PROJECT REPORT

CI/CD pipeline for .NET Serverless Application on the AWS Cloud

TRAINING STREAM: CLOUD-AWS
TRAINING PARTNER: RPS CONSULTANCY
BATCH: 12

Team Members:

Shivangi
Sanskar Gupta
Utkarsh Yashvardhan
Indrani Sengupta
Vishal Subhash Bhadak

Project Description:

What we'll Build:

This Quick Start builds a continuous integration and delivery (CI/CD) pipeline for serverless .NET applications on the Amazon Web Services (AWS) Cloud. This scalable deployment helps you deliver features and updates rapidly and reliably. You have no build servers to manage, and you pay only for what you use. It deploys an AWS CodeCommit repository with a sample AWS Lambda application defined in an AWS Serverless Application Model (AWS SAM) template. Pushing changes to modify the application invokes AWS CodePipeline. The pipeline builds and deploys the application using AWS CloudFormation and exposes it to the internet with Amazon API Gateway.

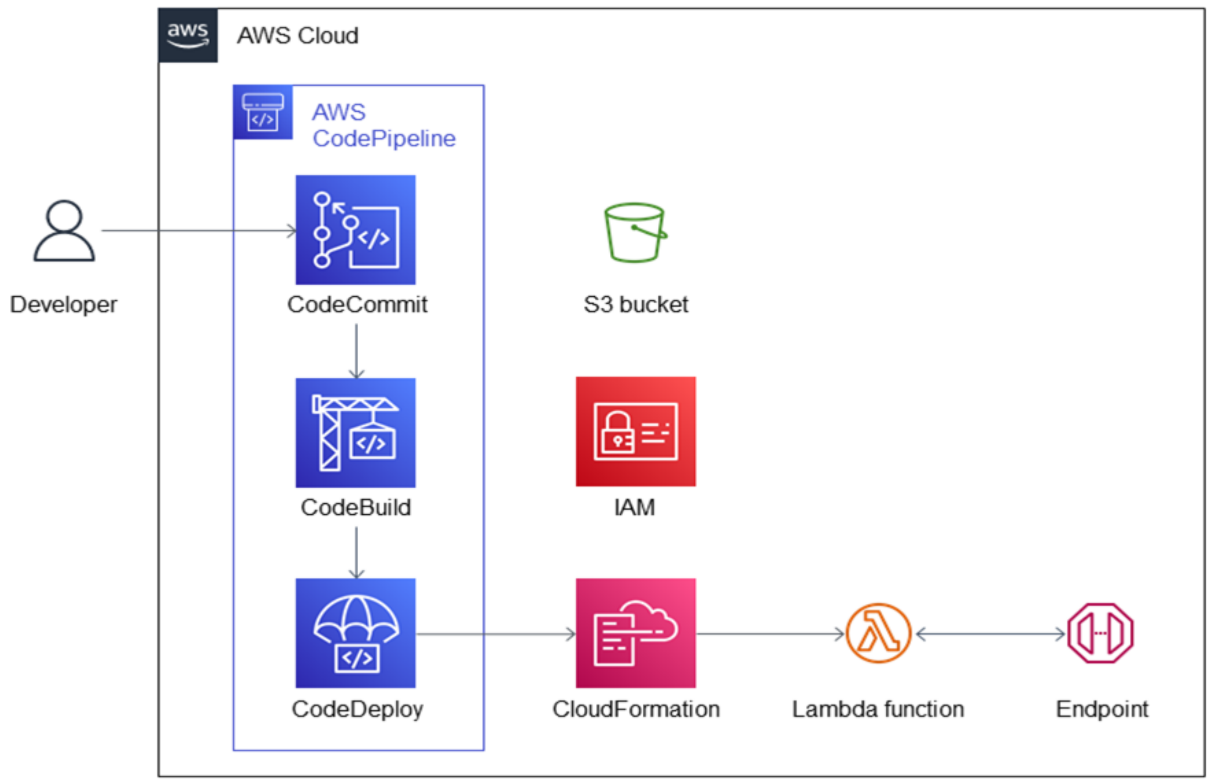
This Quick Start sets up the following:

- CodePipeline to create a pipeline with source, build, and deploy stages. The pipeline is invoked when a developer commits a code change to the CodeCommit repository.
  - A CodeCommit repository to host the application source code.
  - AWS CodeBuild to generate a CloudFormation template from the source code.
  - AWS CodeDeploy to deploy the template.
- An Amazon Simple Storage Service (Amazon S3) bucket to store pipeline artifacts.
- AWS Identification and Access Management (IAM) for IAM roles used by CodeBuild, CodeDeploy, and CloudFormation.
- CloudFormation to provision and configure the AWS Lambda application.
- An AWS Lambda function to run the application.
- API Gateway endpoint to expose the application to the internet.

How to deploy:

1.Sign in to your AWS account. If you don't have an account, sign up at https://aws.amazon.com.
2.Launch the Quick Start. Before you create the stack, choose the Region from the top toolbar.
3.Test the deployment.

Architecture:



- CodePipeline to create a pipeline with source, build, and deploy stages. The pipeline is invoked when a developer commits a code change to the CodeCommit repository.
    - A CodeCommit repository to host the application source code.
    - AWS CodeBuild to generate an AWS CloudFormation template from the source code.
    - AWS CodeDeploy to deploy the template.
- CloudFormation to provision and configure the AWS Lambda application.

- An AWS Lambda function to run the code.
- API Gateway endpoint to expose the application to the internet.
- An Amazon Simple Storage Service (Amazon S3) bucket to store pipeline artifacts.
- AWS Identification and Access Management (IAM) for IAM roles used by CodeBuild, CodeDeploy, and CloudFormation.

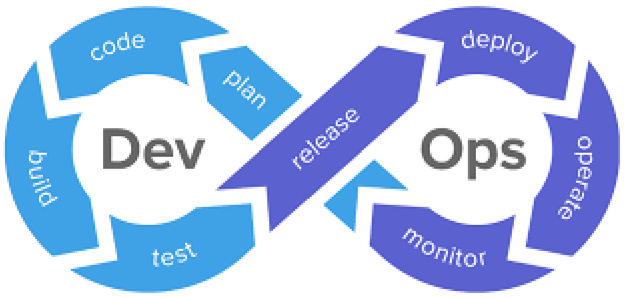
What is AWS?

AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon that includes a mixture of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings. AWS services can offer an organization tools such as compute power, database storage and content delivery services.

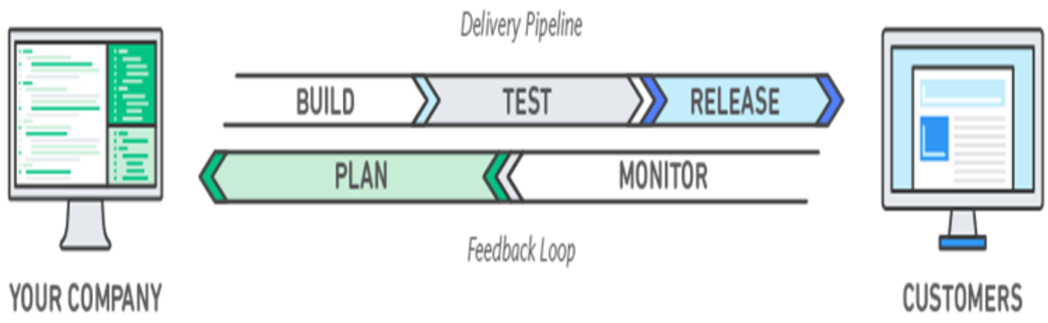
AWS launched in 2006 from the internal infrastructure that Amazon.com built to handle its online retail operations. AWS was one of the first companies to introduce a pay-as-you-go cloud computing model that scales to provide users with compute, storage or throughput as needed.

AWS offers many different tools and solutions for enterprises and software developers that can be used in data centers in up to 190 countries. Groups such as government agencies, education institutions, nonprofits and private organizations can use AWS services.

What Is DevOps?



DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.



How DevOps Works?

Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.
In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.
These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team’s velocity.

Benefits of DevOps

- Speed

Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results. For example, microservices and continuous delivery let teams take ownership of services and then release updates to them quicker.

- Rapid Delivery

Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs and build competitive advantage. Continuous integration and continuous delivery are practices that automate the software release process, from build to deploy.

- Reliability

Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like continuous integration and continuous delivery to test that each change is functional and safe. Monitoring and logging practices help you stay informed of performance in real-time.

- Scale

Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

- Improved Collaboration

Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

- Security

Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and policy as code, you can define and then track compliance at scale.

CI/CD :

CI/CD falls under DevOps (the joining of development and operations) and combines the practices of continuous integration and continuous delivery. CI/CD automates much or all of the manual human intervention traditionally needed to get new code from a commit into production such as build, test, and deploy, as well as infrastructure provisioning. With a CI/CD pipeline, developers can make changes to code that are then automatically tested and pushed out for delivery and deployment. Get CI/CD right and downtime is minimized and code releases happen faster.

Continuous integration (CI)

Continuous integration is the practice of integrating all your code changes into the main branch of a shared source code repository early and often, automatically testing each change when you commit or merge them, and automatically kicking off a build. With continuous integration, errors and security issues can be identified and fixed more easily, and much earlier in the software development lifecycle.

By merging changes frequently and triggering automatic testing and validation processes, you minimize the possibility of code conflict, even with multiple developers working on the same application. A secondary advantage is that you don't have to wait long for answers and can, if necessary, fix bugs and security issues while the topic is still fresh in your mind.

Common code validation processes start with a static code analysis that verifies the quality of the code. Once the code passes the static tests, automated CI routines package and compile the code for further automated testing. CI processes should have a version control system that tracks changes, so you know the version of the code used.

Continuous Delivery (CD)

Continuous delivery is a software development practice that works in conjunction with continuous integration to automate the infrastructure provisioning and application release process.

Once code has been tested and built as part of the CI process, continuous delivery takes over during the final stages to ensure it can be deployed's packaged with everything it needs to deploy to any environment at any time. Continuous delivery can cover everything from provisioning the infrastructure to deploying the application to the testing or production environment.

With continuous delivery, the software is built so that it can be deployed to production at any time. Then you can trigger the deployments manually or move to continuous deployment where deployments are automated as well.

Continuous Deployment

Continuous deployment enables organizations to automatically deploy their applications – eliminating the need for human intervention. With continuous deployment, DevOps teams set the criteria for code releases ahead of time and when those criteria are met and validated, the code is deployed into the production environment. Thanks to this type of automation, organizations are able to be more nimble and get new features into the hands of users faster.

CI/CD fundamentals

There are eight fundamental elements of CI/CD that help ensure maximum efficiency for your development lifecycle. They span development and deployment. Include these fundamentals in your pipeline to improve your DevOps workflow and software delivery:

1. **A single source repository**
   Source code management (SCM) that houses all necessary files and scripts to create builds. The repository should contain everything needed for the build. This includes source code, database structure, libraries, properties files and version control. It should also contain test scripts and scripts to build applications.
2. **Frequent check-ins to main branch**
   Integrating code in your trunk, mainline or master branch — i.e. trunk-based development — early and often. Avoid sub-branches and work with the main branch only. Use small segments of code and merge them into the branch as frequently as possible. Don't merge more than one change at a time.

3. **Automated builds**
   Scripts should include everything you need to build from a single command. This includes web server files, database scripts and application software. The CI processes should automatically package and compile the code into a usable application.
4. **Self-testing builds**
   Testing scripts should ensure that the failure of a test results in a failed build. Use static pre-build testing scripts to check code for integrity, quality and security compliance. Only allow code that passes static tests into the build.
5. **Frequent iterations**
   Multiple commits to the repository results in fewer places for conflicts to hide. Make small, frequent iterations rather than major changes. By doing this, it's possible to roll changes back easily if there's a problem or conflict.
6. **Stable testing environments**
   Code should be tested in a cloned version of the production environment. You can't test new code in the live production version. Create a cloned environment that's as close as possible to the real environment. Use rigorous testing scripts to detect and identify bugs that slipped through the initial pre-build testing process.
7. **Maximum visibility**
   Every developer should be able to access the latest executables and see any changes made to the repository. Information in the repository should be visible to all. Use version control to manage handoffs, so developers know which is the latest version. Maximum visibility means everyone can monitor progress and identify potential concerns.
8. **Predictable deployments anytime**
   Deployments are so routine and low-risk that the team’s comfortable doing them anytime. CI/CD testing and verification processes should be rigorous and reliable. This gives the team confidence to deploy updates at any time. Frequent deployments incorporating limited changes also pose lower risks and can be easily rolled back.

The benefits of CI/CD implementation

Companies and organizations that adopt CI/CD tend to notice a lot of positive changes. Here are some of the benefits you can look forward to as you implement CI/CD:

- **Happier users and customers:** Fewer bugs and errors make it into production, so your users and customers have a better experience. This leads to improved levels of customer satisfaction, confidence and reputation.
- **Accelerated time-to-value:** When you can deploy anytime, you can bring products and new features to market faster. Your development costs are lower, and a faster turnaround frees your team for other work. Customers get results faster and gain a competitive edge.
- **Less fire fighting:** Testing code more often, in smaller batches, and earlier in the development cycle can seriously cut down on fire drills. This results in a smoother development cycle and less team stress. Results are more predictable, and it's easier to find and fix bugs.
- **Hit dates more reliably:** Removing deployment bottlenecks and making deployments predictable can remove a lot of the uncertainty around hitting key dates. Breaking work into smaller, manageable bites means it's easier to complete each stage on time and track progress. This approach gives plenty of time to monitor overall progress and determine completion dates more accurately.
- **Free up developers' time:** With more of the deployment process automated, the team has time for more rewarding projects. It's estimated that developers spend between 35% and 50% of their time testing, validating and debugging code. By automating these processes, developers significantly improve their productivity.
- **Less context switching:** Getting real-time feedback on the code developers commit makes it easier to work on one thing at a time and minimize cognitive load. By working with small sections of code that are

automatically tested, developers can debug code quickly while their minds are still fresh from programming. Finding bugs is easier because there's less code to review.

- **Reduce burnout:** Research shows that continuous delivery measurably reduces deployment pain and team burnout. Developers experience less frustration and strain when working with CI/-CD processes. This directly leads to happier and healthier employees and less burnout.
- **Recover faster:** CI/CD makes it easier to fix issues and recover from incidents (MTTR). Continuous deployment practices mean frequent small software updates so when bugs appear, it's easier to pin them down. Developers have the option of fixing bugs quickly or rolling back the change so that the customer can get back to work quickly.

CI/CD Pipeline:

The CI/CD Pipeline is built using AWS CodePipeline. Thay are the top-level component of continuous integration, delivery, and deployment. Pipelines are a collection of jobs that are divided by stages.A pipeline is a process that drives software development through a path of building, testing, and deploying code, also known as CI/CD. By automating the process, the objective is to minimize human error and maintain a consistent process for how software is released. Tools that are included in the pipeline could include compiling code, unit tests, code analysis, security, and binaries creation. For containerized environments, this pipeline would also include packaging the code into a container image to be deployed across a hybrid cloud.
CI/CD is the backbone of a DevOps methodology, bringing developers and IT operations teams together to deploy software. As custom applications become key to how companies differentiate, the rate at which code can be released has become a competitive differentiator.

Benefits of CI/CD pipeline:

- Software Release Process can be Automated
- Increased in the Productivity of Developers
- Improving Code Quality
- Faster Updates Delivery
- View Progress
- View Pipeline

Services Used:

AWS CodePipeline:

AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously. For information about pricing for CodePipeline.

You can use CodePipeline to help you automatically build, test, and deploy your applications in the cloud. Specifically, you can:

- Automate your release processes: CodePipeline fully automates your release process from end to end, starting from your source repository through build, test, and deployment.

- You can prevent changes from moving through a pipeline by including a manual approval action in any stage except a Source stage.

- You can release when you want, in the way you want, on the systems of your choice, across one instance or multiple instances.

- Establish a consistent release process: Define a consistent set of steps for every code change.

- CodePipeline runs each stage of your release according to your criteria.

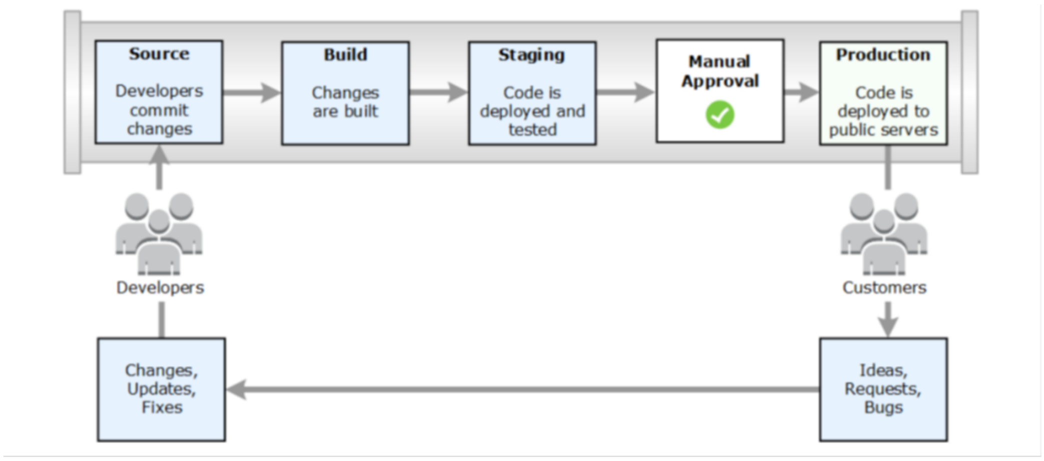
- Speed up delivery while improving quality: You can automate your release process to allow your developers to test and release code incrementally and speed up the release of new features to your customers.

- Use your favorite tools: You can incorporate your existing source, build, and deployment tools into your pipeline. For a full list of AWS services and third-party tools currently supported by CodePipeline,

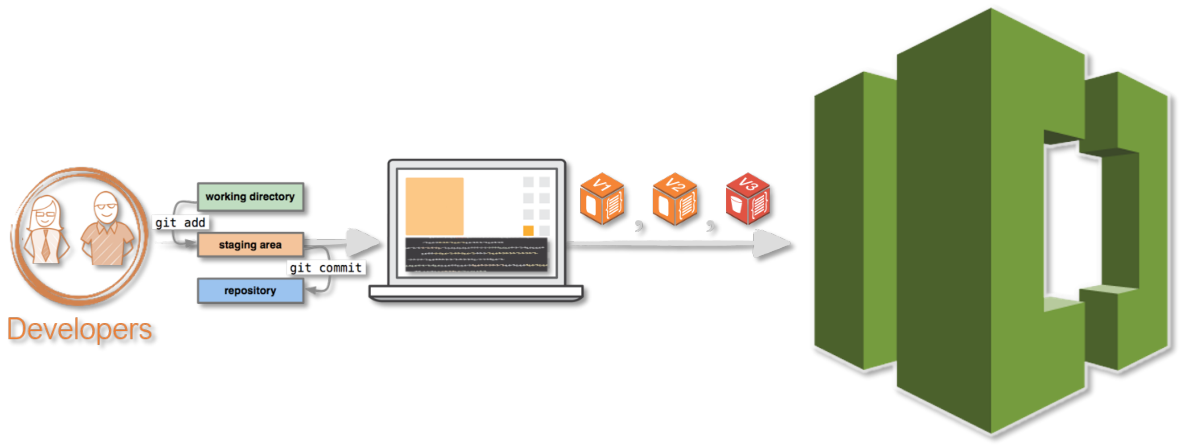
- View progress at a glance: You can review real-time status of your pipelines, check the details of any alerts, retry failed actions, view details about the source revisions used in the latest pipeline execution in each stage, and manually rerun any pipeline.

- View pipeline history details: You can view details about executions of a pipeline, including start and end times, run duration, and execution IDs.

The following diagram shows an example release process using CodePipeline:



CodeCommit:



CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure. You can use CodeCommit to store anything from code to binaries. It supports the standard functionality of Git, so it works seamlessly with your existing Git-based tools.AWS CodeCommit is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets (such as documents, source code, and binary files) in the cloud.

With CodeCommit, you can:

- Benefit from a fully managed service hosted by AWS. CodeCommit provides high service availability and durability and eliminates the administrative overhead of managing your own hardware and software. There is no hardware to provision and scale and no server software to install, configure, and update.
- Store your code securely. CodeCommit repositories are encrypted at rest as well as in transit.
- Work collaboratively on code. CodeCommit repositories support pull requests, where users can review and comment on each other's code changes before merging them to branches; notifications that automatically send emails to users about pull requests and comments; and more.
- Easily scale your version control projects. CodeCommit repositories can scale up to meet your development needs. The service can handle repositories with large numbers of files or branches, large file sizes, and lengthy revision stories.

Code Build:

AWS CodeBuild

AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy. CodeBuild eliminates the need to provision, manage, and scale your own build servers. It provides prepackaged build environments for popular programming languages and build tools such as Apache Maven, Gradle, and more. You can also customize build environments in CodeBuild to use your own build tools. CodeBuild scales automatically to meet peak build requests.

CodeBuild provides these benefits:

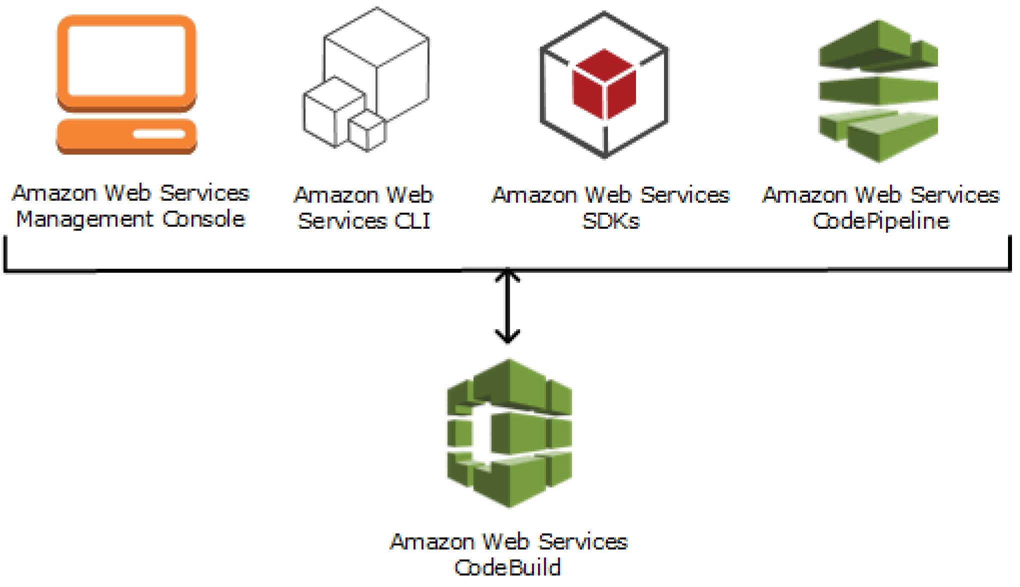
- Fully managed – CodeBuild eliminates the need to set up, patch, update, and manage your own build servers.

- On demand – CodeBuild scales on demand to meet your build needs. You pay only for the number of build minutes you consume.

- Out of the box – CodeBuild provides preconfigured build environments for the most popular programming languages. All you need to do is point to your build script to start your first build.

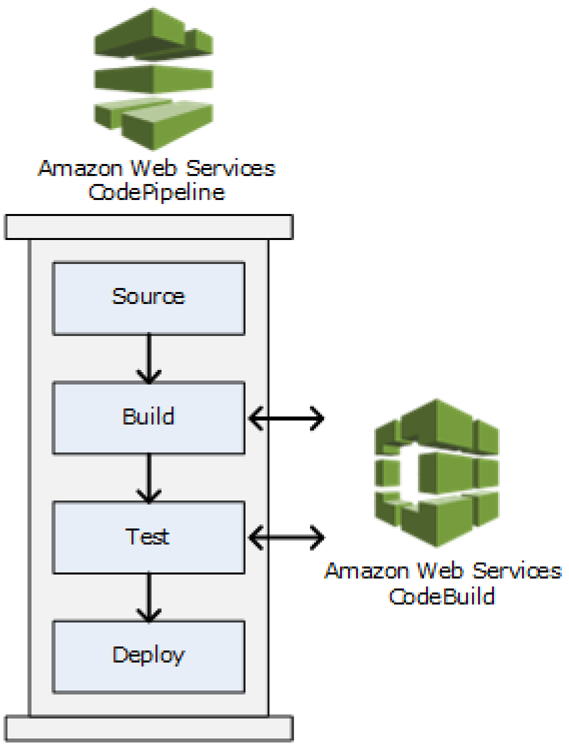
How to run CodeBuild

You can use the AWS CodeBuild or AWS CodePipeline console to run CodeBuild. You can also automate the running of CodeBuild by using the AWS Command Line Interface (AWS CLI) or the AWS SDKs.


Amazon Web Services Management Console
Amazon Web Services CLI
Amazon Web Services SDKs
Amazon Web Services CodePipeline
Amazon Web Services CodeBuild

To run CodeBuild by using the CodeBuild console, AWS CLI, or AWS SDKs, see Run AWS CodeBuild directly.

As the following diagram shows, you can add CodeBuild as a build or test action to the build or test stage of a pipeline in AWS CodePipeline. AWS CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your code. This includes building your code. A *pipeline* is a workflow construct that describes how code changes go through a release process.


Amazon Web Services CodePipeline
Source
Build
Test
Deploy
Amazon Web Services CodeBuild

To use CodePipeline to create a pipeline and then add a CodeBuild build or test action, see Use CodePipeline with CodeBuild. For more information about CodePipeline, see the AWS CodePipeline User Guide.

The CodeBuild console also provides a way to quickly search for your resources, such as repositories, build projects, deployment applications, and pipelines. Choose Go to resource or press the / key, and then enter the name of the resource. Any matches appear in the list. Searches are case insensitive. You only see resources that you have permissions to view. For more information, see Viewing resources in the console

Code Deploy:

AWS CodeDeploy

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premises servers. AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications. You can use AWS CodeDeploy to automate software deployments, eliminating the need for error-prone manual operations. The service scales to match your deployment needs.

You can deploy a nearly unlimited variety of application content, including:

- Code
- Serverless AWS Lambda functions

- Web and configuration files
- Executables
- Packages
- Scripts
- Multimedia files

AWS CodeDeploy Features

AWS CodeDeploy is a service that automates application deployments to a variety of compute services including Amazon EC2, AWS Fargate, AWS Lambda, and on-premises instances. CodeDeploy fully automates your application deployments eliminating the need for manual operations. CodeDeploy protects your application from downtime during deployments through rolling updates and deployment health tracking. CodeDeploy gives you centralized control of your deployments through the AWS Management Console, CLI, SDKs, or APIs allowing you to launch, control, and monitor your deployments. You can view the deployment progress down to individual setup events. CodeDeploy tracks and stores the recent history of your deployments, so you can investigate the timeline and change history of past deployments.

AWS CodeDeploy is platform and language agnostic and works with any application, so you can reuse your existing setup code. You can also easily integrate your application deployments with your existing software delivery process or into a continuous delivery toolchain by using the CodeDeploy APIs.

S3 Bucket:

Amazon Simple Storage Service is an object storage service that offers
industry-leading
scalability, data availability, security, and performance. Customers of all sizes and
industries can use
Amazon S3 to store and protect any amount of data for a range of use cases, such
as data lakes,
websites, mobile applications, backup and restore, archive, enterprise applications,
IoT devices, and
big data analytics. Amazon S3 provides management features so that you can
optimize, organize,
and configure access to your data to meet your specific business, organizational,
and compliance Requirements.

Amazon S3 offers a range of storage classes designed for different use cases. For example, you can store mission-critical production data in S3 Standard for frequent access, save costs by storing infrequently accessed data in S3 Standard-IA or S3 One Zone-IA, and archive data at the lowest costs in S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive.

You can store data with changing or unknown access patterns in S3 Intelligent-Tiering, which optimizes storage costs by automatically moving your data between four access tiers when your access patterns change. These four access tiers include two low-latency access tiers optimized for frequent and infrequent access, and two opt-in archive access tiers designed for asynchronous access for rarely accessed data.

For more information, see Using Amazon S3 storage classes. For more information about S3 Glacier Flexible Retrieval, see the *Amazon S3 Glacier Developer Guide*.

IAM(AWS Identity and Access Management):

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

AWS CloudFormation:

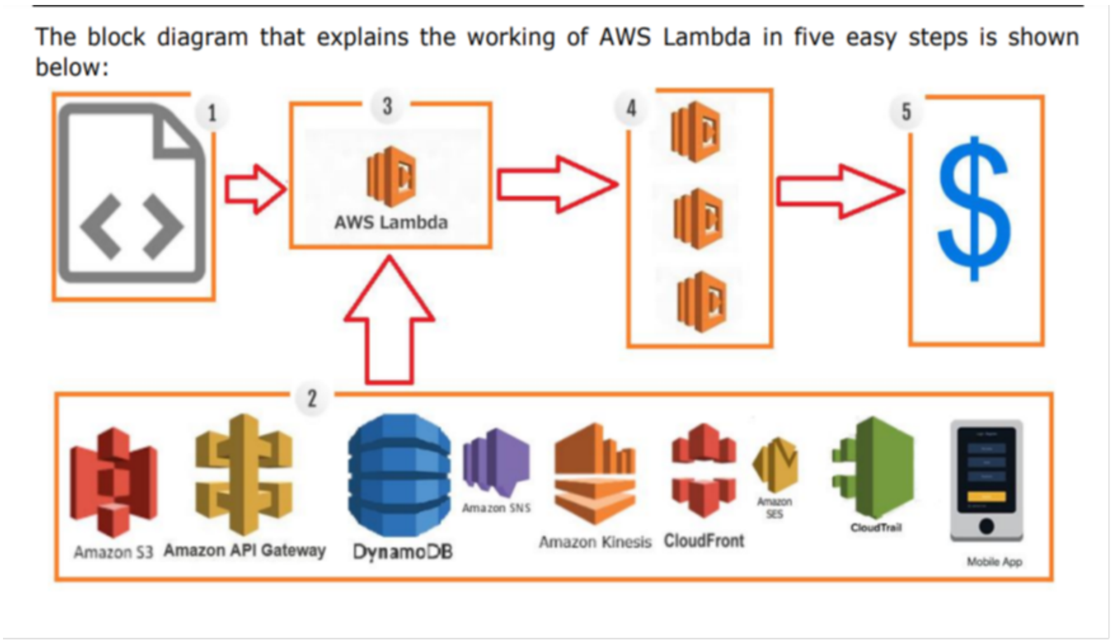
AWS CloudFormation

AWS CloudFormation is a service that gives developers and businesses an easy way to create a collection of related AWS and third-party resources, and provision and manage them in an orderly and predictable fashion.
A CloudFormation template consists of 6 sections – Description, Parameters, Mappings, Conditions, Resources and Outputs.
Developers can deploy and update compute, database, and many other resources in a simple, declarative style that abstracts away the complexity of specific resource APIs. AWS CloudFormation is designed to allow resource lifecycles to be managed repeatably, predictable, and safely, while allowing for automatic rollbacks, automated state management, and management of resources across accounts and regions. Recent enhancements and options allow for multiple ways to create resources, including using AWS CDK for coding in higher-level languages, importing existing resources, detecting configuration drift, and a new Registry that makes it easier to create custom types that inherit many core CloudFormation benefits.

Lambda Function:

AWS Lambda

AWS Lambda is a service which performs serverless computing, which involves computing without any server. The code is executed based on the response of events in AWS services such as adding/removing files in S3 bucket, updating Amazon dynamo dB tables, HTTP request from Amazon API gateway etc.
To get working with AWS Lambda, we just have to push the code in AWS Lambda service.
All other tasks and resources such as infrastructure, operating system, maintenance of server, code monitoring, logs and security is taken care by AWS.
AWS Lambda supports languages such as Java, NodeJS, Python, C# and Go. Note that AWS Lambda will work only with AWS services.


The block diagram that explains the working of AWS Lambda in five easy steps is shown below:

API Gateway:

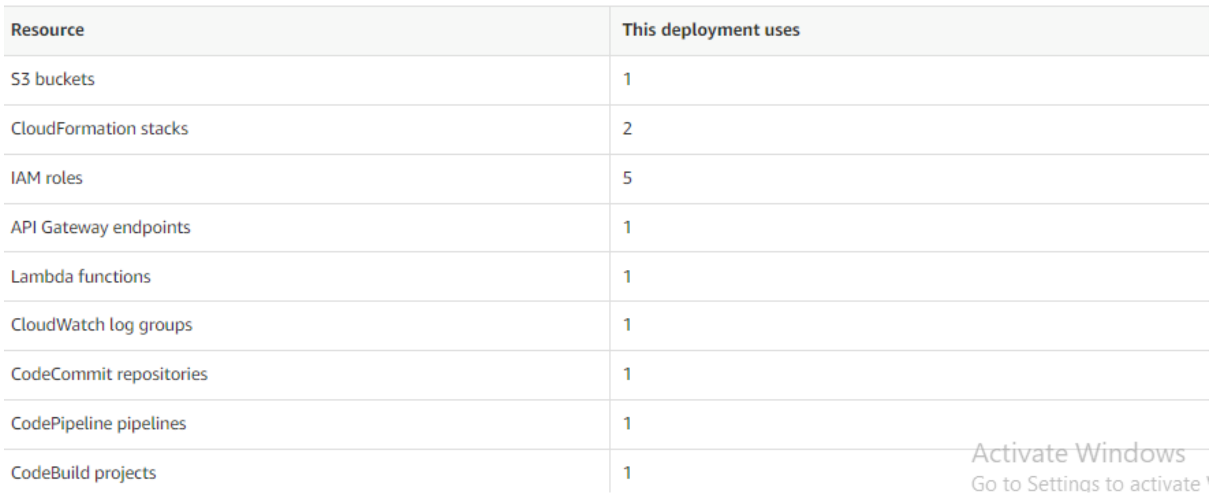
Amazon
API Gateway

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.

API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, CORS support, authorization and access control, throttling, monitoring, and API version management. API Gateway has no minimum fees or startup costs. You pay for the API calls you receive and the amount of data transferred out and, with the API Gateway tiered pricing model, you can reduce your cost as your API usage scales.

Technical Requirements:

- Resource quotas



Resource	This deployment uses
S3 buckets	1
CloudFormation stacks	2
IAM roles	5
API Gateway endpoints	1
Lambda functions	1
CloudWatch log groups	1
CodeCommit repositories	1
CodePipeline pipelines	1
CodeBuild projects	1

If necessary, request service quota increases for the following resources. You might need to request increases if your existing deployment currently uses these resources and if this Quick Start deployment could result in exceeding the default quotas. The Service Quotas console displays your usage and quotas for some aspects of some services.

- Supported AWS Regions

For any Quick Start to work in a Region other than its default Region, all the services it deploys must be supported in that Region. You can launch a Quick Start in any Region and see if it works. If you get an error such as “Unrecognized resource type,” the Quick Start is not supported in that Region.

- ## IAM permissions

Before launching the Quick Start, you must sign in to the AWS Management Console with IAM permissions for the resources that the templates deploy. The *AdministratorAccess* managed policy within IAM provides sufficient permissions, although your organization may choose to use a custom policy with more restrictions.

- ## Deployment options

This Quick Start uses AWS SAM to deploy a serverless application into an AWS account. As part of this deployment, it will create resources such as IAM roles, an API Gateway, and a Lambda function.

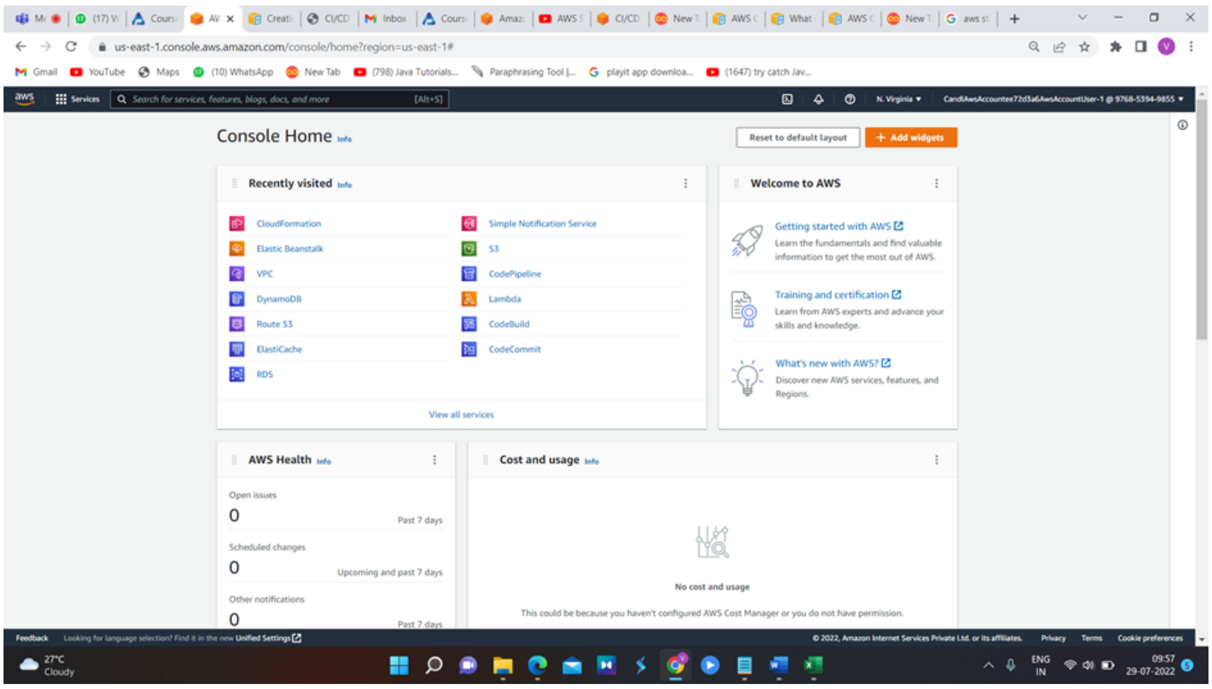
Procedure:

Step1:

AWS MANAGEMENT CONSOLE

Console home:

The AWS Management Console is a web application that comprises and refers to a board collection of services consoles for managing AWS resources.



Learn the fundamentals and find valuable information to get the most out of AWS.

Learn from AWS experts and advance your skills and knowledge.

Discover new AWS services, features, and Regions.

This could be because you haven't configured AWS Cost Manager or you do not have permission.

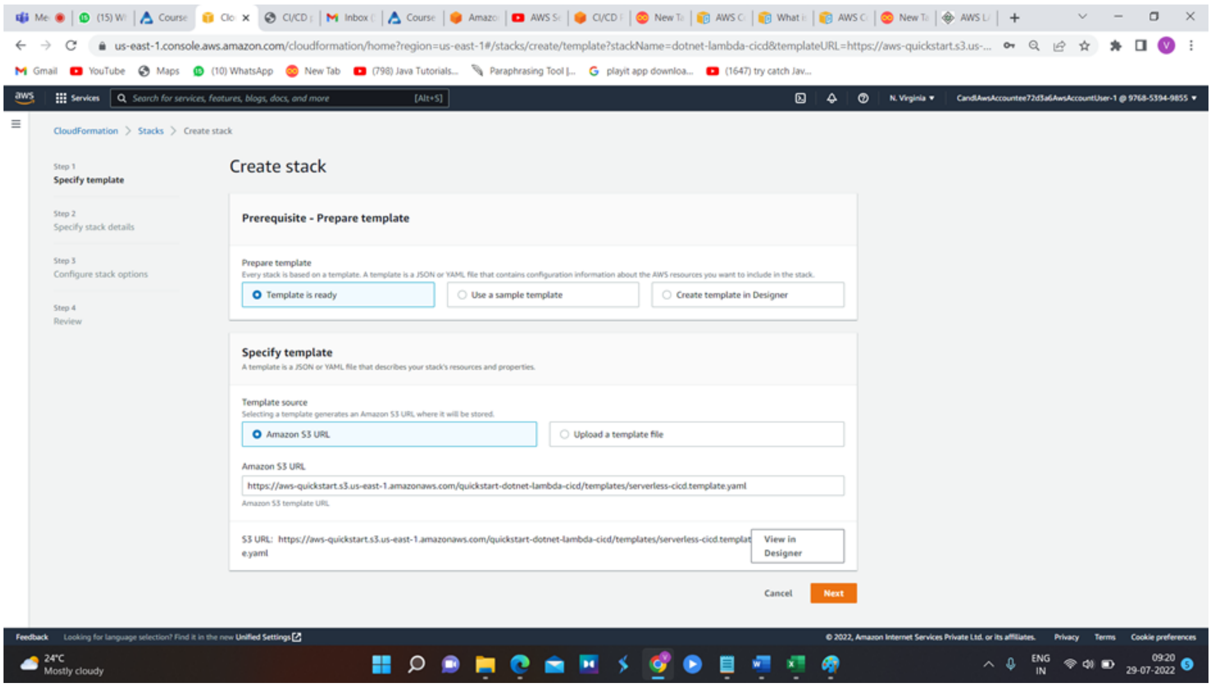
We have to open the sign in page, select root user and sign in using account root user credentials. It also lets us customise the console home experience by adding, removing, and rearranging widgets such as recently visited. By using the management console we can launch various services such as creating instances using Amazon EC2 or creating buckets. Users need to remember only one identity to sign in to AD and AWS Management console. AWS Management helps customers to manage their cloud storage, cloud computing and other resources.

Step 2:

Create Stack :

Creating a stack on the AWS CloudFormation console.

On the Create stack page, keep the default setting for the template URL, and then choose Next.



CloudFormation > Stacks > Create stack

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Create stack

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

[✓ Template is ready] Use a sample template Create template in Designer

Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

[✓ Amazon S3 URL] Upload a template file

Amazon S3 URL
https://aws-quickstart.s3.us-east-1.amazonaws.com/quickstart-dotnet-lambda-cicd/templates/serverless-cicd.template.yaml
Amazon S3 template URL

S3 URL: https://aws-quickstart.s3.us-east-1.amazonaws.com/quickstart-dotnet-lambda-cicd/templates/serverless-cicd.template.yaml

The aws cloudformation list-stacks command enables you to get a list of any of the stacks you have created. You can use an option to filter results by stack status, such as CREATE_COMPLETE and DELETE_COMPLETE. The aws cloudformation list-stacks command returns summary information about any of your running or deleted stacks, including the name, stack identifier, template, and status.

We have to select the proper Prerequisite - Prepare template, In that there are 3 options which are- Template is ready, Use a sample template , Create template in Designer.

Selecting a template generates an Amazon S3 URL where it will be stored.

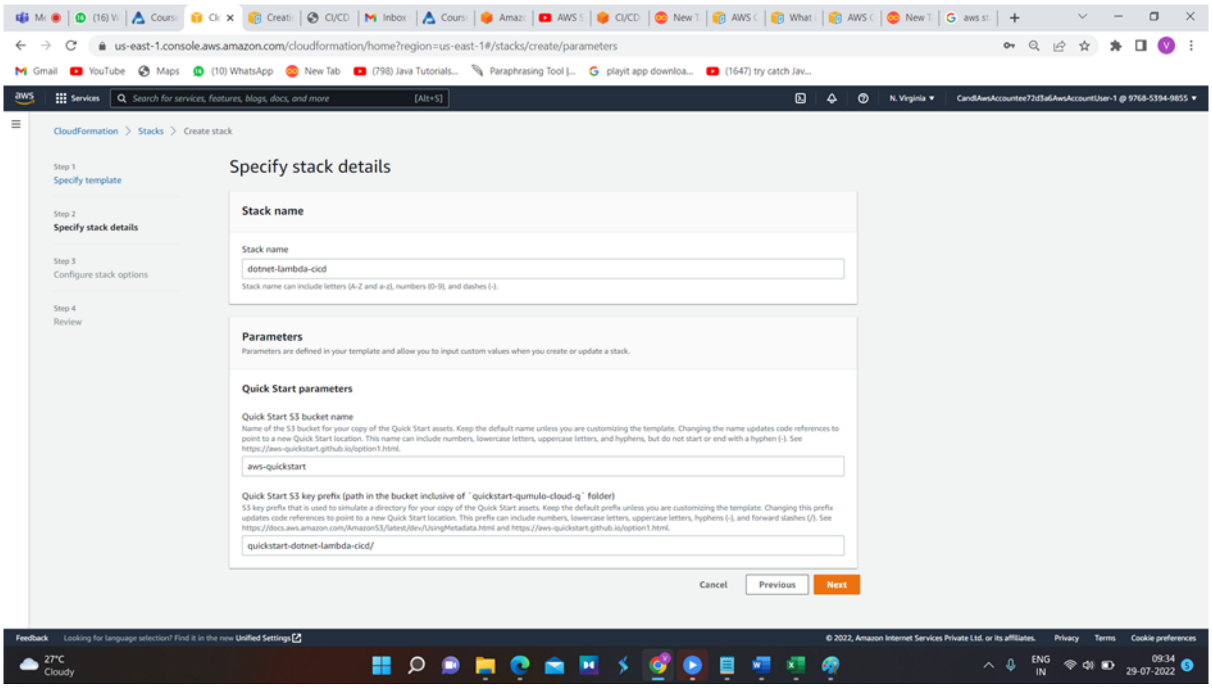
Select Amazon S3 URL.

Upload a template file

Step 3:

Specify stack details:

We have given the name to Stack.



Specify stack details

Stack name

Stack name

dotnet-lambda-cicd

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Quick Start parameters

Quick Start S3 bucket name

Name of the S3 bucket for your copy of the Quick Start assets. Keep the default name unless you are customizing the template. Changing the name updates code references to point to a new Quick Start location. This name can include numbers, lowercase letters, uppercase letters, and hyphens, but do not start or end with a hyphen (-). See https://aws-quickstart.github.io/option1.html.

aws-quickstart

Quick Start S3 key prefix (path in the bucket inclusive of `quickstart-qumulo-cloud-q` folder)

S3 key prefix that is used to simulate a directory for your copy of the Quick Start assets. Keep the default prefix unless you are customizing the template. Changing this prefix updates code references to point to a new Quick Start location. This prefix can include numbers, lowercase letters, uppercase letters, hyphens (-), and forward slashes (/). See https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMetadata.html and https://aws-quickstart.github.io/option1.html.

quickstart-dotnet-lambda-cicd/

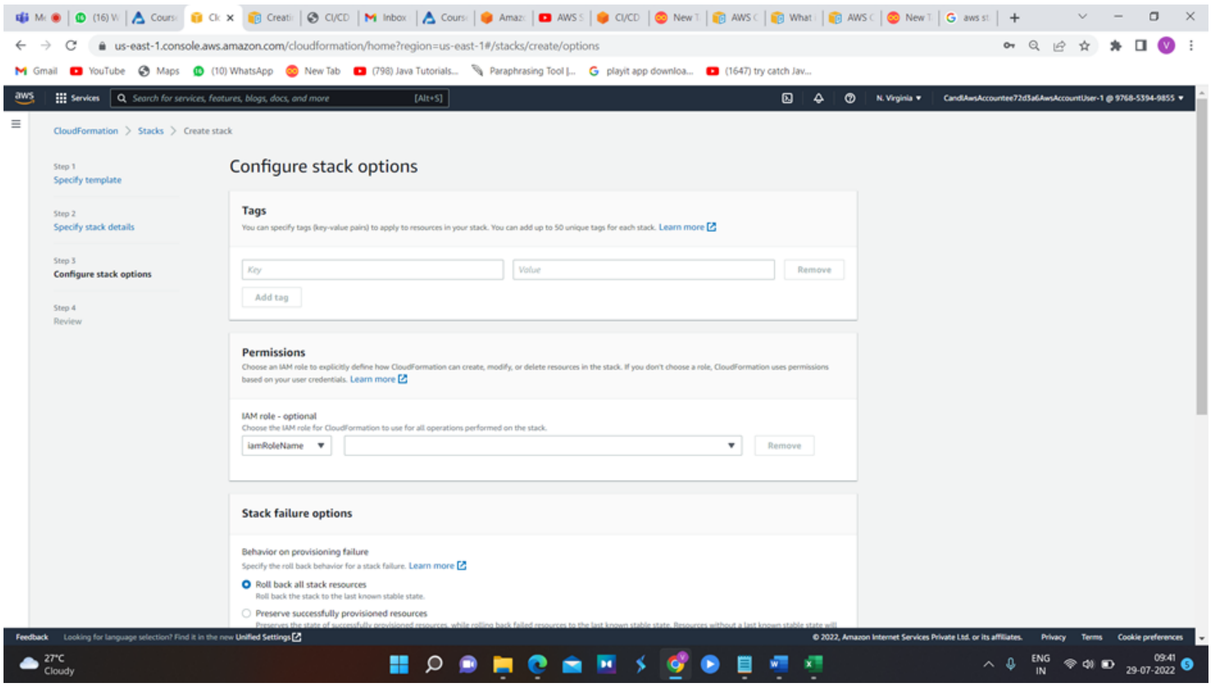
us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/create/parameters
CloudFormation > Stacks > Create stack
Step 1 Specify template
Step 2 Specify stack details
Step 3 Configure stack options
Step 4 Review
Cancel
Previous
Next

Giving proper name to Quick Start S3 bucket & then select Quick Start S3 key prefix. We can track the status of the resources AWS CloudFormation is creating and deleting with the aws cloudformation describe-stack-events command. The amount of time to create or delete a stack depends on the complexity of your stack.

On this Specify stack details page, change the stack name if needed. Review the parameters for the template. Provide values for the parameters that require input. For all other parameters, review the default settings and customise them as necessary. For details on each parameter, see the Parameter reference section of this guide. When you finish reviewing and customising the parameters, choose Next.

Step 4:

Configure Stack Option:
AWS CloudFormation stores the template we use to create our stack as part of the stack. To retrieve the template from CloudFormation use the aws cloudformation get-template command.


CloudFormation > Stacks > Create stack

Step 1 Specify template
Step 2 Specify stack details
Step 3 Configure stack options
Step 4 Review

Configure stack options

Tags
You can specify tags (key-value pairs) to apply to resources in your stack. You can add up to 50 unique tags for each stack. Learn more

Key | Value | Remove
Add tag

Permissions
Choose an IAM role to explicitly define how CloudFormation can create, modify, or delete resources in the stack. If you don't choose a role, CloudFormation uses permissions based on your user credentials. Learn more

IAM role - optional
Choose the IAM role for CloudFormation to use for all operations performed on the stack.
iamRoleName | Remove

Stack failure options
Behavior on provisioning failure
Specify the roll back behavior for a stack failure. Learn more
[✓ Roll back all stack resources]
Roll back the stack to the last known stable state.
Preserve successfully provisioned resources
Preserves the state of successfully provisioned resources, while rolling back failed resources to the last known stable state. Resources without a last known stable state will

You can specify tags to apply to resources in your stack. You can add up to 50 unique tags for each stack.
Choose an IAM role to explicitly define how CloudFormation can create, modify, or delete resources in the stack. If you don't choose a role, CloudFormation uses permissions based on your user credentials.
Choose the IAM role for CloudFormation to use for all operations performed on the stack.
Select Stack failure options.
If you want to modify just the parameters or settings of a stack (like a stack's Amazon SNS topic), you can reuse the existing stack template. You don't need to get a copy of the stack template or make modifications to the stack template.

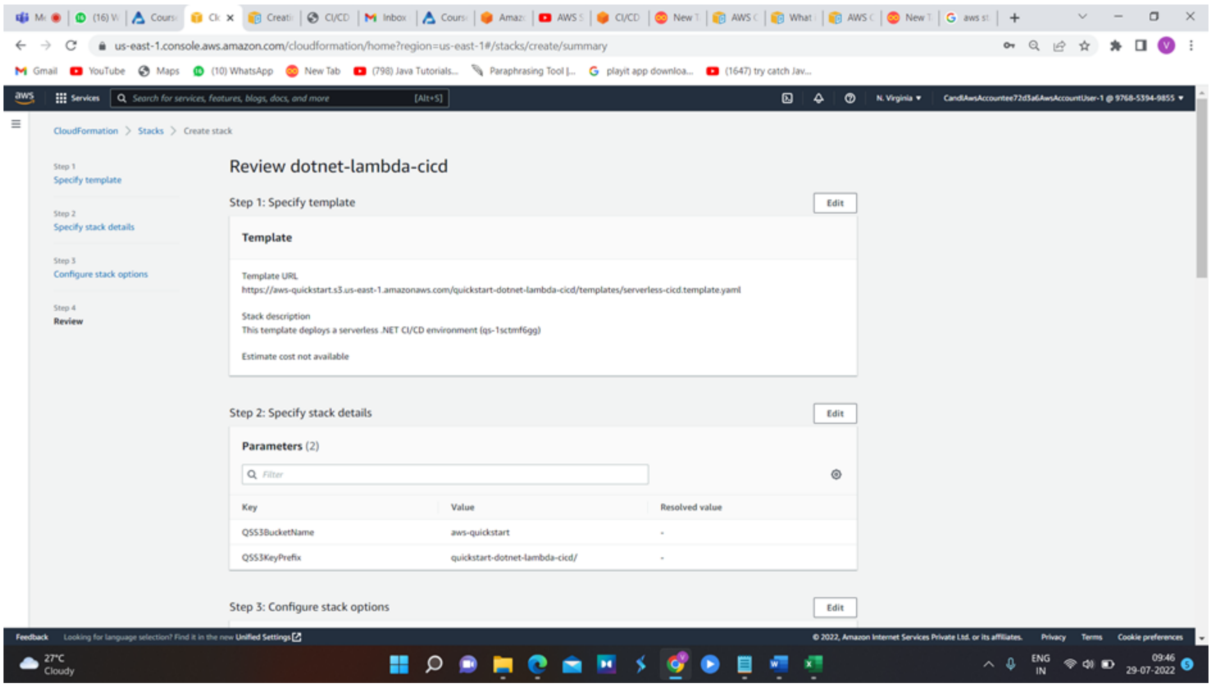
Step 5:

Review dotnet-lambda-cicd Pipeline:

On the Review page, review and confirm the template settings. Under Capabilities, select the two check boxes to acknowledge that the template creates IAM resources and might require the ability to automatically expand macros.

Step 1: Specify Template

Step 2: Specify stack details



us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/create/summary

CloudFormation > Stacks > Create stack

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Review dotnet-lambda-cicd

Step 1: Specify template

Template

Template URL
https://aws-quickstart.s3.us-east-1.amazonaws.com/quickstart-dotnet-lambda-cicd/templates/serverless-cicd.template.yaml

Stack description
This template deploys a serverless .NET CI/CD environment (qs-1sctmf6gg)

Estimate cost not available

Step 2: Specify stack details

Parameters (2)

Key	Value	Resolved value
QSS3BucketName	aws-quickstart	-
QSS3KeyPrefix	quickstart-dotnet-lambda-cicd/	-

Step 3: Configure stack options

Step 3: Configure stack options

- **Tags**

Tags are arbitrary key-value pairs that can be used to identify your stack for purposes such as cost allocation. For more information about what tags are and how they can be used, see Tagging your resources in the *Amazon EC2 User Guide*.

A Key consists of any alphanumeric characters or spaces. Tag keys can be up to 127 characters long.

A Value consists of any alphanumeric characters or spaces. Tag values can be up to 255 characters long.

- **Permissions**

An existing AWS Identity and Access Management (IAM) service role that CloudFormation can assume.

Instead of using your account credentials, CloudFormation uses the role's credentials to create your stack. For more information, see AWS CloudFormation service role.

- **Stack failure options**

Specifies the provision failure options for all stack deployments and change set operations. For more information, see Stack failure options.

The Roll back all stack resources option will roll back all resources specified in the template when the stack status is `CREATE_FAILED` or `UPDATE_FAILED`.

For create operations, the Preserve successfully provisioned resources option preserves the state of successful resources, while failed resources will stay in a failed state until the next update operation is performed.

For update and change set operations, the Preserve successfully provisioned resources option to preserve the state of successful resources while rolling back failed resources to the last known stable state. Failed resources will be in an `UPDATE_FAILED` state. Resources without a last known stable state will be deleted upon the next stack operation.

You can also set the following advanced options for stack creation:

- **<u>Stack policy</u>**

Defines the resources that you want to protect from unintentional updates during a stack update. By default, all resources can be updated during a stack update.

You can enter the stack policy directly as JSON, or upload a JSON file containing the stack policy. For more information, see Prevent updates to stack resources.

- **<u>Rollback Configuration</u>**

Enables you to have CloudFormation monitor the state of your stack during stack creation and updating, and to roll back that operation if the stack breaches the threshold of any of the alarms you've specified. Specify the CloudWatch alarms that CloudFormation should monitor. If any of the alarms goes to `ALARM` state during the stack operation or the monitoring period, CloudFormation rolls back the entire stack operation. For more information, see Monitor and roll back stack operations.

- **<u>Notification options</u>**

You can specify a new or existing Amazon Simple Notification Service topic where notifications about stack events are sent.

If you create an Amazon SNS topic, you must specify a name and an email address where stack event notifications are to be sent.

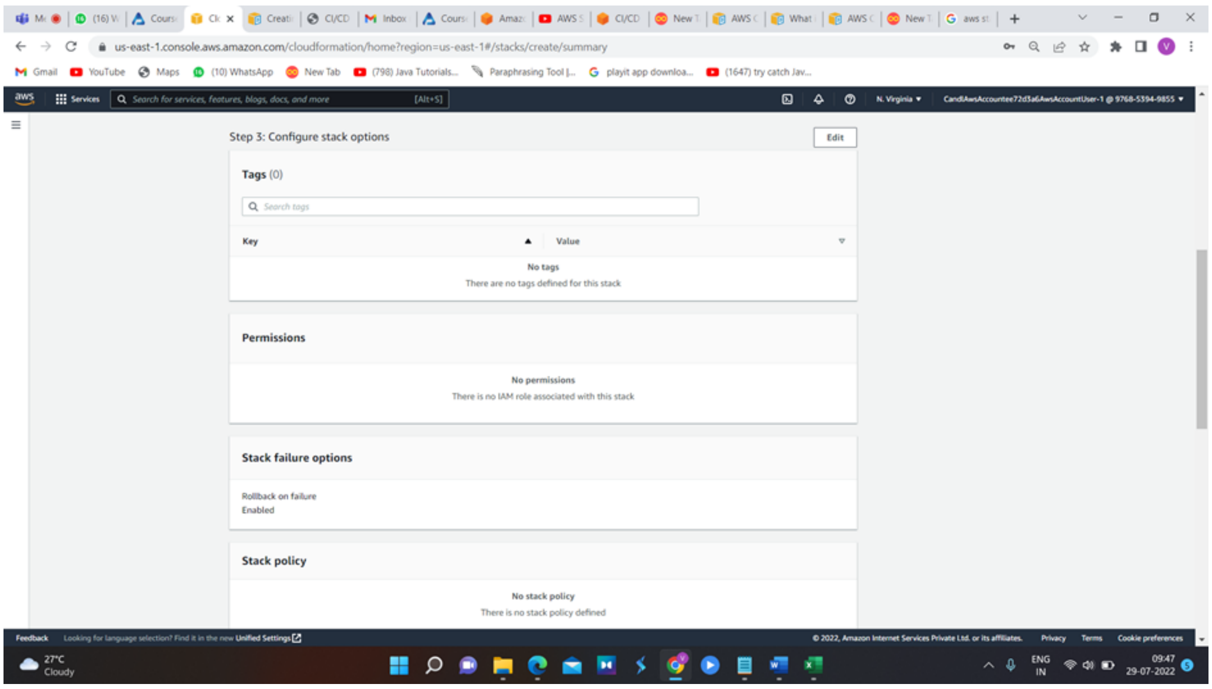
- **<u>Stack creation options</u>**

The following options are included for stack creation, but aren't available as part of stack updates.

- **<u>Timeout</u>**

Specifies the amount of time, in minutes, that CloudFormation should allot before timing out stack creation operations. If CloudFormation can't create the entire stack in the time allotted, it fails the stack creation due to timeout and rolls back the stack.

By default, there is no timeout for stack creation. However, individual resources may have their own timeouts based on the nature of the service they implement. For example, if an individual resource in your stack times out, stack creation also times out even if the timeout you specified for stack creation hasn't yet been reached.



us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/create/summary

Step 3: Configure stack options

Tags (0)

Key	Value
No tags There are no tags defined for this stack	

Permissions

No permissions
There is no IAM role associated with this stack

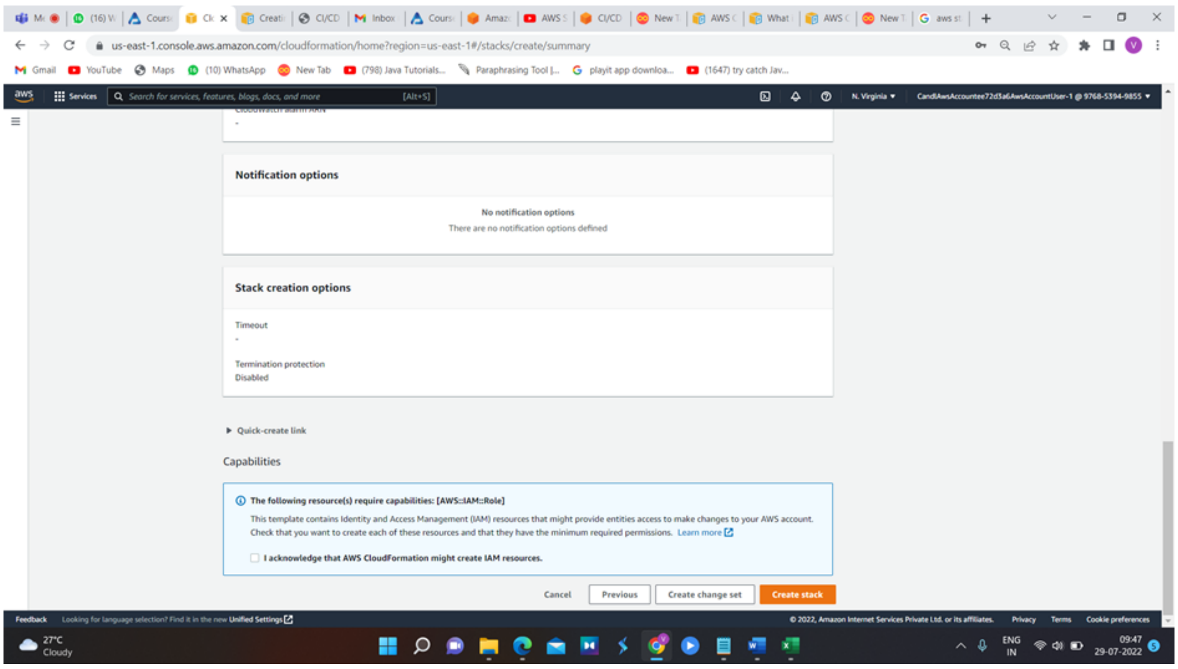
Stack failure options

Rollback on failure
Enabled

Stack policy

No stack policy
There is no stack policy defined

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions.


Notification options

No notification options
There are no notification options defined

Stack creation options

Timeout
-

Termination protection
Disabled

▶ Quick-create link

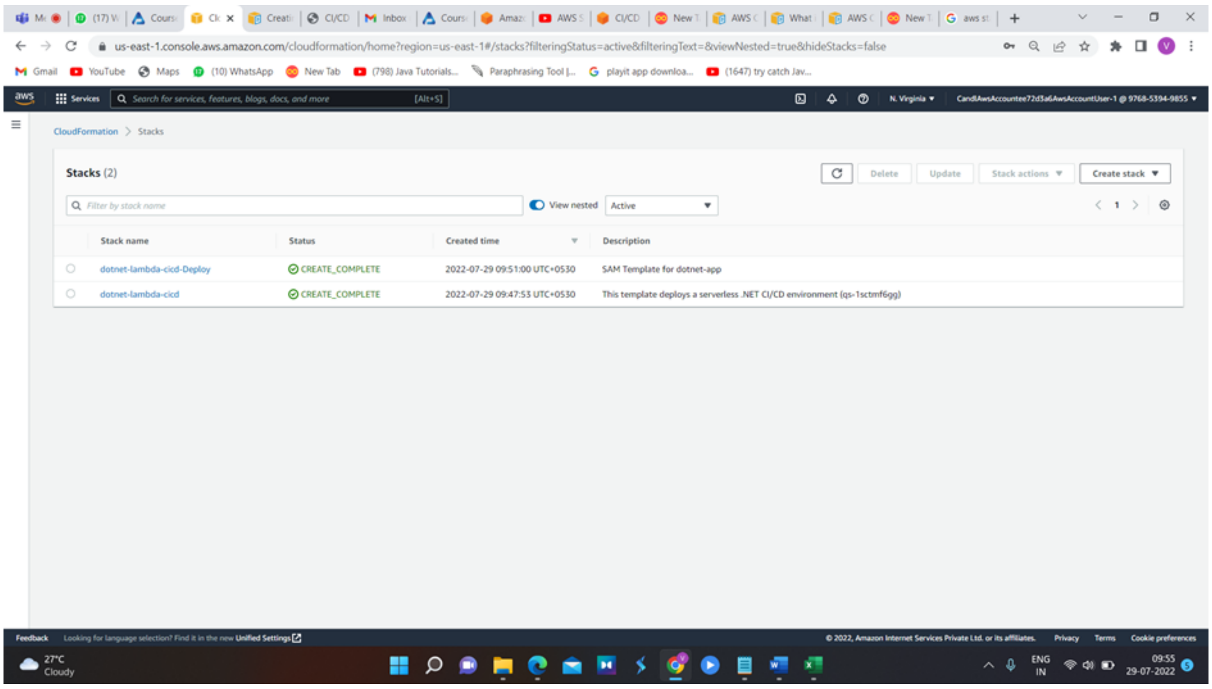
Capabilities

The following resource(s) require capabilities: [AWS::IAM::Role]
This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. Learn more

☐ I acknowledge that AWS CloudFormation might create IAM resources.

Step 6:

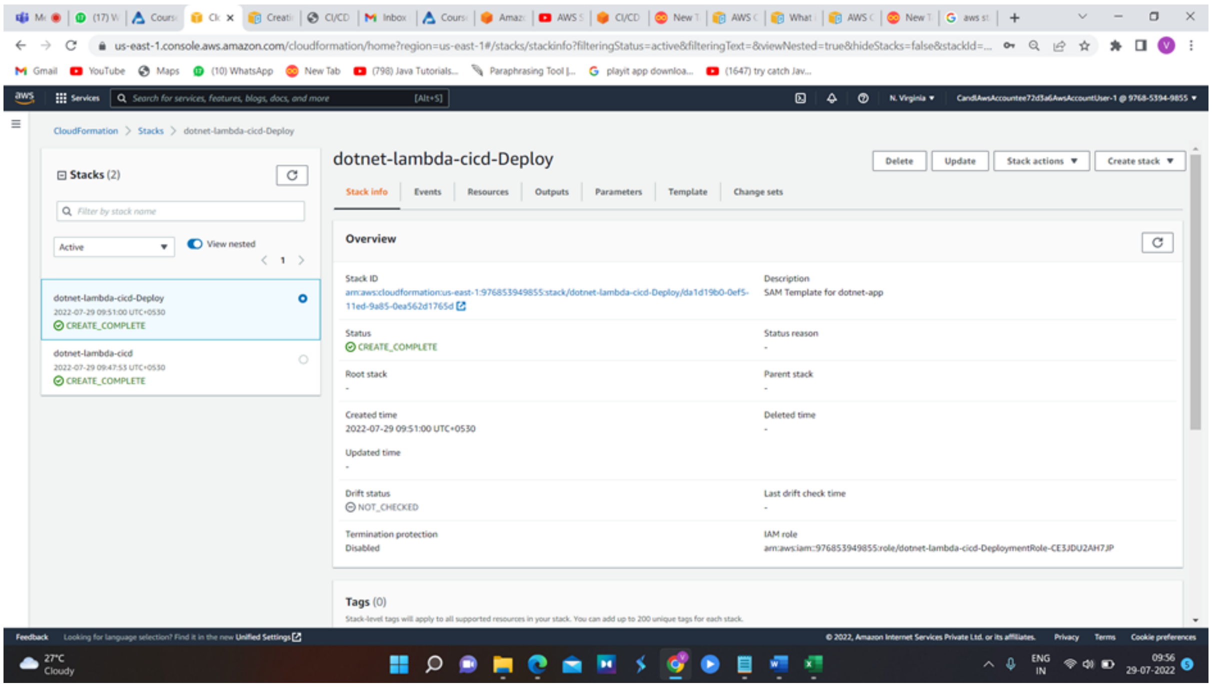
dotnet-lambda-CICD Deploy
Monitor the status of the stack. When the status is CREATE_COMPLETE, the CI/CD pipeline for .NET Serverless Applications deployment is ready.


CloudFormation > Stacks

Stacks (2)

Stack name	Status	Created time	Description
dotnet-lambda-cicd-Deploy	CREATE_COMPLETE	2022-07-29 09:51:00 UTC+0530	SAM Template for dotnet-app
dotnet-lambda-cicd	CREATE_COMPLETE	2022-07-29 09:47:53 UTC+0530	This template deploys a serverless .NET CI/CD environment (qs-1sctmf6gg)

To view the created resources, see the values displayed in the Outputs tab for the stack.



CloudFormation > Stacks > dotnet-lambda-cicd-Deploy

Stacks (2)

dotnet-lambda-cicd-Deploy
2022-07-29 09:51:00 UTC+0530
CREATE_COMPLETE

dotnet-lambda-cicd
2022-07-29 09:47:53 UTC+0530
CREATE_COMPLETE

dotnet-lambda-cicd-Deploy

Stack info | Events | Resources | Outputs | Parameters | Template | Change sets

Overview

Stack ID
arn:aws:cloudformation:us-east-1:976853949855:stack/dotnet-lambda-cicd-Deploy/da1d19b0-0ef5-11ed-9a85-0ea562d1765d

Description
SAM Template for dotnet-app

Status
CREATE_COMPLETE

Status reason
-

Root stack
-

Parent stack
-

Created time
2022-07-29 09:51:00 UTC+0530

Deleted time
-

Updated time
-

Drift status
NOT_CHECKED

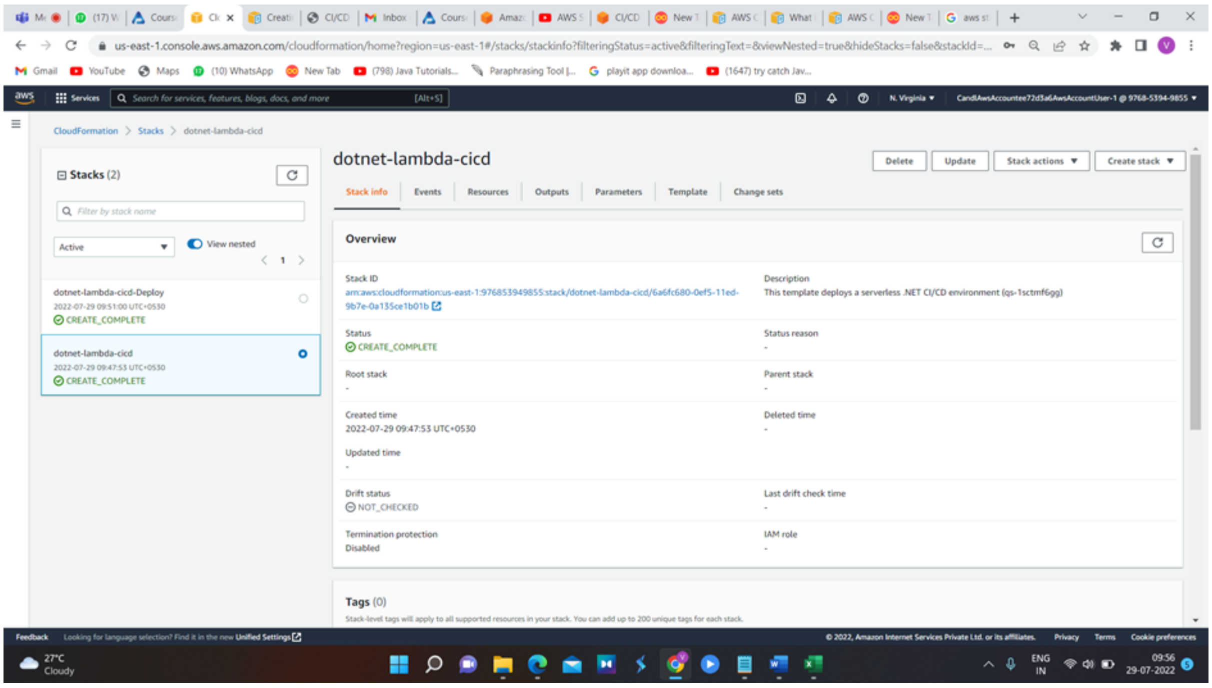
Last drift check time
-

Termination protection
Disabled

IAM role
arn:aws:iam::976853949855:role/dotnet-lambda-cicd-DeploymentRole-CE3JDU2AH7JP

Tags (0)
Stack-level tags will apply to all supported resources in your stack. You can add up to 200 unique tags for each stack.

dotnet-lambda-cicd:



CloudFormation > Stacks > dotnet-lambda-cicd

Stacks (2)

dotnet-lambda-cicd-Deploy
2022-07-29 09:51:00 UTC+0530
CREATE_COMPLETE

dotnet-lambda-cicd
2022-07-29 09:47:53 UTC+0530
CREATE_COMPLETE

dotnet-lambda-cicd

Stack info | Events | Resources | Outputs | Parameters | Template | Change sets

Overview

Stack ID
arn:aws:cloudformation:us-east-1:976853949855:stack/dotnet-lambda-cicd/6a6fc680-0ef5-11ed-9b7e-0a135ce1b01b

Description
This template deploys a serverless .NET CI/CD environment (qs-1sctmf6gg)

Status
CREATE_COMPLETE

Status reason
-

Root stack
-

Parent stack
-

Created time
2022-07-29 09:47:53 UTC+0530

Deleted time
-

Updated time
-

Drift status
NOT_CHECKED

Last drift check time
-

Termination protection
Disabled

IAM role
-

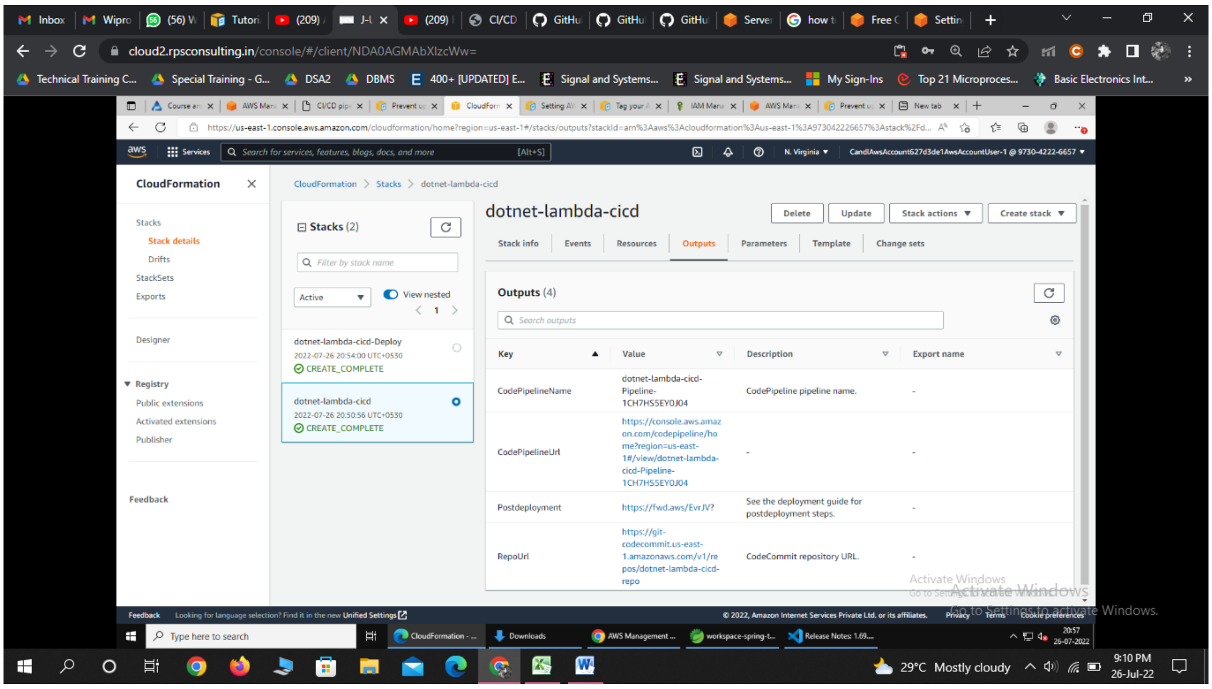
Tags (0)
Stack-level tags will apply to all supported resources in your stack. You can add up to 200 unique tags for each stack.

Outputs:

After the stack operation completes, the following output values display on the **Outputs** tab of the AWS CloudFormation console.

Output	Description
`RepoUrl`	URL of the CodeCommit repository that you can clone and edit to invoke the pipeline and application deployment.
`CodePipelineUrl`	URL of the CodePipeline pipeline.

After deploying this Quick Start, a build of the application starts. You can use your browser to navigate to the CodePipeline URL to observe the build process. After the build completes, the deployment process for the application creates a CloudFormation named `<Quick Start stack name>-Deploy` using the name you enter when launching this Quick Start.



dotnet-lambda-cicd

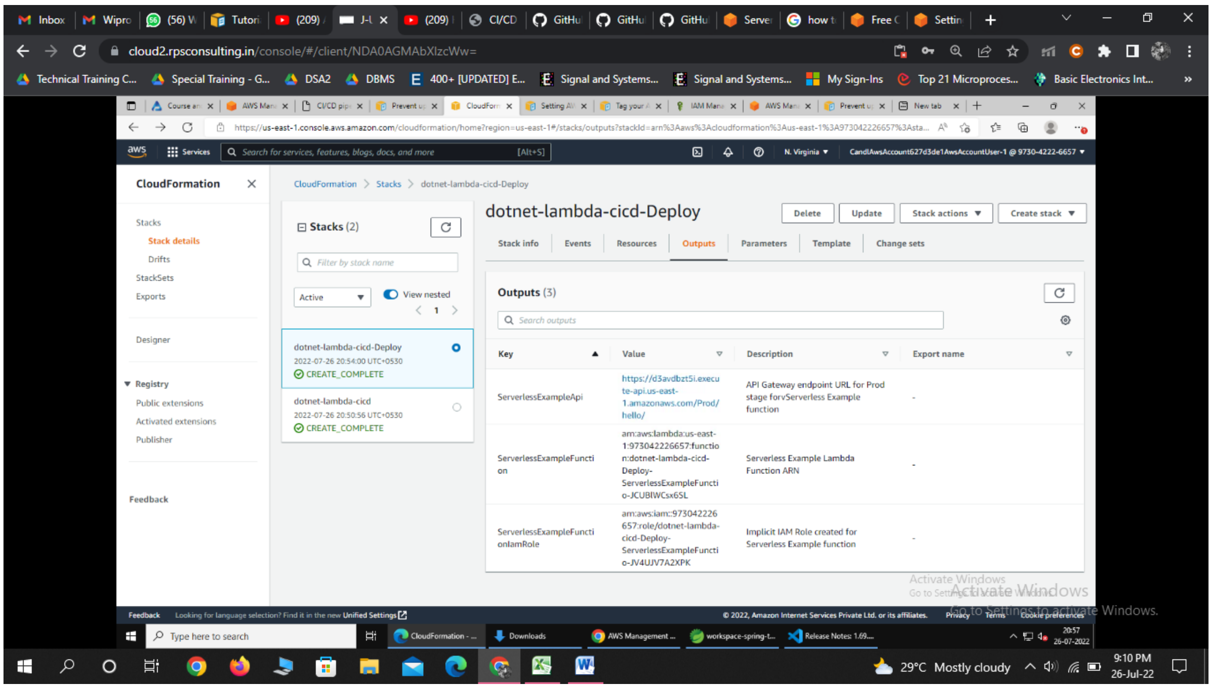
Outputs (4)

Key	Value	Description	Export name
CodePipelineName	dotnet-lambda-cicd-Pipeline-1CH7HS5EY0J04	CodePipeline pipeline name.	-
CodePipelineUrl	https://console.aws.amazon.com/codepipeline/home?region=us-east-1#/view/dotnet-lambda-cicd-Pipeline-1CH7HS5EY0J04	-	-
Postdeployment	https://fwd.aws/EvrJV?	See the deployment guide for postdeployment steps.	-
RepoUrl	https://git-codecommit.us-east-1.amazonaws.com/v1/repos/dotnet-lambda-cicd-repo	CodeCommit repository URL.	-

After deploying this Quick Start, a build of the application starts. You can use your browser to navigate to the CodePipeline URL to observe the build process. After the build completes, the deployment process for the application creates a CloudFormation named `<Quick Start stack name>-Deploy` using the name you enter when launching this Quick Start.

The stack also includes an output parameter that can be used to view your deployed application:

Output	**Description**
`ServerlessExampleApi`	API Gateway URL that exposes the application.



dotnet-lambda-cicd-Deploy

Stacks (2)

dotnet-lambda-cicd-Deploy
2022-07-26 20:54:00 UTC+0530
CREATE_COMPLETE

dotnet-lambda-cicd
2022-07-26 20:50:56 UTC+0530
CREATE_COMPLETE

Outputs (3)

Key	Value	Description	Export name
ServerlessExampleApi	https://d3avdbzt5i.execute-api.us-east-1.amazonaws.com/Prod/hello/	API Gateway endpoint URL for Prod stage forvServerless Example function	-
ServerlessExampleFunction	arn:aws:lambda:us-east-1:973042226657:function:dotnet-lambda-cicd-Deploy-ServerlessExampleFunctio-JCUBIWCsx6SL	Serverless Example Lambda Function ARN	-
ServerlessExampleFunctionlamRole	arn:aws:iam::973042226657:role/dotnet-lambda-cicd-Deploy-ServerlessExampleFunctio-JV4UJV7A2XPK	Implicit IAM Role created for Serverless Example function	-