



CI/CD Configurations for Backend Applications

Agenda:

CodeBuild:

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don't need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools.

CodePipeline:

AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. You can easily integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plugin.

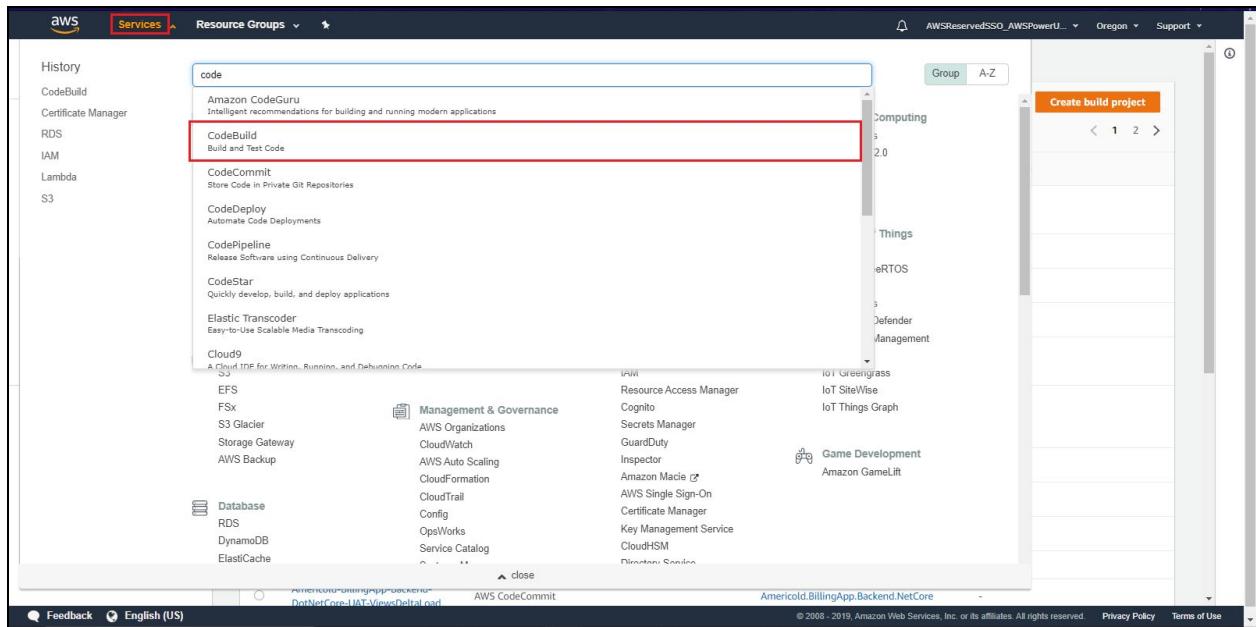
Serverless Deployment:

The Serverless Framework was designed to provision your AWS Lambda Functions, Events and infrastructure Resources safely and quickly.

The Serverless Framework translates all syntax in **serverless.yml** to a single AWS CloudFormation template. By depending on CloudFormation for deployments, users of the Serverless Framework get the safety and reliability of CloudFormation.

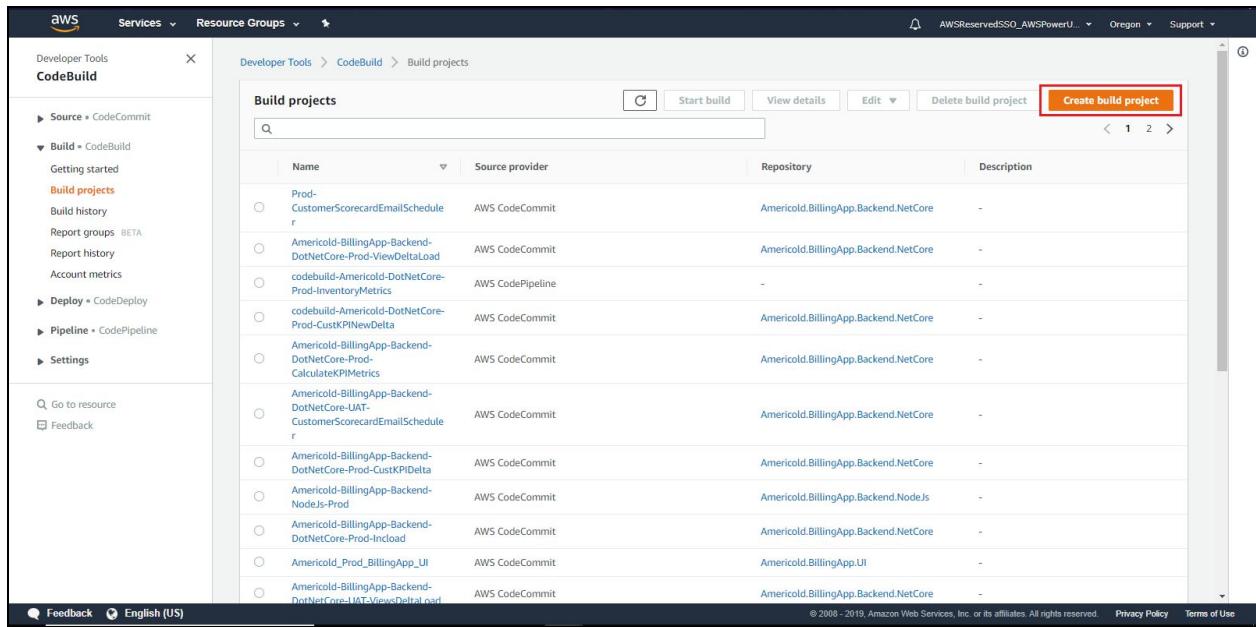
By the end of the document, we learn how to create a CI-CD pipeline for NodeJs and ASPNetCore applications.

First we need to login into the AWS console and search for **CodeBuild** service.



The screenshot shows the AWS Services menu with 'Services' selected. A search bar at the top contains the text 'code'. Below the search bar, a list of services is displayed, with 'CodeBuild' being the second item and highlighted with a red box. Other services listed include Amazon CodeGuru, CodeCommit, CodeDeploy, CodePipeline, CodeStar, Elastic Transcoder, Cloud9, EFS, FSx, S3 Glacier, Storage Gateway, AWS Backup, Database, RDS, DynamoDB, and ElastiCache. To the right of the service list, there is a sidebar with various AWS categories like Computing, Things, eRTOS, Defender, Management, IAM, Resource Access Manager, Cognito, Secrets Manager, GuardDuty, Inspector, Amazon Macie, CloudTrail, AWS Single Sign-On, Certificate Manager, Key Management Service, CloudHSM, Direct Connect, and Game Development. At the bottom right of the main area, there is a 'Create build project' button.

Goto build projects and click on **create build project** to create a new build project for integration.



The screenshot shows the 'Build projects' page within the AWS CodeBuild service. On the left, a navigation sidebar lists 'Developer Tools', 'Source', 'Build', 'Report groups', 'Deploy', 'Pipeline', and 'Settings'. Under 'Build', 'Build projects' is selected. The main area displays a table titled 'Build projects' with columns for 'Name', 'Source provider', 'Repository', and 'Description'. There are 12 entries in the table, each with a circular icon and a link to its details. At the top of the main area, there are buttons for 'Start build', 'View details', 'Edit', 'Delete build project', and 'Create build project' (which is highlighted with a red box). The bottom of the screen includes standard AWS footer links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

Enter the name of the project.

The screenshot shows the 'Create build project' page in the AWS CodeBuild console. In the 'Project configuration' section, the 'Project name' field is highlighted with a red box and contains the value 'Americold-BillingApp-Backend-NodeJs-dev'. Below it, a note states: 'A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.' The 'Description - optional' and 'Build badge - optional' sections are also visible. The 'Source' section is partially visible at the bottom.

Under the source section, select the source provider as **AWS CodeCommit**, and select the repository and branch as shown below.

The screenshot shows the 'Source' configuration page. Under 'Source 1 - Primary', the 'Source provider' dropdown is set to 'AWS CodeCommit'. The 'Repository' field contains 'Q_Americold.BillingApp.Backend.NodeJs'. The 'Reference type' section has 'Branch' selected. The 'Branch' dropdown shows 'Dev'. The 'Commit ID - optional' field is empty. The 'Source version info' section shows 'refs/heads/Dev' and a commit history entry '2e212004 [Chandra Nuguri]: Updated the table name in Saved Search service'. The 'Environment' section is partially visible at the bottom.

Under Environment section we took the Managed Image that is provided by AWS, Operating system as Ubuntu, runtime as standard, Image as standard 2.0 version, image version as always latest, environment type as Linux and created new service role for the code build .

Under the Buildspec section, choose an option as **Insert build commands** to edit the **buildspec.yml** file from console. buildspec.yml file has a set of commands for integration and deployment.

```

28 >   commands:
29     - cd Inventory_Consolidation
30     - pwd
31     - npm install
32     - npm install -g serverless
33     # - command
34     # - command
35 >   post_build:
36     commands:
37     - serverless deploy --stage dev
38     # - command
39     # - command
40 >   #artifacts:
41     #files:
42     # - location
43     # - location
44     #name: ${date +%Y-%m-%d}
45     #discard-paths: yes
46     #base-directory: location
47     #cache:
48     #paths:
49     # - paths

```

Configure the respective stages as per requirement in **buildspec.yml** file as shown below.

Below is the step by step explanation for the above script:

Install stage:

- **runtime-verions:**

nodejs : 10:It defines the runtime environment and version for the application

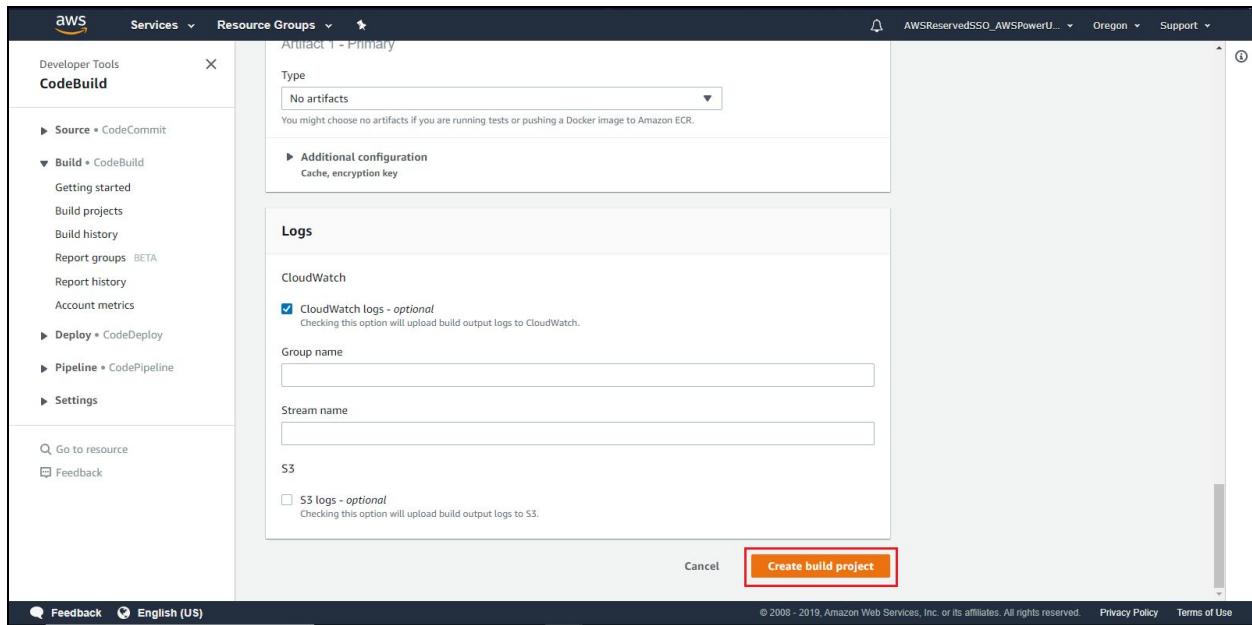
Build Stage:

- **cd Inventory_consolidation:** Go to the folder called Inventory_consolidation to install the necessary dependencies for integration.
 - **npm install:** Install node modules using the following command
 - **npm install -g serverless:** Install serverless module using the following command

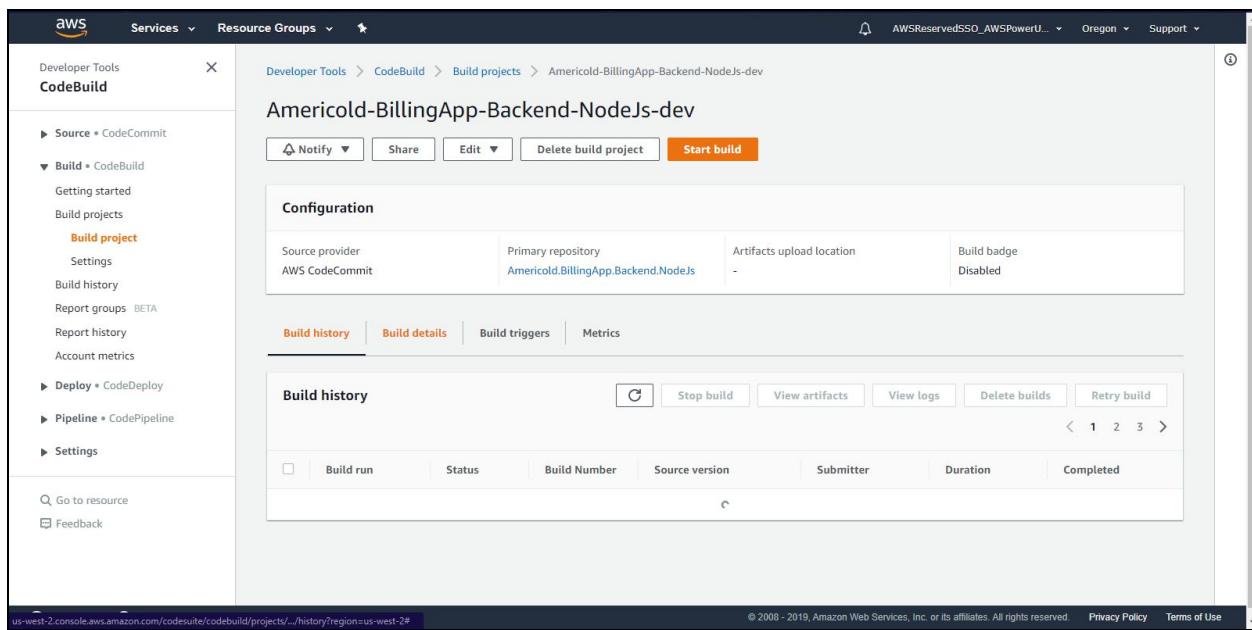
Post-Build stage:

- **serverless deploy --stage dev**: To deploy an application using serverless.yml template file we will run the following command

We are enabling cloud watch logs to check the build logs. Once all the configurations were done, click on **create build project**.



New build project has been created and the dashboard is shown as below.



Goto details and goto **environment section** in build details.

AWS Services Resource Groups AWSReservedSSO_AWSPowerU... Oregon Support

Developer Tools CodeBuild

Source • CodeCommit

Build • CodeBuild

Getting started Build projects

Build project

Settings

Build history

Report groups BETA

Report history

Account metrics

Deploy • CodeDeploy

Pipeline • CodePipeline

Settings

Go to resource Feedback

Build history Build details Build triggers Metrics

Configuration

Source provider Primary repository Artifacts upload location Build badge

AWS CodeCommit Americold.BillingApp.Backend.NodeJs - Disabled

Build history Build details Build triggers Metrics

Project configuration

Name Description

Americold-BillingApp-Backend-NodeJs-dev -

Project ARN

arn:aws:codebuild:us-west-2:646632670602:project/Americold-BillingApp-Backend-NodeJs-dev

Tags

Source

Edit

Feedback English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Click on service role which is new service role created while configuring code build. It will redirect us to **IAM console**.

AWS Services Resource Groups AWSReservedSSO_AWSPowerU... Oregon Support

Developer Tools CodeBuild

Source • CodeCommit

Build • CodeBuild

Getting started Build projects

Build project

Settings

Build history

Report groups BETA

Report history

Account metrics

Deploy • CodeDeploy

Pipeline • CodePipeline

Settings

Go to resource Feedback

Image Environment type Compute Privileged

aws/codebuild/standard:2.0 Linux 3 GB memory, 2 vCPUs False

Service role Timeout Queued timeout Certificate

arn:aws:iam::646632670602:role/service-role/codebuild-Amercold-BillingApp-Backend-NodeJs-dev-service-role 1 hour 0 minutes 8 hours 0 minutes -

Registry credential

VPC

Environment variables

Buildspec

Edit

https://us-west-2.console.aws.amazon.com/iam/home?region=us-west-2#roles/codebuild-Ame...

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

The IAM role dashboard will be as shown below and we need to add the required policies to access the AWS services for application deployment through code build.

The screenshot shows the AWS IAM Role Summary page for a role named "codebuild-Americold BillingApp Backend.NodeJs dev-service-role". The "Permissions" tab is selected, displaying a list of attached policies. Two policies are currently listed: "AWSLambdaFullAccess" and "IAMFullAccess". A red box highlights the "Attach policies" button, which is used to add more policies to the role.

Click on **attach policies** to add the policies for the IAM role.

The screenshot shows the same IAM Role Summary page as above, but with a much larger list of policies under the "Attach policies" dropdown. The list includes numerous AWS managed policies such as "AWSLambdaFullAccess", "IAMFullAccess", "AmazonS3FullAccess", "AmazonAPIGatewayAdministrator", "AmazonSSMFullAccess", "AmazonVPCFullAccess", and "AWSCloudFormationFullAccess". A red box highlights the "Attach policies" button.

Search for the required policy and select the check boxes and click on Attach Policies.

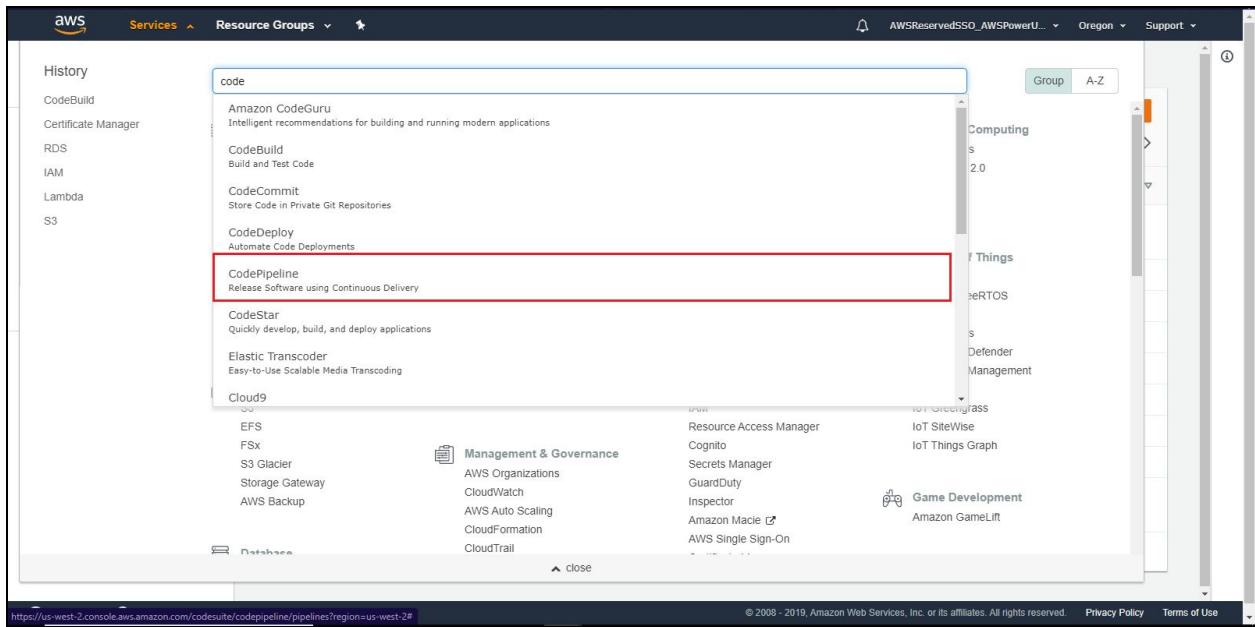
The screenshot shows the 'Attach Permissions' step in the AWS IAM console. A search bar is highlighted with a red box. Below it is a table listing various AWS managed policies, each with a checkbox, a policy name, its type, and how many times it's used. The table includes columns for 'Policy name', 'Type', and 'Used as'. Policies listed include AdministratorAccess, AlexaForBusinessDeviceSetup, AlexaForBusinessFullAccess, AlexaForBusinessGatewayExecution, AlexaForBusinessPolyDelegatedAccessPolicy, AlexaForBusinessReadOnlyAccess, AmazonAPIGatewayInvokeFullAccess, AmazonAPIGatewayPushToCloudWatchLogs, AmazonAppStreamFullAccess, and AmazonAppStreamReadOnlyAccess. At the bottom right of the dialog are 'Cancel' and 'Attach policy' buttons, with 'Attach policy' also highlighted by a red box.

Here we have attached the policies for **Lambda, IAM, S3, APIGateway, SSM, VPC and CloudFormation**.

The screenshot shows the 'Permissions' tab for a specific IAM role. On the left, there's a navigation menu with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Roles' is also selected. The main area displays the role's ARN, description, instance profile ARNs, path, creation time, last activity, and maximum session duration. Below this, the 'Permissions' tab is active, showing a list of 10 policies applied to the role. These policies are: AWSLambdaFullAccess, IAMFullAccess, AmazonS3FullAccess, AmazonAPIGatewayAdministrator, AmazonSSMFullAccess, AmazonVPCFullAccess, and AWSCloudFormationFullAccess. Each policy is listed with its name, type (AWS managed policy), and a delete icon. At the top of the permissions section is an 'Attach policies' button, which is highlighted with a red box. Other tabs like 'Trust relationships', 'Tags', 'Access Advisor', and 'Revoke sessions' are also visible.

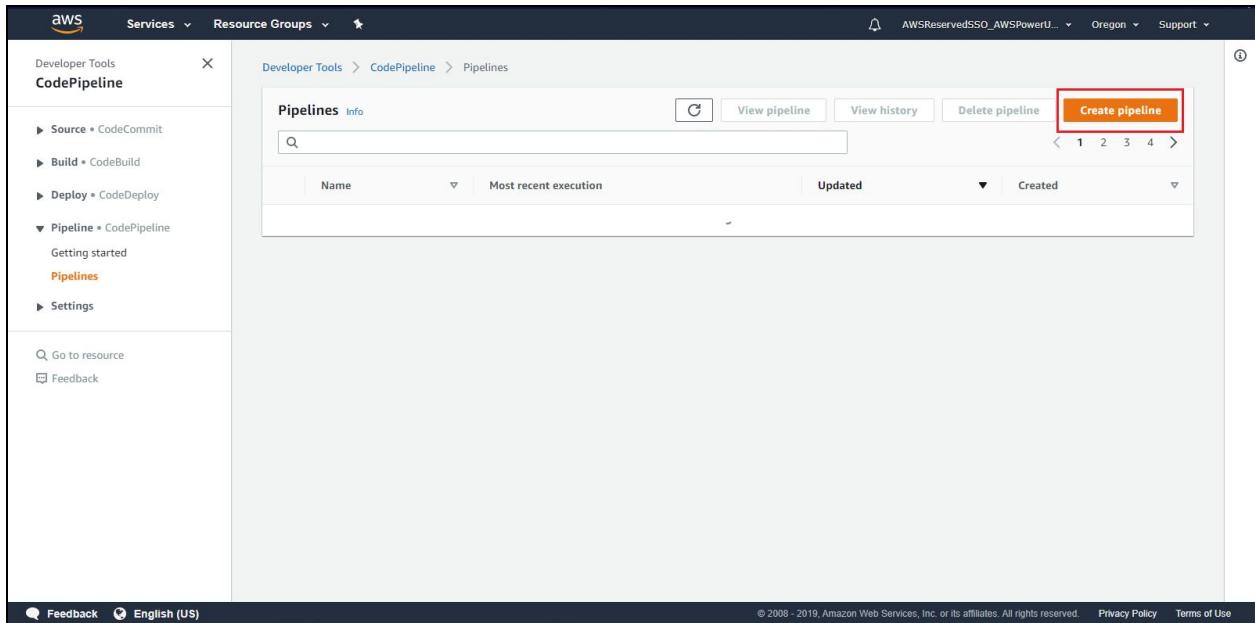
We are going to automate the process for integration and deployment using CodePipeline.

Goto **CodePipeline service** in AWS console.



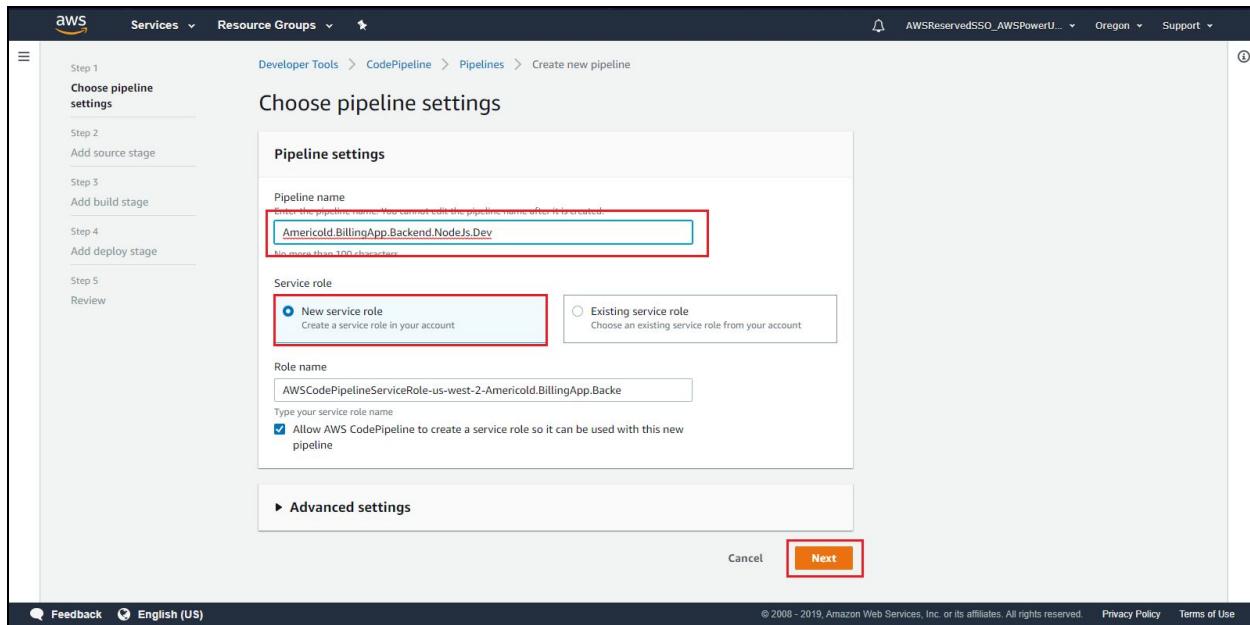
The screenshot shows the AWS Services menu with the search bar containing "code". The "CodePipeline" service is highlighted with a red box and selected. Other services listed include Amazon CodeGuru, CodeBuild, CodeCommit, CodeDeploy, CodeStar, Elastic Transcoder, Cloud9, EFS, FSx, S3 Glacier, Storage Gateway, AWS Backup, and various Management & Governance services like Resource Access Manager, Cognito, Secrets Manager, GuardDuty, Inspector, Amazon Macie, AWS Single Sign-On, AWS Organizations, CloudWatch, AWS Auto Scaling, CloudFormation, and CloudTrail. The URL at the bottom is <https://us-west-2.console.aws.amazon.com/codesuite/codepipeline/pipelines?region=us-west-2#>.

Click on **Create pipeline** to create new pipeline for automated CI/CD.

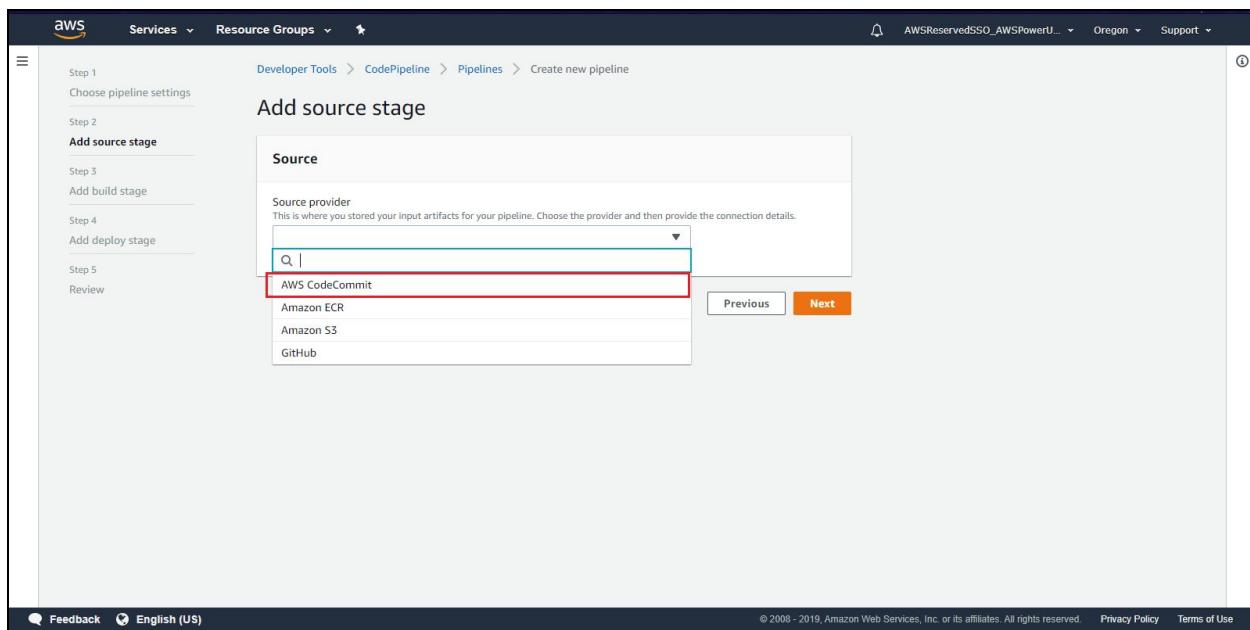


The screenshot shows the "Pipelines" page within the CodePipeline service. The left sidebar has sections for Source (CodeCommit), Build (CodeBuild), Deploy (CodeDeploy), Pipeline (CodePipeline), Getting started, Pipelines (selected), and Settings. The main area displays a table of pipelines with columns for Name, Most recent execution, Updated, and Created. A search bar is at the top of the table. At the top right, there are buttons for View pipeline, View history, Delete pipeline, and Create pipeline, with the "Create pipeline" button highlighted with a red box. The URL at the bottom is <https://us-west-2.console.aws.amazon.com/codesuite/codepipeline/pipelines?region=us-west-2#>.

Enter the name of the Pipeline and Choose the Service role as New Service role and Enter Click on **Next** as shown below. Here a new IAM service role will be created for code pipeline.



Under the Add source stage, Select the Source provider as **AWS CodeCommit**.



Select the name of the Repository and Branch name. Choose the Change detection options as Amazon CloudWatch Events to enable cloudwatch logs for the pipeline and click on **Next** to goto build configuration.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Add source stage

Source

Source provider

AWS CodeCommit

Repository name

Americold.BillingApp.Backend.NodeJs

Branch name

Dev

Change detection options

Amazon CloudWatch Events (recommended)

AWS CodePipeline

Cancel Previous Next

Under Add build stage, Select the Build provider as **AWS CodeBuild**, Region as **US West – (Oregon)**, Select the Build Project name that we created in the above steps and click on **Next** as shown below.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Add build stage

Build - optional

Build provider

AWS CodeBuild

Region

US West - (Oregon)

Project name

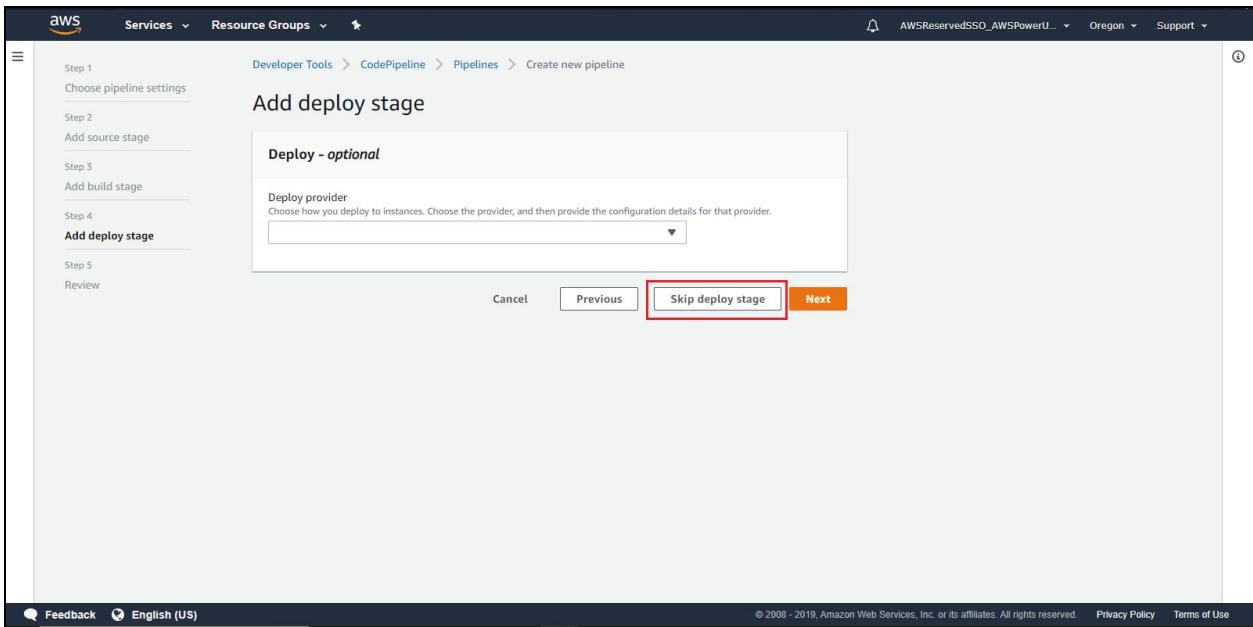
Americold-BillingApp-Backend-NodeJs-dev

Environment variables - optional

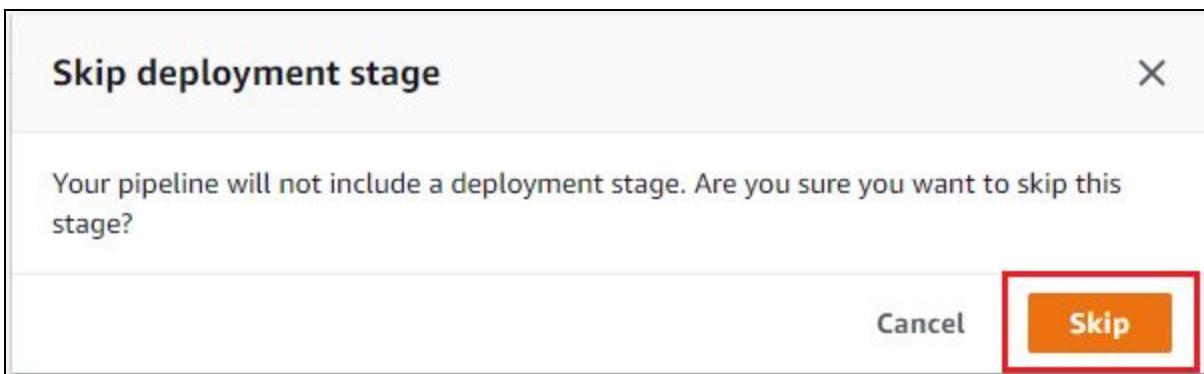
Add environment variable

Cancel Previous Skip build stage Next

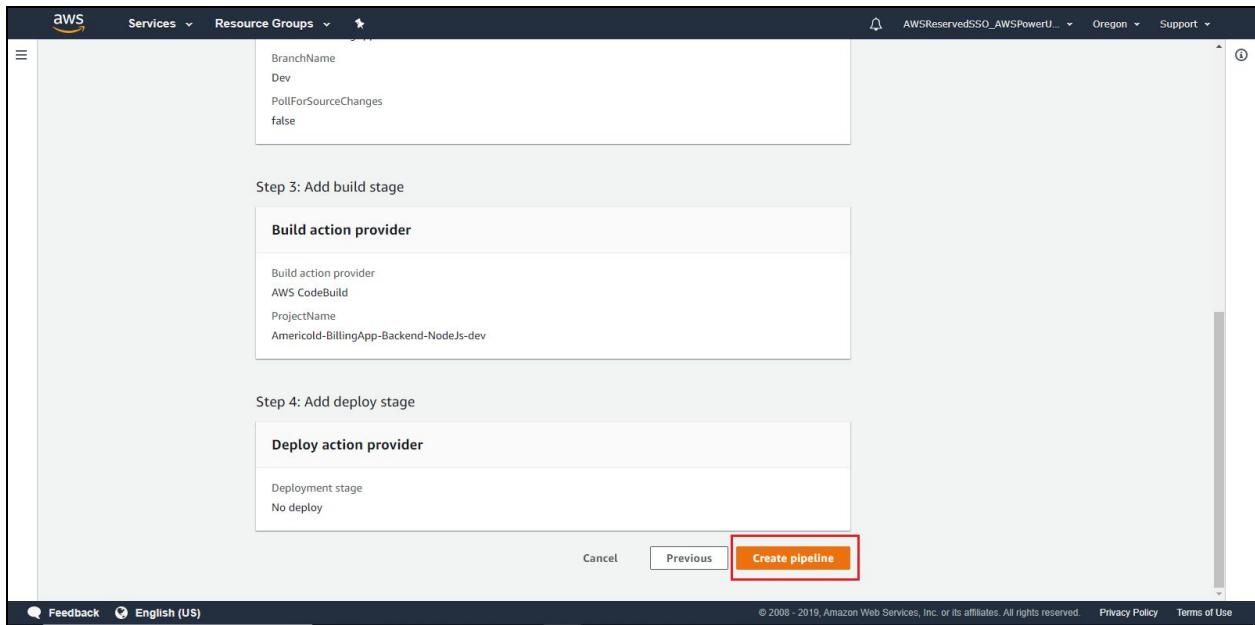
We are deploying through the commands. So, we can **skip** the deploy stage.



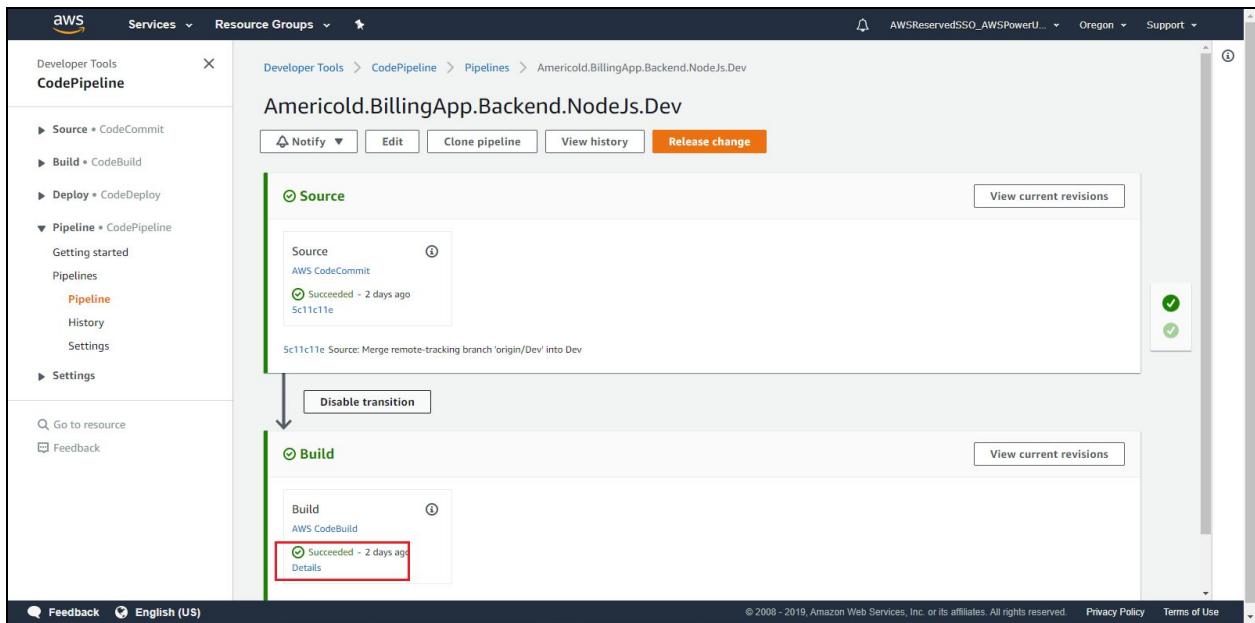
Click on **Skip**.



Review the pipeline configuration and Click on **Create pipeline**.



We can see the process is automated for CI/CD. We can check the stage logs by clicking on details. Pipeline will trigger automatically for the first time to release changes and later it will be triggered when the new changes are committed to repository.



We can check the **build logs** here.

The screenshot shows the AWS CodeBuild console. On the left, the navigation pane includes sections for Source, Build, Deploy, Pipeline, and Settings. Under Build, 'Build project' is selected. The main area displays the build status for 'Americold_BillingApp_UI:3ff22054-4a71-4fc1-8dbd-25e09e9fe9ab'. The status is 'Succeeded'. Below the status, it shows the initiator as 'codepipeline/Americold.Dev.BillingApp.U...', build ARN, and resolved source version. The build started at Dec 11, 2019, 2:58 AM and ended at Dec 11, 2019, 3:01 AM. The build number is 174. The 'Build logs' tab is active, showing the log output. A red box highlights the 'Tail logs' button at the top right of the log viewer. The log output shows the build process starting with a ping, then downloading source code, and finally executing the 'serverless deploy' command, which completes successfully.

For backend applications, while running the **serverless deploy** command it will automatically Create/Updates the Cloudformation stack using a serverless.yml template file.

This screenshot shows the AWS CodeBuild console with the 'Build project' section selected. The build logs for a specific run are displayed. A large red box highlights a portion of the log output where the 'serverless deploy' command is being executed. The log shows the deployment process, including the creation and updating of the CloudFormation stack, and the final message 'Serverless: Stack update finished...'. The log also includes details about the service, region, stack, resources, and API keys used during the deployment.

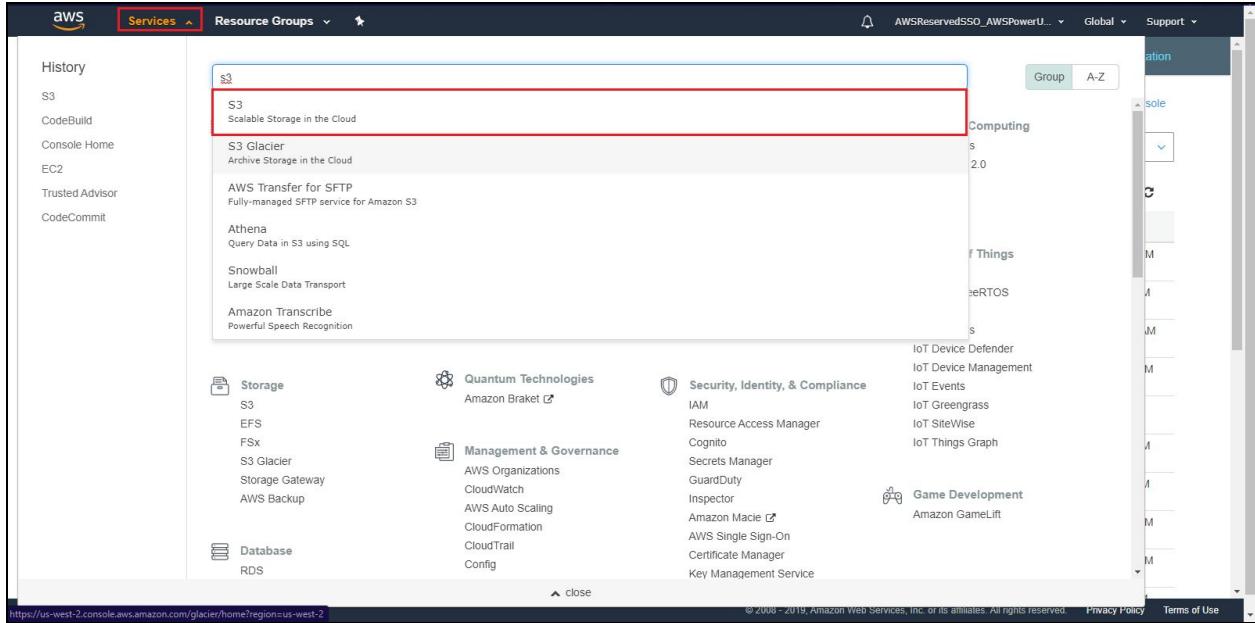
Through the Cloudformation template file, the artifacts will automatically deploy into **S3 bucket**, updates the Lambda Function and APIs

```

83 Serverless: Validating template...
84 Serverless: Updating Stack...
85 Serverless: Checking Stack update progress...
86 .....
87 Serverless: Stack update finished...
88 Service Information
89 service: americold-billingapp-nodeapis
90 stage: dev
91 region: us-west-2
92 stack: americold-billingapp-nodeapis-dev
93 resources: 38
94 api keys:
95 None
96 endpoints:
97 POST - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/inventoryConsolidation
98 GET - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/facilityData
99 GET - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/customerData
100 POST - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/inventoryTransaction
101 POST - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/savedSearch
102 GET - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/savedSearchApply
103 GET - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/savedSearchView
104 DELETE - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/deleteSavedSearch/{Search_Name}
105 GET - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/userInfo
106 GET - https://l4r9iowml.execute-api.us-west-2.amazonaws.com/dev/api/usermenu
107 functions:
108   Login: americold-billingapp-nodeapis-dev-Login
109 layers:
110 None
111 Serverless: Prune: Running post-deployment pruning
112 Serverless: Prune: Querying for deployed function versions
113 Serverless: Prune: americold-billingapp-nodeapis-dev-login has 3 additional versions published and 0 aliases, 1 version selected for deletion
114 Serverless: Prune: Selection selected americold-billingapp-nodeapis-dev-login v42...
115 Serverless: Prune: Pruning complete
116 Serverless: Removing old service artifacts from S3...
117 Serverless: Run the "serverless" command to setup monitoring, troubleshooting and testing.
118
119 [Container] 2019/12/16 00:54:03 Phase complete: POST_BUILD State: SUCCEEDED
120 [Container] 2019/12/16 00:54:03 Phase context status code: Message:
121

```

Goto S3 service to check whether the artifacts in the S3 bucket are updated or not.



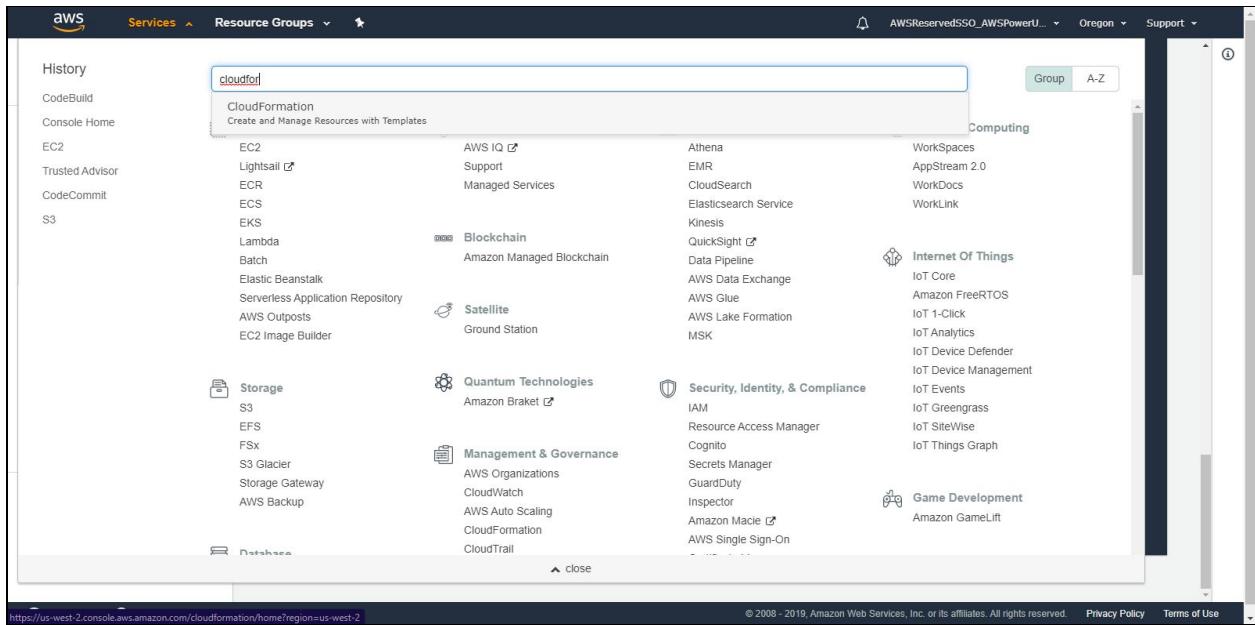
Following is the **S3 bucket** where we are storing our artifacts for backend Node.js application.

The screenshot shows the AWS S3 service page. On the left, there's a sidebar with options like 'Buckets', 'Batch operations', and 'Access analyzer for S3'. The main area is titled 'S3 buckets' and contains a search bar and buttons for 'Create bucket', 'Edit public access settings', 'Empty', and 'Delete'. Below this, a table lists 20 buckets across 1 region. The columns include 'Bucket name', 'Access', 'Region', and 'Date created'. One specific bucket, 'americold.dev.services.billingapp', is highlighted with a red box. Other buckets listed include 'americold-billingapp-ui', 'americold.billing.lambda.incloudonpremtocloud', 'americold.dev.netcore.schedulerlambdafunction', 'americold.dev.services.billingapp', 'americold.dev.services.billingapp.netcore', 'americold.log.ui', 'americoldlambdafunctions', 'aws-glue-scripts-646632670602-us-west-2', and 'aws-glue-temporary-646632670602-us-west-2'.

Here we can check our **latest artifact** in the S3 bucket.

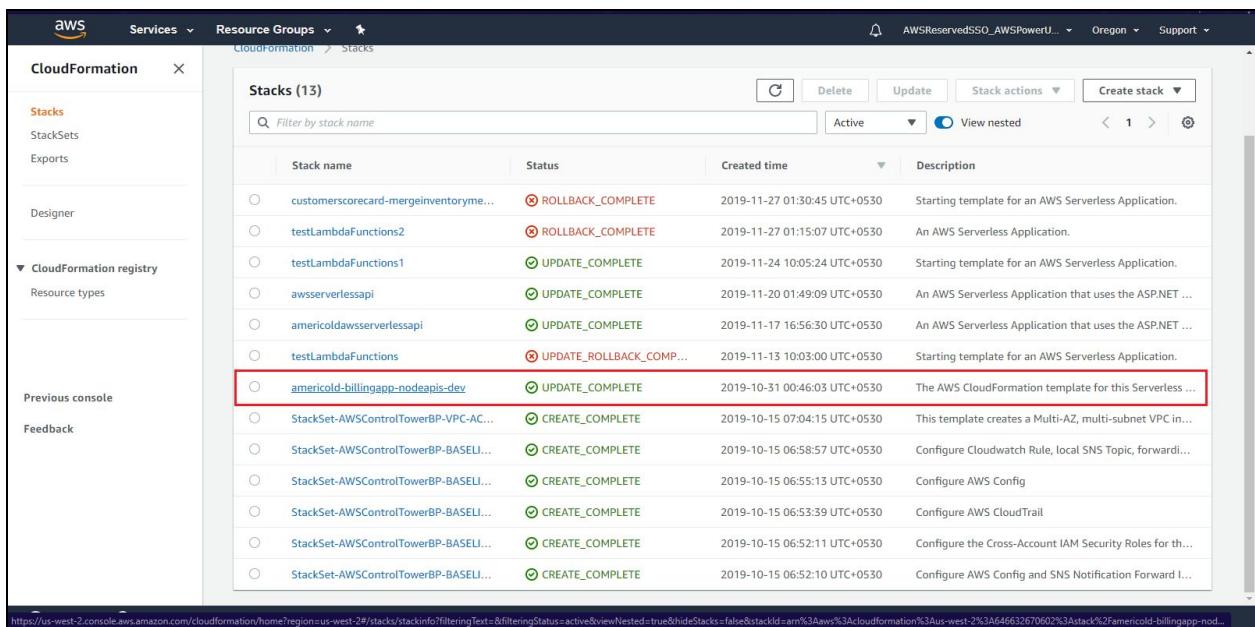
This screenshot shows the details for the 'americold.dev.services.billingapp' bucket. The navigation path is 'Amazon S3 > americold.dev.services.billingapp > serverless > americold-billingapp-nodeapis > dev'. The main view is the 'Overview' tab, which includes a search bar, upload/download buttons, and an actions dropdown. Below is a table listing objects in the 'dev' folder. The first five objects are highlighted with a red box. The columns are 'Name', 'Last modified', 'Size', and 'Storage class'. The objects listed are: '1576043779074-2019-12-11T05:56:19.074Z', '1576168398988-2019-12-12T16:33:18.988Z', '1576396653890-2019-12-15T07:57:33.890Z', '1576400253630-2019-12-15T08:57:33.630Z', and '1576457624322-2019-12-16T00:53:44.322Z'. At the bottom, there are two 'Viewing 1 to 5' notices.

Goto **CloudFormation** to check whether the events got triggered or not.



The screenshot shows the AWS Services menu with the search bar containing "cloudfor". The "CloudFormation" service is highlighted in the search results. The menu also includes links for History, CodeBuild, Trusted Advisor, CodeCommit, S3, and various computing, storage, and management services.

Here we can check our **cloud formation stack** updated time.



The screenshot shows the AWS CloudFormation Stacks page. The left sidebar includes options for CloudFormation (Stacks, StackSets, Exports), Designer, CloudFormation registry (Resource types), Previous console, and Feedback. The main area displays a table of stacks:

Stack name	Status	Created time	Description
customerscorecard-mergeinventoryme...	ROLLBACK_COMPLETE	2019-11-27 01:30:45 UTC+0530	Starting template for an AWS Serverless Application.
testLambdaFunctions2	ROLLBACK_COMPLETE	2019-11-27 01:15:07 UTC+0530	An AWS Serverless Application.
testLambdaFunctions1	UPDATE_COMPLETE	2019-11-24 10:05:24 UTC+0530	Starting template for an AWS Serverless Application.
awsserverlessapi	UPDATE_COMPLETE	2019-11-20 01:49:09 UTC+0530	An AWS Serverless Application that uses the ASP.NET ...
americoldawsserverlessapi	UPDATE_COMPLETE	2019-11-17 16:56:30 UTC+0530	An AWS Serverless Application that uses the ASP.NET ...
testLambdaFunctions	UPDATE_ROLLBACK_COMP...	2019-11-13 10:03:00 UTC+0530	Starting template for an AWS Serverless Application.
americold-billingapp-nodeapis-dev	UPDATE_COMPLETE	2019-10-31 00:46:03 UTC+0530	The AWS CloudFormation template for this Serverless ...
StackSet-AWSControlTowerBP-VPC-AC...	CREATE_COMPLETE	2019-10-15 07:04:15 UTC+0530	This template creates a Multi-AZ, multi-subnet VPC in...
StackSet-AWSControlTowerBP-BASELI...	CREATE_COMPLETE	2019-10-15 06:58:57 UTC+0530	Configure Cloudwatch Rule, local SNS Topic, forwardi...
StackSet-AWSControlTowerBP-BASELI...	CREATE_COMPLETE	2019-10-15 06:55:13 UTC+0530	Configure AWS Config
StackSet-AWSControlTowerBP-BASELI...	CREATE_COMPLETE	2019-10-15 06:53:39 UTC+0530	Configure AWS CloudTrail
StackSet-AWSControlTowerBP-BASELI...	CREATE_COMPLETE	2019-10-15 06:52:11 UTC+0530	Configure the Cross-Account IAM Security Roles for th...
StackSet-AWSControlTowerBP-BASELI...	CREATE_COMPLETE	2019-10-15 06:52:10 UTC+0530	Configure AWS Config and SNS Notification Forward i...

The previous console is being deprecated in favor of the new console

The previous console is being deprecated in favor of this new AWS CloudFormation Console, that we are continually improving based on customer feedback. There will be no new features added or improvements made to the previous console. We will continue to support bug fixes and patches until 03/31/2020 after which the previous console will no longer be accessible.

CloudFormation

Stacks

Stack details

- Drifts
- StackSets
- Exports

Designer

CloudFormation registry

Resource types

Previous console

Feedback

CloudFormation

Stacks (13)

Filter by stack name

Active

View nested

Overview

Stack ID

am:aws:cloudformation:us-west-2:646632670602:stack/americold-billingapp-nodeapis-dev/b6ed2c0-fb49-1e68-a648-0a13a2454304

Description

The AWS CloudFormation template for this Serverless application

Status

UPDATE_COMPLETE

Root stack

-

Created time

2019-10-31 00:46:03 UTC+0530

Updated time

2019-12-16 06:23:46 UTC+0530

Drift status

NOT_CHECKED

Deleted time

-

Last drift check time

-

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

We can check our **events triggered** by our cloud formation stack.

The previous console is being deprecated in favor of the new console

The previous console is being deprecated in favor of this new AWS CloudFormation Console, that we are continually improving based on customer feedback. There will be no new features added or improvements made to the previous console. We will continue to support bug fixes and patches until 03/31/2020 after which the previous console will no longer be accessible.

CloudFormation

Stacks

Stack details

- Drifts
- StackSets
- Exports

Designer

CloudFormation registry

Resource types

Previous console

Feedback

CloudFormation

Stacks (13)

Filter by stack name

Active

View nested

Events

Search events

Timestamp	Logical ID	Status	Status reason
2019-12-16 06:23:59 UTC+0530	americold-billingapp-nodeapis-dev	UPDATE_COMPLETE	-
2019-12-16 06:23:59 UTC+0530	ApiGatewayDeployment 1576400242335	DELETE_COMPLETE	-
2019-12-16 06:23:59 UTC+0530	LoginLambdaVersionCe eWFrRfisThcnr111yu hPvacp7n3q5lUYUFKGP YM	DELETE_SKIPPED	-
2019-12-16 06:23:58 UTC+0530	ApiGatewayDeployment 1576400242335	DELETE_IN_PROGRESS	-
2019-12-16 06:23:57 UTC+0530	americold-billingapp-nodeapis-dev	UPDATE_COMPLETE_CLEANUP_IN_PROGRESS	-

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

Go to the Lambda service.

The screenshot shows the AWS Services dashboard. The 'Services' button is highlighted with a red box. In the search bar, 'lamb' is typed, and 'Lambda' is the top result, also highlighted with a red box. The Lambda service card is displayed, showing its description: 'Run Code without Thinking about Servers'. To the right, there's a grid of other services like CloudWatch Metrics, AWS Lambda, AWS Data Exchange, and IoT Core. A sidebar on the left lists services like History, CloudFormation, S3, etc. A sidebar on the right shows recently added services and a navigation menu.

Following is the **Lambda function** that got updated.

The screenshot shows the AWS Lambda Functions list. The 'Functions' tab is selected in the sidebar. There are 19 functions listed. One function, 'americold-billingapp-nodeapis-dev-Login', is highlighted with a red box. The table columns include Function name, Description, Runtime, Code size, and Last modified. The function details show it's a Node.js 10.x function with 21.5 MB code size, last modified 3 days ago.

Function name	Description	Runtime	Code size	Last modified
testLambdaFunctions-DefaultFunction-W33NN2DD3ZKL	Default function	.NET Core 2.1 (C#/PowerShell)	487.4 kB	last month
americold-billingapp-nodeapis-dev-Login		Node.js 10.x	21.5 MB	3 days ago
UserSearchScheduler		.NET Core 2.1 (C#/PowerShell)	3.7 MB	14 hours ago
customerscorecardcalculatemetrics		.NET Core 2.1 (C#/PowerShell)	1.0 MB	9 days ago
processcustomerscorecardviewsdelta		.NET Core 2.1 (C#/PowerShell)	631.5 kB	17 days ago
customerscorecardmergeinventorymetrics		.NET Core 2.1 (C#/PowerShell)	488.7 kB	13 days ago
awsserverlessapi-AspNetCoreFunction-10Z8VWV1YKYY1		.NET Core 2.1 (C#/PowerShell)	8.0 MB	21 days ago
testLambdaFunctions1-DefaultFunction-OPOW8IKAYCXE	Default function	.NET Core 2.1 (C#/PowerShell)	630.9 kB	22 days ago
PowerBIRefreshIntervalNotifications		.NET Core 2.1 (C#/PowerShell)	4.4 MB	15 days ago
Americold_CloudFront_Headers		Node.js 10.x	637 bytes	last month

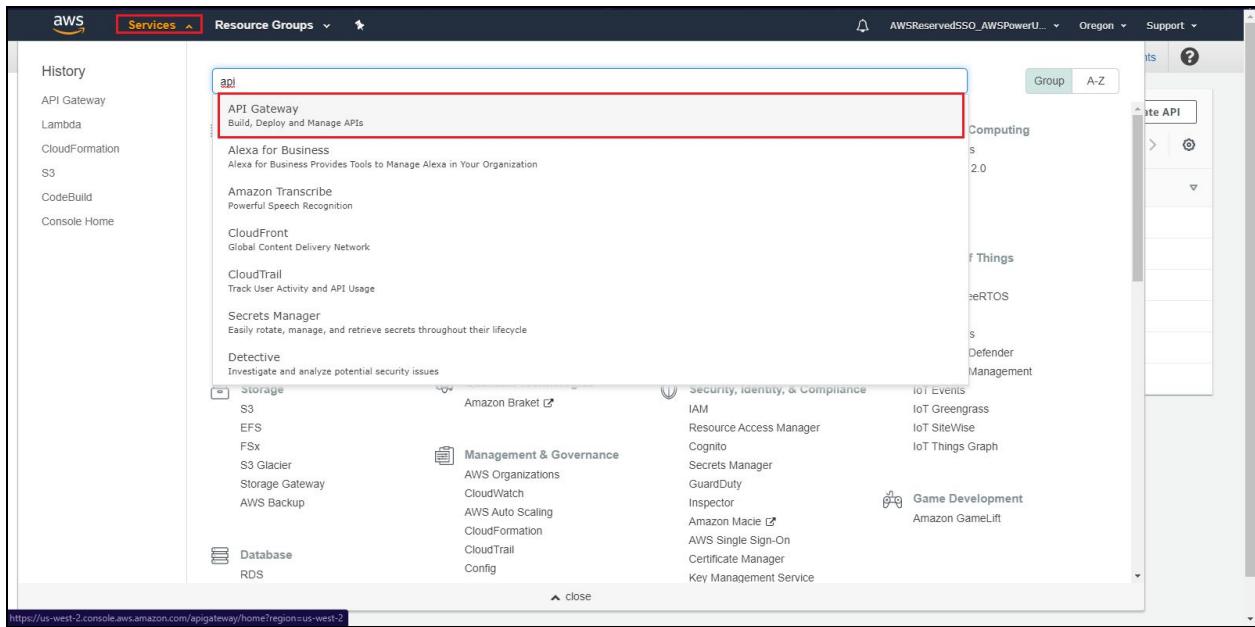
The screenshot shows the AWS Lambda Functions console. The top navigation bar includes 'Services', 'Resource Groups', and 'Oregon'. The main title is 'americold-billingapp-nodeapis-dev-Login'. Below it, there are tabs for 'Throttle', 'Qualifiers', 'Actions', 'Select a test event', 'Test', and 'Save'. A message says 'This function belongs to an application. Click here to manage it.' The 'Configuration' tab is selected. Under 'Designer', there is a section titled 'Go back to application americold-billingapp-nodeapis-dev'. It shows an 'API Gateway' icon with a red box around it, indicating it's the current destination. There is also a 'Layers' section with '(0)' items. A button '+ Add destination' is visible. At the bottom, there are links for 'Feedback', 'English (US)', and 'Privacy Policy Terms of Use'.

Here is the **endpoint** of API gateway.

The screenshot shows the AWS Lambda Functions console. The top navigation bar includes 'Services', 'Resource Groups', and 'Oregon'. The main title is 'americold-billingapp-nodeapis-dev-Login'. Below it, there are tabs for 'Throttle', 'Qualifiers', 'Actions', 'Select a test event', 'Test', and 'Save'. A message says 'This function belongs to an application. Click here to manage it.' The 'Configuration' tab is selected. Under 'Designer', there is a section titled 'Go back to application americold-billingapp-nodeapis-dev'. It shows an 'API Gateway' icon with a red box around it, indicating it's the current destination. There is also a 'Layers' section with '(0)' items. A button '+ Add destination' is visible. At the bottom, there are links for 'Feedback', 'English (US)', and 'Privacy Policy Terms of Use'. A red box highlights the 'API Gateway' section, which contains the following details:

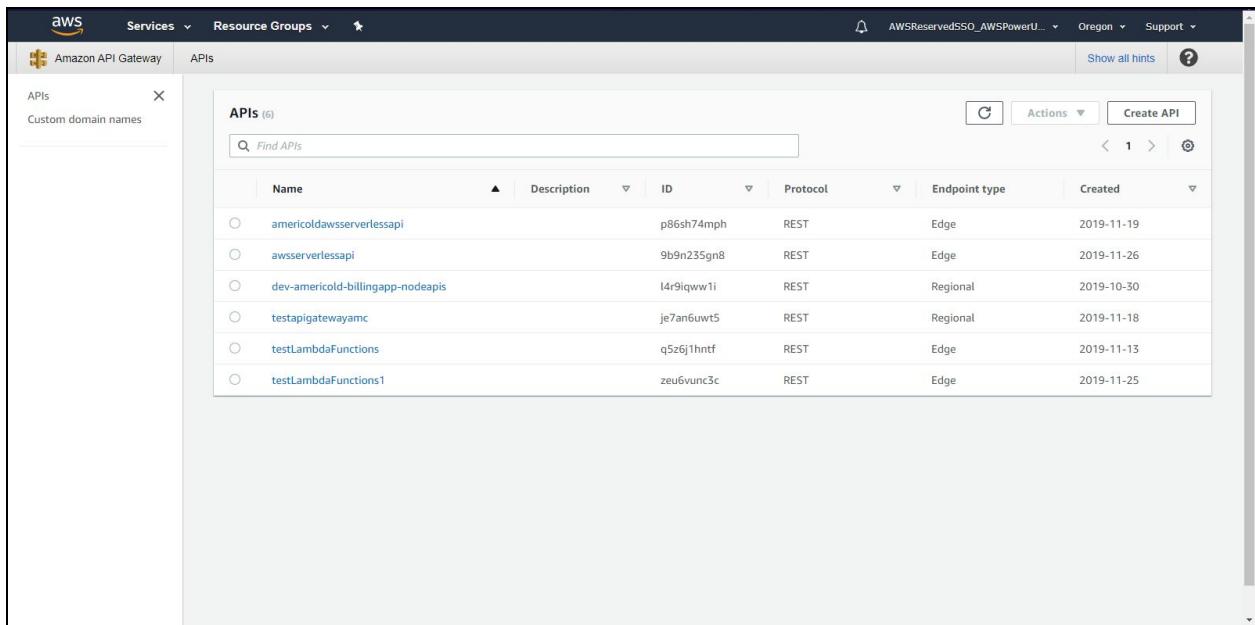
API Gateway
dev-americold-billingapp-nodeapis
arn:aws:execute-api:us-west-2:646632670602:14r9iqww1/*
► API endpoint: https://14r9iqww1.execute-api.us-west-2.amazonaws.com/dev/api/inventoryConsolidation API Type: rest Authorization: NONE
Enabled <input type="button" value="Delete"/>

Goto AWS console and select **API gateway** service.



The screenshot shows the AWS Services menu with the search bar set to "api". The "API Gateway" service is highlighted with a red box. The menu also lists other services like Alexa for Business, Amazon Transcribe, CloudFront, CloudTrail, Secrets Manager, Detective, Storage (S3, EFS, FSx, S3 Glacier, Storage Gateway, AWS Backup), Database (RDS), and various AWS Management services (IAM, Resource Access Manager, Cognito, Secrets Manager, GuardDuty, Inspector, Amazon Macie, AWS Single Sign-On, Certificate Manager, Key Management Service). The URL at the bottom is https://us-west-2.console.aws.amazon.com/apigateway/home?region=us-west-2.

Select the **API gateway** which is created/updated.



The screenshot shows the AWS API Gateway service page. The left sidebar has "Amazon API Gateway" selected. The main area displays a table titled "APIs (6)" with columns: Name, Description, ID, Protocol, Endpoint type, and Created. The table lists six APIs:

Name	Description	ID	Protocol	Endpoint type	Created
americoldawsserverlessapi		p86sh74mph	REST	Edge	2019-11-19
awsserverlessapi		9b9n235gn8	REST	Edge	2019-11-26
dev-americold-billingapp-nodeapis		l4r9iqwv1i	REST	Regional	2019-10-30
testapigatewayamc		je7an6uwrt5	REST	Regional	2019-11-18
testLambdaFunctions		q5z6j1hntf	REST	Edge	2019-11-13
testLambdaFunctions1		zeu6vunc3c	REST	Edge	2019-11-25

Here is the dashboard of the API gateway.

The screenshot shows the AWS API Gateway interface. The left sidebar has a red box around the 'Resources' link under the 'APIs' section. The main content area shows a tree structure for the resource path `/api`. Under `/api`, there are several methods listed: `/customerData` (GET, OPTIONS), `/deleteSavedSearch` (DELETE, OPTIONS), `/facilityData` (GET, OPTIONS), `/inventoryConsolidation` (OPTIONS, POST), `/inventoryTransaction` (OPTIONS, POST), `/savedSearch` (OPTIONS, POST), `/savedSearchApply` (GET, OPTIONS), `/savedSearchView` (GET, OPTIONS), and `/userinfo` (GET). A message at the top right says "No methods defined for the resource." The bottom of the screen includes standard AWS footer links: Feedback, English (US), Privacy Policy, and Terms of Use.

For API stage URL click on the stages and click on dev

The screenshot shows the AWS API Gateway interface. The left sidebar has a red box around the 'Stages' link under the 'APIs' section. The main content area shows a list of stages for the API. There is one stage named `dev` with a small icon next to it. A button labeled "Create" is visible above the stage list. A message at the top right says "Select a stage". The bottom of the screen includes standard AWS footer links: https://us-west-2.console.aws.amazon.com/apigateway/home?region=us-west-2#apis/449icqww1/stages, Privacy Policy, and Terms of Use.

Here we can seed the Node.js API url.

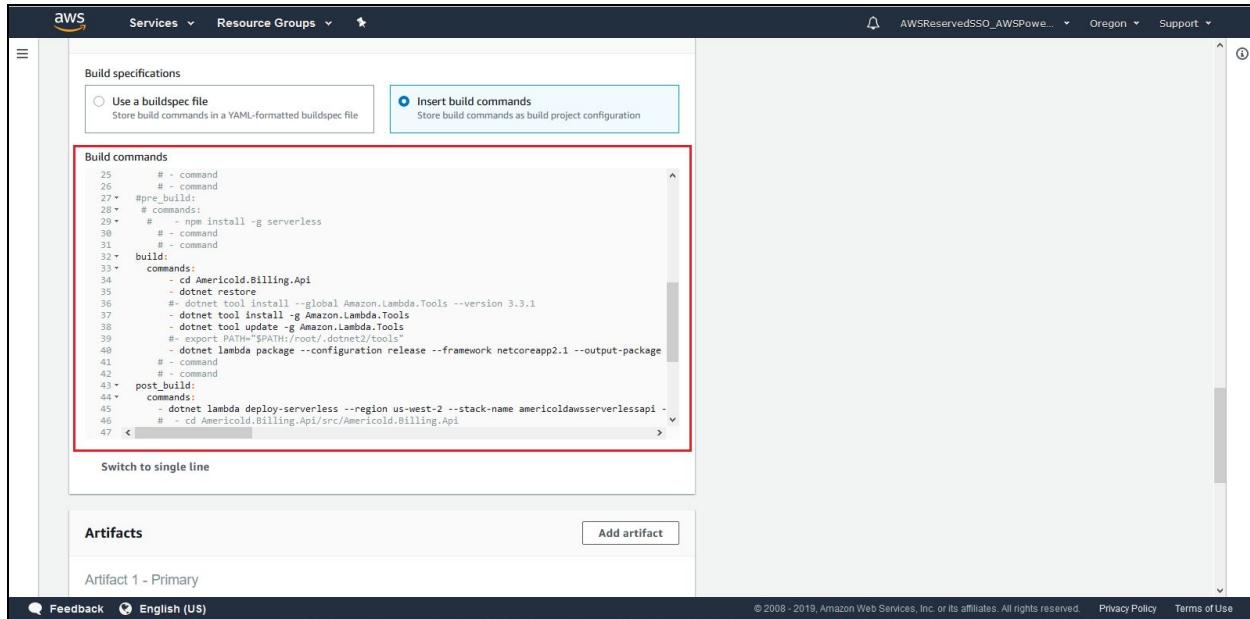
The screenshot shows the AWS API Gateway Stage Editor for the 'dev' stage of the 'dev-americold...' API. The left sidebar lists various API resources like Resources, Stages, Authorizers, etc. The main panel shows the 'dev Stage Editor' with the 'Invoke URL' field highlighted. The 'Settings' tab is active, displaying cache settings, default method throttling (rate 10000 requests per second, burst 5000 requests), and a WAF section. A red box highlights the 'Invoke URL' field.

For backend API's we are configuring AWS WAF to provide security.

The screenshot shows the AWS API Gateway Stage Editor for the 'dev' stage of the 'dev-americold...' API. The left sidebar lists various API resources like Resources, Stages, Authorizers, etc. The main panel shows the 'dev Stage Editor' with the 'Web ACL' dropdown open, showing 'Americold_Backend' and 'None'. The 'Americold_Backend' option is selected and highlighted with a red box. The 'Save Changes' button is visible at the bottom right.

Similarly, we are creating the CI/CD pipeline for the ASP dotnet core application. Following is the buildspec.yml file that we are using for integration and deployment for the ASP dotnetcore application.

Following is the **buildspec.yml** file for ASP .net core application.



The screenshot shows the AWS Lambda Build Spec configuration interface. At the top, there are two radio button options: "Use a buildspec file" (selected) and "Insert build commands". Below this is a code editor containing the buildspec.yml file content. A red box highlights the "Build commands" section. At the bottom of the code editor, there is a "Switch to single line" link. To the right of the code editor, there is an "Artifacts" section with an "Add artifact" button. The bottom of the screen shows standard AWS navigation links: Feedback, English (US), Copyright notice (© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.), Privacy Policy, and Terms of Use.

```
25      # - command
26      # - command
27  ▼ # pre_build:
28  ▼ # commands:
29  ▶   # - npm install -g serverless
30      # - command
31      # - command
32  ▶   build:
33    ▶   commands:
34      - cd Americold.Billing.Api
35      - dotnet restore
36      # -dotnet tool install --global Amazon.Lambda.Tools --version 3.3.1
37      - dotnet tool install -g Amazon.Lambda.Tools
38      - dotnet tool update -g Amazon.Lambda.Tools
39      # - export PATH=$PATH:/root/.dotnet/tools
40      - dotnet lambda package --configuration release --framework netcoreapp2.1 --output-package
41      # - command
42      # - command
43  ▶   post_build:
44    ▶   commands:
45      - dotnet lambda deploy-serverless --region us-west-2 --stack-name americoldawsserverlessapi -
46      # - cd Americold.Billing.Api/src/Americold.Billing.Api
```

Below is the step by step explanation for the above script:

Install stage:

- **runtime-versions:**

dotnet : 2.2:It defines the runtime environment and version for the application

Build Stage:

- **cd Americold.Billing.Api:**Go to the folder called Americold.Billing.Api to install the dependencies
- **dotnet restore:**Following command uses NuGet to **restore** dependencies
- **dotnet tool install -g Amazon.Lambda.Tools**
dotnet tool update -g Amazon.Lambda.Tools
dotnet clean
dotnet build

Following above commands used for installing required lambda tools for dotnet application.

Post-Build stage:

- `dotnet lambda deploy-serverless --region us-west-2 --stack-name americoldawsserverlessapi --s3-bucket americold.dev.services.billingapp.netcore --s3-prefix Americold.Billing.Api/ --template-parameters "EnvironmentName=Dev;LambdaSecurityGroups=\"sg-02578287bc42d72ea\";Lambda SubnetID=\"subnet-07459ad3400f384b1,subnet-026ad089d2e15eb1f,subnet-0681558 37bd958f7c\""`

To deploy ASP .net core application using serverless.template file we will run the following above command. Here we have added the additional parameters to define the AWS region to deploy, name of the cloudformation stack, AWS S3 bucket name, the name of the environment and VPC.