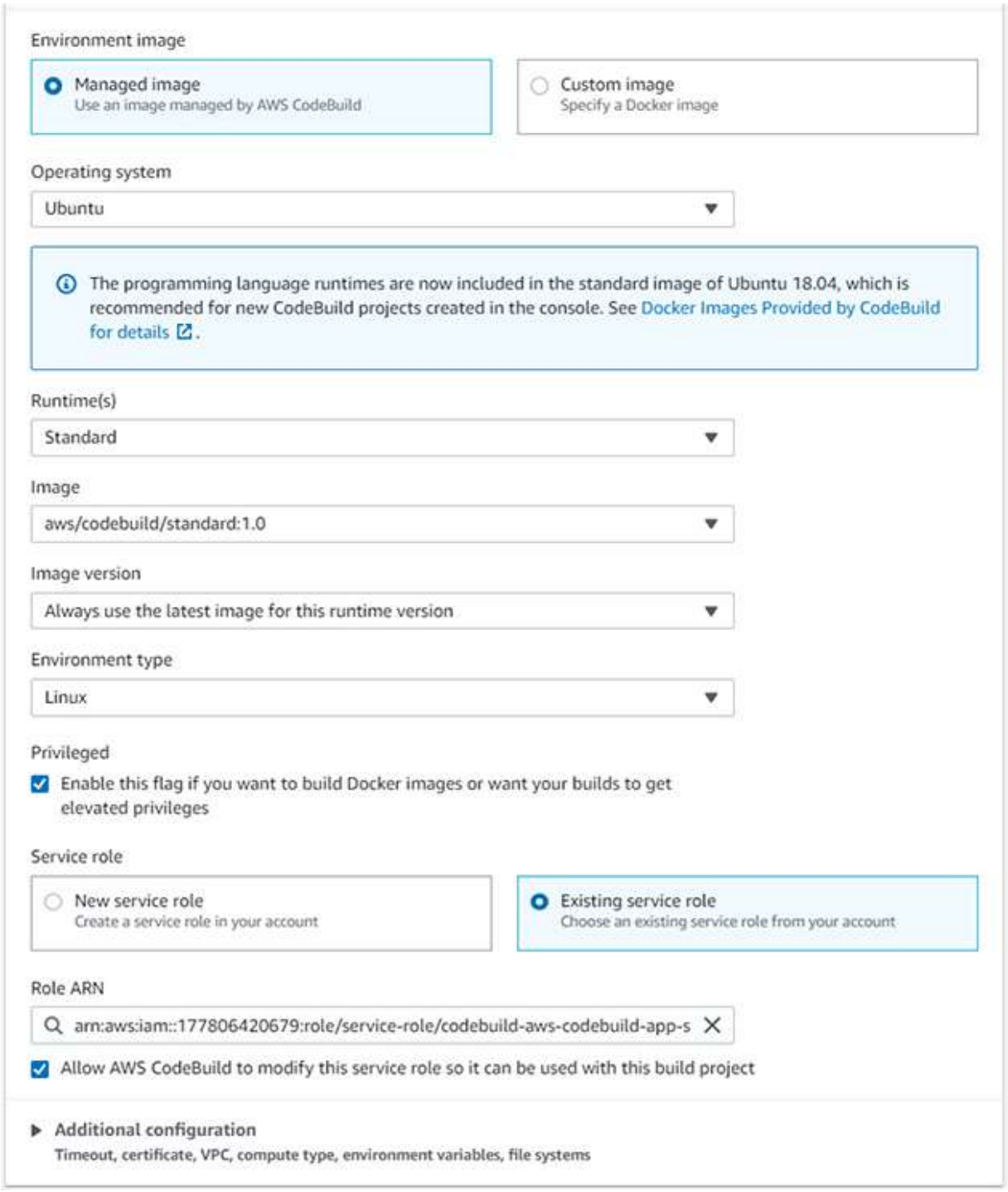
Build Project button.



Enter the project name as **docker-image-build** and provide a description. Next, you need to add the source, so choose AWS CodeCommit as the source provider from the dropdown and choose aws-codebuild-repo for the repository.



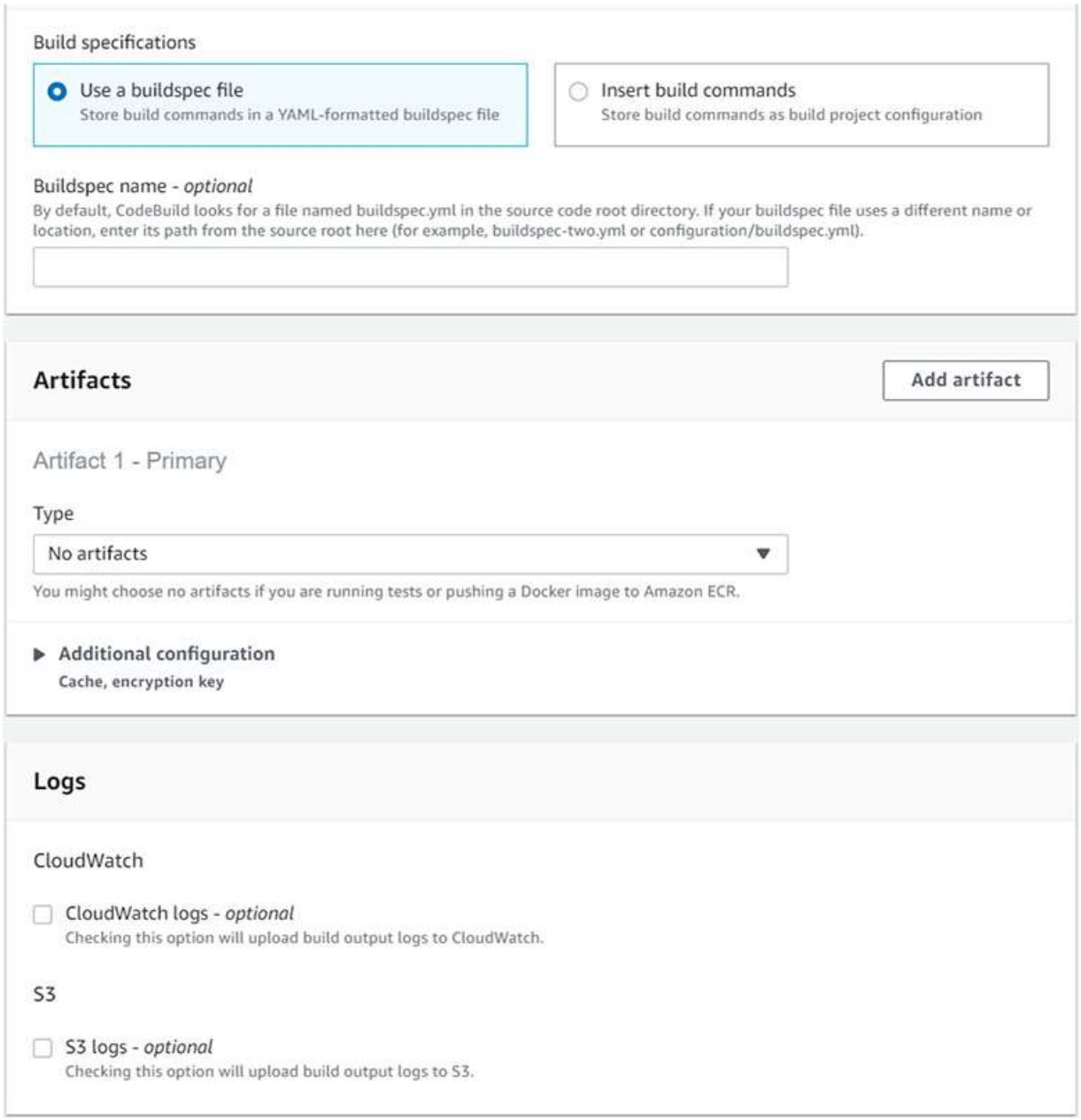
Now you can choose either a managed image or custom image for the environment image. Choose Managed Image and Ubuntu for the operating system, as shown next. Then choose Runtime as the standard and aws/codebuild/standard:1.0 for the image. The runtime is standard for Amazon Linux 2 and Ubuntu and base for the Windows operating system. You can either choose the image version from the dropdown menu or leave the default option to always use the latest image for this runtime version. Choose Linux as the environment type from the dropdown. You need to select the checkbox for Privileged to build Docker images.



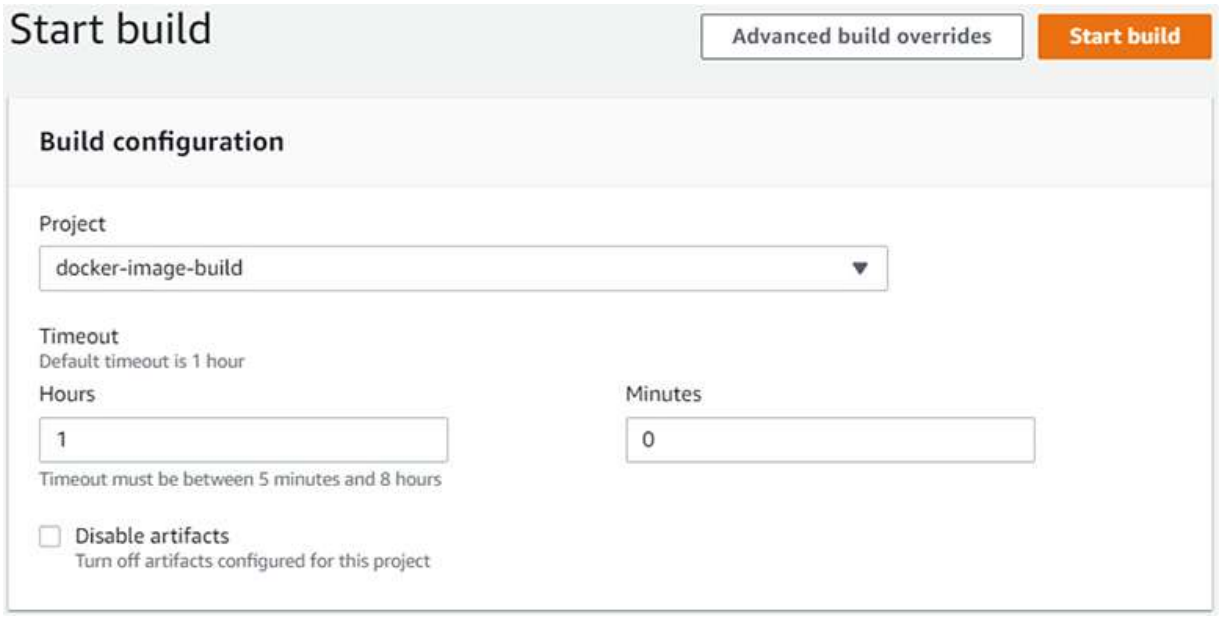
Next, choose an existing service role with required access or create to create a new service role. If you require your AWS CodeBuild to access resources inside your Virtual Private Cloud (VPC), then choose Additional Configuration and provide the VPC, subnet, security group, etc. Because, typically AWS CodeBuild cannot access your resources like Amazon EC2,

Amazon RDS is defined inside a VPC. By enabling VPC connectivity, you can test your build against data in your Amazon RDS database or interact with web services hosted on Amazon EC2 in a private subnet. In addition, you have the option to add a timeout between 5 minutes and 8 hours (the default timeout is 1 hour). Also, you can install a self-signed certificate or a certificate signed by a certification authority, which is stored on your Amazon S3 bucket.

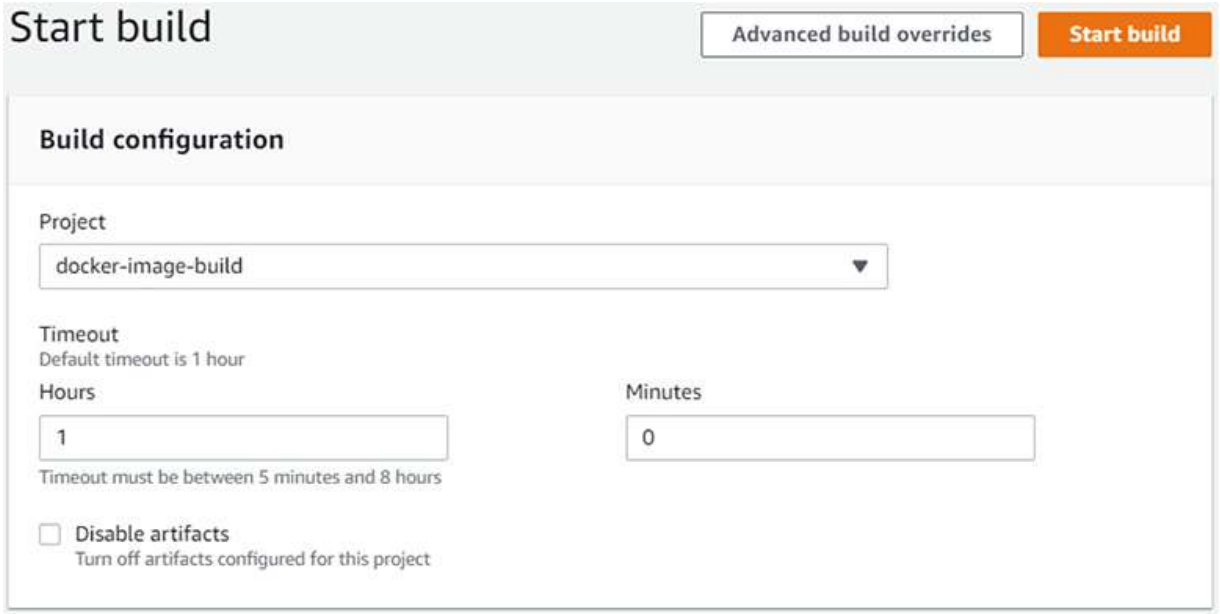
Now you can either use the buildspec YAML file that was created already or insert build commands to create a new buildspec YAML file, and you will provided with a sample script template when you click on Switch To Editor. Since we created the YAML file already, choose Use A Buildspec File. Since you are pushing the Docker image to ECR, choose No Artifacts. Choose Build ID for the namespace and the packaging as None or Zip to compress the file before storing it in Amazon S3. If you select Amazon S3, you will also have an option to provide the encryption key. You can either supply the AWS KMS customer master key or leave the default setting. Finally, you have option to choose to store the build output logs. You can choose Amazon CloudWatch logs to upload build output logs to Amazon CloudWatch or choose Amazon S3 to upload build output logs to the Amazon S3 bucket. Click Create Build Project.



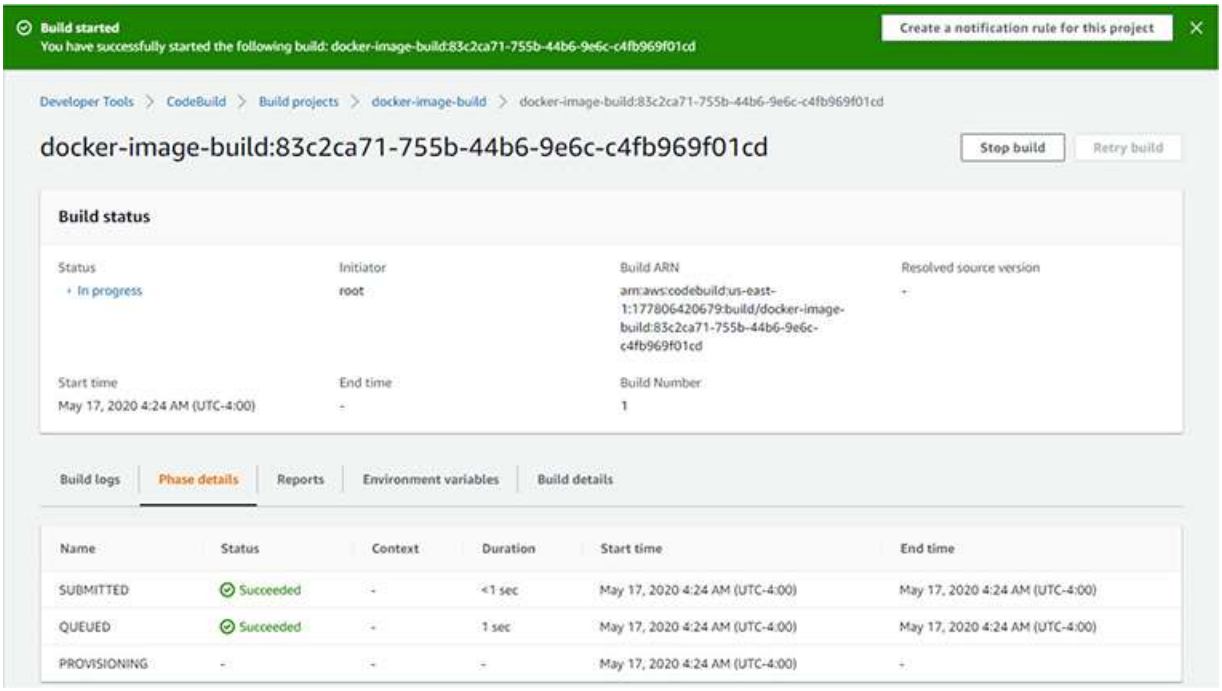
Now you are ready to build, so click Start Build.



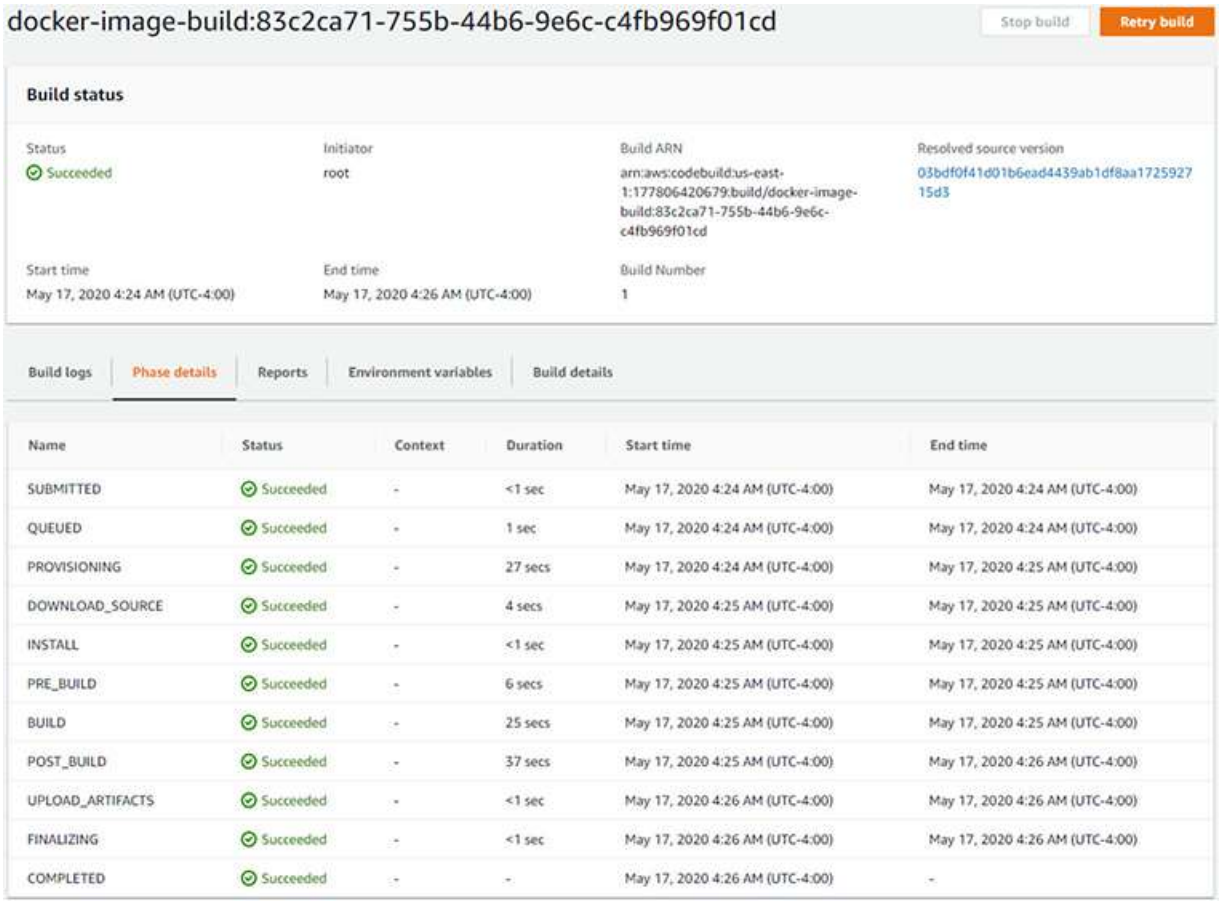
Provide the project name as **docker-image-build** and leave the other options at the default settings. Click the Start Build button again.



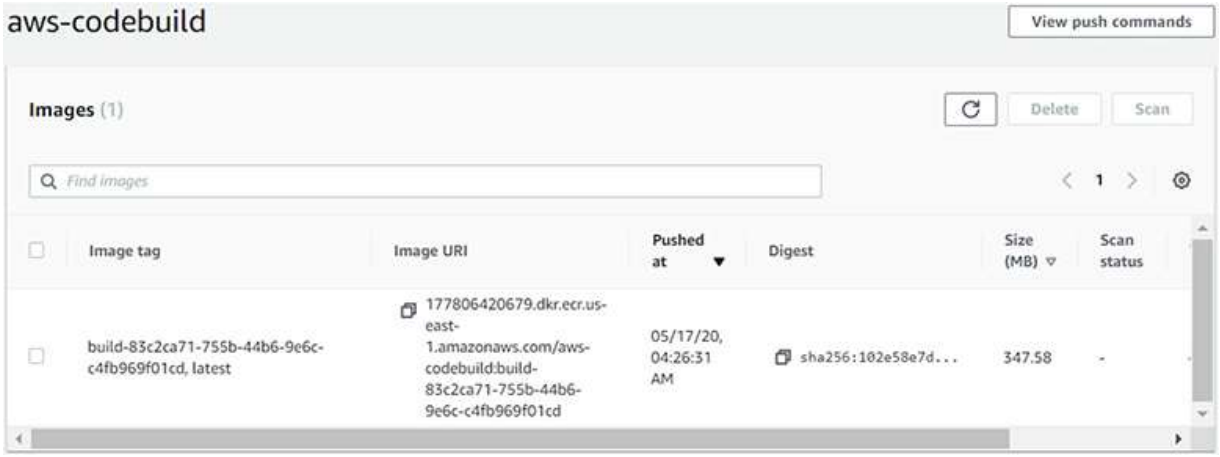
Your code build is successfully started, and you can see various stage names as submitted, queued, and provisioning.



In few minutes, you can see all the build phases have succeeded. You can see the overall build status is “Succeeded” and the build number is 1.



Now, navigate to the aws-codebuild repository in Amazon ECR and see a new image listed.



Let us make a simple change in the buildspec file from your local machine. Change the aws-codebuild-app-v1.0 to **aws-codebuild-app-v2.0** in

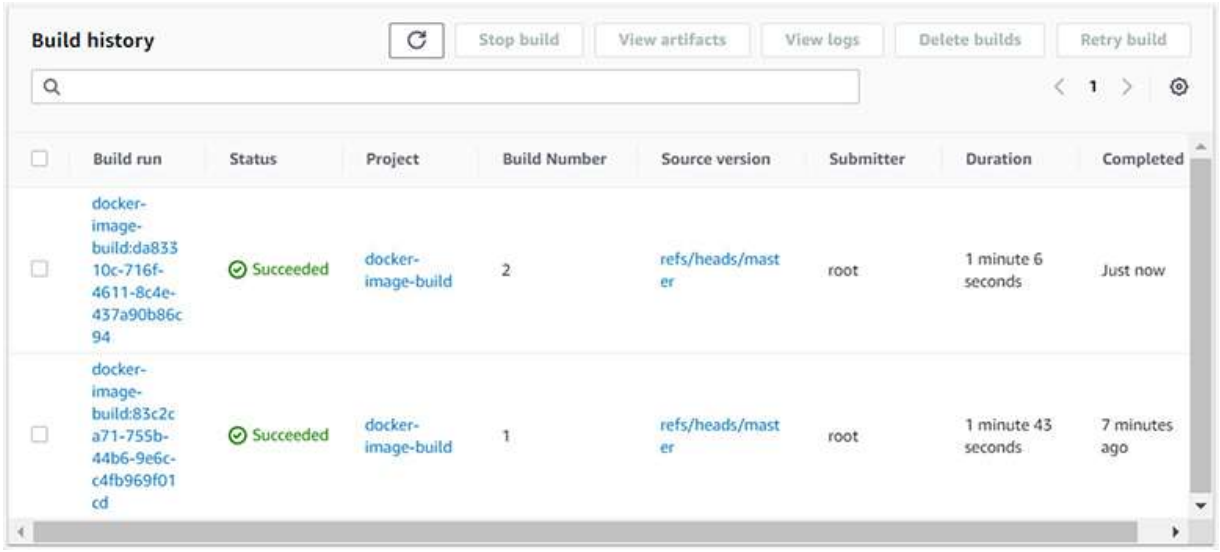
the printf line.

Add the updated buildspec file using the git add command and commit the changes using the git commit -m “comment” command. Then push the code to your remote repository in AWS CodeCommit using the git push command.

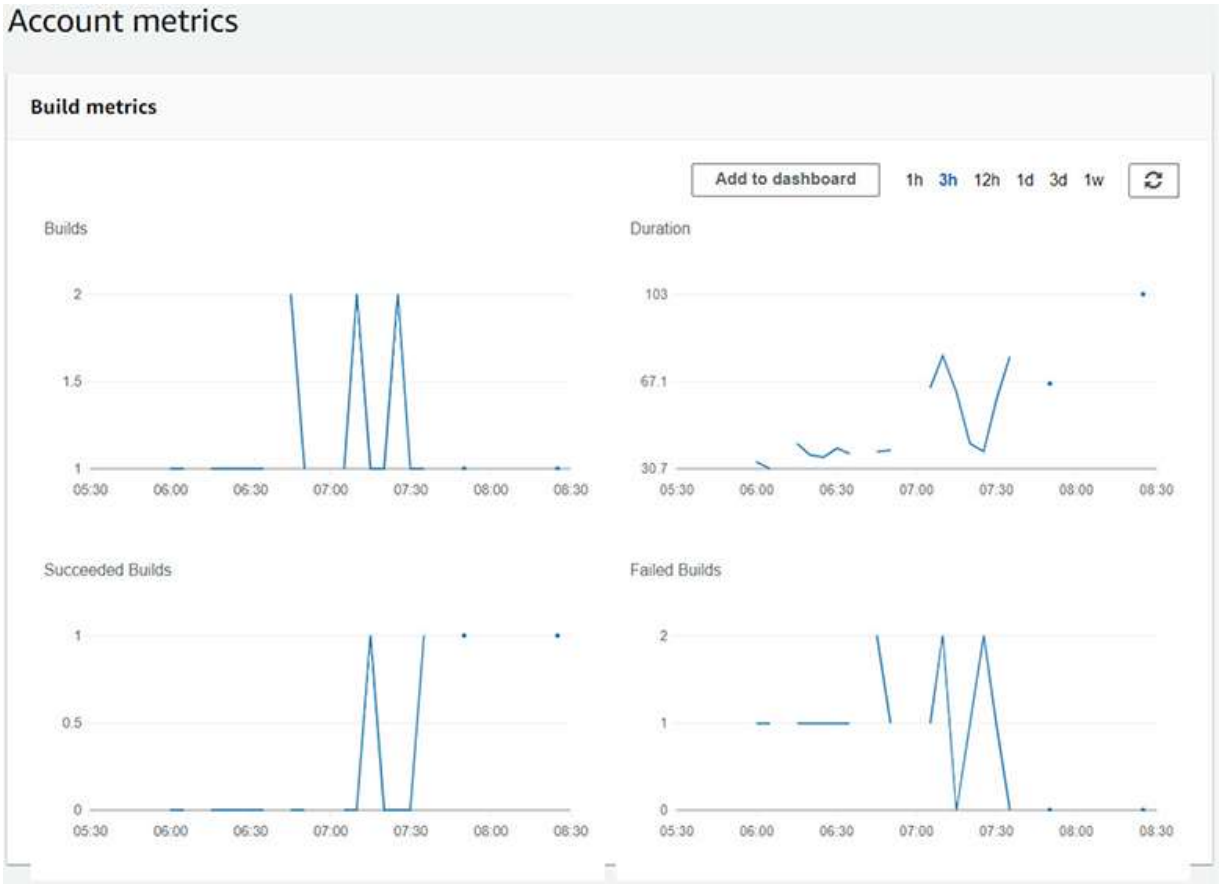
```
MINGW64:/c/Users/user/aws-codebuild-repo
user@DESKTOP-5B87H96 MINGW64 ~/aws-codebuild-repo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   buildspec.yml

no changes added to commit (use "git add" and/or "git commit -a")

user@DESKTOP-5B87H96 MINGW64 ~/aws-codebuild-repo (master)
$ git add .
warning: LF will be replaced by CRLF in buildspec.yml.
The file will have its original line endings in your working directory

user@DESKTOP-5B87H96 MINGW64 ~/aws-codebuild-repo (master)
$ git commit -m "application version 2.0"
[master b583f80] application version 2.0
 1 file changed, 1 insertion(+), 1 deletion(-)

user@DESKTOP-5B87H96 MINGW64 ~/aws-codebuild-repo (master)
$ 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), 306 bytes | 306.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/aws-codebuild-repo
   03bdf0f..b583f80  master -> master

user@DESKTOP-5B87H96 MINGW64 ~/aws-codebuild-repo (master)
$
```

Navigate again to your build project in AWS CodeBuild and click on Start Build. In the next few minutes, the overall build status is “Succeeded” and the build number changes to 2.

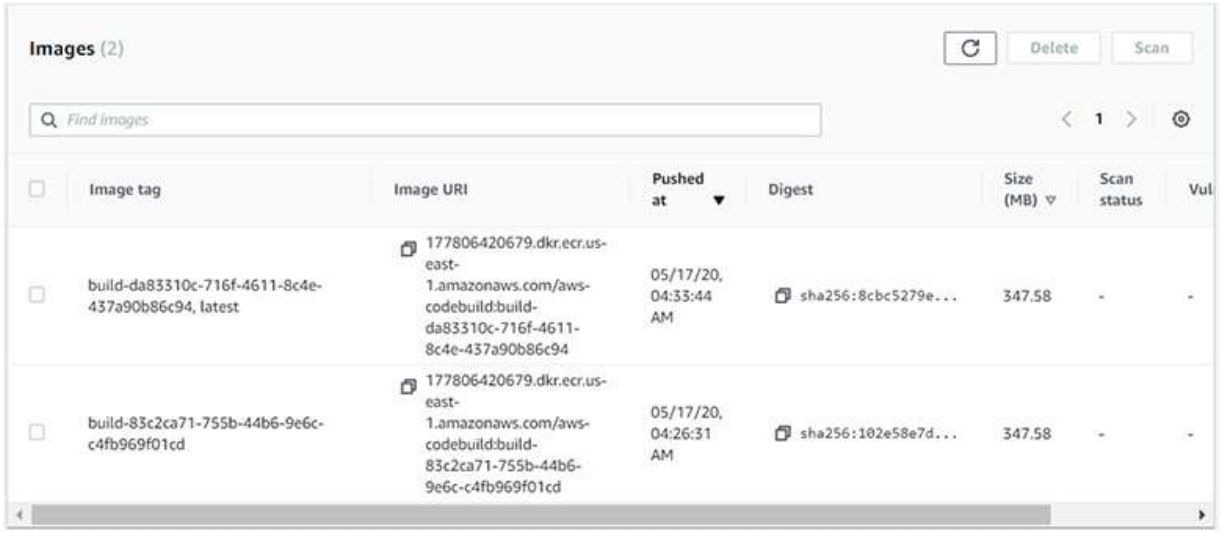
You can see the build history details by choosing Build History from the left pane.



You can see the build metrics details on the Account metrics dashboard by choosing Account Metrics from the left navigation pane.



Finally, you can navigate to your aws-codebuild repository on the Amazon ECR service page. You can see that both the images you built using AWS CodeBuild are successfully stored here.



AWS CodeBuild integrates well with AWS CodeCommit and AWS CodeDeploy in addition to Gradle, Apache Maven, etc. You just need to pay for the build minutes in AWS CodeBuild instead of running and managing a dedicated build server.

Test Reporting

You can create reports for unit tests and functional tests that run during the AWS CodeBuild build, with test result details in JSON, XML, and TRX formats. You need to add a report group name in the buildspec file, and test reports are created when you run the build project. You can either specify a new report group name or choose an existing group name in the buildspec file. A new test report with the new results is created every time a new build is executed. The build test reports help you optimize your build by reviewing the test success and failure trends, and it helps you troubleshoot any problem that you encounter during a build run. You need to export the test results file to Amazon S3 if you want to keep it for more than 30 days because the test reports expire after this period.

Chapter Review

This chapter began by explaining what AWS CodeBuild is. It then provided a step-by-step procedure to create a build specification file for the Docker image and pushed the source code to the AWS CodeCommit remote repository. Then, we created a build project in AWS CodeBuild and built two Docker images successfully, which were stored in Amazon ECR.

Exercises

The following exercises will help you practice using AWS CodeBuild. You need to create an AWS account, as explained earlier, before performing these exercises. You can use the Free Tier when launching AWS resources, but make sure to terminate them at the end.

Exercise 24-1: Delete the AWS CodeBuild Project Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Build Projects.
4. Choose the radio button next to your build project.
5. Choose Delete to delete your build project.

Exercise 24-2: Delete Build Images from Amazon ECR Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS Amazon ECR console at https://console.aws.amazon.com/ecr/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Repositories.

4. From the Repositories page, choose your repository.
5. Choose Images from the left navigation pane.
6. Select all the images that you want to delete.
7. Choose Delete to delete your images.

Exercise 24-3: Delete the AWS CodeCommit Repository Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CodeCommit console at https://console.aws.amazon.com/codecommit/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Repositories.
4. Choose Settings.
5. Navigate to Delete Repository on the General tab.
6. Choose Delete Repository.
7. Enter **delete** in the popup window and choose to delete.

Questions

The following questions will help you gauge your understanding of the contents in this chapter. Read all the answers carefully because there might be more than one correct answer. Choose the best responses for each question.

1. Your company is having issues managing its build server, which is hosted on an Amazon EC2 instance that is used by many teams. Which of the following AWS service provides a managed code build service?

   A. AWS CodeCommit

   B. AWS CodeDeploy

   C. AWS CodePipeline

   D. AWS CodeBuild

2. Your company security policy mandates that all the build artifacts be stored in an Amazon S3 bucket as encrypted using a client encryption key. How can you encrypt all the build artifacts using a client master AWS KMS key?

   A. Supply the customer master AWS KMS key in the Artifact section during build project creation

   B. Upload the customer encryption key during build project creation in AWS CodeBuild

   C. Check the enryption checkbox in AWS CodeCommit

   D. Supply the AWS KMS key during the build process

3. Your team is using AWS CodeBuild to compile source code, and when they try to run a test with their Amazon RDS hosted on their private subnet, they are having an access issue. How can you resolve this?

   A. Update the private subnet route table to allow AWS CodeBuild

   B. Update the private subnet security group to allow AWS CodeBuild

   C. Add an Internet Gateway to the VPC and access it through the Internet

   D. When creating a build project, they need to select appropriate VPC and private subnet in the Additional Configuration section.

4. What two environment image options are available during build project creation in AWS CodeBuild? (Choose two.)

   A. Managed image

   B. Docker image

   C. Custom image

   D. Windows image

5. What three operating systems are available when you choose managed image during build project creation in AWS CodeBuild? (Choose three.)

   A. Amazon Linux 2

   B. Ubuntu

   C. Windows Server

D. CentOS

6. A DevOps engineer in your company created a build project in AWS CodeBuild to build Docker images and store them in Amazon ECR. The build is failing in the Docker build phase. How can you resolve this issue?

   A. The Privileged checkbox should be selected while creating build projects to provide the elevated privilege to run Docker commands

   B. Create an AWS IAM user and give it read-only access to the Amazon ECR service

   C. Create an AWS IAM user and give it full access to Amazon EC2

   D. Create an AWS IAM user and give it full access to the Amazon S3 service

7. Your team is running a build project, but they are getting a timeout error after 1 hour. They asked you to resolve the issue and requested that their build project not timeout even after 7 hours. How can you achieve this?

   A. Configure the timeout to 7 hours—the default is 1 hour

   B. Remove the timeout from the build project

   C. Update the timeout to 0—the default is 12 hours

   D. Create a support ticket in AWS

8. Your company audit team mandates all the build logs be stored in redundant places. What you can enable to store logs in multiple places during build project creation?

   A. Choose Amazon CloudWatch and Amazon RDS as your log destination

   B. Choose Amazon S3 and Amazon RDS as your log destination

   C. Choose Amazon S3 and AWS CodeCommit as your log destination

   D. Choose Amazon CloudWatch and Amazon S3 as your log destination

9. Your security engineer found that AWS CodeBuild runs all your build commands as the root user. What is the reason for this?

   A. AWS CodeBuild uses the root user to run all build commands by default

B. You need to create new a AWS IAM user and grant it full AWS CodeBuild access

C. Delete the root user from the AWS Management Console

D. Use a different AWS account that does not have a root user

10. What pricing model is used by AWS CodeBuild?

A. It is free to use for your entire workload

B. Pay up-front pricing model

C. Pay hourly pricing model

D. Pay-as-you-go pricing model

Answers

1. **D.** AWS CodeBuild is the managed build service from AWS.
2. **A.** You can supply the customer master AWS KMS key in the Artifact section during build project creation.
3. **D.** When creating a build project, they need to select the appropriate VPC and private subnet in the Additional Configuration section.
4. **A, C.** AWS CodeBuild offers managed images and custom images.
5. **A, B, C.** Amazon Linux 2, Ubuntu, and Windows Server operating systems are available in AWS CodeBuild.
6. **A.** The Privileged checkbox should be selected when creating build projects to provide the elevated privilege to run Docker commands.
7. **A.** You need to add the timeout to 7 hours. The timeout can be set from 5 minutes to 8 hours (the default is 1 hour).
8. **D.** Choose Amazon CloudWatch and Amazon S3 as your log destination.
9. **A.** AWS CodeBuild uses the root user to run all build commands by default.
10. **D.** AWS CodeBuild uses a pay-as-you-go pricing model.

Additional Resources

- **AWS CodeBuild** AWS CodeBuild official documentation is the place to get all the latest information about this service in addition to all the updates to its features and new features.
  `https://docs.aws.amazon.com/codebuild/index.html`
- **AWS CodeBuild Blogs** These blogs help you with step-by-step details of using AWS CodeBuild for different use cases and security best practices.
  `https://aws.amazon.com/blogs/devops/category/developer-tools/aws-codebuild/`

CHAPTER 25

Deploying Applications Using CodeDeploy and CodePipeline

In this chapter, you will learn

- AWS CodeDeploy
- AWS CodePipeline

This chapter will provide a practical approach to learn AWS CodeDeploy and AWS CodePipeline.

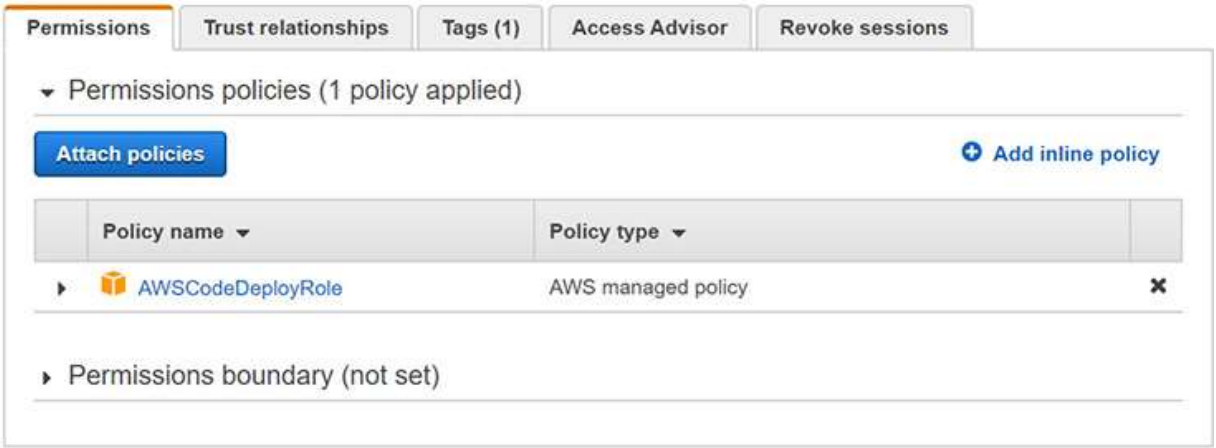
AWS CodeDeploy

As a developer, you might have used automated code deployment tools such as Jenkins, Bamboo, TeamCity, etc., including AWS CodeDeploy. AWS CodeDeploy automates your application deployments to Amazon EC2, Amazon ECS, AWS Lambda, and on-premises instances. You can deploy your application code, serverless functions, packages, executables, and web and configuration files from your git repositories and Amazon S3 buckets.

You need to provide an application name to uniquely identify the deployment and specify the compute platform as EC2/On-premises, AWS Lambda, or Amazon ECS. The CodeDeploy agent is necessary when you use the EC2/On-Premises compute platform, but it is not required for AWS Lambda or Amazon ECS. You also need to specify the deployment configuration, such as canary, linear, or all at once to indicate how the traffic is routed during the deployment.

The next component is the deployment, where you can choose either in-place deployment or blue/green deployment, based on whether your application needs to be available even during the application deployment. With in-place deployment, the application is stopped and the updated application is installed, tested, and started again. In blue/green deployment, the updated application is deployed to the deployment group while the application is online and the load balancer reroutes the traffic to the new deployment group without any downtime to the application. However, it works only with Amazon EC2 instances, not on-premises instances. The in-place deployment type is not available for Amazon ECS and AWS Lambda. AWS CodeDeploy APIs, AWS CLI, AWS SDKs, and AWS Management Console can be used to access and manage AWS CodeDeploy.

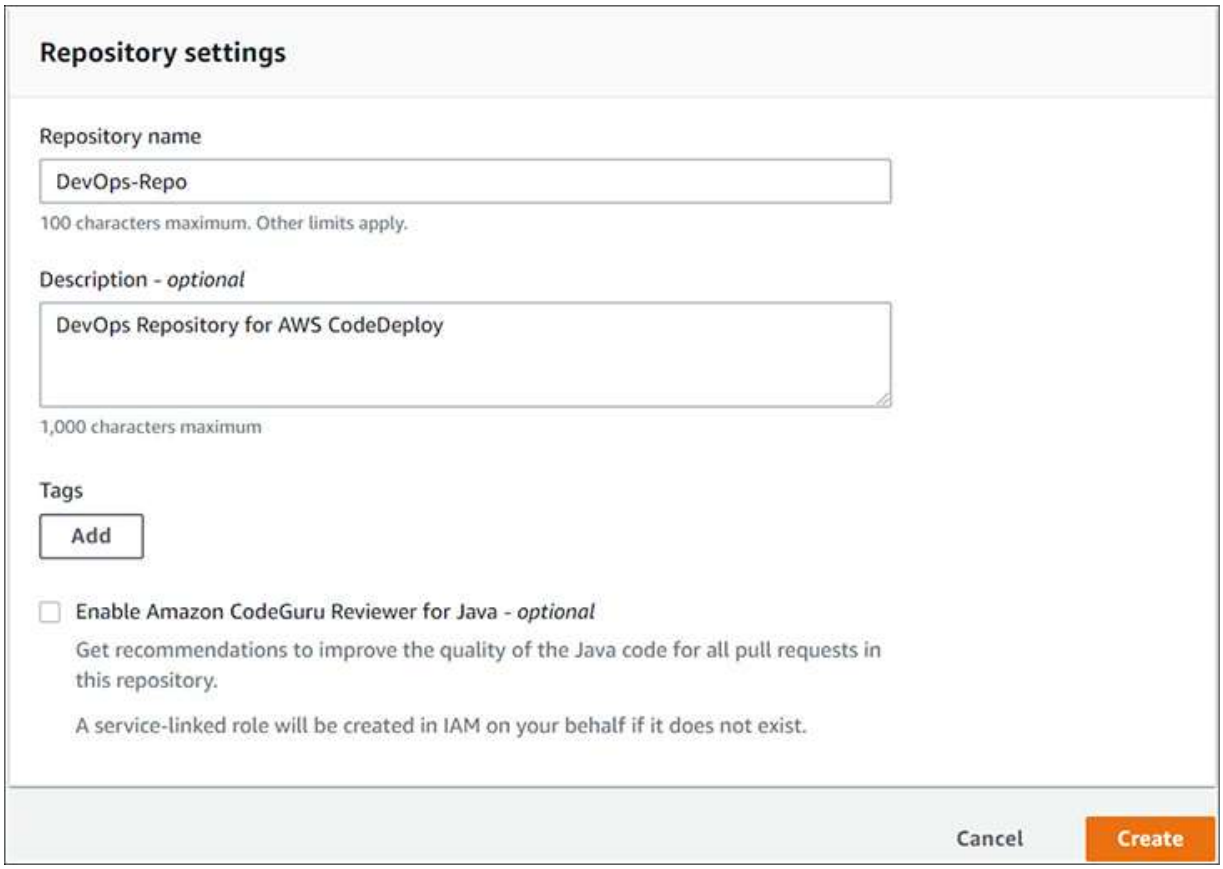
Now let us explore the step-by-step procedure to automatically deploy a sample application using AWS CodeDeploy and add AWS CodePipeline to automatically pull source code changes from the code repository, such as AWS CodeCommit, and deploy it to an Amazon EC2 instance using AWS CodeDeploy. You need to log in to your AWS account and navigate to the AWS Identity And Access Management (IAM) page. But you first need to create a CodeDeployRole and attach the AWS managed policy AWSCodeDeployRole.



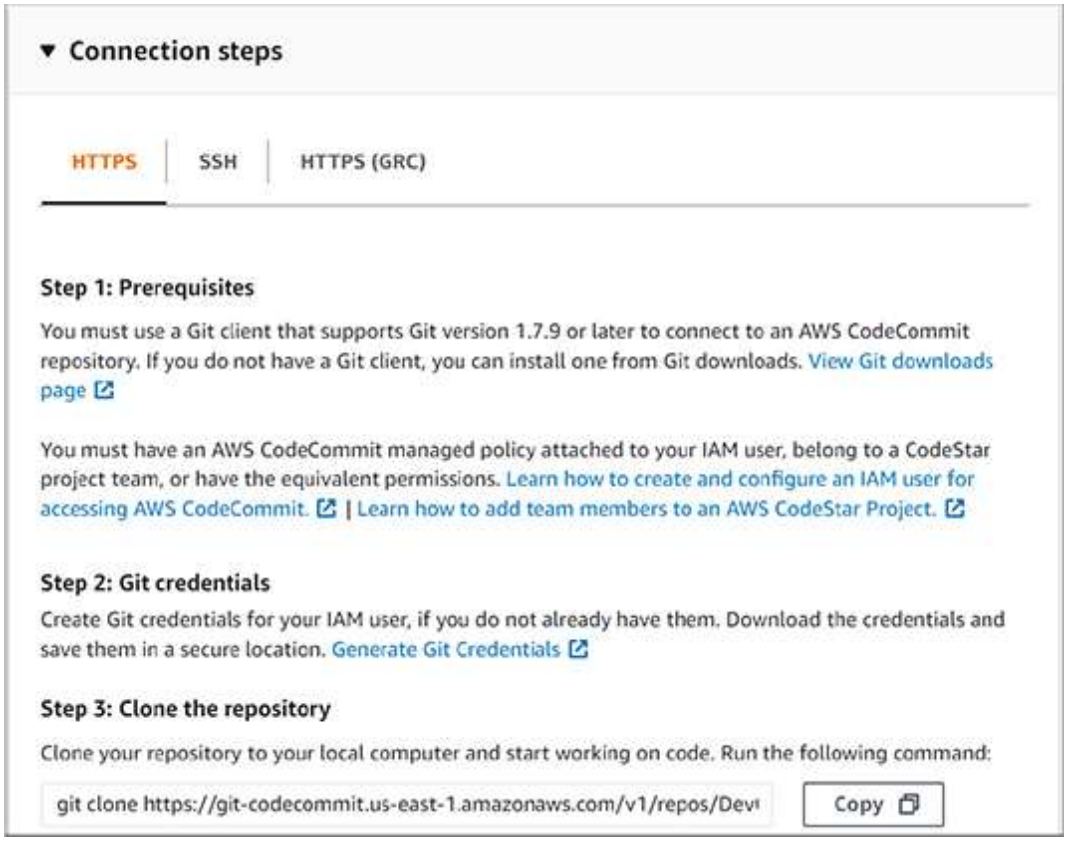
You need to create another service role—EC2-to-S3-Read-only—and attach the AWS managed policy AmazonS3ReadOnlyAccess to enable the Amazon EC2 instance to access Amazon S3 and read the objects.

You then need to navigate to the AWS CodeCommit service page and create a repository—for example, DevOps-Repo—and click the Create

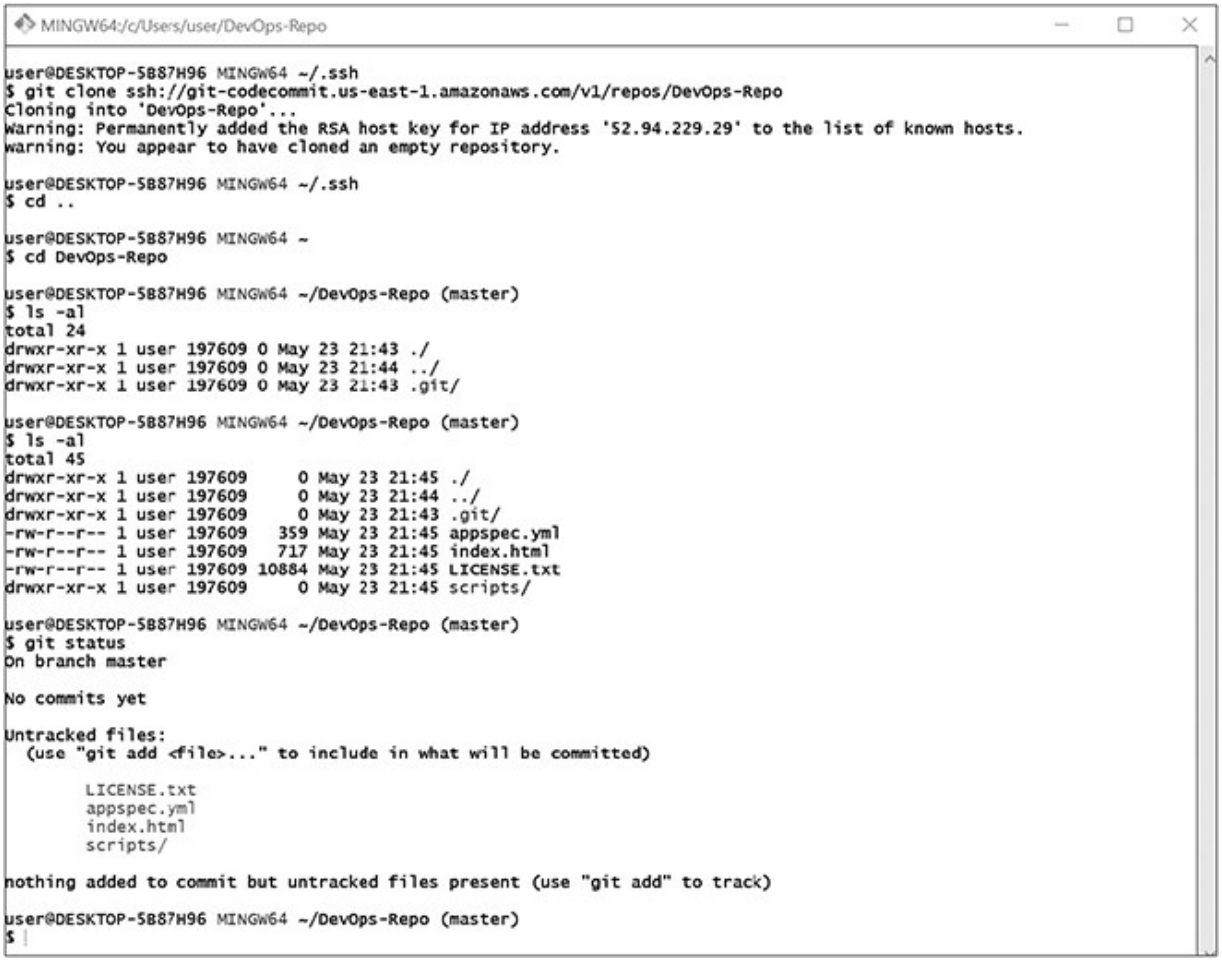
button.



Note either the HTTPS or SSH URL from this page to clone it in your local machine. Click the Copy button.



On your local machine, open git.bash, if it is Windows OS, or Terminal, if it is macOS. Clone the repository and then go to the DevOpsRepo directory. You can use any sample application code or use this Amazon sample application: https://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip, and move the contents of this ZIP file to the DevOps-Repo folder.



Add the new files to the remote repository using the git add command, and commit the changes using the git commit command. Then push the code to the AWS CodeCommit repository using the git push command.

```
MINGW64:/c/Users/user/DevOps-Repo

user@DESKTOP-5B87H96 MINGW64 ~/DevOps-Repo (master)
$ git add -A
warning: LF will be replaced by CRLF in LICENSE.txt.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in appspec.yml.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in index.html.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in scripts/install_dependencies.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in scripts/start_server.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in scripts/stop_server.
The file will have its original line endings in your working directory

user@DESKTOP-5B87H96 MINGW64 ~/DevOps-Repo (master)
$ git commit -m "uploading to remote DevOps Repo"
[master (root-commit) 4903fb9] uploading to remote DevOps Repo
 6 files changed, 266 insertions(+)
 create mode 100644 LICENSE.txt
 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

user@DESKTOP-5B87H96 MINGW64 ~/DevOps-Repo (master)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (9/9), 5.04 KiB | 1.01 MiB/s, done.
Total 9 (delta 0), reused 0 (delta 0)
To ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/DevOps-Repo
 * [new branch]      master -> master

user@DESKTOP-5B87H96 MINGW64 ~/DevOps-Repo (master)
$
```

You can verify the source code changes by navigating to the AWS CodeCommit page and choosing Code Repository.

You need an Amazon EC2 instance to deploy the code. You can use your existing Amazon EC2 instance by just adding the tag Name with value DevOps or create a new instance by clicking the Launch Instance button.

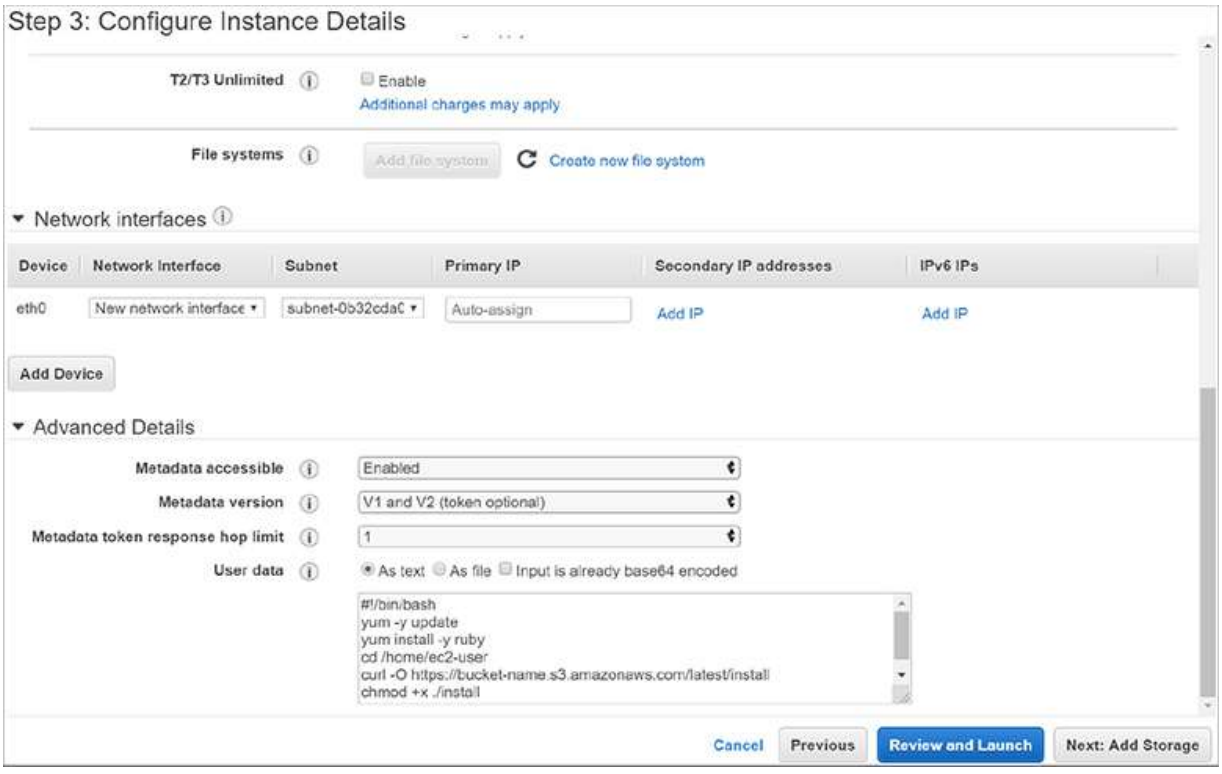
Choose the latest Amazon Linux 2 AMI from the list and click Next.

We can use the Free Tier–eligible instance type for this example. Select t2.micro and click the Next: Configure Instance Details button.

Choose either your default VPC or custom VPC that you have in your account, and choose a public subnet, since we are going to deploy our sample web application. You need to attach the EC2-to-S3-Read-Only service role.

Scroll down and click on the Advanced Details section. Enter the following code in the user data to install the AWS CodeDeploy agent and click Next:

```
#!/bin/bash
yum -y update
yum install -y ruby
cd /home/ec2-user
curl -O https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/
install
chmod +x ./install
./install auto
```

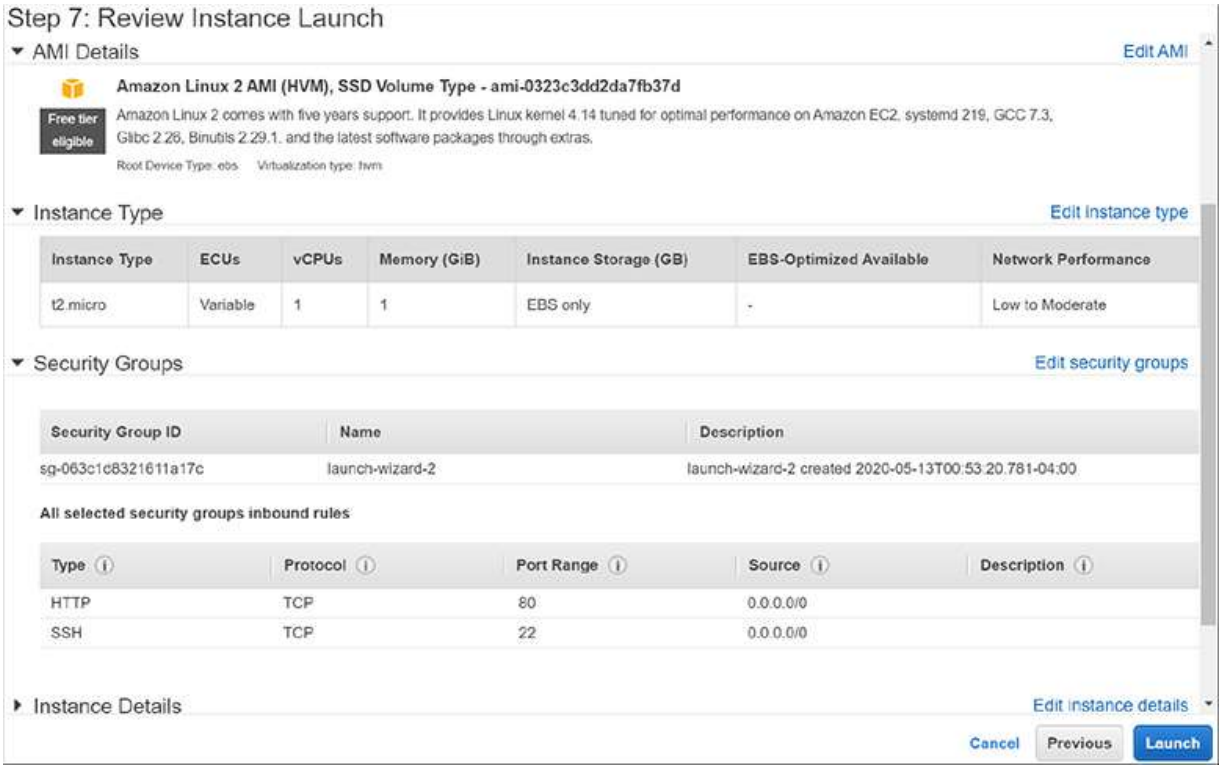


You can add more storage if needed, but make sure Delete On Termination is checked to ensure this storage volume will be deleted when you terminate the instance.

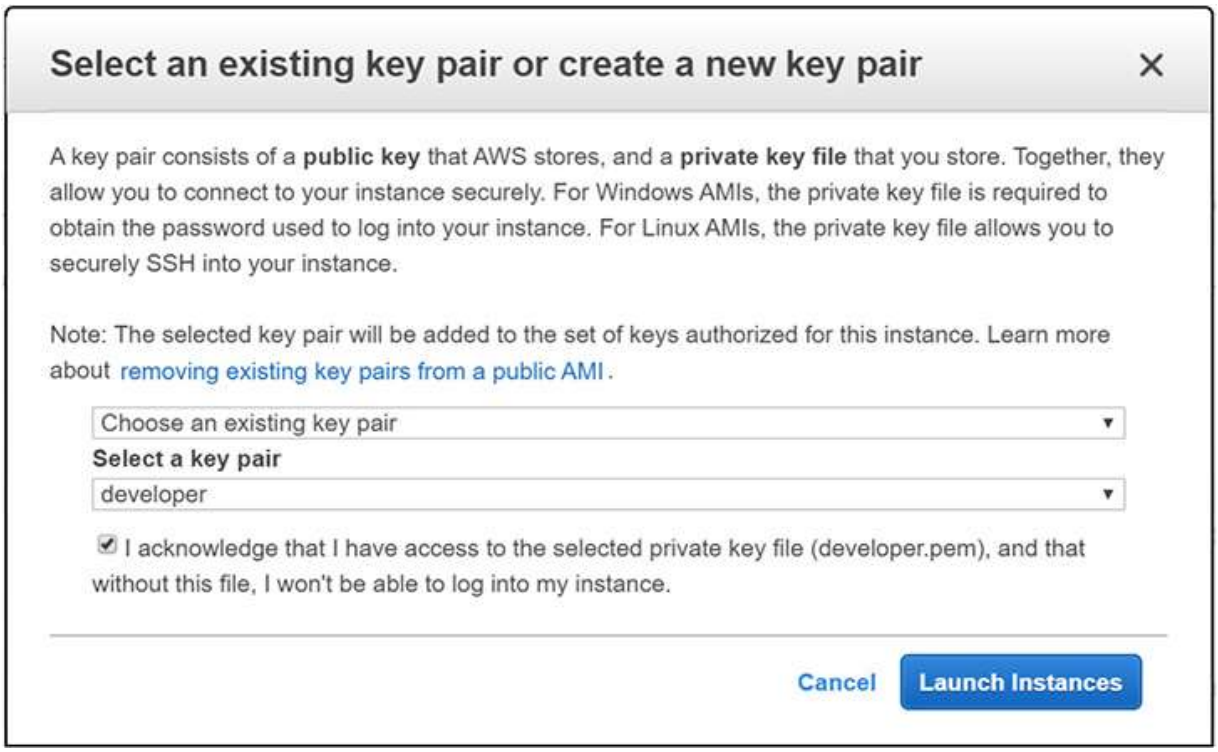
As per the best practice, you need to always add appropriate tags to each AWS resource. Here you must add a tag for the AWS CodeDeploy to identify your instance. Add a tag name with the value **DevOps**.

You can create a new security group or select an existing security group. It is not advisable to open the traffic to 0.0.0.0/0, which is open to the public in your enterprise environment. You do need to open HTTP and SSH in case you need to log in to the instance later.

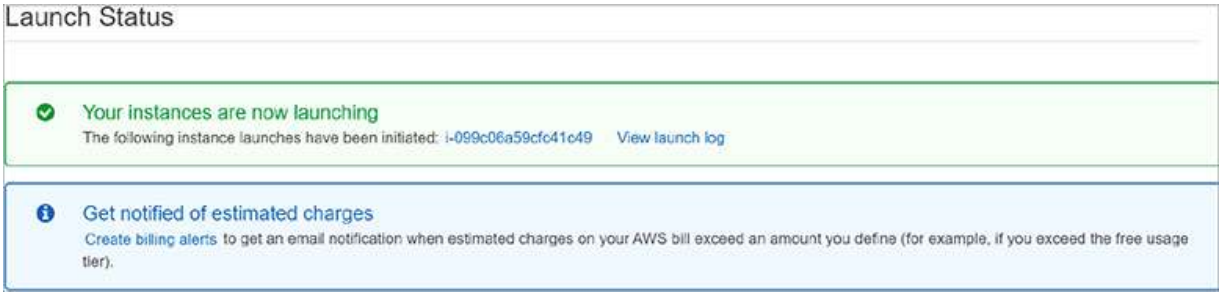
Review all the options and click the Launch button.



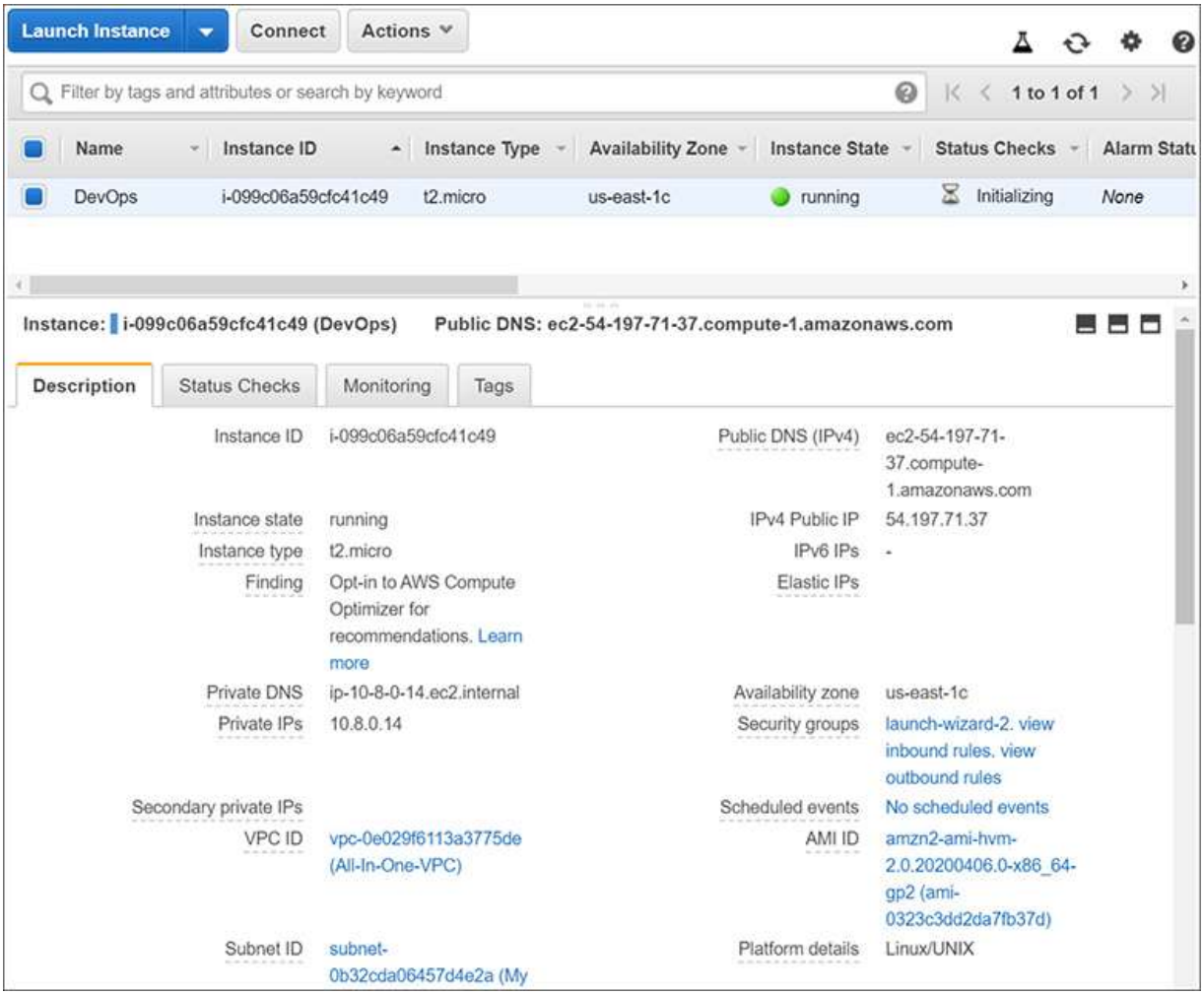
You can either use your existing key pair or create a new key pair, which will be used to log in to the instance later.



If everything is successful, you will see a message such as "your instances are now launching." Click the View Instances button to verify the status.

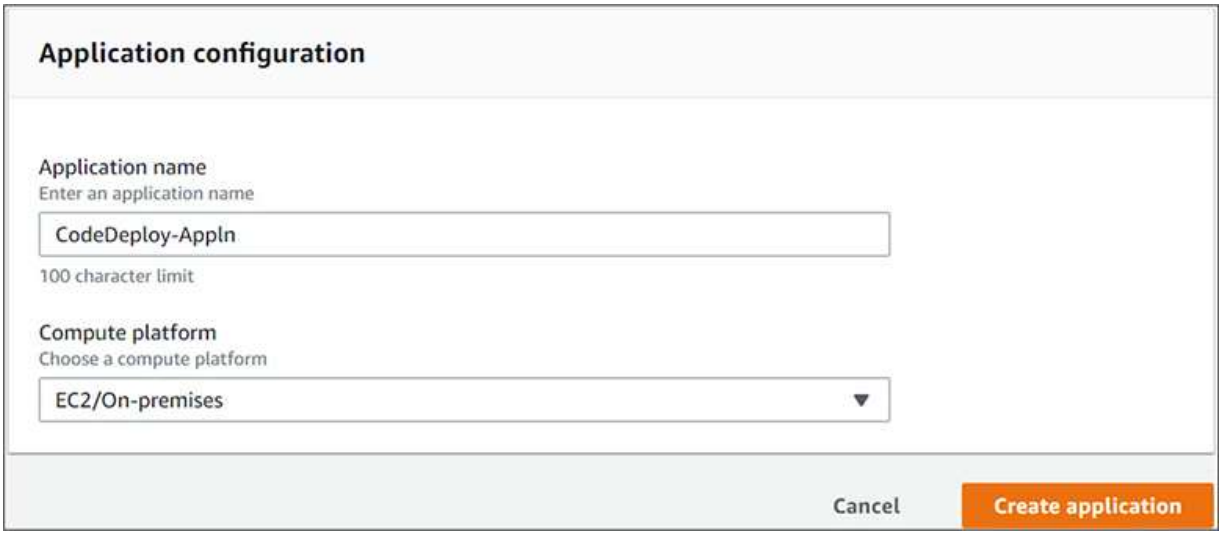


Note down your public IP and verify the status checks are okay.

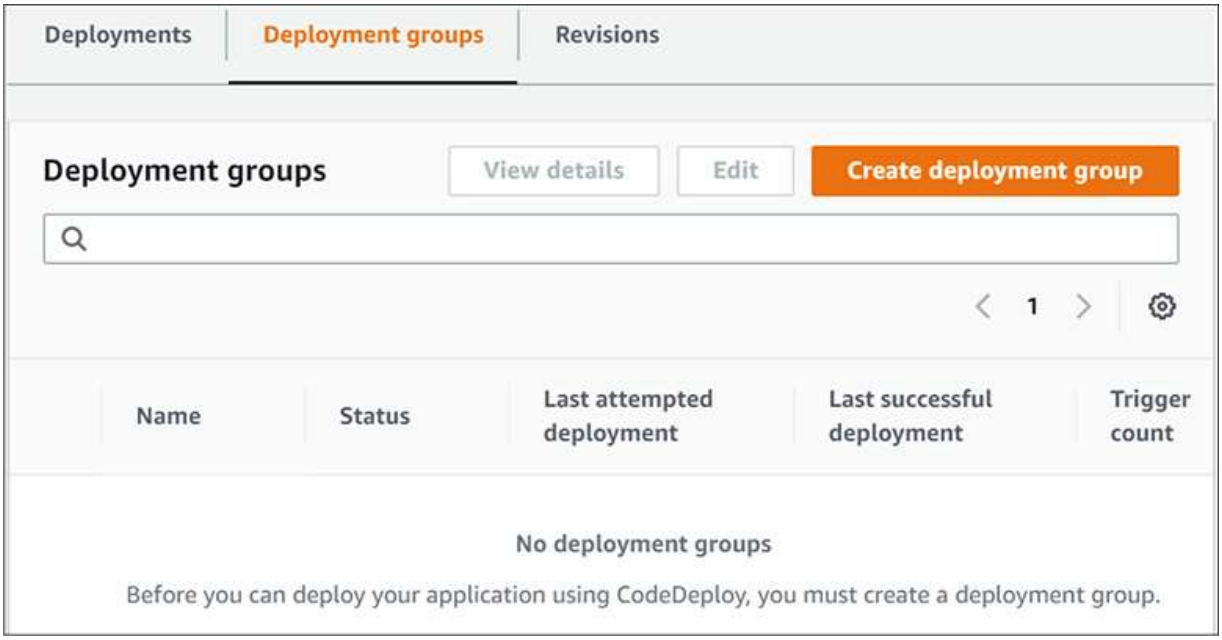


All the prerequisites are completed to get started with AWS CodeDeploy. Navigate to the AWS CodeDeploy service in the AWS Management Console and click the Create Application button.

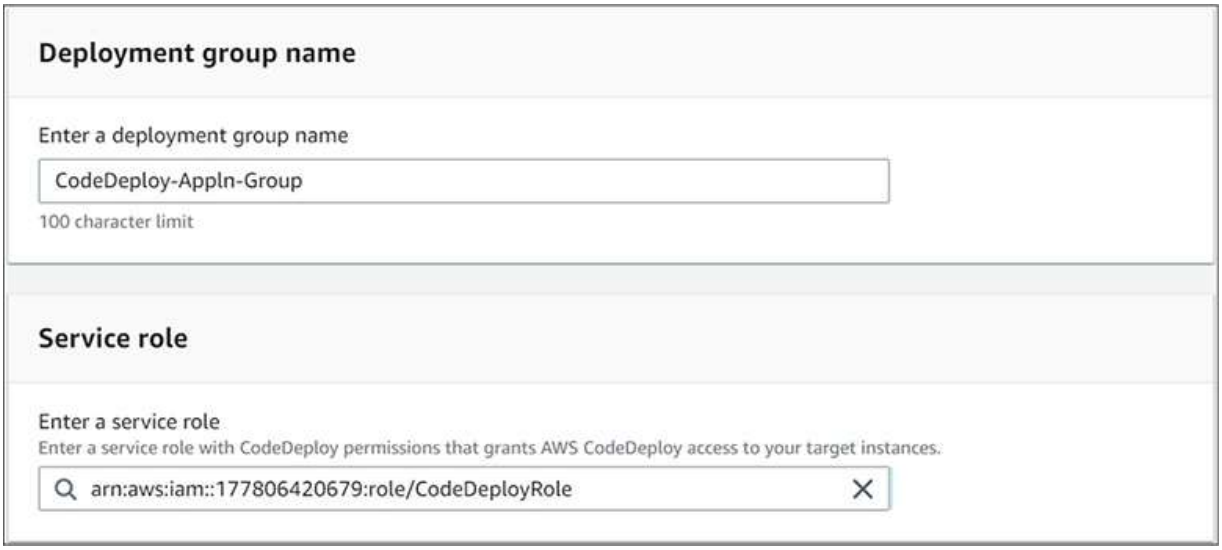
Enter the application name as **CodeDeply-Appln** and select the compute platform as EC2/On-premises from the dropdown.



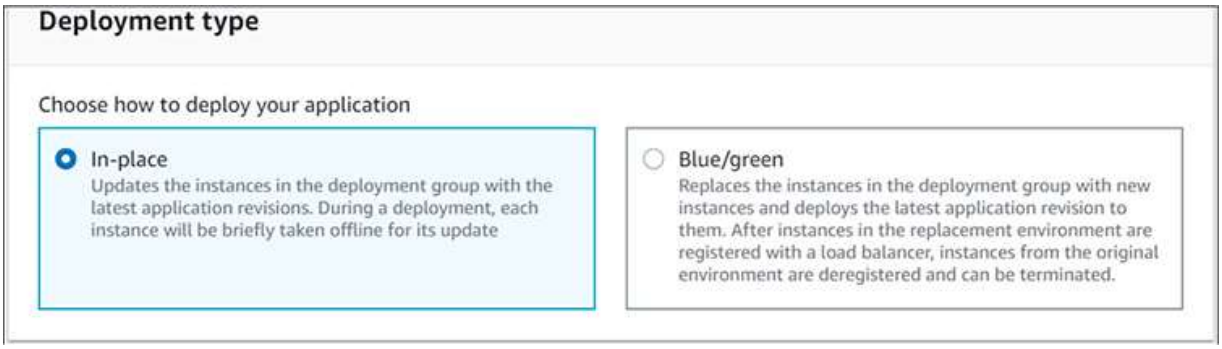
Next click the Create Deployment Group button.



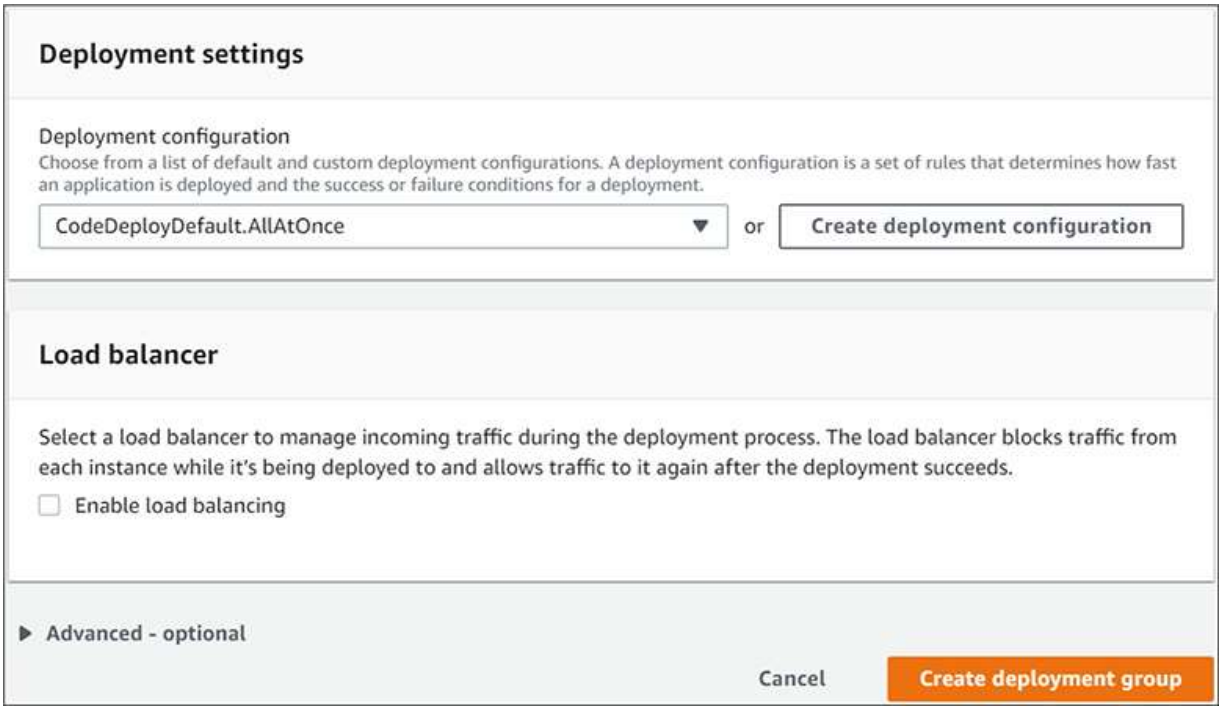
Enter the deployment group name as **CodeDeploy-Appln-group** and select the CodeDeployRole that you created initially.



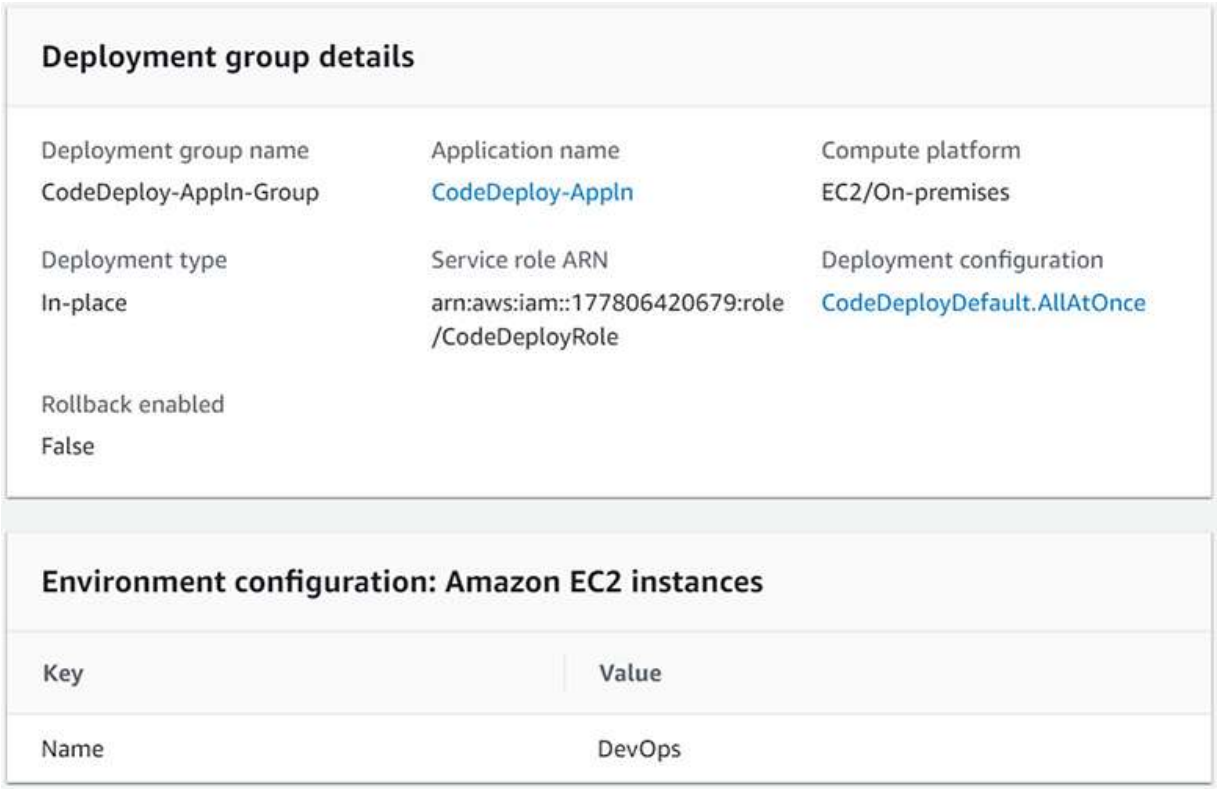
Choose the in-place deployment type and select Amazon EC2 instances for the environment configuration. Enter the tag that you specified on the instance, and note that it identified the instance with a message such as "1 unique matched instance."



The deployment settings can be all at once, which is default, but you can also choose half at a time and one at a time based on your requirements. Also, you can place the deployment behind a load balancer if you choose. For simplicity, choose AllAtOnce for Deployment Settings, uncheck Enable Load Balancing, and click Create Deployment Group.



Verify the deployment group details, such as application name, deployment type, compute platform, service role, and deployment configuration.



AWS CodePipeline

Many tools are available for creating continuous integration/continuous delivery (CI/CD) pipelines, such as Jenkins, CircleCI, Spinnaker, Buddy, and Bamboo, to name a few. AWS CodePipeline provides easy integration with other AWS developer tools like AWS CodeCommit, AWS CodeBuild, and AWS CodeDeploy. You can configure, visualize, and automate different stages of your software release process, such as building, testing, and deployment, using AWS CodePipeline. You can use the AWS Management Console, AWS SDK, and AWS CLI to configure and manage your CI/CD pipeline in AWS CodePipeline. AWS CodePipeline provides a consistent set of release processes, which helps to speed up the delivery of new features to your application, resolve bug fixes with improved quality, and avoid manual errors.

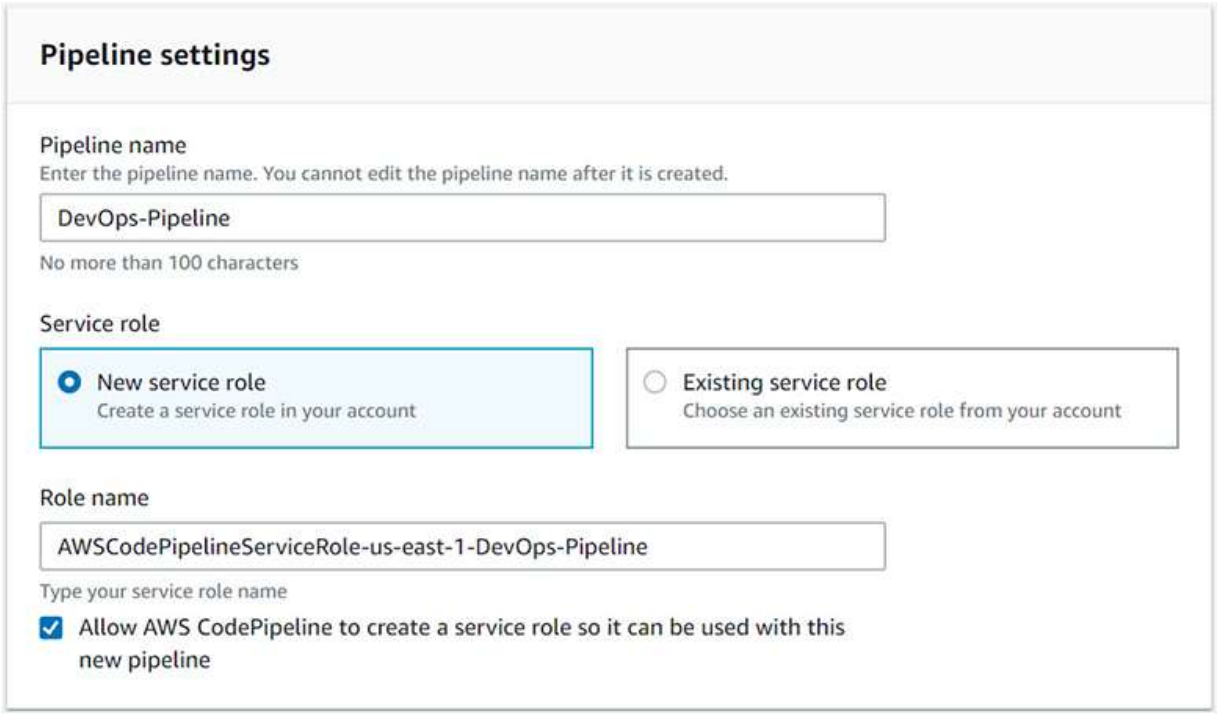
You can use your existing source code repository or AWS CodeCommit to push source code changes, which will be picked up automatically by the build process using tools such as AWS CodeBuild. The release cycle then

moves to the deploy and testing stage by using tools such as AWS CodeDeploy.

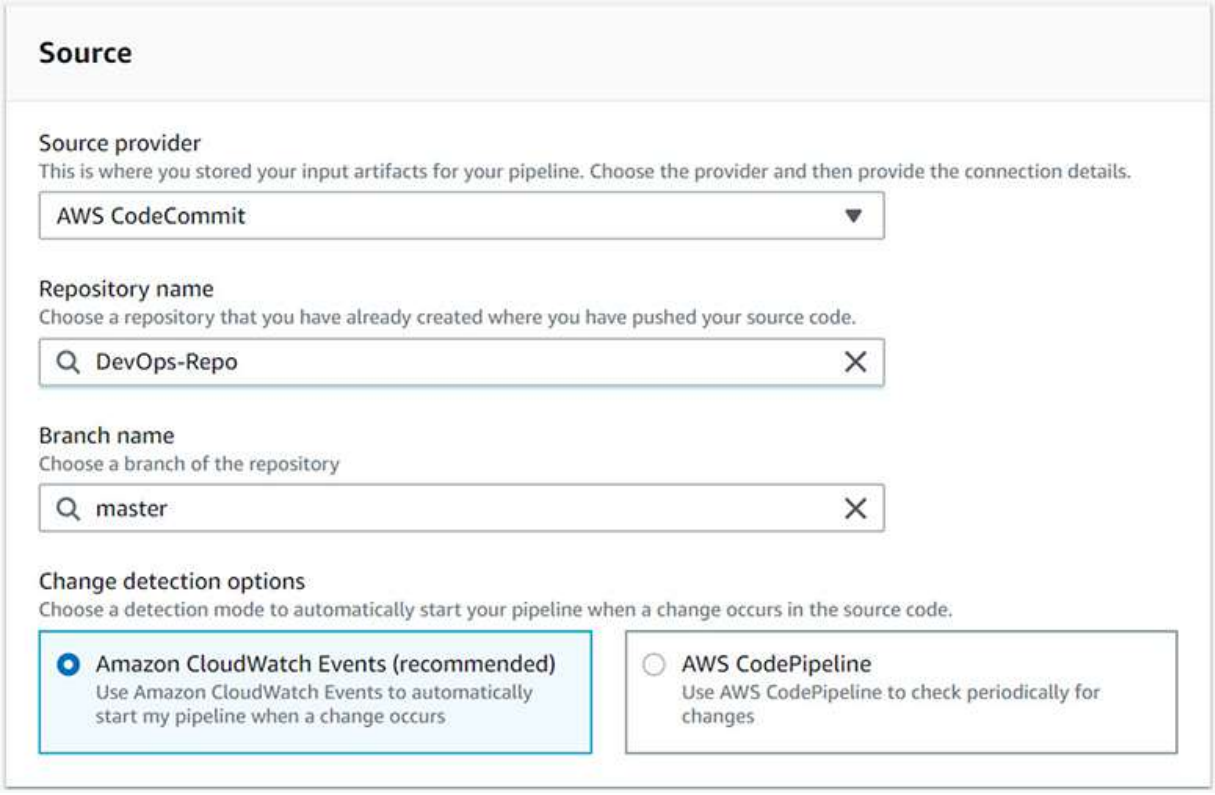
A stage is a logical unit, such as a build stage or deployment stage, which contains the actions that are performed on an application artifact, such as source code. An action is a group of operations that can be performed on your source code change or when deploying an application to instances. A transition is the point where the execution moves from one stage to another, like moving from commit to build to deploy. A pipeline can be defined as a workflow that specifies how the different stages progress in the software release process, and it tracks each execution. A pipeline can be triggered either manually or automatically when the source code is changed by your developer, or you can schedule pipeline execution using Amazon CloudWatch events. You can stop pipeline execution in two ways: stop and wait or stop and abandon. When you use stop and wait, the actions in progress will be completed and all the remaining actions will not be started. When you use stop and abandon, all the actions in progress, including any future actions, will be stopped. You can always use the retry option to start the pipeline execution again. The best practice is to group all the related actions for each application environment, such as development, QA, or production. You can create notification rules to alert you when any important changes, like pipeline execution, are completed.

Navigate to the AWS CodePipeline service in the AWS Management Console and click on Create Pipeline.

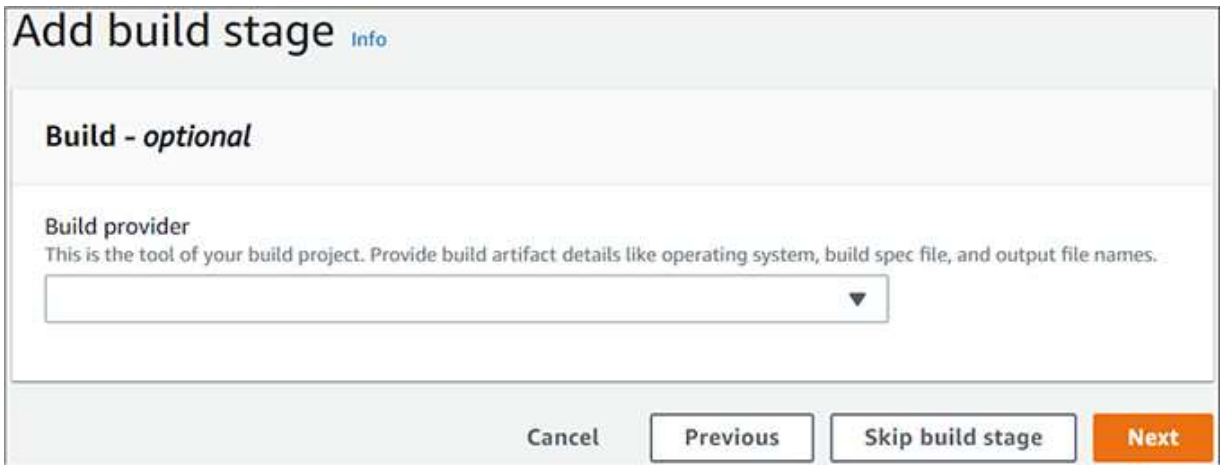
Enter a name for the pipeline—for example, **DevOps-Pipeline**—and choose a service role, if it already exists. If it does not, choose to create a new service role and check Allow AWS CodePipeline To Create A Service Role So It Can Be Used With This New Pipeline.



Now, add the source stage by choosing AWS CodeCommit from the dropdown and choose the repository name, DevOps-Repo, with its branch name. Choose Amazon CloudWatch Events to detect changes and click Next.

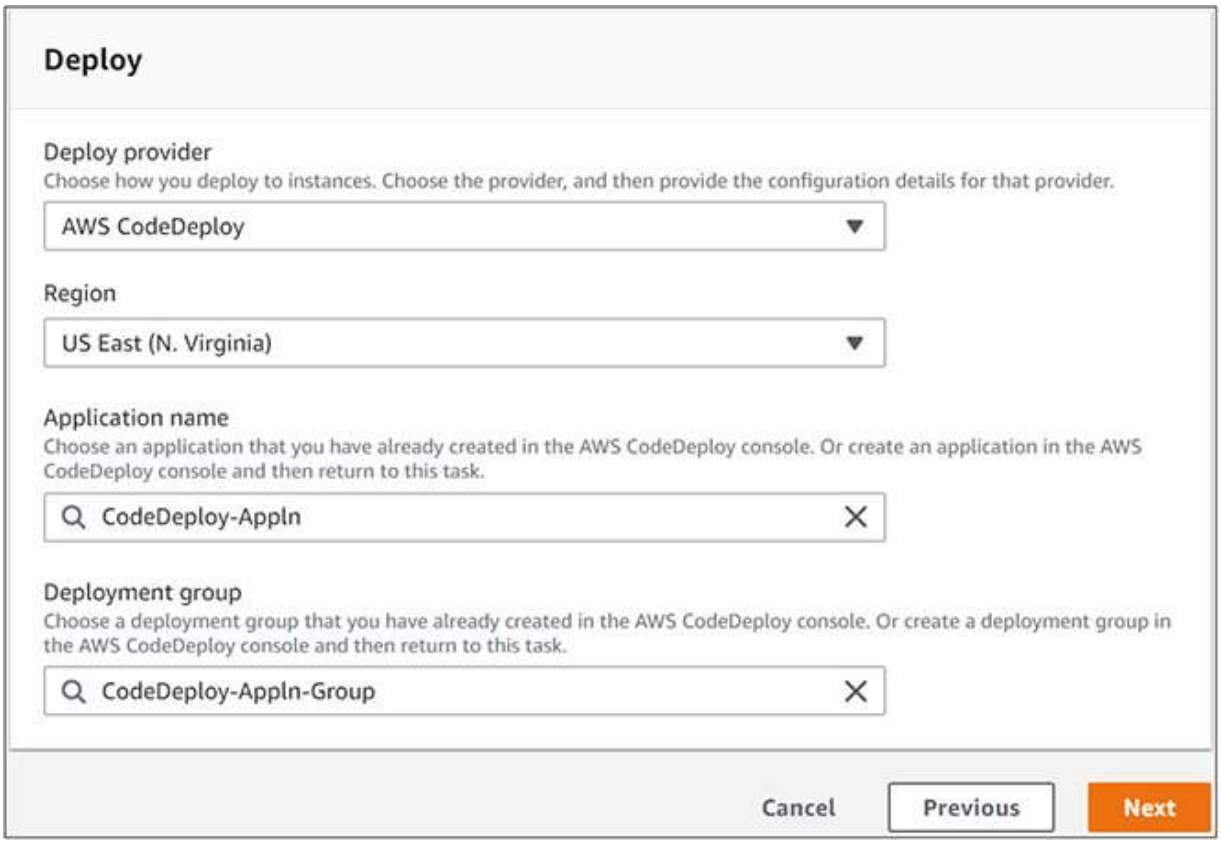


In this exercise you are not going to build using the pipeline, since you created the EC2 instance already. So, click the Skip Build Stage button.



In the popup window, click Skip.

Add the deploy stage to your pipeline by selecting AWS CodeDeploy from the dropdown Deploy Provider menu. Select your region, select your AWS CodeDeploy application name as CodeDeploy-Appln and your deployment group as CodeDeploy-Appln-Group, and click Next.



Review all the options that you selected and click Create Pipeline.

Review Info

Step 1: Choose pipeline settings

Pipeline settings

Pipeline name

DevOps-Pipeline

Artifact location

A new Amazon S3 bucket will be created as the default artifact store for your pipeline

Service role name

AWSCodePipelineServiceRole-us-east-1-DevOps-Pipeline

Step 2: Add source stage

Source action provider

Source action provider

AWS CodeCommit

RepositoryName

DevOps-Repo

BranchName

master

PollForSourceChanges

false

Step 3: Add build stage

Build action provider

Build stage

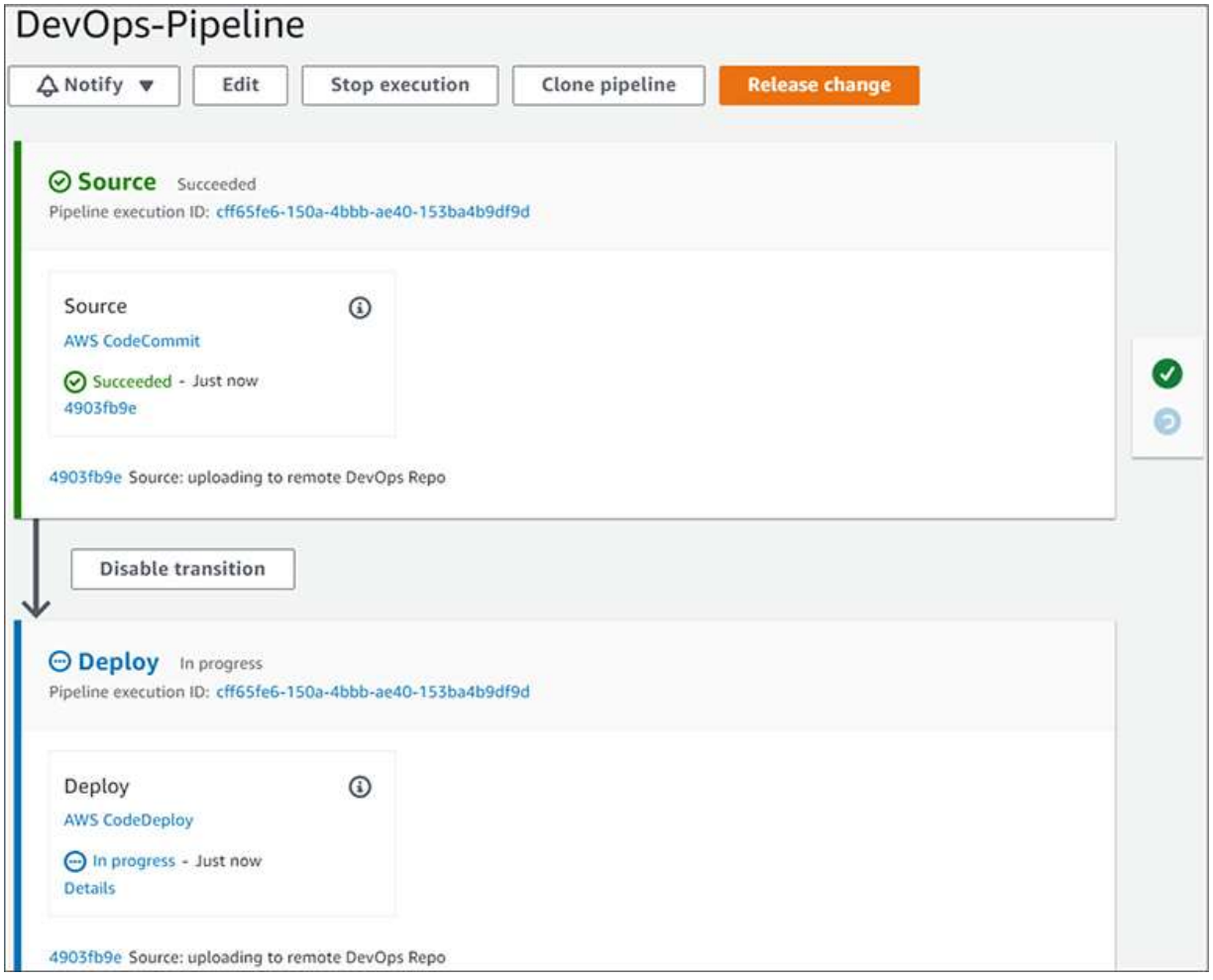
No build

Step 4: Add deploy stage

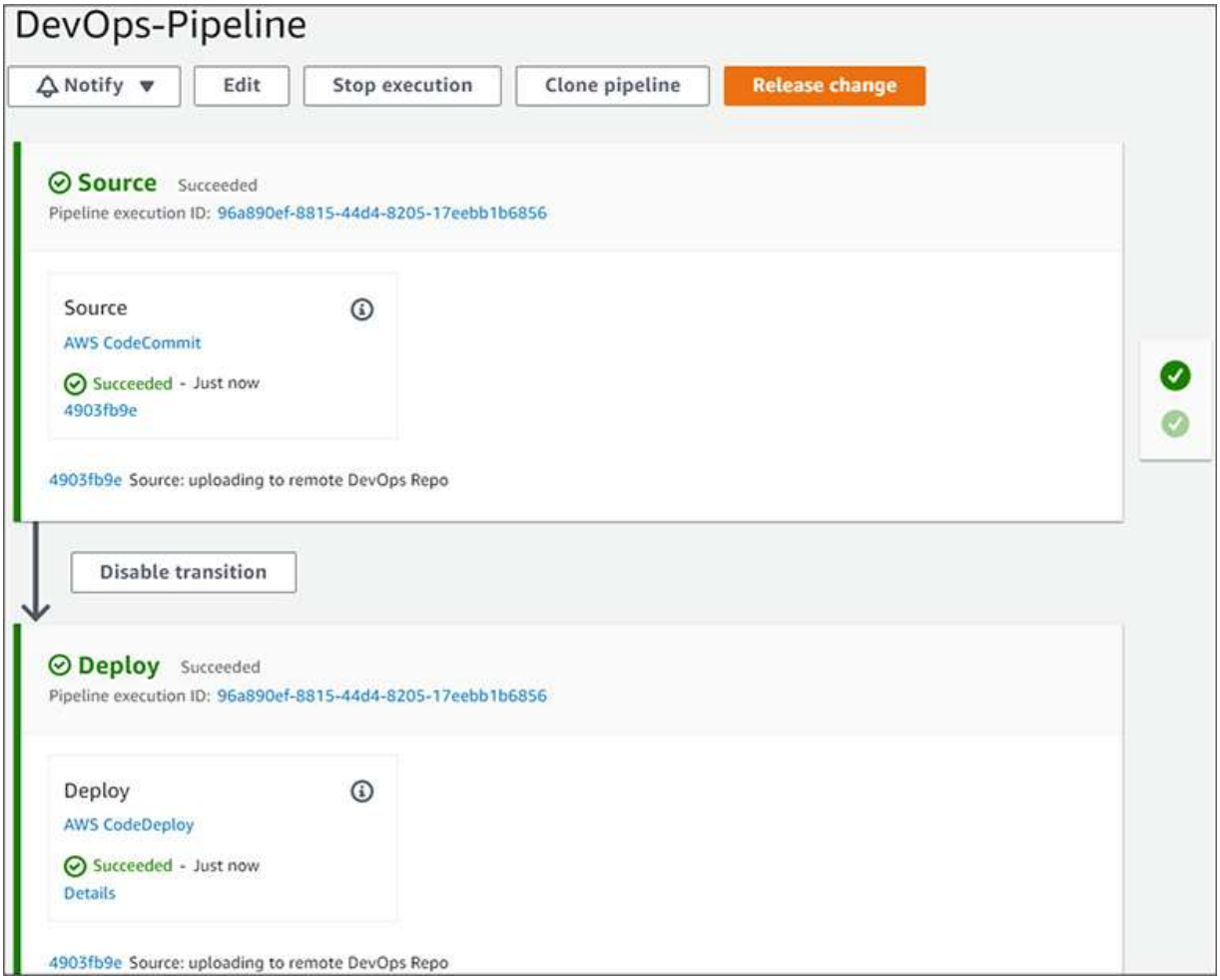
Deploy action provider

Deploy action provider

The application source code is automatically pulled from AWS CodeCommit and the source stage is completed successfully. The deploy stage is in progress.



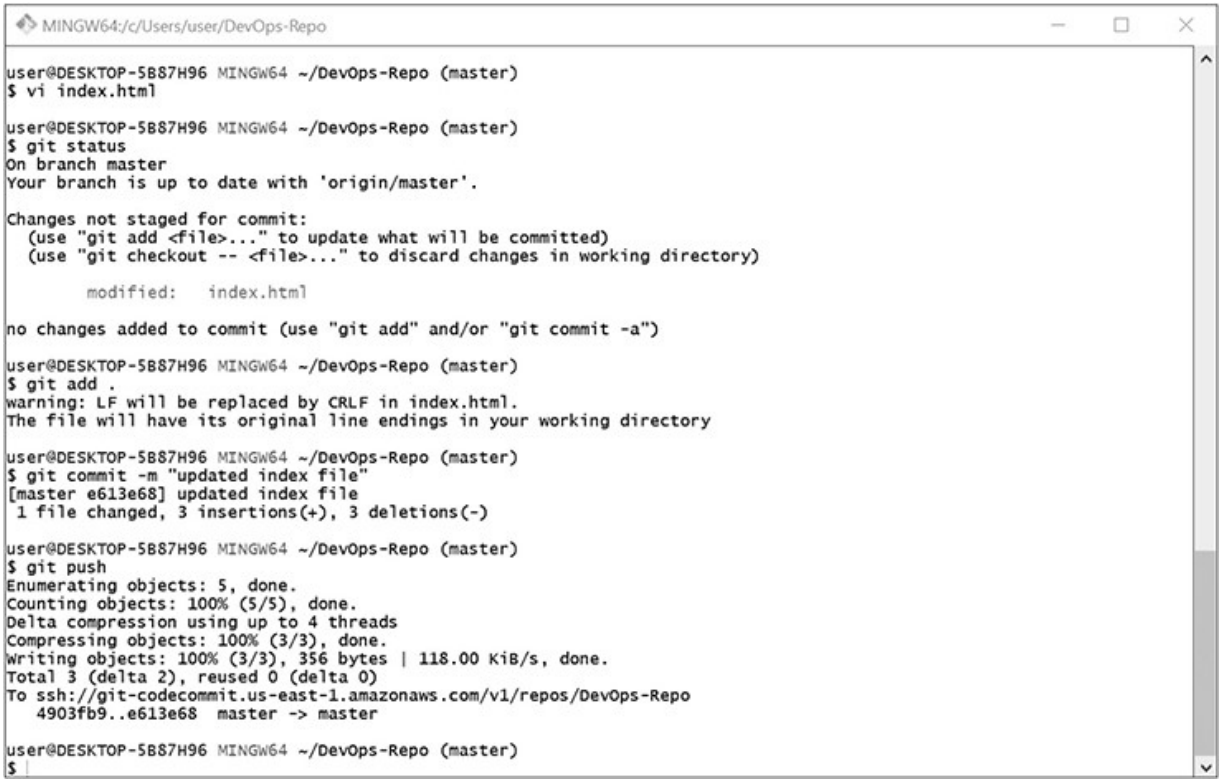
Within a minute, the deploy stage is also successfully completed.



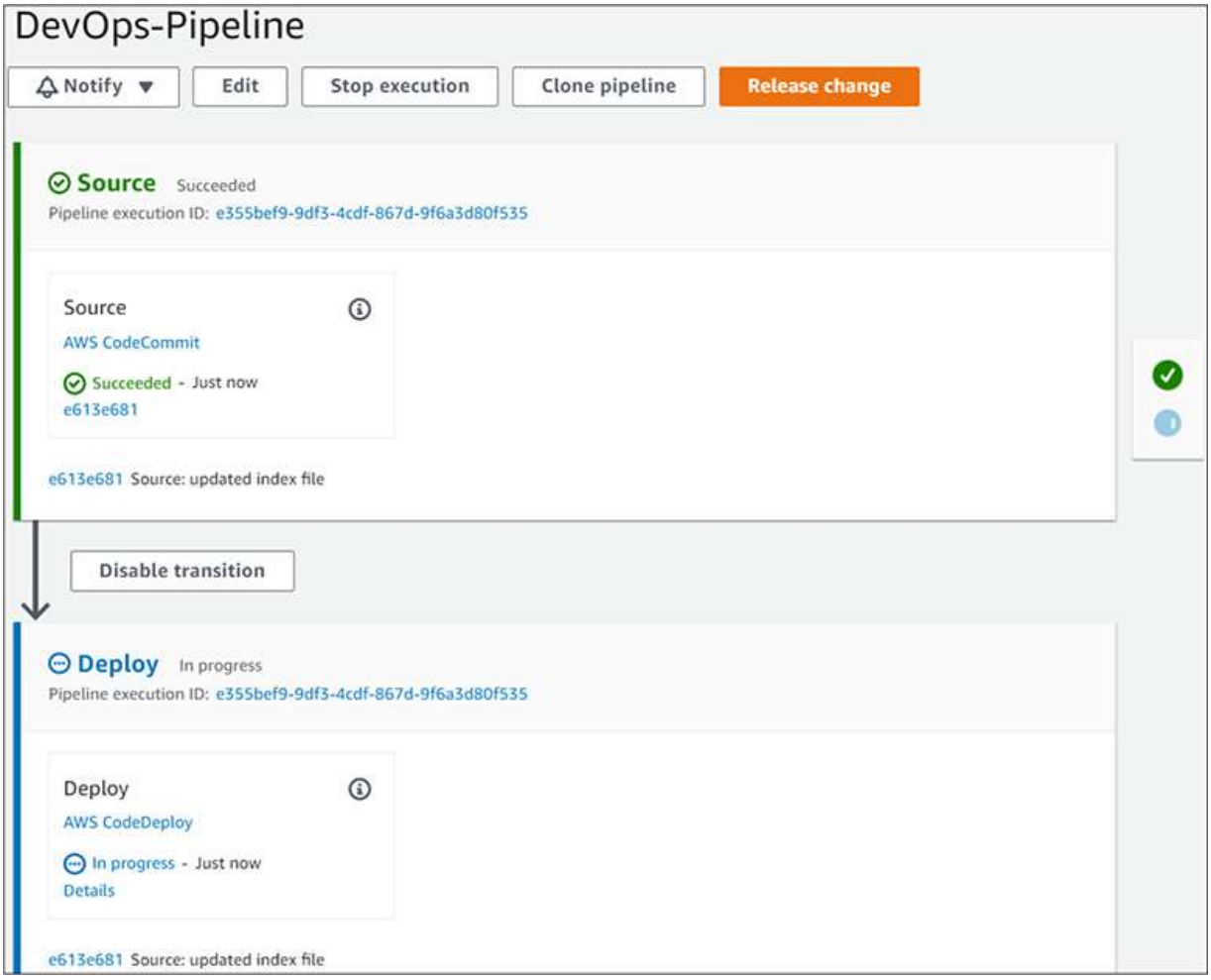
Open the browser of your choice and enter the public IP of your EC2 instance. When you press the ENTER key, you can see the "Congratulations" message.

Now, go to your local machine and edit the index.html file to change the header-1 to **Hello World!** and header-2 to **This application was deployed using AWS CodeDeploy and AWS Pipeline.**

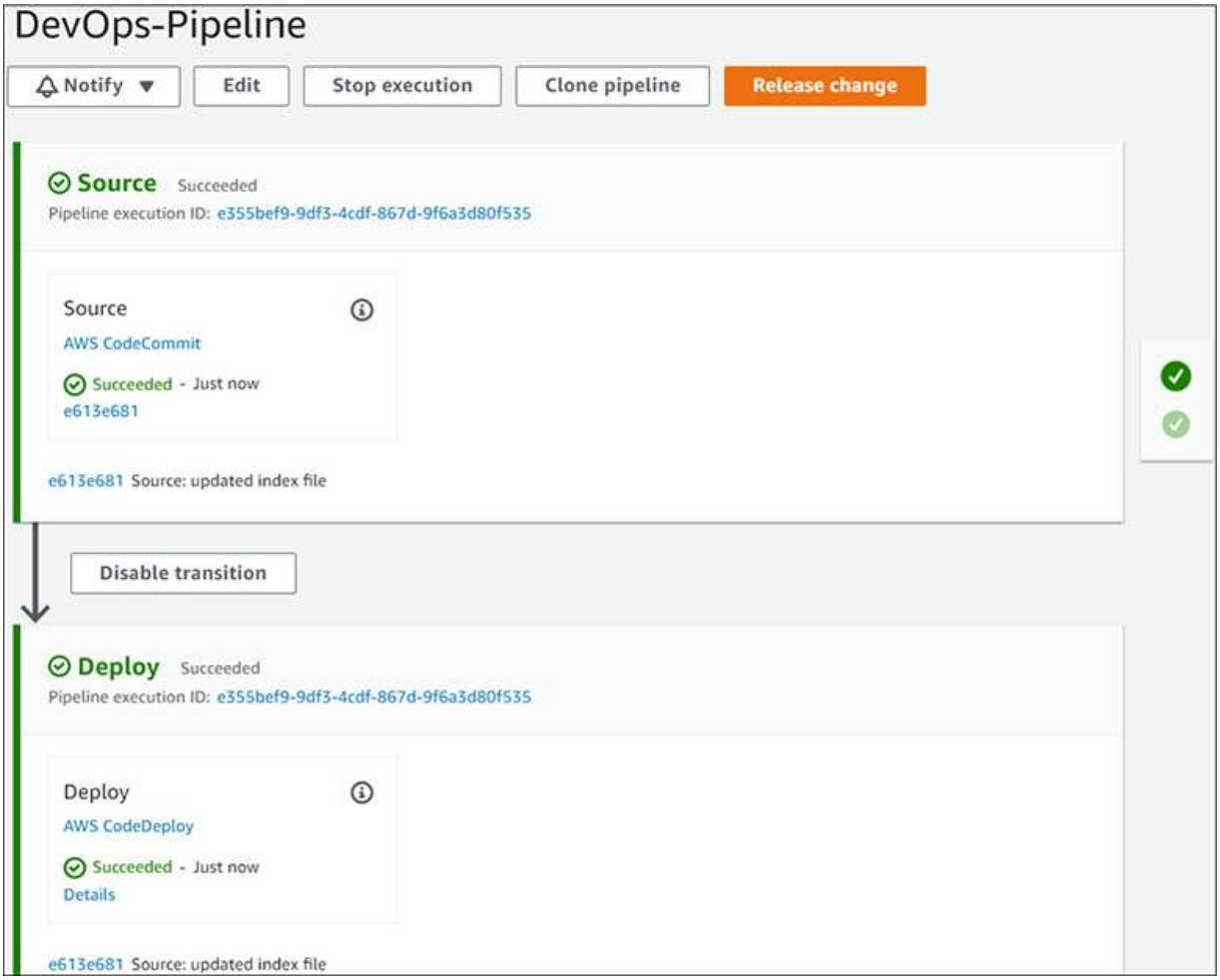
Save the source code change and add it to the repository using the git add command. Then commit the changes using the git commit command and push the updated source code to the remote repository using the git push command.



Navigate to the AWS CodePipeline and verify that the source for the updated index file has completed successfully. You will notice that the deploy stage is in progress for the updated index file.

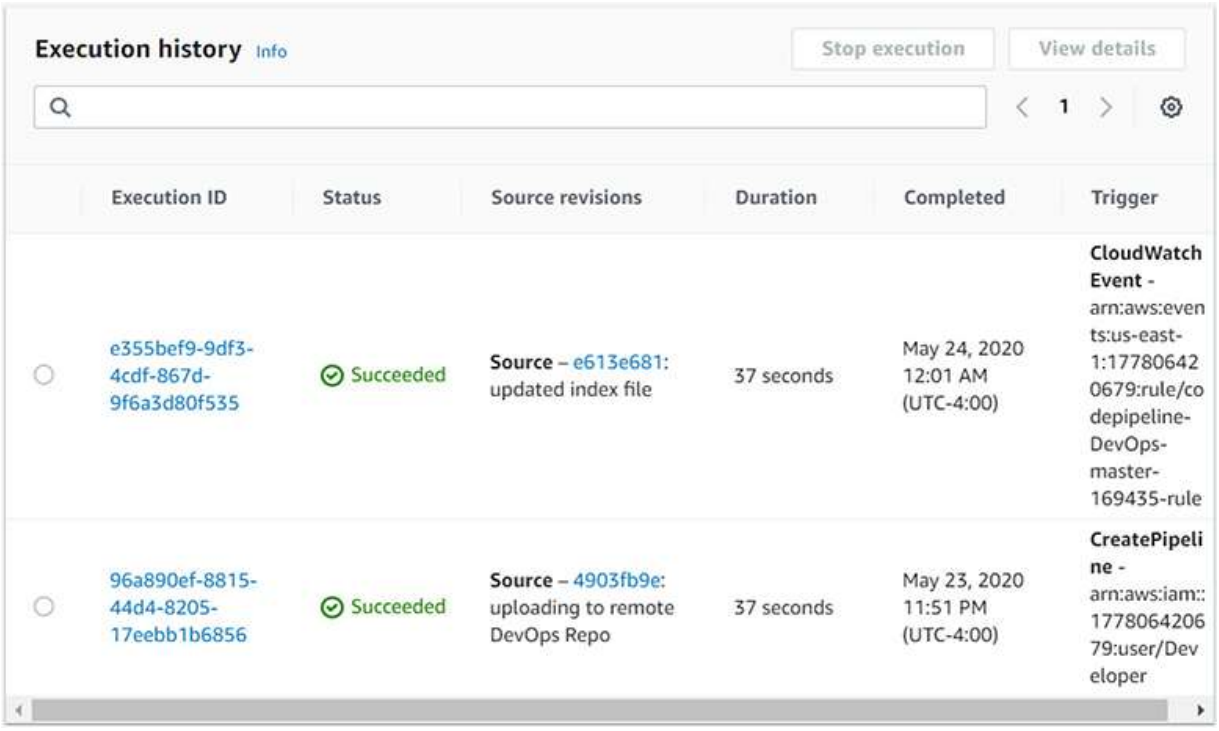


Within a few seconds, the deploy stage is completed successfully.



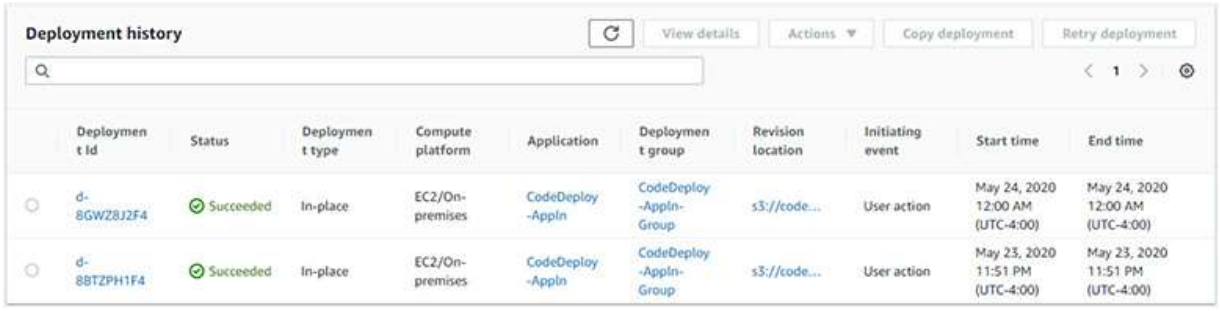
Open a browser and enter the public IP of your Amazon EC2 instance. You will see the updated application code deployed successfully.

You can review the pipeline's execution history. It shows the source revisions, duration, and triggers, etc.



When you click on Settings for the pipeline, you can see the general details, notifications (if configured), and pipeline tags. You also can see the artifact store, version, and region details.

Navigate to the AWS CodeDeploy deployment history and review the deployment details.



You have successfully created a pipeline and automatically deployed the changes. You need to clean up all the resources that you created as a best practice for your Free Tier to avoid any charges.

Chapter Review

This chapter began by describing AWS CodeDeploy and how it helps you automate application deployments to your Amazon EC2 instances. AWS CodePipeline was introduced, which is used to create continuous integration and continuous deployment pipelines and visualize the execution. This is another practical chapter where we created an Amazon EC2 instance and automatically deployed the sample application using AWS CodePipeline. Then we changed the source code and observed the automatic execution of source code deployment into the application. You gained important experience in terms of your certification and how it will relate to the real world in this chapter.

Exercises

The following exercises will help you practice using AWS CodeDeploy and AWS CodePipeline. You need to create an AWS account, as explained earlier, before performing these exercises. You can use the Free Tier when launching AWS resources, but make sure to terminate them at the end.

Exercise 25-1: Delete the AWS CodeDeploy Application Using the AWS Management Console

1. Use your AWS account email address and password to sign in to the AWS account and then navigate to the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane, expand Deploy. Then choose Applications.
4. Choose the DevOps-Appln that you created in this chapter and click on the application name.
5. On the Application Details page, click Delete Application.
6. In response to the prompt, confirm you want to delete.

Exercise 25-2: Delete the AWS CodePipeline Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Name.
4. Choose the DevOps-Pipeline that you created in this chapter.
5. Click on Edit on the Details page.
6. Click on Delete from the Edit page.
7. In response to the prompt, confirm you want to delete the pipeline.

Exercise 25-3: Delete the AWS CodeCommit Repository Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CodeCommit console at https://console.aws.amazon.com/codecommit/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.

3. From the navigation pane on the left, choose Repositories.
4. Choose Settings.
5. Navigate to Delete Repository on the General tab.
6. Choose Delete Repository.
7. Enter **delete** in the popup window and choose to delete.

Exercise 25-4: Delete the Amazon EC2 Instance Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Instances.
4. Choose the DevOps instance and select Actions from the dropdown.
5. Click Instance State and choose Terminate.
6. When prompted, choose Yes to terminate the instance.

Questions

The following questions will help you gauge your understanding of the contents in this chapter. Read all the answers carefully because there might be more than one correct answer. Choose the best responses for each question.

1. Your company wants to improve the application deployment process in AWS and asked you to find an appropriate tool that makes it easy to rapidly release new features and avoid any downtime during deployment and scale the deployment to thousands of instances when required. Which of the following AWS service provides those features?

   A. AWS CodeCommit

   B. AWS CodeBuild

   C. AWS CodeDeploy

D. AWS CodePipeline

2. Which of the following compute platforms does AWS CodeDeploy use to deploy applications? (Choose three.)

   A. AWS Lambda

   B. Amazon ECS

   C. Amazon EC2/On-Premises

   D. Amazon S3

3. Which of the following deployment configurations does AWS CodeDeploy use to specify how traffic is routed during deployment? (Choose three.)

   A. Canary

   B. Linear

   C. All-at-once

   D. Rolling

4. Which of the following compute platforms requires the CodeDeploy agent in AWS CodeDeploy deployment?

   A. Amazon EC2/On-Premises

   B. Amazon ECS

   C. AWS Lambda

   D. Amazon S3

5. What two deployment types are available in AWS CodeDeploy? (Choose two.)

   A. Reckless deployment

   B. In-place deployment

   C. Shadow deployment

   D. Blue/green deployment

6. A company started using AWS CodeDeploy to automate application deployments. Which of the following deployments cannot use an in-place deployment type? (Choose two.)

   A. Amazon EC2

B. Amazon ECS

C. AWS Lambda

D. Amazon S3

7. Which of the following can be used to access and manage AWS CodeDeploy? (Choose all that apply.)

   A. AWS CodeDeploy API

   B. AWS SDK

   C. AWS Management Console

   D. AWS CLI

8. Your company has a hybrid environment and has applications with instances on both AWS Cloud and On-Premises. You enabled blue/green deployments on AWS CodeDeploy to make the applications highly available. Which of the following is true?

   A. Blue/green deployments work only with Amazon EC2 instances

   B. Blue/green deployments are not supported in the AWS Lambda compute platform

   C. Blue/green deployments work only with On-Premises instances

   D. Blue/green deployments are not supported in the Amazon ECS compute platform

9. A company wants to automate and visualize the continuous integration and continuous deployment pipeline to release software code to production. Which of the following AWS services has the functionality that provides this solution?

   A. AWS CodePipeline

   B. AWS CodeBuild

   C. AWS CodeDeploy

   D. AWS CodeCommit

10. Which of the following can be used to stop pipeline execution in AWS CodePipeline? (Choose two.)

    A. Stop and wait

    B. Stop and start

C. Stop and abandon

D. Stop and terminate

Answers

1. **C.** AWS CodeDeploy can be used to rapidly release new features to applications and avoid any downtime during deployment and scale the deployment to thousands of instances when required.
2. **A, B, C.** AWS Lambda, Amazon ECS, and Amazon EC2/On-Premises are the compute platforms available in AWS CodeDeploy.
3. **A, B, C.** All-at-once, canary, and linear are the deployment configurations that AWS CodeDeploy uses to specify how traffic is routed during deployment.
4. **A.** Amazon EC2/On-Premises requires the CodeDeploy agent in AWS CodeDeploy deployment.
5. **B, D.** In-place deployment and blue/green deployment are available in AWS CodeDeploy.
6. **B, C.** Amazon ECS and AWS Lambda cannot use an in-place deployment type in AWS CodeDeploy.
7. **A, B, C, D.** AWS CodeDeploy API, AWS SDK, AWS Management Console, and AWS CLI can be used to access and manage AWS CodeDeploy.
8. **A.** Blue/green deployments work only with Amazon EC2 instances.
9. **A.** AWS CodePipeline can be used to automate and visualize the end-to-end continuous integration and continuous delivery pipeline to release software code to production.
10. **A, C.** Stop and wait and stop and abandon are the two ways used to stop pipeline execution in AWS CodePipeline.

Additional Resources

- **AWS CodeDeploy** The official AWS documentation is the best place to get the latest updates and new features for each service, including

AWS CodeDeploy.
`https://docs.aws.amazon.com/codedeploy/index.html`

- **AWS CodeDeploy Blogs** These blogs will help you explore AWS CodeDeploy in more depth and include many best-practice architectures to practice and implement.
`https://aws.amazon.com/blogs/devops/tag/codedeploy/`
- **AWS CodePipeline** The official AWS documentation is the best place to get all the latest information on all the AWS services, including AWS CodePipeline.
`https://docs.aws.amazon.com/codepipeline/index.html`
- **AWS CodePipeline Blogs** These AWS DevOps blogs will help you explore many reference architectures and best practices that can be used for your proof of concept and to practice and implement in your own environment.
`https://aws.amazon.com/blogs/devops/category/developer-tools/aws-codepipeline/`

CHAPTER 26

Building a Scalable and Fault-Tolerant CI/CD Pipeline

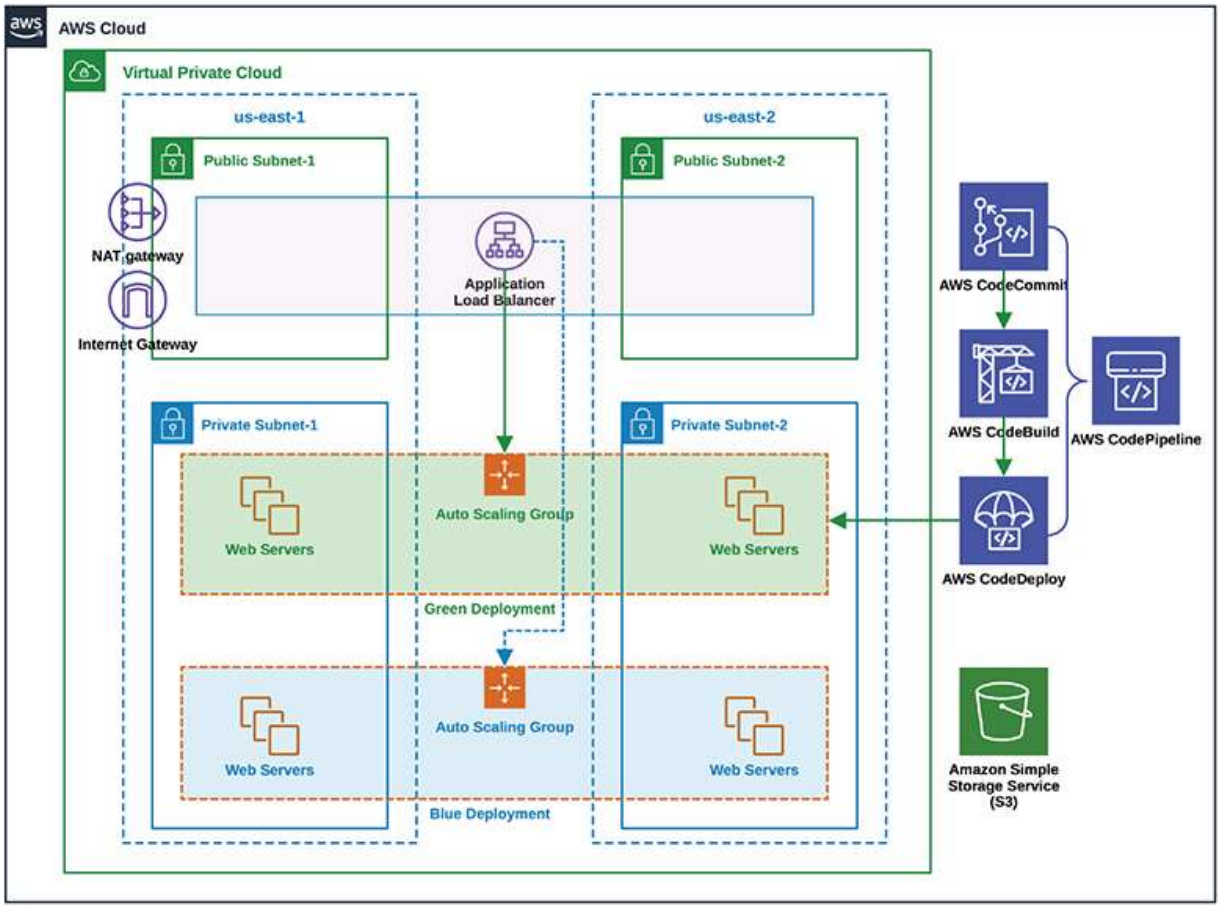
In this chapter, you will learn

- Build a CI/CD pipeline using AWS developer tools

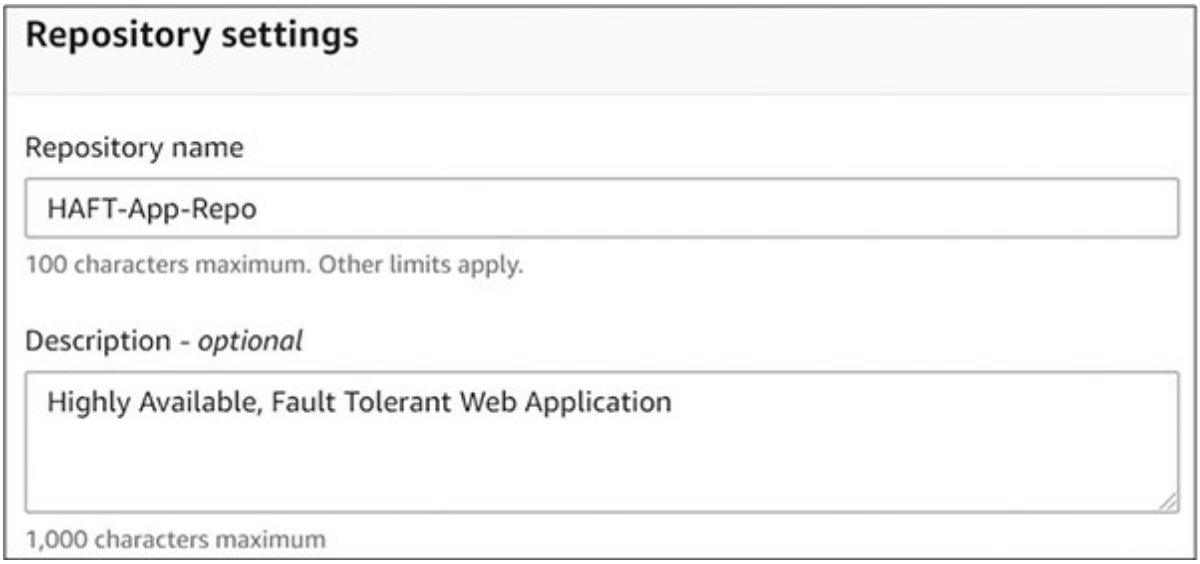
In this chapter, we are going to build highly available, scalable, and fault-tolerant application using AWS developer tools such as AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline.

CI/CD Pipeline

A typical CI/CD pipeline automates the entire software code builds, automated testing, and deployment to a production environment process using DevOps tools. We are going to use AWS developer tools such as AWS CodeCommit for storing application source code, or you can use another source code repository such as GitHub; AWS CodeBuild to build and test the application code; AWS CodeDeploy to automate code deployments using blue/green deployment; and AWS CodePipeline to automate the continuous delivery pipeline.

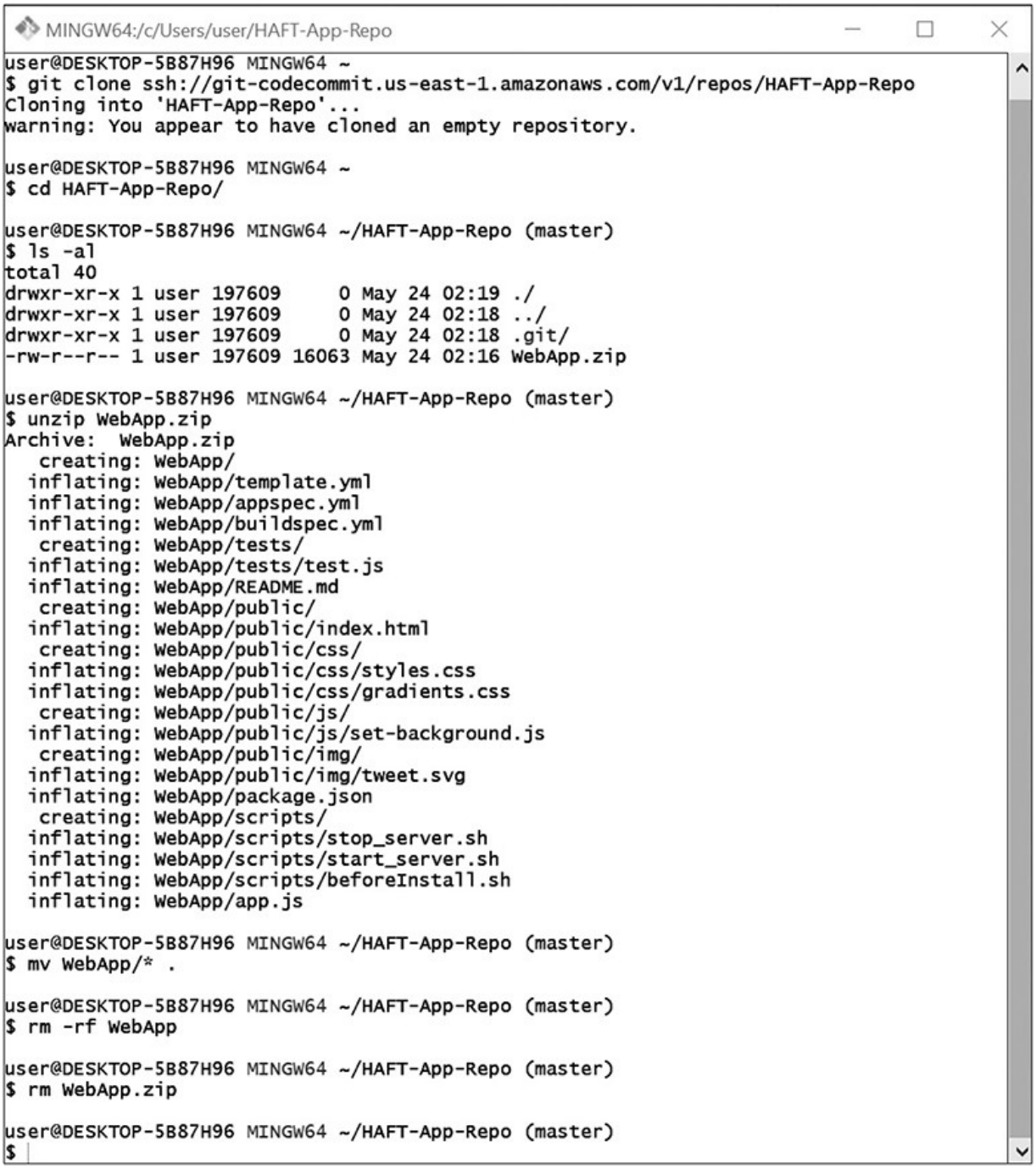


First, you need to log in to your AWS Management Console, navigate to the AWS CodeCommit service, and choose Create Repository. Enter the repository name—for example, **HAFT-App-Repo**—and provide a description of **Highly Available Fault Tolerant Web Application** and then click the Create button.



Note the clone URL for HTTPS or SSH.

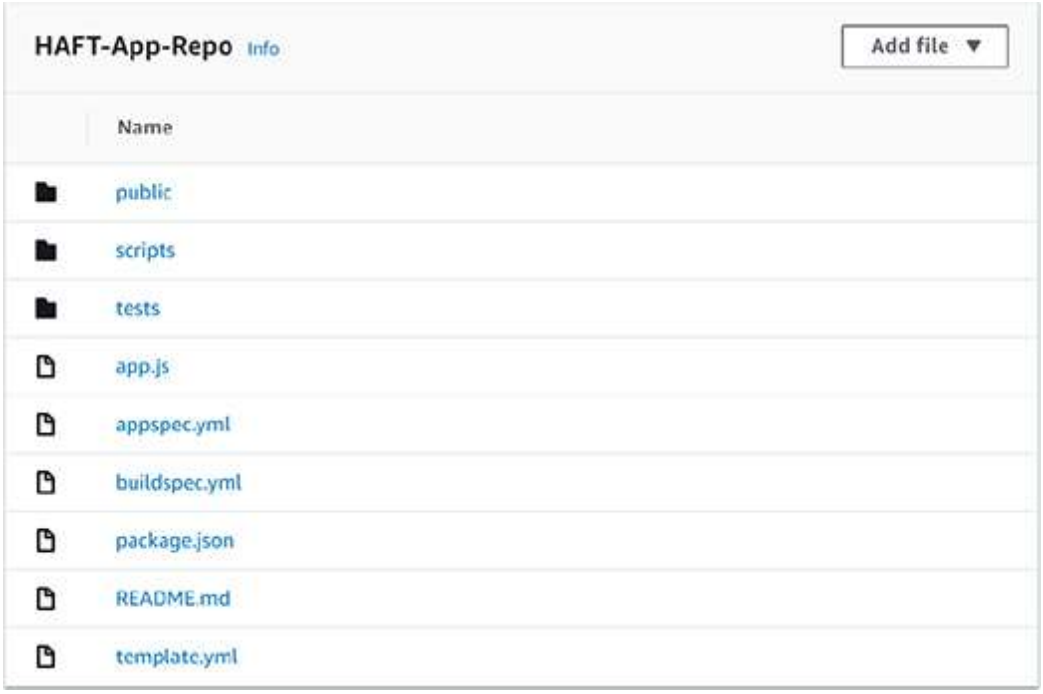
Go to your local machine and use git.bash for Windows and Terminal for macOS. You need to clone the AWS CodeCommit repository using the git clone and go to the HAFT-App-Repo folder. Download the AWS sample web application from https://github.com/aws-samples/aws-cicd-bluegreen/raw/master/WebApp.zip and unzip it to the local repository. You also need to move and remove some files from it.



Add the sample web application files to the repository using the git add command.

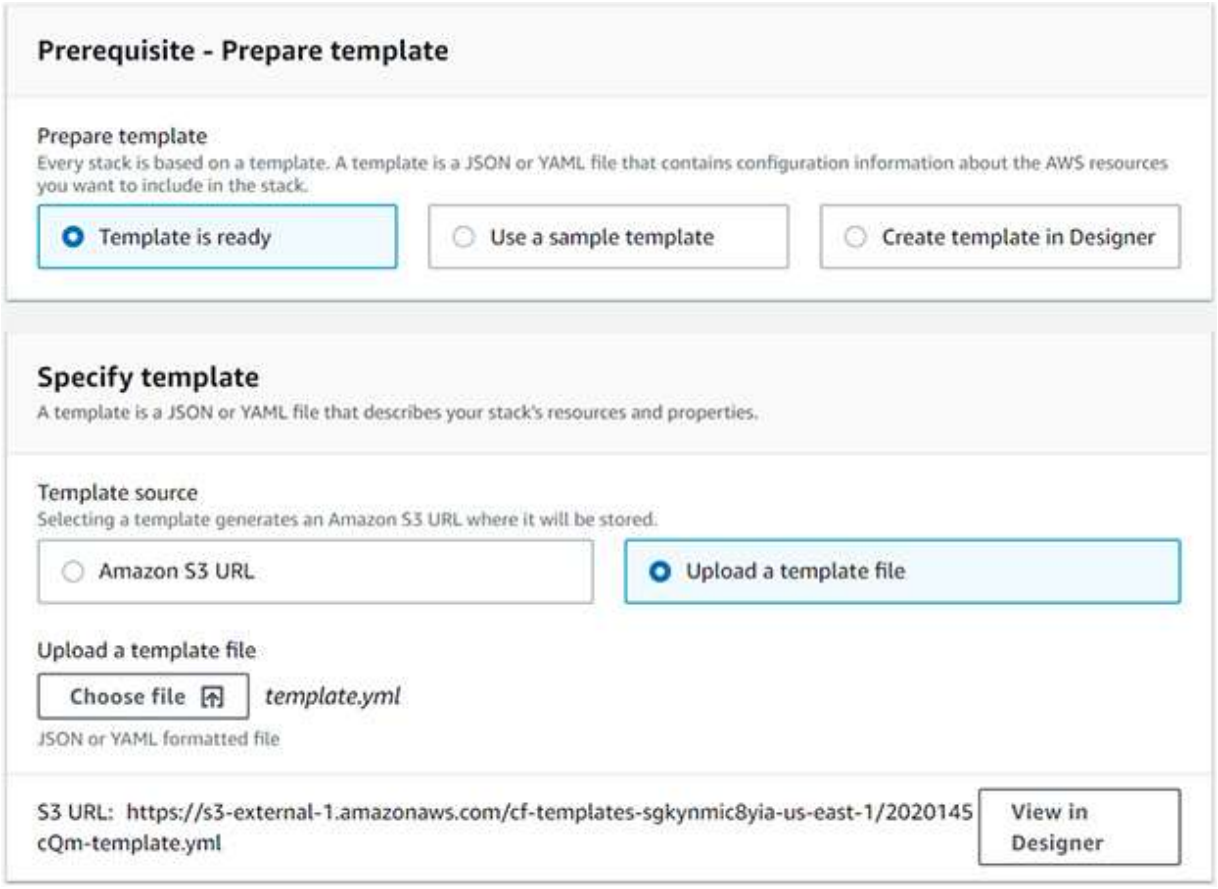
Commit the files using the git commit command, and use the git push command to push the AWS sample web application files to the remote repository in AWS CodeCommit.

Navigate to the AWS CodeCommit service in the AWS Management Console and verify your updated repository.

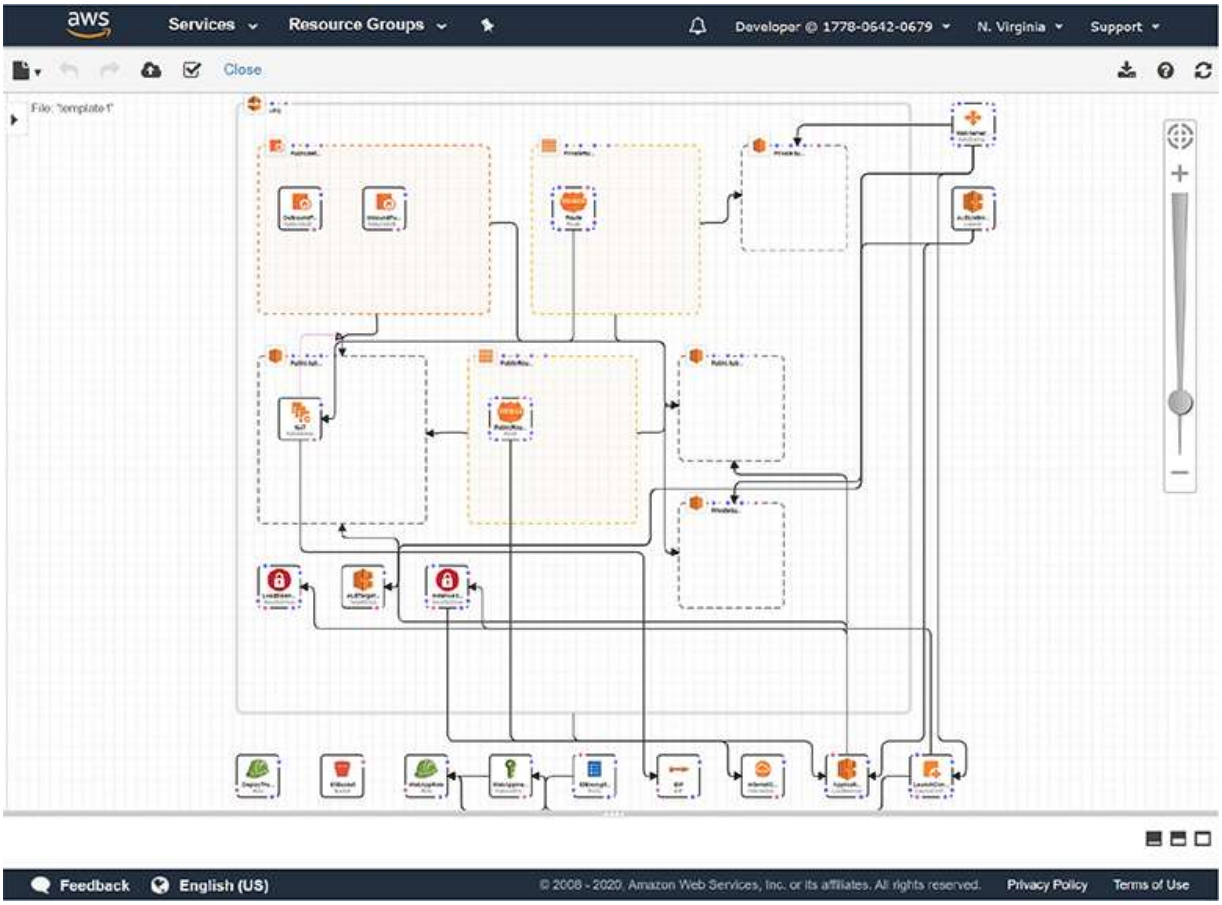


Next, you need to create the infrastructure using the template.yml file, which you extracted from WebApp.zip. Navigate to the AWS CloudFormation service in the AWS Management Console and click on Create Stack.

Choose Template Is Ready in the Prepare Template section and choose Upload A Template File for the Template Source. Upload template.yml from your local machine and click View In Designer.

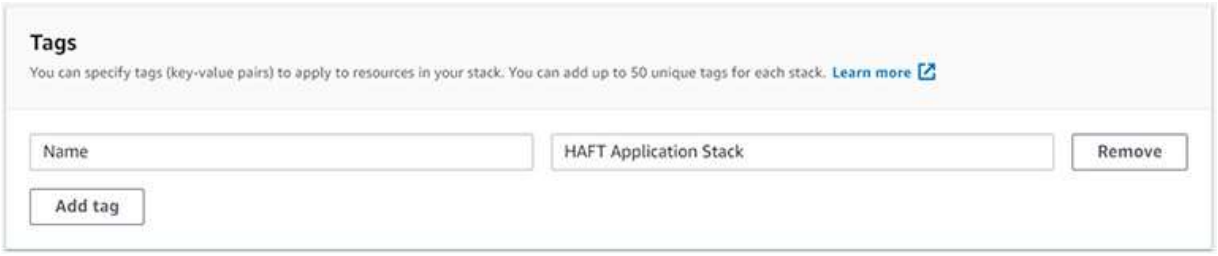


The AWS CloudFormation designer is a graphical tool used to visualize and modify resources using a drag-and-drop interface. You can verify all the resources that will be created as part of this template.



Enter a stack name—for example, **HAFT-Application**—and click Next.

As per best practices, add a tag of **Name** with value of **HAFT Application Stack** and leave other options at their default settings. Click Next to continue.



Review all the options that you selected, acknowledge that the AWS CloudFormation might create IAM resources, and click Create Stack.

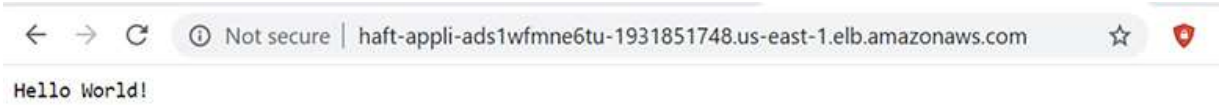
Select the Events tab to see all the events in your template execution, which have a "Resource creation initiated" status. As you recall from Chapter 18, AWS CloudFormation takes care of the dependencies regarding which resource needs to be created first, etc. Scroll down and verify that all the resources are created successfully.

Timestamp	Logical ID	Status	Status reason
2020-05-24 02:42:13 UTC-0400	NAT	CREATE_IN_PROGRESS	Resource creation Initiated
2020-05-24 02:42:13 UTC-0400	NAT	CREATE_IN_PROGRESS	-
2020-05-24 02:42:11 UTC-0400	PublicSubnetNetworkAclAssociation01	CREATE_COMPLETE	-
2020-05-24 02:42:10 UTC-0400	PublicSubnetRTAssociation01	CREATE_COMPLETE	-
2020-05-24 02:42:10 UTC-0400	PrivateSubnetRTAssociation02	CREATE_COMPLETE	-
2020-05-24 02:42:10 UTC-0400	PrivateSubnetRTAssociation01	CREATE_COMPLETE	-
2020-05-24 02:42:09 UTC-0400	PublicSubnetRTAssociation02	CREATE_COMPLETE	-
2020-05-24 02:42:09 UTC-0400	PublicSubnetNetworkAclAssociation02	CREATE_COMPLETE	-
2020-05-24 02:41:56 UTC-0400	ApplicationLoadBalancer	CREATE_IN_PROGRESS	Resource creation Initiated
2020-05-24 02:41:55 UTC-0400	PublicSubnetNetworkAclAssociation01	CREATE_IN_PROGRESS	Resource creation Initiated
2020-05-24 02:41:55 UTC-0400	PublicSubnetRTAssociation01	CREATE_IN_PROGRESS	Resource creation Initiated
2020-05-24 02:41:55 UTC-0400	PublicSubnetNetworkAclAssociation01	CREATE_IN_PROGRESS	-

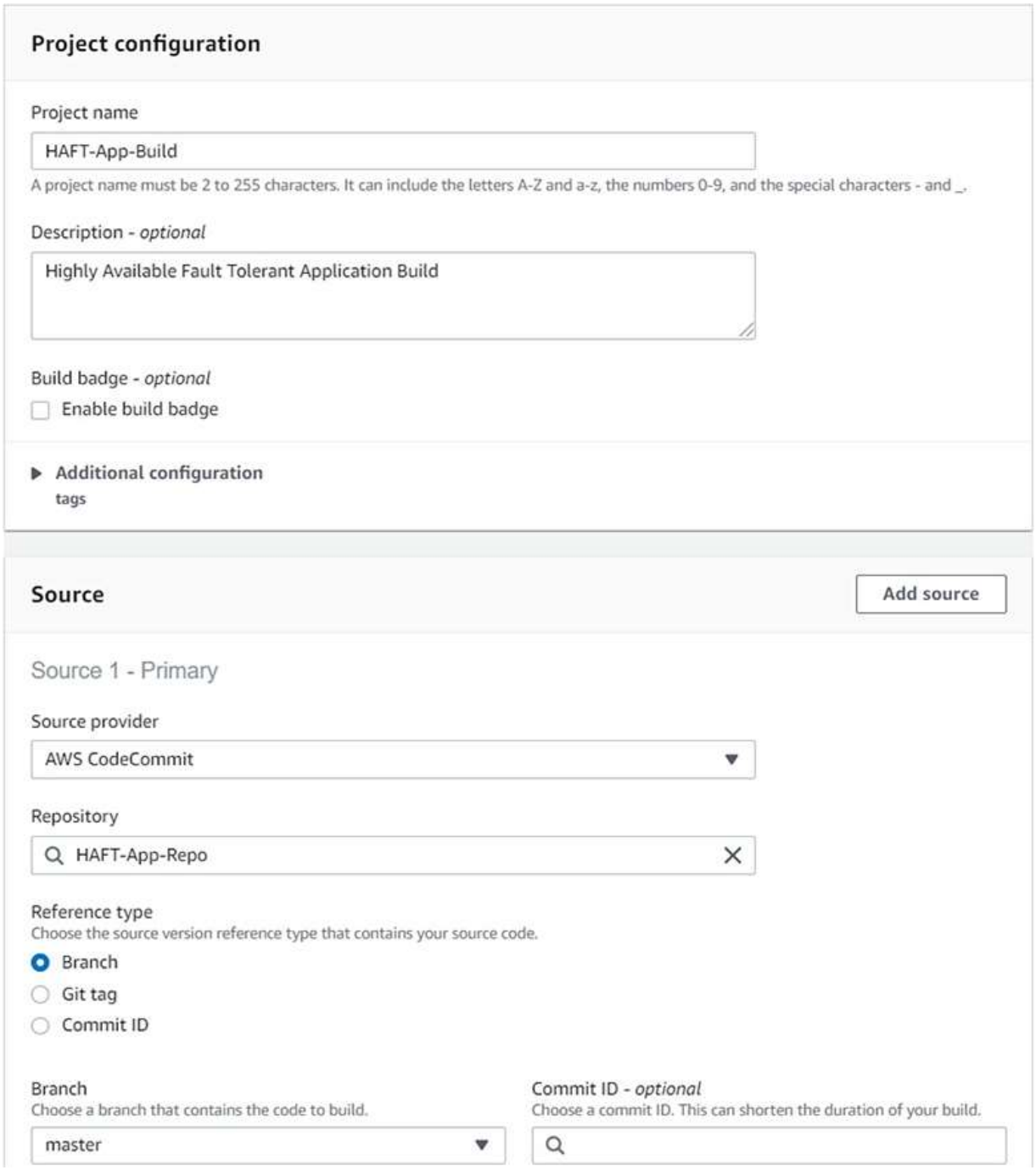
Next, click the Outputs tab and note the Application Load Balancer (ALB) URL.

ALBTargetGroup	BlueGreenTG	TargetGroup Name	-
ASGroup	BlueGreenASGroup	AutoScaling Group Name	-
DeployRoleArn	HAFT-Application-DeployTrustRole-1VF3NE39Y4SR9	CodeDeploy role Arn	-
S3BucketName	build-artifact-bluegreenbucket-us-east-1-177806420679	Bucket to for storing artifacts	-
URL	http://HAFT-Appli-ADS1WFMNE6TU-1931851748.us-east-1.elb.amazonaws.com	URL of the website	-

Open a browser on your local machine, enter the ALB URL, and press ENTER. You will see the Hello World! message. You have successfully created the infrastructure VPC, subnets, web servers, ALB, Internet Gateway, NAT Gateway, etc.

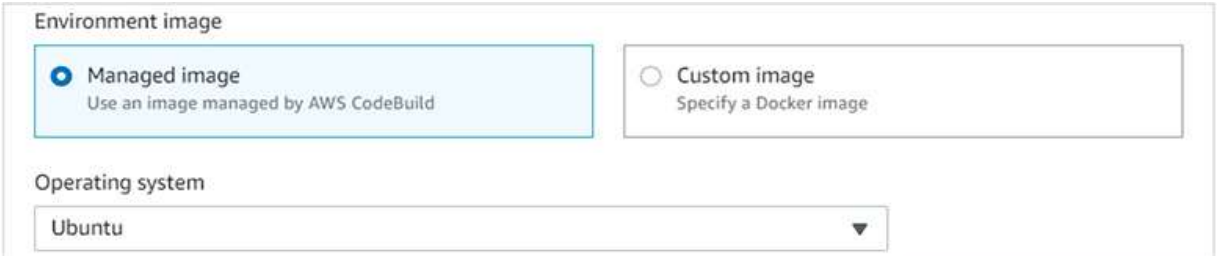


Navigate to the AWS CodeBuild service from the AWS Management Console and choose Create Build Project. Enter **HAFT-App-Build** for the project name and **Highly Available Fault Tolerant Application Build** for the description. In the Source section, choose AWS CodeCommit as the source provider and HAFT-App-Repo as the repository. Then select the reference type as Branch and choose Master.

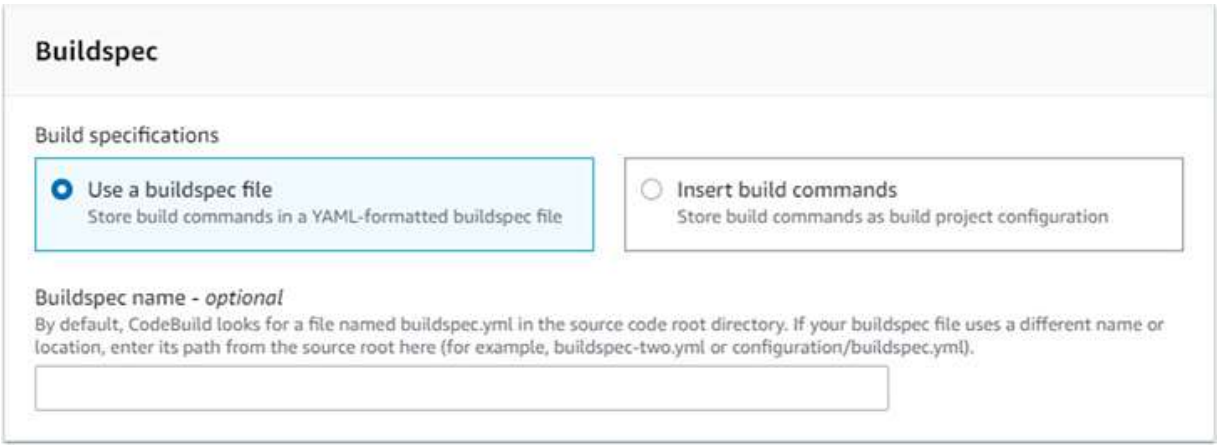


Select the environment image as Managed Image and the operating system as Ubuntu. Choose Runtime for the standard and aws/codebuild/standard:2.0 for the image. Choose Linux for the environment type as Linux and Always Use The Latest Image For This Runtime Version for the image version. Select Privileged if you want to build a Docker image (you can leave it unchecked

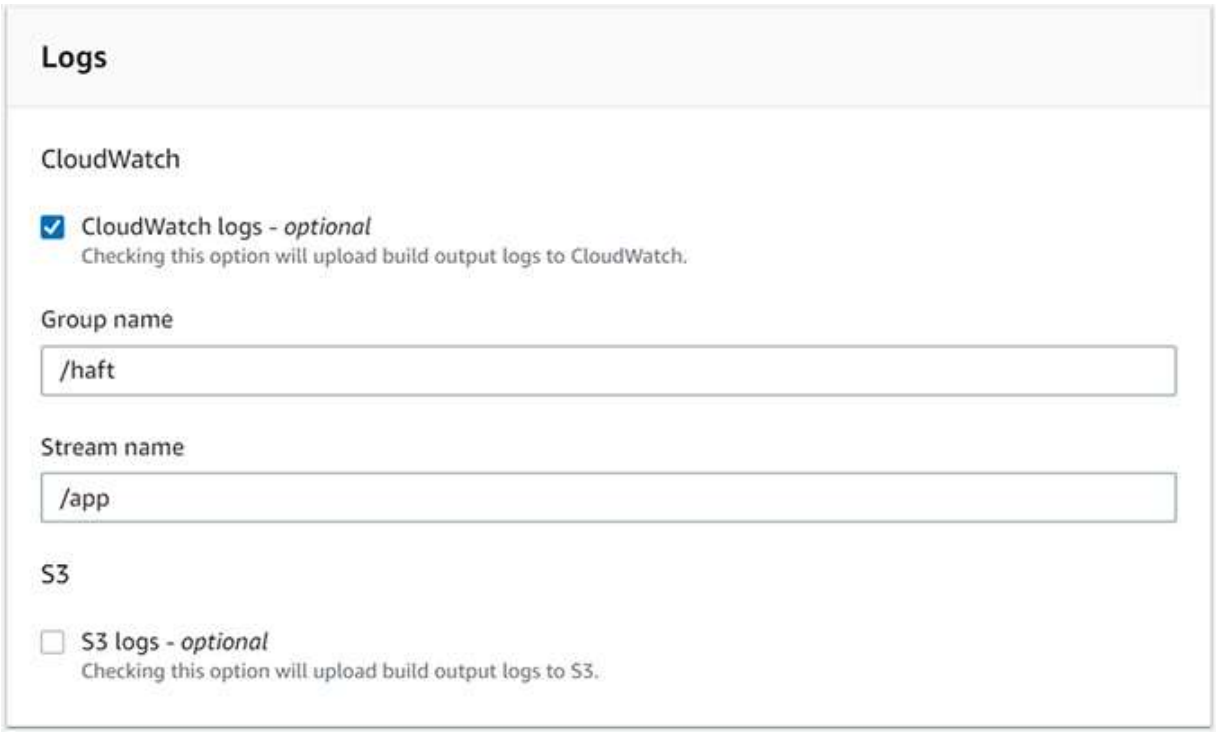
for this application). Select New Service Role to create a new CodeBuild service role in your account.



In the Buildspec section, choose Use A Buildspec File. In the Artifacts section, choose Amazon S3 for the type and select the bucket name and filename. Choose Zip for the artifacts packaging to upload artifacts as a compressed file.



Enable logs and choose either CloudWatch Logs or Amazon S3, or both, which will be helpful to verify detailed steps and for troubleshooting if required. Enter a group name and stream name.



Verify the build project details, such as source provider, primary repository, and artifacts upload location. Click the Start Build button.

Click the Start Build button again to start the initial build process of your application.

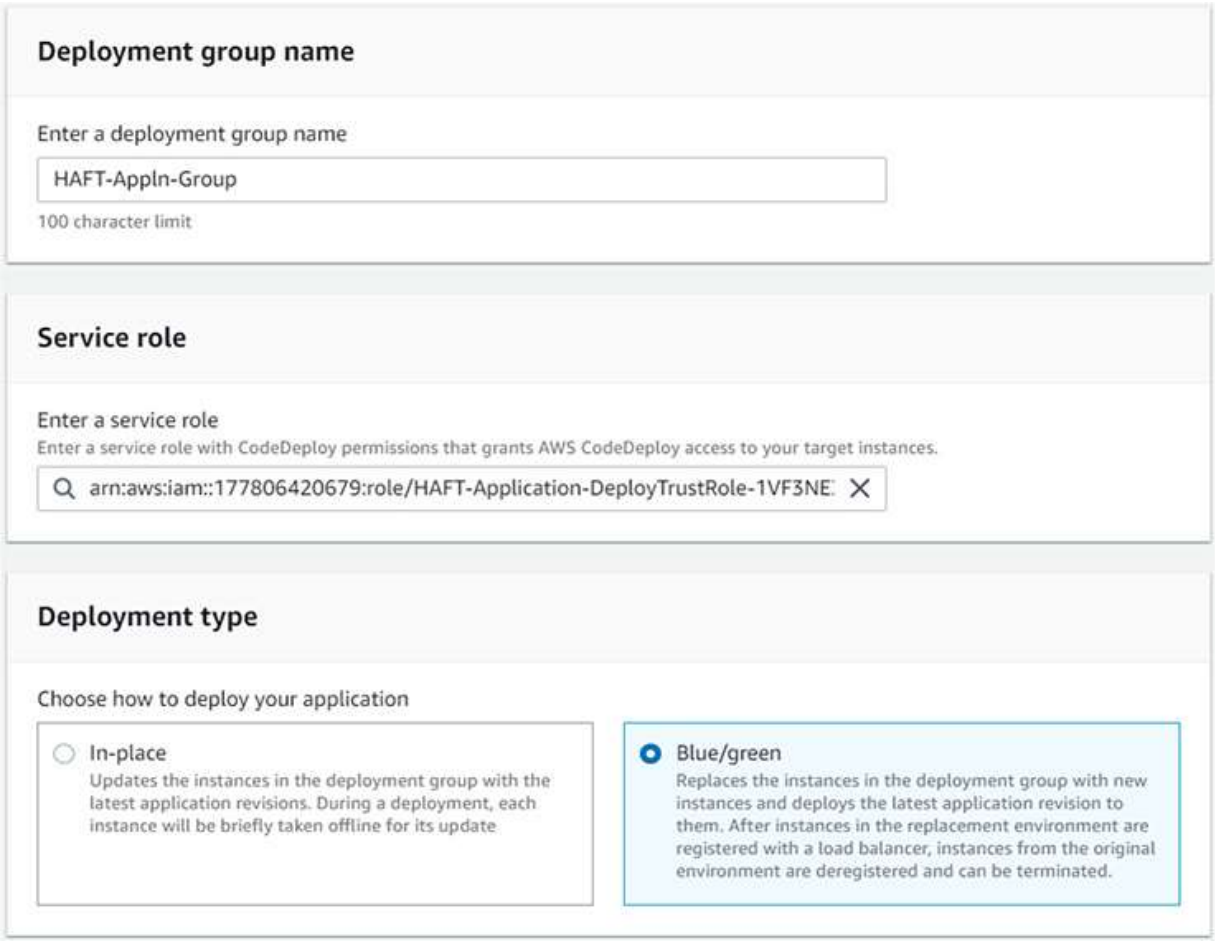
Within a minute, the overall status is “Succeeded” and all the phases are listed, such as Submitted, Queued, Provisioning, Download_source, Install, Pre_build, Build, Post_build, Upload_artifacts, Finalizing, and Completed. You can verify the build number, source version, start time, and end time.

Name	Status	Context	Duration	Start time	End time
SUBMITTED	Succeeded	-	<1 sec	May 24, 2020 3:18 AM (UTC-4:00)	May 24, 2020 3:18 AM (UTC-4:00)
QUEUED	Succeeded	-	1 sec	May 24, 2020 3:18 AM (UTC-4:00)	May 24, 2020 3:18 AM (UTC-4:00)
PROVISIONING	Succeeded	-	18 secs	May 24, 2020 3:18 AM (UTC-4:00)	May 24, 2020 3:18 AM (UTC-4:00)
DOWNLOAD_SOURCE	Succeeded	-	16 secs	May 24, 2020 3:18 AM (UTC-4:00)	May 24, 2020 3:19 AM (UTC-4:00)
INSTALL	Succeeded	-	22 secs	May 24, 2020 3:19 AM (UTC-4:00)	May 24, 2020 3:19 AM (UTC-4:00)
PRE_BUILD	Succeeded	-	<1 sec	May 24, 2020 3:19 AM (UTC-4:00)	May 24, 2020 3:19 AM (UTC-4:00)
BUILD	Succeeded	-	<1 sec	May 24, 2020 3:19 AM (UTC-4:00)	May 24, 2020 3:19 AM (UTC-4:00)
POST_BUILD	Succeeded	-	1 sec	May 24, 2020 3:19 AM (UTC-4:00)	May 24, 2020 3:19 AM (UTC-4:00)
UPLOAD_ARTIFACTS	Succeeded	-	<1 sec	May 24, 2020 3:19 AM (UTC-4:00)	May 24, 2020 3:19 AM (UTC-4:00)
FINALIZING	Succeeded	-	2 secs	May 24, 2020 3:19 AM (UTC-4:00)	May 24, 2020 3:19 AM (UTC-4:00)
COMPLETED	Succeeded	-	-	May 24, 2020 3:19 AM (UTC-4:00)	-

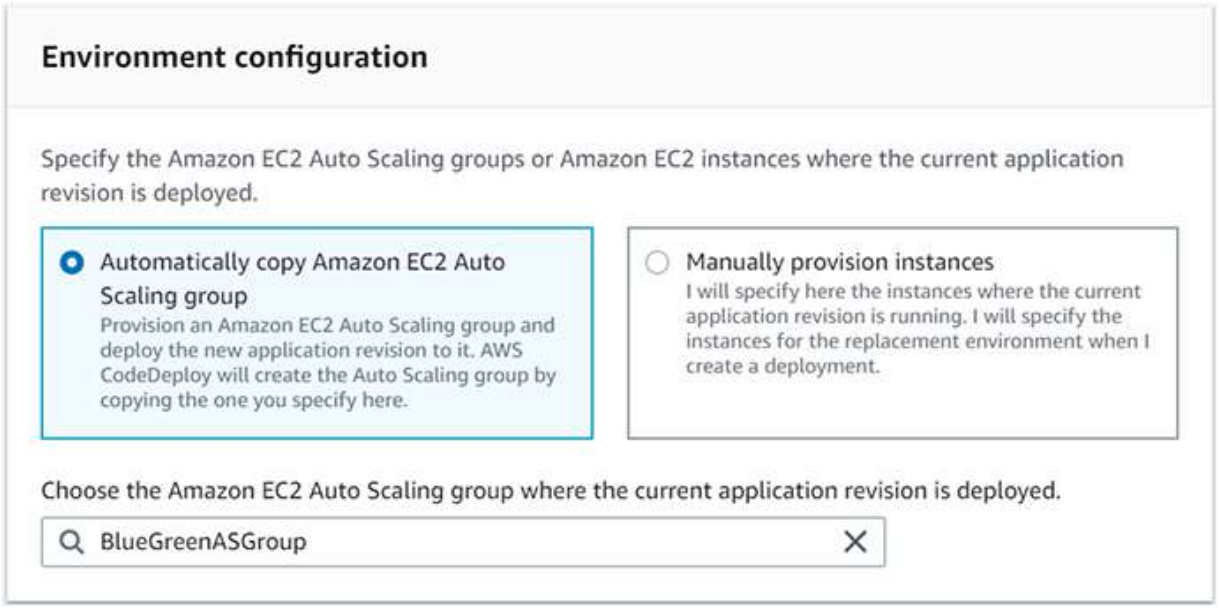
Navigate to AWS CodeDeploy and choose Create Application. Enter an application name, such as **HAFT-Appln**, and select the compute platform as EC2/On-Premises from the dropdown. Click the Create Application button.

Create a deployment group by clicking the Create Deployment Group button.

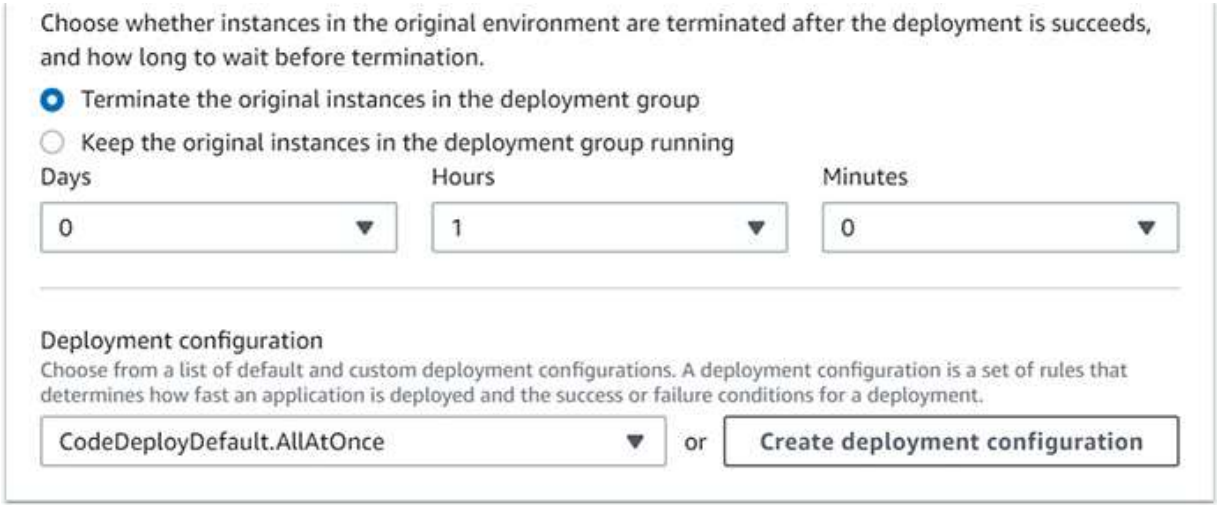
Enter **HAFT-Appln-Group** for the deployment group name and choose a service role that has access to CodeDeploy. Choose Blue/Green for the deployment type, which replaces the instances in the deployment group with new instances that have the latest application revision. Once the new environment is registered with load balancer, the old environment will be deregistered and its instances terminated.



In the Environment Configuration area, choose Automatically Copy Amazon EC2 Auto Scaling Group to deploy the revised application to it. Choose the current Amazon EC2 Auto Scaling group name, such as BlueGreenAsGroup. In Deployment Settings, choose Reroute Traffic Immediately and choose Terminate The Original Instances In The Deployment Group.

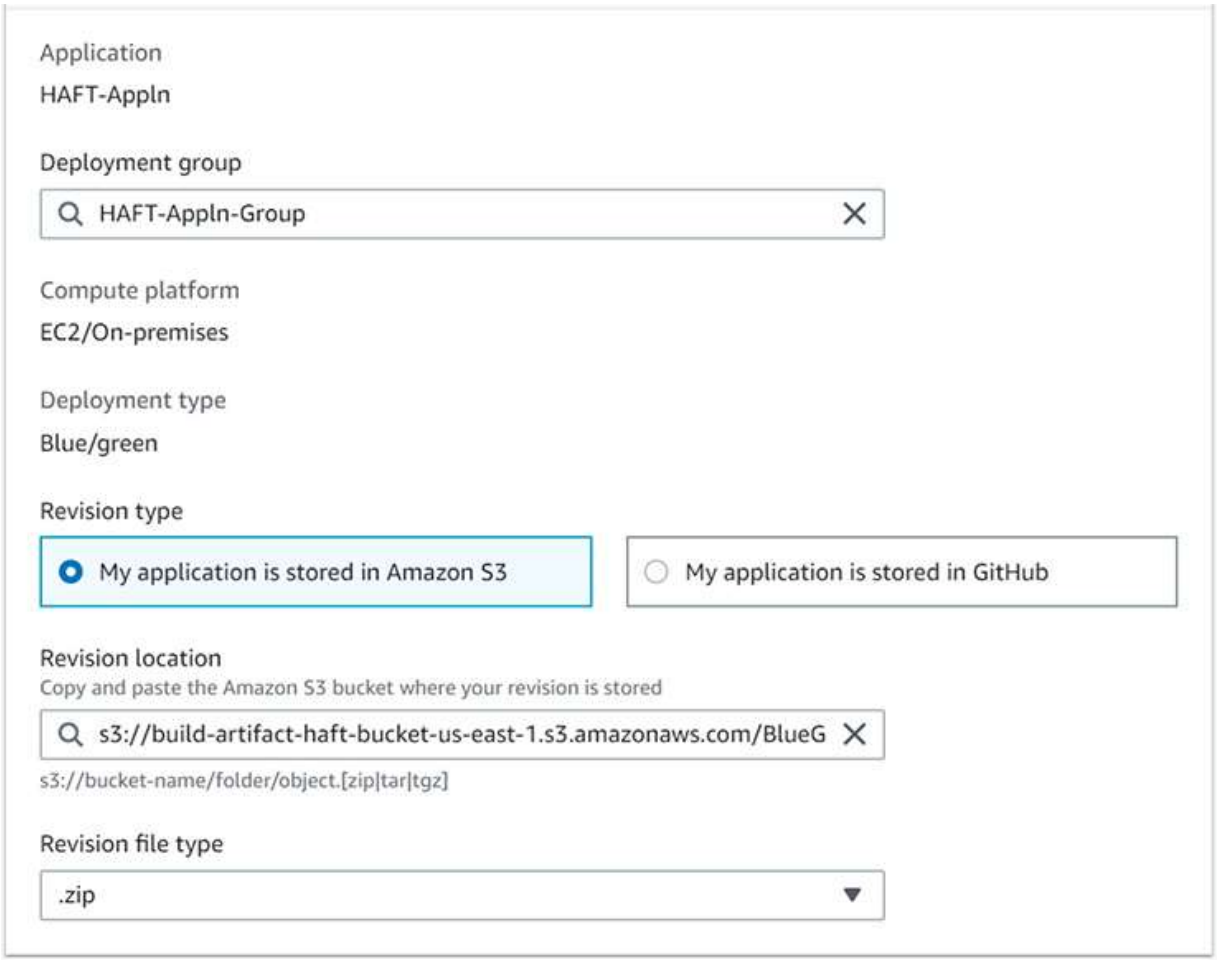


For the deployment configuration choose AllAtOnce, which is the default for CodeDeploy. In the Load Balancer section, choose Application Load Balancer, choose the target group from the dropdown, and click Create Deployment Group.



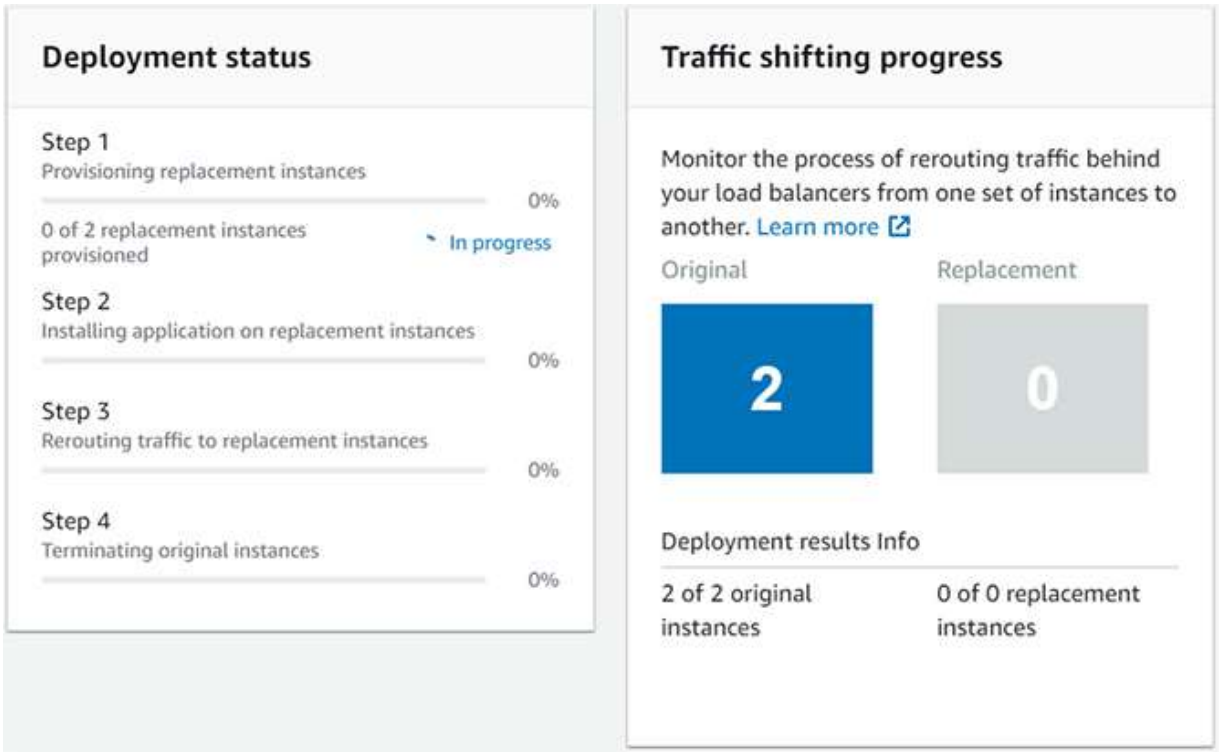
Your deployment group is successfully created, and now you need to create the deployment by clicking the Create Deployment button.

In the Deployment settings, enter **HAFT-Appln-Group** for the deployment group name and choose My Application Is Stored In Amazon S3 for the revision type. Enter your Amazon S3 bucket location in the Revision location, and choose Zip from the dropdown for Revision Type.



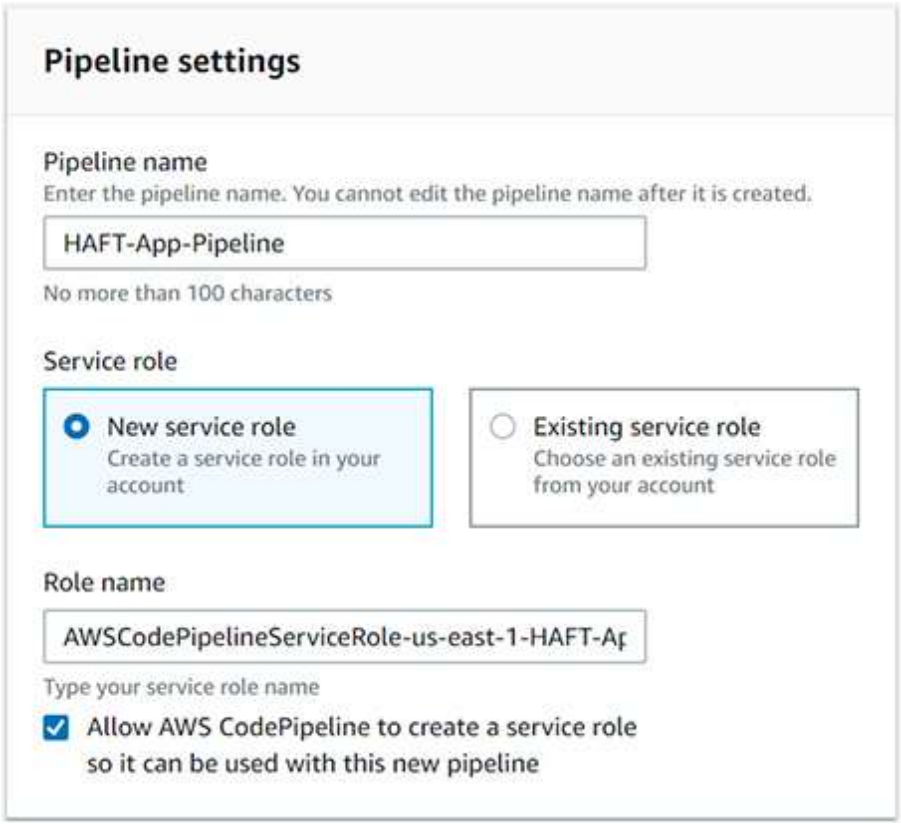
On the next page you can enter additional deployment behavior settings. For this example, leave everything at the default settings and click the Create Deployment button.

In the Deployments section, you can see the status of provisioning replacement instances, installing application on the replacement instances, reroute traffic to replacement instances, and terminating the original instances. You can also view the traffic shifting progress here.

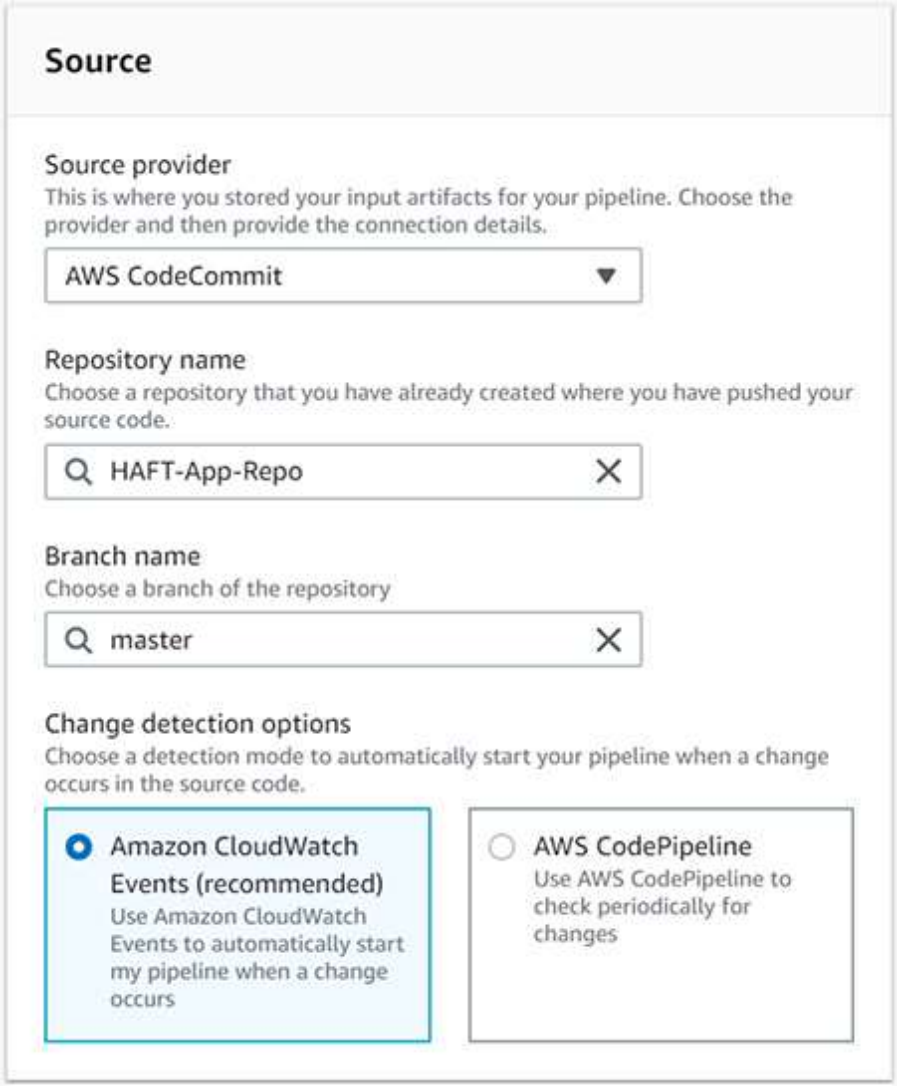


You have successfully configured AWS CodeDeploy, so navigate to AWS CodePipeline and click Create Pipeline.

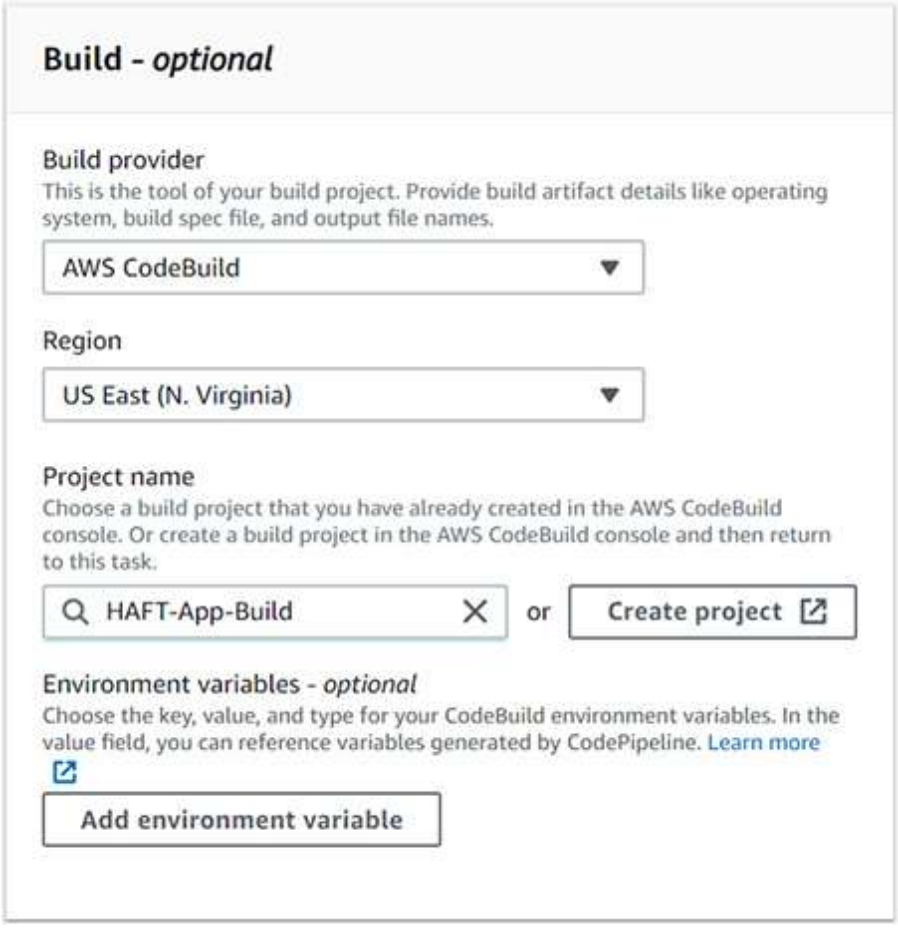
Enter a name for your pipeline, such as **HAFT-APP-Pipeline**, and choose to create a new service role. Check Allow AWS CodePipeline To Create A New Service Role To Be Used In This Pipeline and click Next.



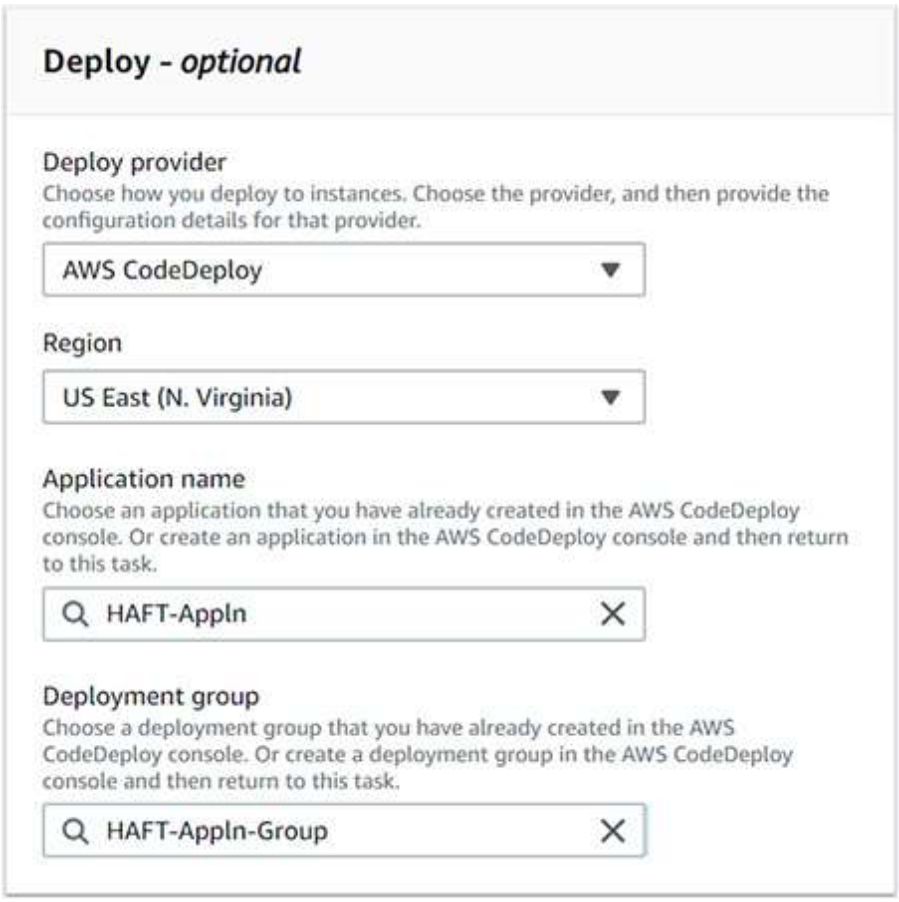
In the Add Source Stage area, choose AWS CodeCommit from the dropdown for the source provider. Enter **HAFT-App-Repo** for the Repository name and select Master for the branch name. Then select the recommended Amazon CloudWatch events for change detection options and click Next.



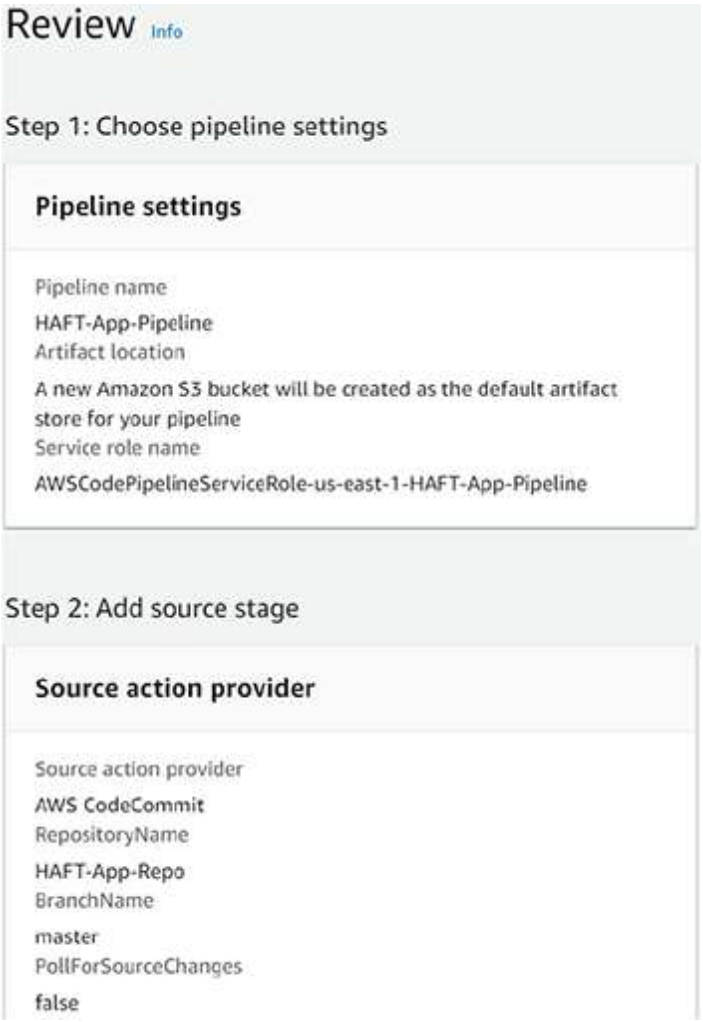
In the Add Build Stage area, choose AWS Codebuild as the build provider from the dropdown and choose your region. Enter the build project name, such as **HAFT-App-Build**, leave the other settings at their defaults, and click Next.



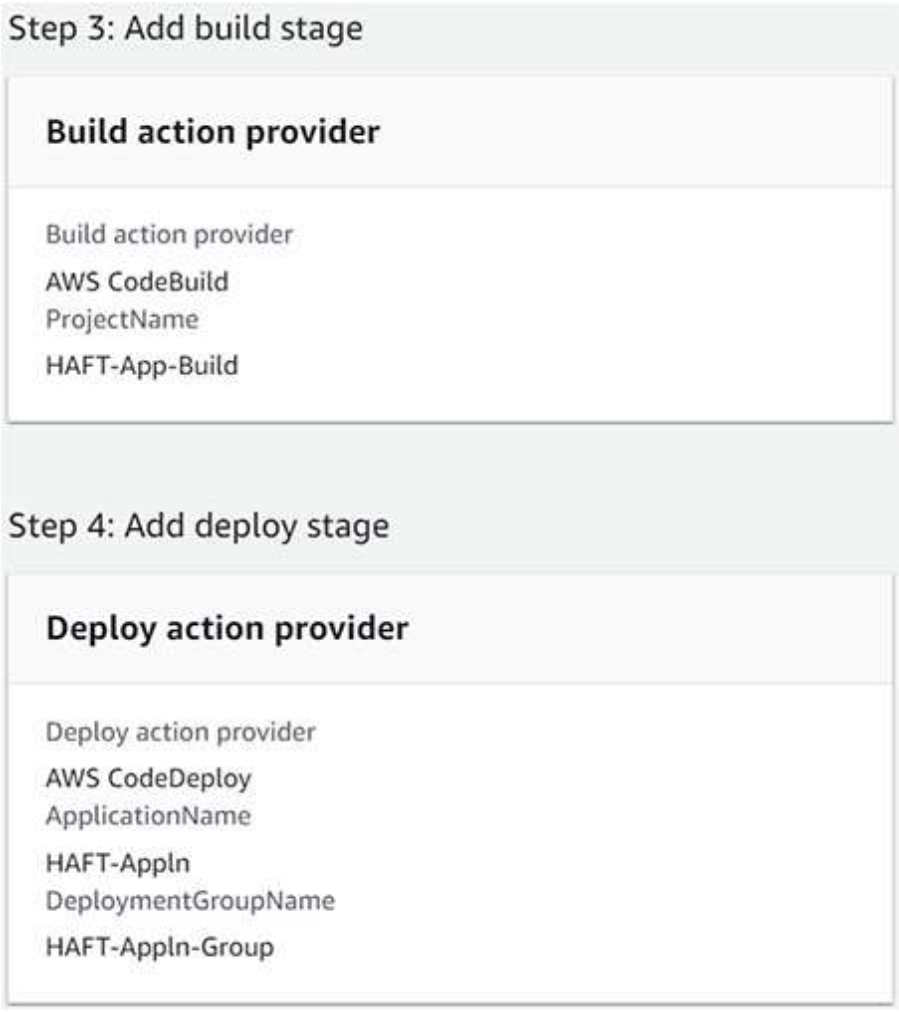
In the Add Deploy Stage area, choose AWS CodeDeploy as the deploy provider from the dropdown and choose your region. Then enter the CodeDeploy application name, such as **HAFT-Appln**, and enter **HAFT-Appln-Group** for the deployment group. Click Next to continue.



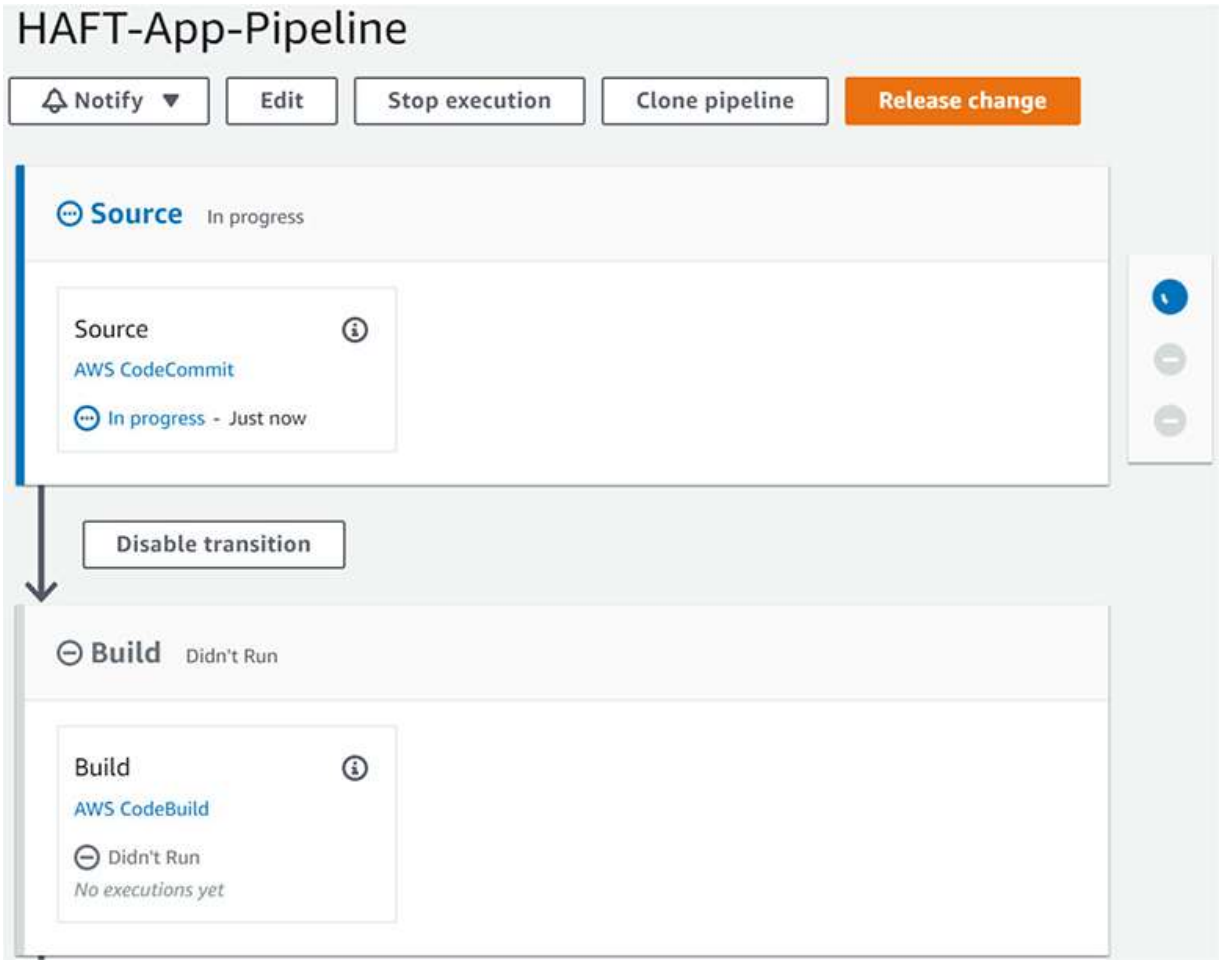
Review the pipeline settings and verify the source provider details.



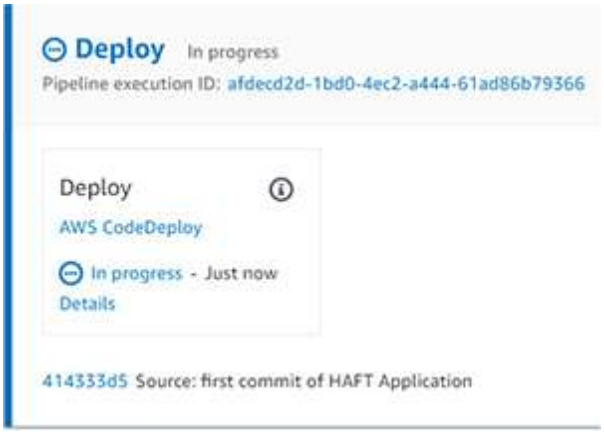
Review the build stage details and the deploy action provider details. Click Create Pipeline once you are done.



You have successfully created the HAFT-App-Pipeline, and you can visually see the progress of your CI/CD pipeline. The source stage is in progress and the build stage is waiting.

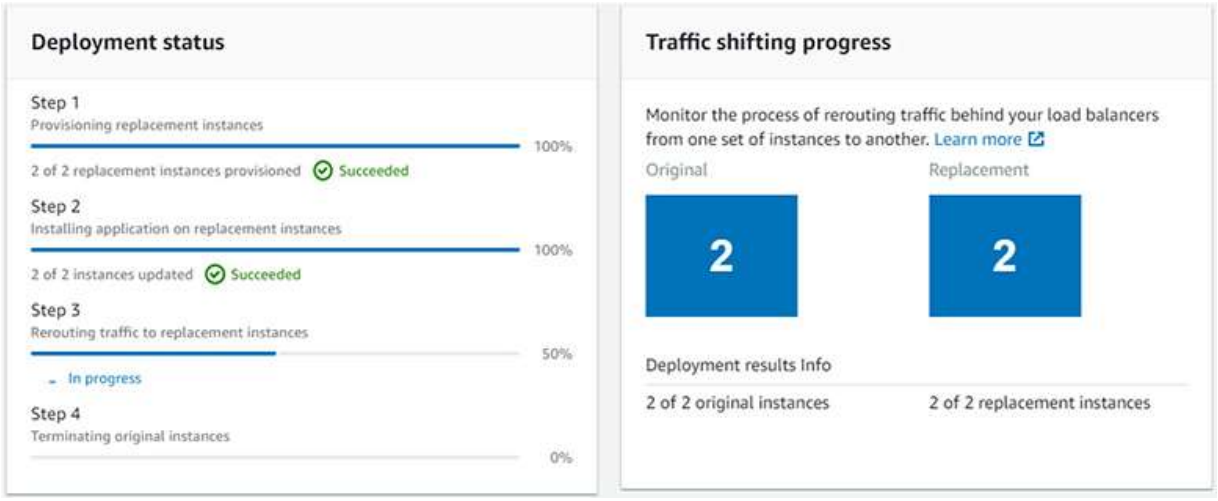


Within a few minutes the source stage and build stage have completed successfully. Now the deploy stage is in progress.

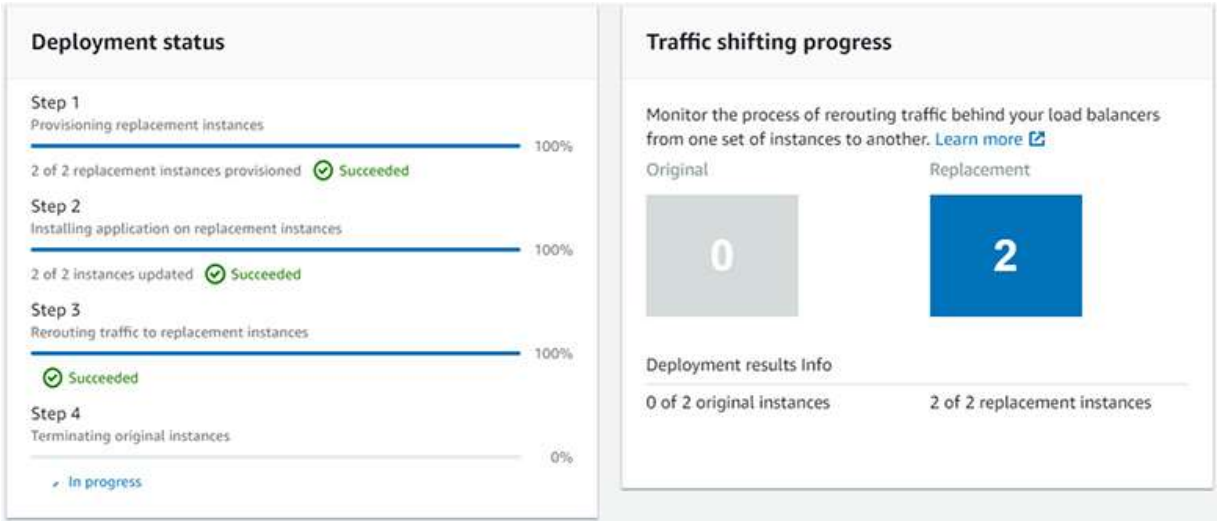


When you click Deploy Stage Details, you can visually see the traffic staging progress where the two instances are running in the original

deployment and another two instances are provisioned as replacement instances. Also, you can see the deployment status and verify that the provisioning replacement instances completed 100 percent and installing applications on replacement instances completed 100 percent. The rerouting traffic to replacement instances is halfway through, with 50 percent progress.

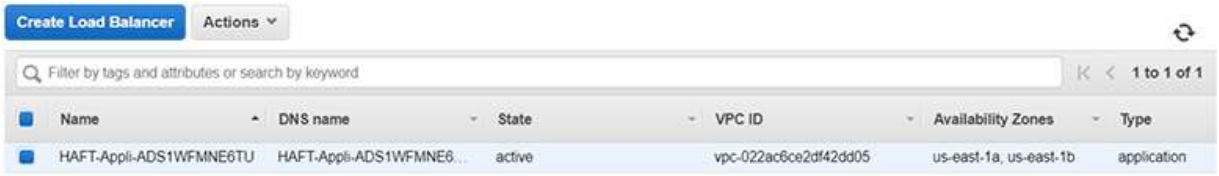


In a few more minutes, the rerouting traffic to replacement instances has completed 100 percent and terminating original instances is in progress. You can also view the traffic shifting progress is completed and moved to replacement instances.

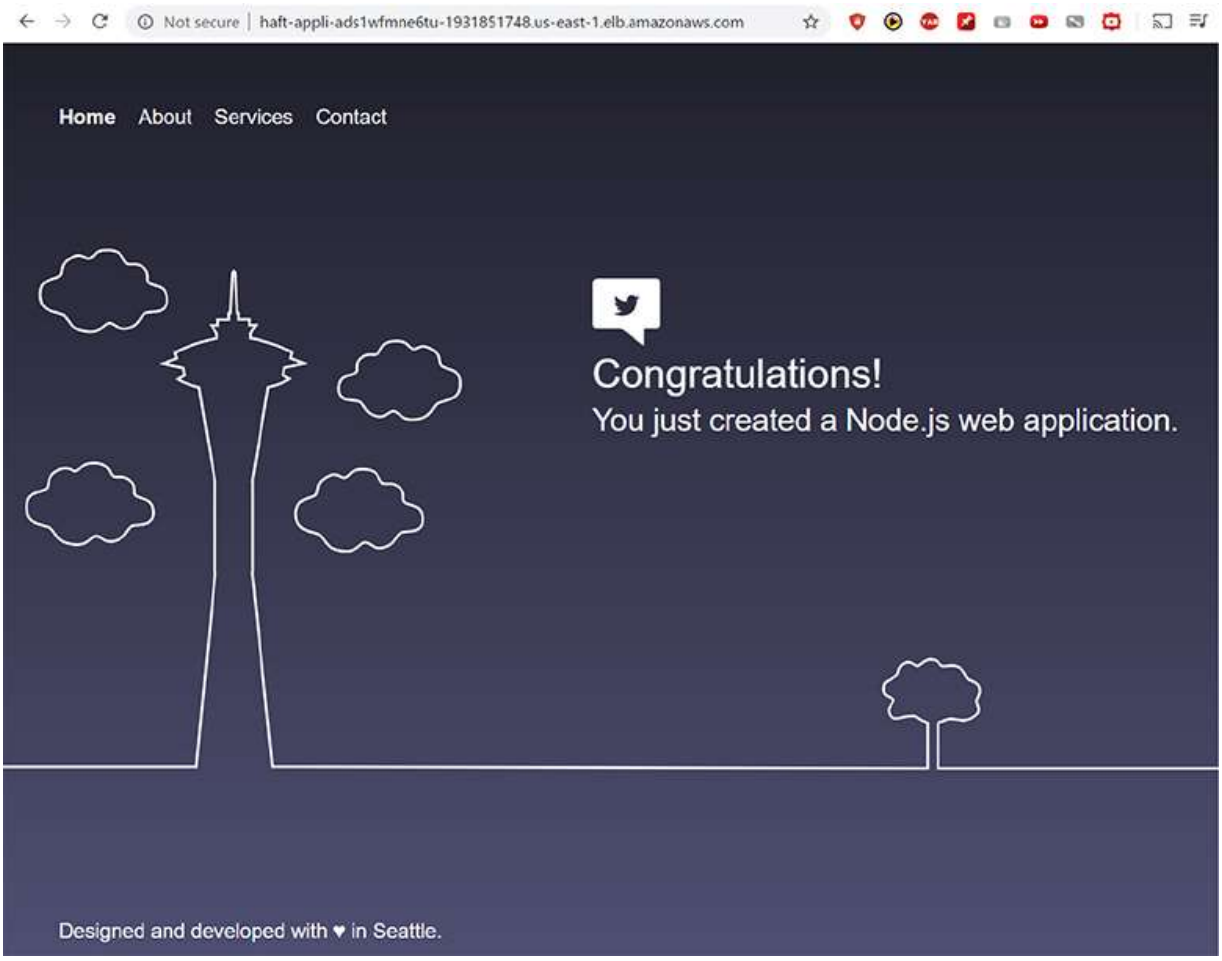


From your AWS Management Console, navigate to your ALB, which will be on the left navigation pane in the Amazon EC2 dashboard. Select your

ALB and note the DNS name.



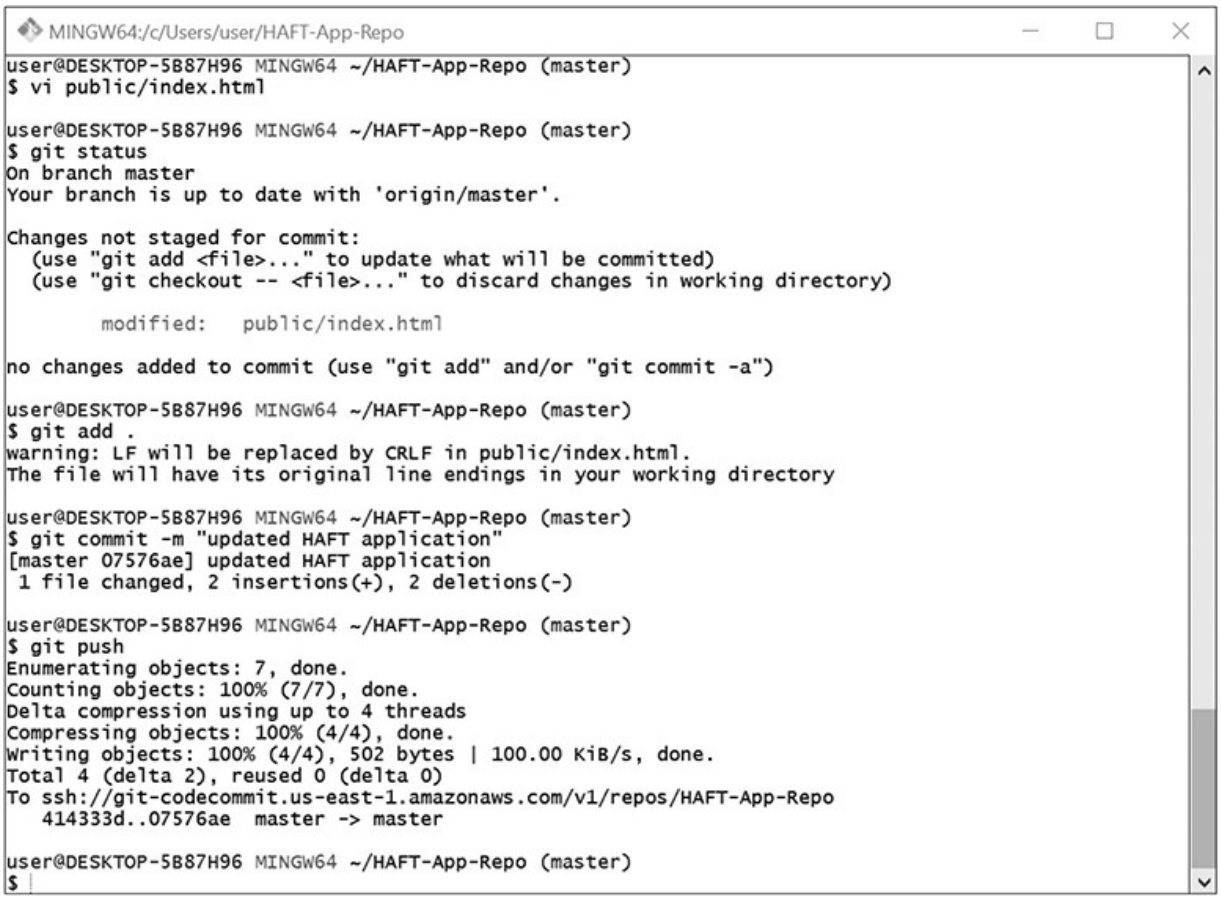
Now on your local machine, open a browser of your choice, enter the ALB DNS name, and press the ENTER key. Congratulations! You just created a scalable, highly available, fault-tolerant web application by using an automated software release pipeline.



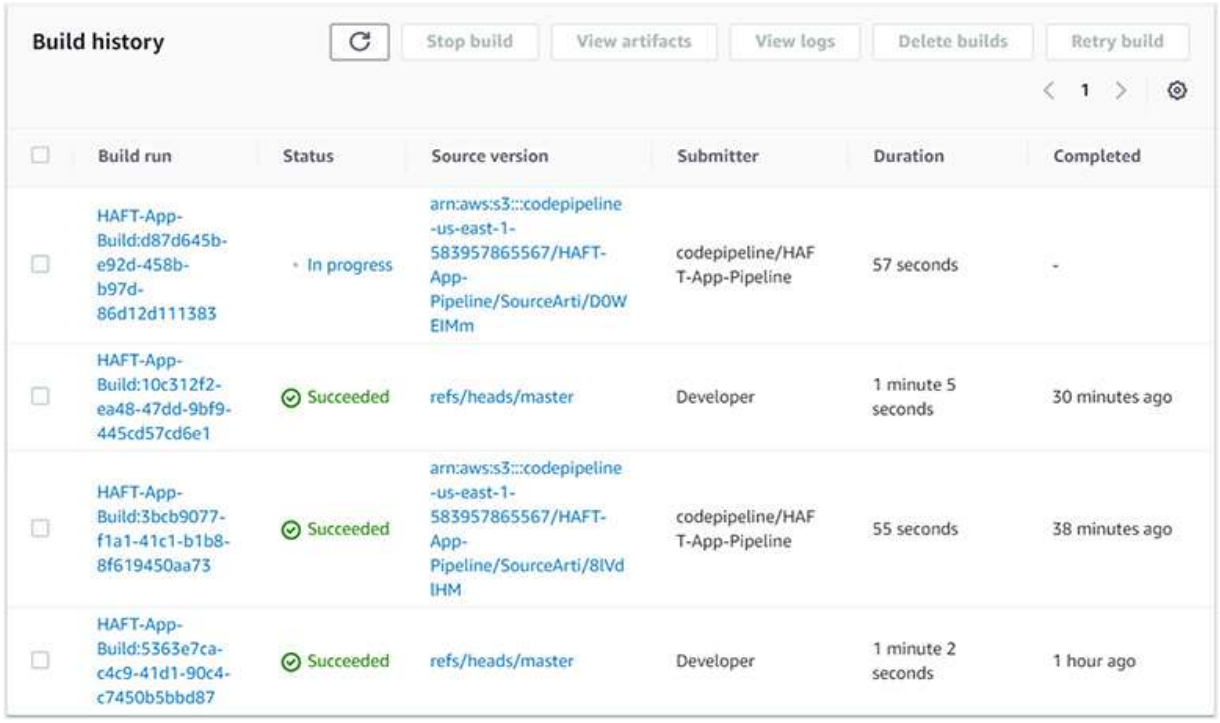
It is easy to add a new feature or update your source code and deploy it in production automatically using the CI/CD pipeline that you created using AWS developer tools. Go to your source code repository in your local

machine and edit index.html, which is stored in the public folder. For simplicity, change Congratulations to **Hello World!** and change the second line to **You just deployed HAFT Node.js web application V2 successfully!!**

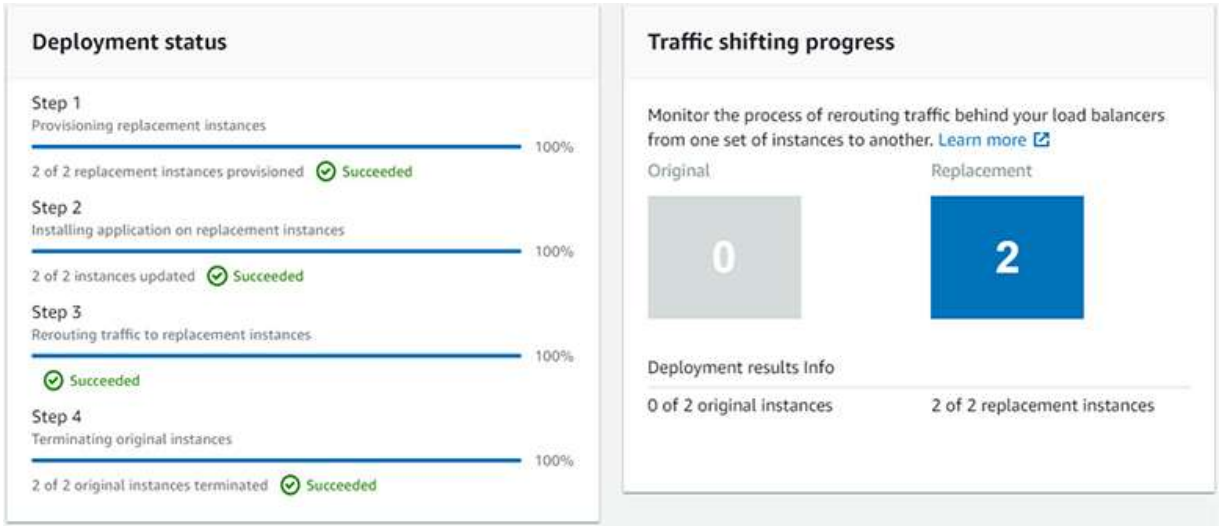
Now add the index file using the git add command and commit the change using the git commit command. Next push your source code changes to your remote repository in AWS CodeCommit using the git push command.



Navigate to AWS CodeBuild in the AWS Management Console and verify the build history. You can see the build process is in progress using the updated source code from AWS Commit.

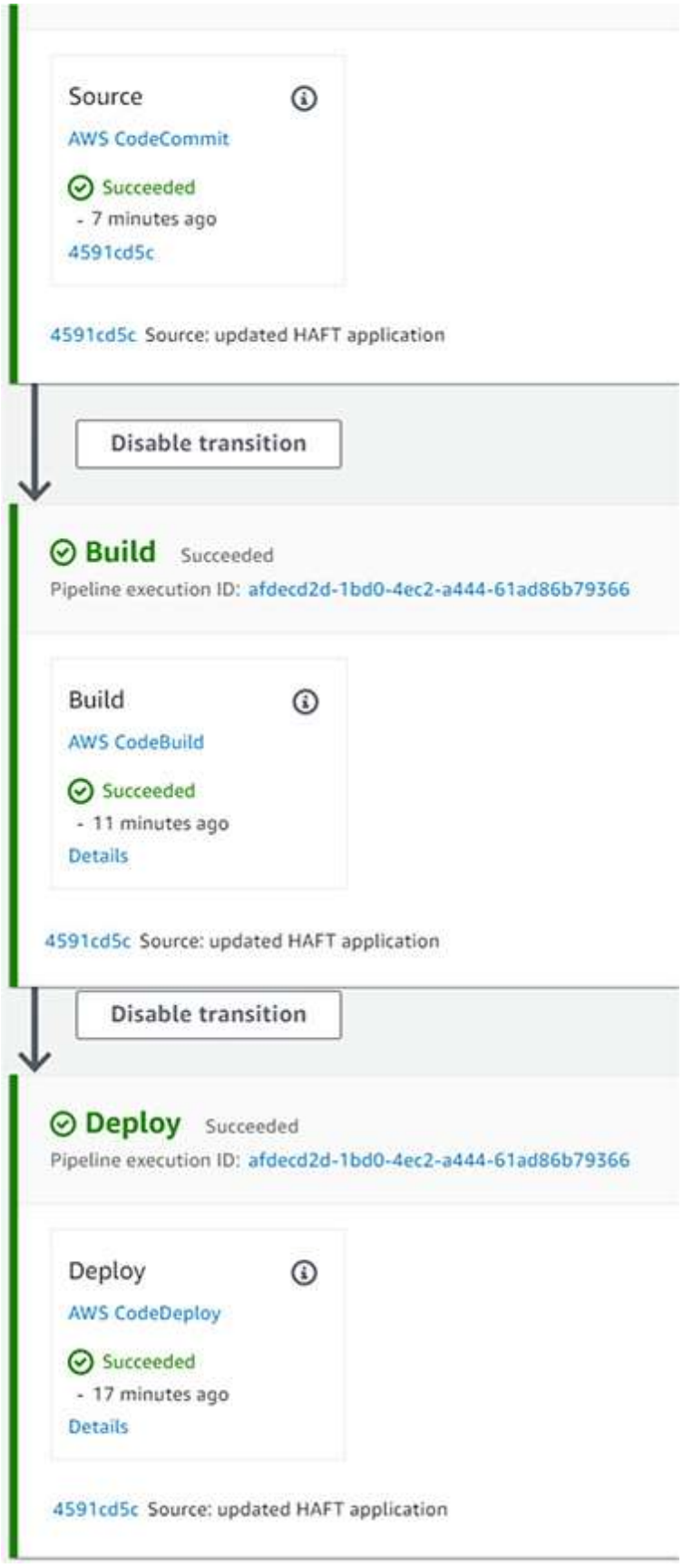


In few minutes, you can see that the traffic is shifted to new instances. Also, you can verify from the deployment status that all the steps are completed 100 percent successfully.

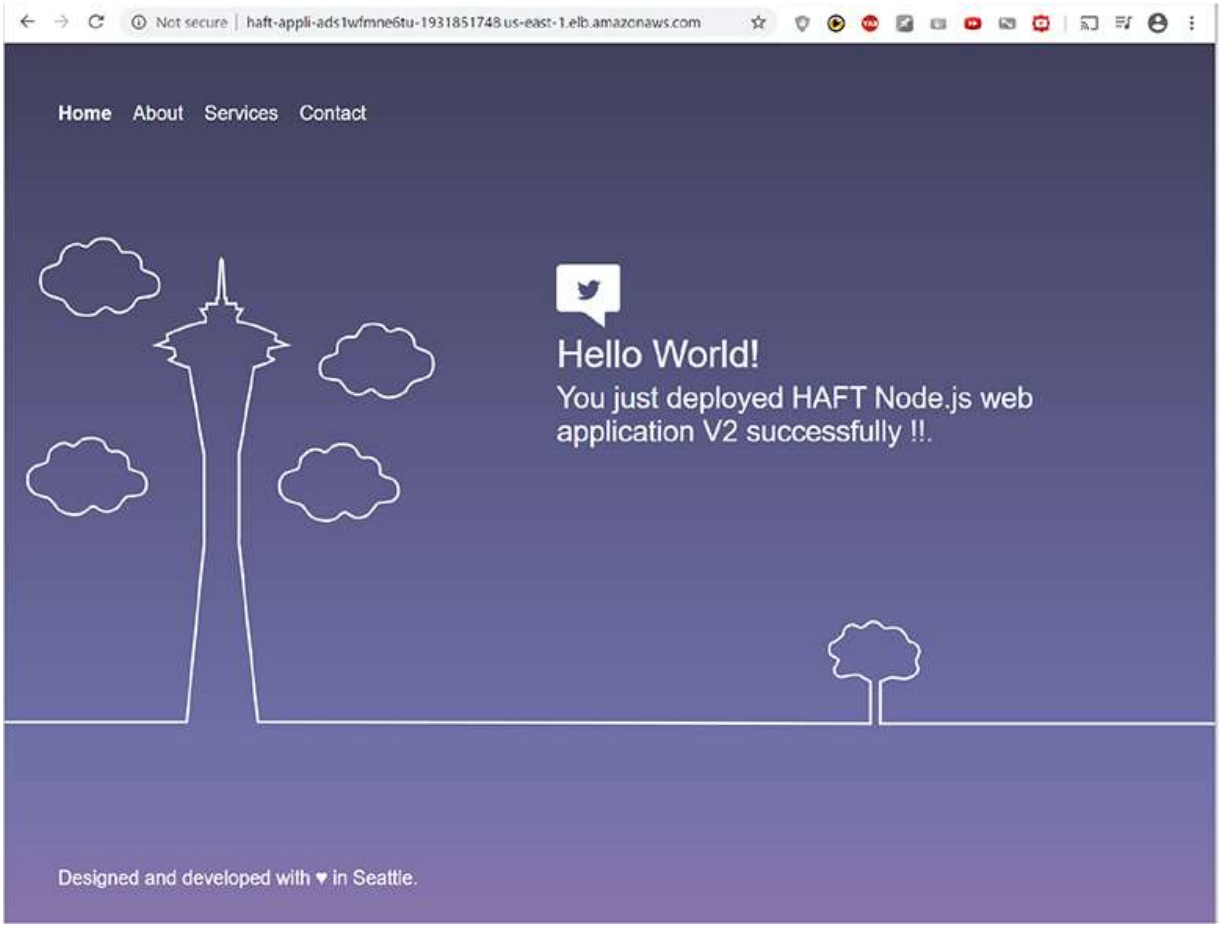


Next navigate to your CI/CD pipeline and verify that the updated HAFT application is in AWS CodeCommit. Next, you can see the build has succeeded using the updated HAFT application source. Finally, you can see

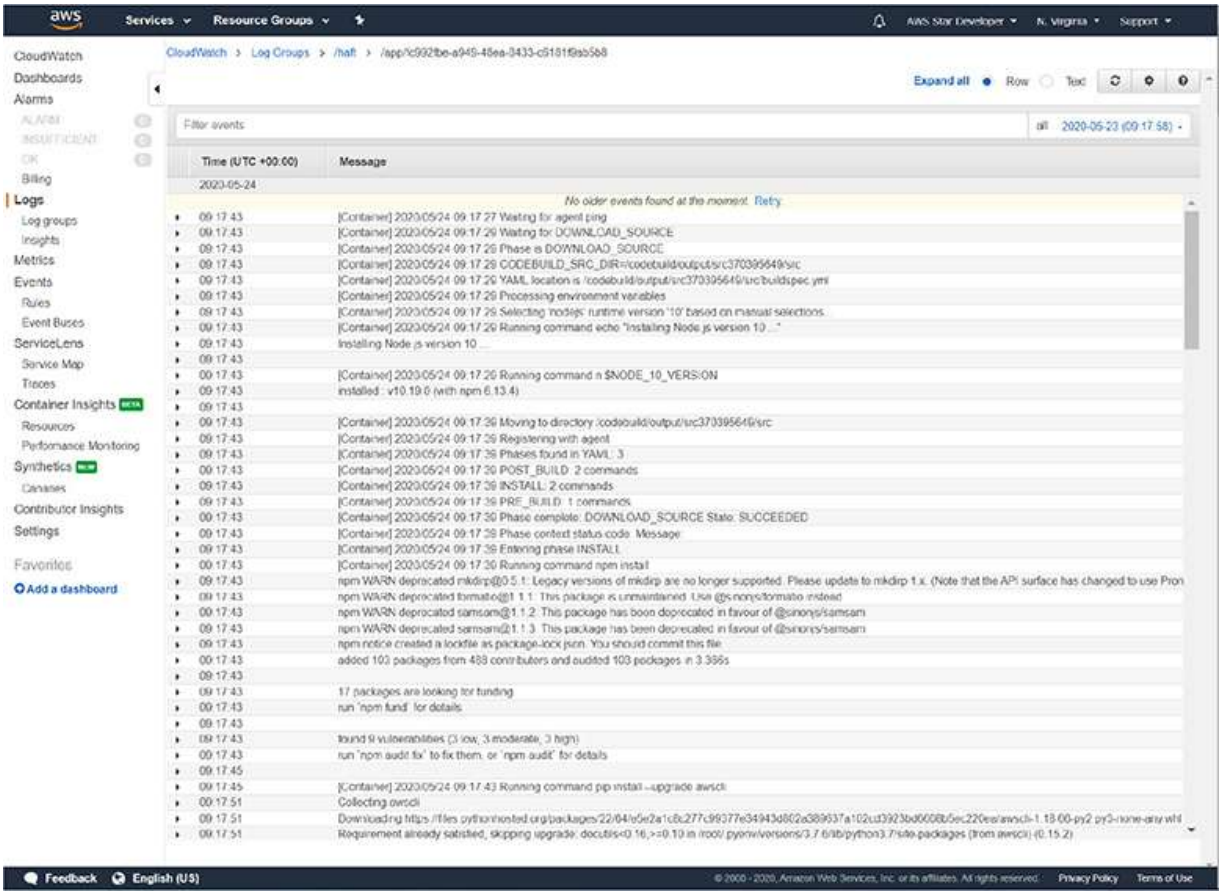
the successful deployment in the deploy stage using the updated HAFT application.


Source
AWS CodeCommit
Succeeded
- 7 minutes ago
4591cd5c
4591cd5c Source: updated HAFT application
Disable transition
Build Succeeded
Pipeline execution ID: afdecd2d-1bd0-4ec2-a444-61ad86b79366
Build
AWS CodeBuild
Succeeded
- 11 minutes ago
Details
4591cd5c Source: updated HAFT application
Disable transition
Deploy Succeeded
Pipeline execution ID: afdecd2d-1bd0-4ec2-a444-61ad86b79366
Deploy
AWS CodeDeploy
Succeeded
- 17 minutes ago
Details
4591cd5c Source: updated HAFT application

On your local machine, open a browser and enter the ALB URL that you noted down previously. When you press ENTER, you can see the updated application is displayed successfully.



Navigate to Amazon CloudWatch and click on Logs in the navigation pane. Choose Log Groups and click on the haft log group. You can verify all the messages that were created during the execution. This is helpful when troubleshooting any issues later to easily identify the root cause.



You have observed the demonstration of building a CI/CD pipeline using AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline. Using this CI/CD pipeline, you created a highly scalable, available, and fault-tolerant sample web application quickly. You also changed the application source code and observed from the AWS CodePipeline page the different stages of your software build, testing, and application deployment. This experience will help you in your certification exam, and you can practice in your test environment and possibly implement it in a real environment once you gain confidence. You can release new application software and additional features to your software rapidly by implementing continuous integration and continuous delivery using AWS developer tools, which accelerates your software development, testing, and release cycle.

Chapter Review

This chapter began by explaining what a CI/CD pipeline is and how it automates the entire software code build, automated testing, and deploying to a different environment. We then used AWS developer tools to build a CI/CD pipeline. The AWS CodeCommit service is used for storing sample application source code. We used AWS CodeBuild to build and test the web application code. We also used AWS CodeDeploy to automate code deployments using the blue/green deployment type. The chapter showed how to use AWS CodePipeline to automate the continuous delivery pipeline. You also learned how to host your application code, build, test, and automatically deploy applications in AWS Cloud in this chapter.

Exercises

The following exercises will help you practice using AWS development tools. You need to create an AWS account, as explained earlier, before performing these exercises. You can use the Free Tier when launching AWS resources, but make sure to terminate them at the end.

Exercise 26-1: Delete the AWS CodeCommit Repository Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CodeCommit console at https://console.aws.amazon.com/codecommit/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Repositories.
4. Select HAFT-App-Repo and choose Settings.
5. Navigate to Delete Repository on the General tab.
6. Choose Delete Repository.
7. Enter delete in the popup window and choose to delete.

Exercise 26-2: Delete the AWS CodeBuild Project Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Build Projects.
4. Choose the radio button next to HAFT-App-Build project.
5. Choose Delete to delete your build project.

Exercise 26-3: Delete the AWS CodeDeploy Application Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, expand Deploy. Then choose Applications.
4. Choose HAFT-Appln that you created in this chapter.
5. Click on the application name.
6. On the Application Details page, click Delete Application.
7. In response to the prompt, confirm that you want to delete.

Exercise 26-4: Delete the AWS CodePipeline Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Name.

4. Choose the HAFT-App-Pipeline that you created in this chapter.
5. Click Edit on the Details page.
6. Click Delete from Edit page.
7. In response to the prompt, confirm you want to delete the pipeline.

Exercise 26-5: Delete the AWS CloudFormation Stack Using the AWS Management Console

1. Use your AWS account e-mail address and password to sign in to the AWS account and then navigate to the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Verify the AWS region by using the Region selector in the upper-right corner of the page.
3. From the navigation pane on the left, choose Stacks. Select the stack that you created in this chapter.
4. From the Stack Details page, choose Delete
5. When prompted, choose Delete Stack to delete the entire stack.

Questions

The following questions will help you gauge your understanding of the contents in this chapter. Read all the answers carefully because there might be more than one correct answer. Choose the best responses for each question.

1. What four AWS developer tools can be used to build a CI/CD pipeline? (Choose four.)

   A. AWS CodeCommit

   B. AWS CodeBuild

   C. AWS CodeDeploy

   D. AWS CodePipeline

2. Which of the following AWS developer tools provides a fully managed solution to host your application source code by creating a code repository and uses git commands to interact with your code repository?

A. AWS CLI

B. AWS Cloud9

C. AWS CodeCommit

D. AWS CodePipeline

3. Which of the following AWS developer tools provides the integrated development environment (IDE) for creating, executing, and debugging your source code from the AWS CodeCommit repository?

A. Amazon Corretto

B. AWS CDK

C. AWS CLI

D. AWS Cloud9

4. Which of the following AWS developer tools can be used to build your application and test your source code with continuous scaling?

A. AWS Cloud9

B. AWS CodeDeploy

C. AWS CodeBuild

D. AWS X-Ray

5. Which of the following AWS developer tools can be used to automate your application code deployment and maintain your application uptime using blue/green deployment?

A. AWS CodeCommit

B. AWS CodeBuild

C. AWS CodeDeploy

D. AWS Device Farm

6. Which of the following AWS developer tools can be used to automate the software delivery using automated continuous delivery pipelines and also used for fast and reliable software code updates by visually monitoring the flow?

A. AWS CodeCommit

B. AWS Cloud9

C. AWS CodeDeploy

D. AWS CodePipeline

7. Your company is planning to deploy an application to the production environment frequently using AWS CodeDeploy. The application, which is hosted on Amazon EC2, needs to be always available even during application code change implementations. How you can meet your company's requirement?

   A. Use in-place deployment

   B. Use blue/green deployment

   C. Use rolling deployment

   D. Use canary deployment

8. Your company uses the AWS Lambda platform and uses AWS CodeDeploy to deploy serverless applications to production. They want to direct the traffic in increments to the new AWS Lambda functions instead of directing all the traffic at once. How can you meet this requirement?

   A. Use an all-at-once deployment configuration

   B. Use a linear deployment configuration

   C. Use a canary deployment configuration

   D. Use a rolling deployment configuration

9. A company wants to remove the manual process in their software delivery process and use an automated process end to end from development, build, and deployment into production. They want to utilize AWS services to build the pipeline, and they use GitHub as their source code repository. What services stack do they need to build the CI/CD pipeline?

   A. GitHub, AWS CodeBuild, AWS CodeDeploy, AWS CodePipeline

   B. GitHub, AWS CodeBuild, AWS CodeDeploy, AWS CodeCommit

   C. AWS Cloud9, AWS CodeBuild, AWS CodeDeploy, AWS CodePipeline

   D. AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, AWS Cloud9

10. A company has enabled all the available options to store logs in AWS CodeDeploy. What are the two destinations where the logs will be stored? (Choose two.)

    A. Amazon CloudWatch logs

    B. Amazon EC2

    C. Amazon S3

    D. Amazon RDS

Answers

1. **A, B, C, D.** You can use AWS CodeCommit to store the code repository, AWS CodeBuild to build your application, AWS CodeDeploy to automatically deploy your application, and AWS CodePipeline to build the CI/CD pipeline.
2. **C.** AWS CodeCommit provides a fully managed service to host your application source code by creating a code repository, and you can use git commands to interact with your code repository.
3. **D.** AWS Cloud9 provides the IDE for creating, executing, and debugging your source code from the AWS CodeCommit repository.
4. **C.** AWS CodeBuild can be used to build your application and test your source code with continuous scaling.
5. **C.** AWS CodeBuild can be used to automate your application code deployment and maintain your application uptime using blue/green deployment.
6. **D.** AWS CodePipeline can be used to automate the software delivery using automated continuous delivery pipelines.
7. **B.** Use blue/green deployment when the application needs to be always available even during application code change implementations.
8. **C.** Use a canary deployment configuration to direct the traffic in increments to the new AWS Lambda functions.
9. **A.** You can use GitHub, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline to build the pipeline.

10. **A, C.** You can use Amazon CloudWatch logs and Amazon S3 as log destinations.

Additional Resources

- **Blue/Green Deployment** This reference deployment guide has a quick-start guide with an architecture diagram and scripts to quickly deploy the CI/CD pipeline using AWS CodePipeline.
  `https://aws.amazon.com/quickstart/architecture/blue-green-deployment/`
- **Automate Infrastructure Deployments** This blog demonstrates how to use AWS CodePipeline to orchestrate the end-to-end infrastructure and application deployment.
  `https://aws.amazon.com/blogs/devops/bluegreen-infrastructure-application-deployment-blog/`
- **Build Containers with AWS CodePipeline** This blog explains the approach to building Windows Server containers using the CI/CD pipeline.
  `https://aws.amazon.com/blogs/devops/building-windows-containers-with-aws-codepipeline-and-custom-actions/`
- **Continuous Kubernetes Deployment** This blog explains the steps to automatically build a Kubernetes cluster using AWS CodeCommit, AWS CodeBuild, and AWS CodePipeline.
  `https://aws.amazon.com/blogs/devops/continuous-deployment-to-kubernetes-using-aws-codepipeline-aws-codecommit-aws-codebuild-amazon-ecr-and-aws-lambda/`