



CI/CD flow for ANgular application using cross-account deployment

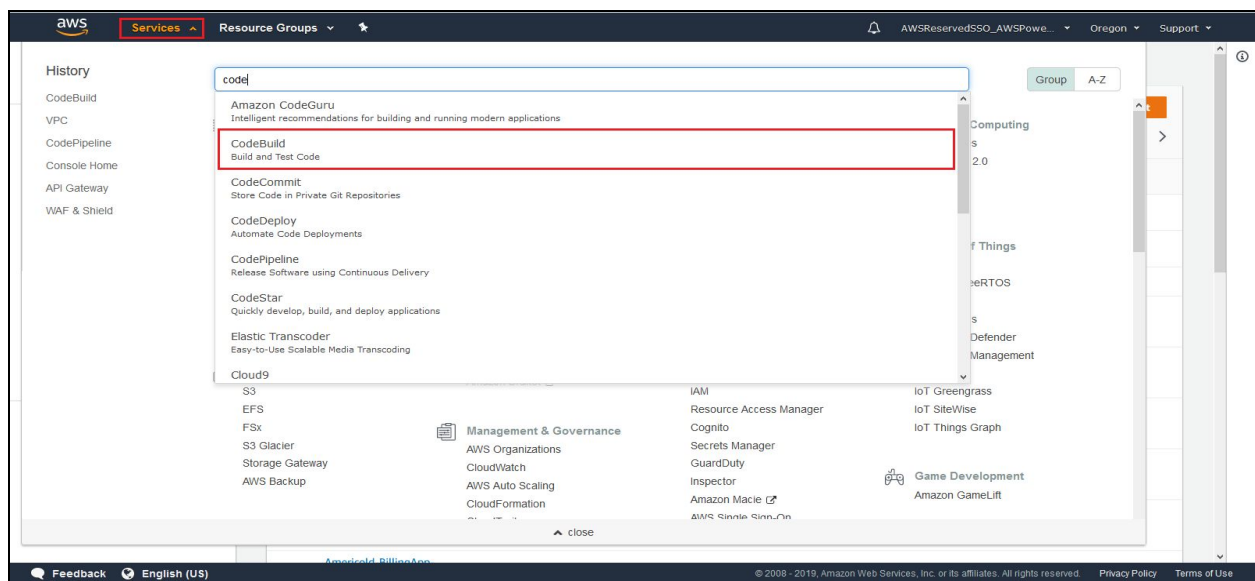
Agenda:

CodeBuild: AWS code build is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With code build, you don't need to provision, manage, and scale your own build servers. code build 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. With code build, you are charged by the minute for the compute resources you use.

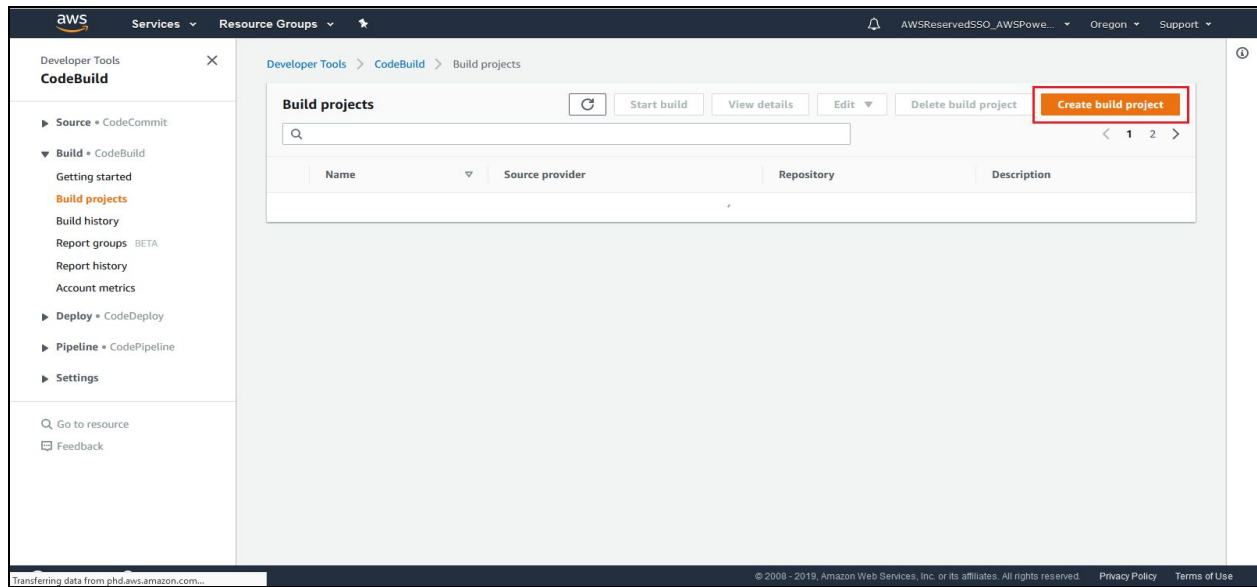
CodePipeline: AWS code pipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. code pipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. You can easily integrate the AWS code pipeline with third-party services such as git hub or with your own custom plugin. With the AWS code pipeline, you only pay for what you use. There are no upfront fees or long-term commitments.

By the end of the document, we learn how to create a CI-CD pipeline for Angular application using the cross-account.

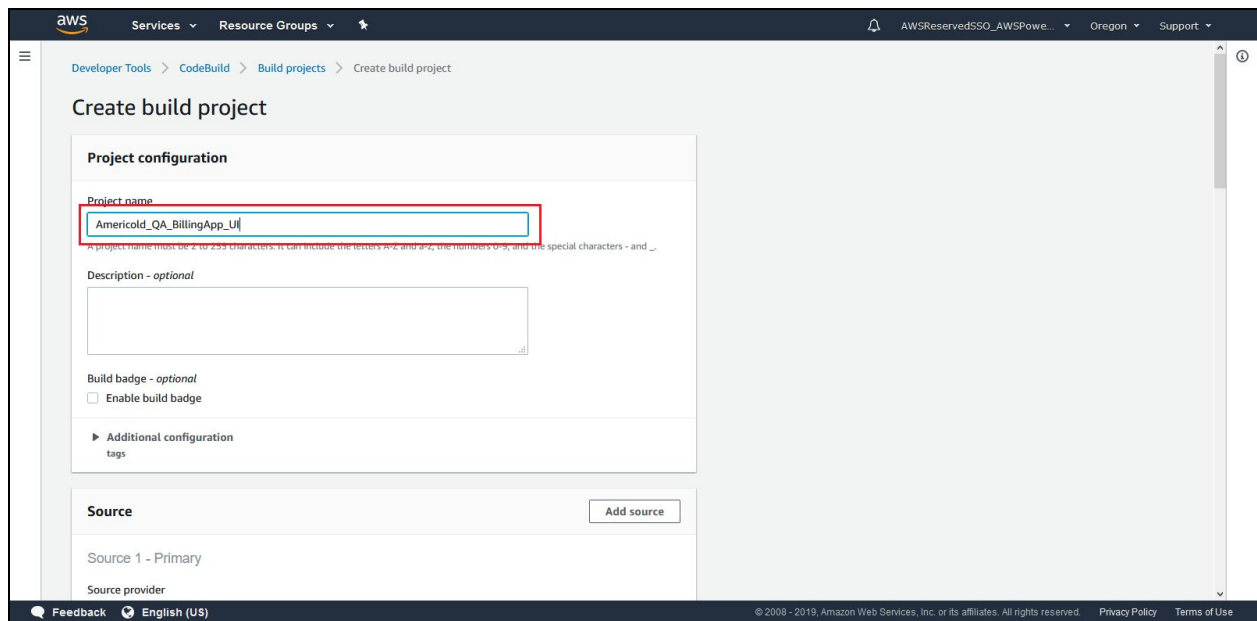
First, we need to login to the AWS console and search for **code build** service.



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



Enter the name of the project.



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

The screenshot shows the 'Source' configuration page in the AWS CodeBuild console. The 'Source provider' is set to 'AWS CodeCommit'. The 'Repository' is 'Americold.BillingApp.UI'. The 'Reference type' is 'Branch'. The 'Branch' is 'QA'. The 'Commit ID' is optional and empty. The 'Source version' is 'refs/heads/QA' with a commit hash '57f5fc84'. The 'Additional configuration' section is expanded, showing 'Git clone depth' and 'Git submodules'.

Source

Source 1 - Primary

Source provider: AWS CodeCommit

Repository: Americold.BillingApp.UI

Reference type: Branch

Branch: QA

Commit ID - optional:

Source version info: refs/heads/QA, 57f5fc84 Conflicts resolved in the console.

Additional configuration: Git clone depth, Git submodules

Under the 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.

The screenshot shows the 'Environment' configuration page in the AWS CodeBuild console. The 'Environment image' is 'Managed image'. The 'Operating system' is 'Ubuntu'. The 'Runtime(s)' is 'Standard'. The 'Image' is 'aws/codebuild/standard:2.0'. The 'Image version' is 'Always use the latest image for this runtime version'. The 'Environment type' is 'Linux'. The 'Privileged' checkbox is unchecked.

Environment

Environment image: Managed image

Operating system: Ubuntu

Runtime(s): Standard

Image: aws/codebuild/standard:2.0

Image version: Always use the latest image for this runtime version

Environment type: Linux

Privileged:

The screenshot shows the AWS CodeBuild console interface. At the top, there's a navigation bar with 'aws' logo, 'Services', 'Resource Groups', and a user profile. Below the navigation bar, the main content area is divided into sections. The 'Service role' section has two radio buttons: 'New service role' (selected) and 'Existing service role'. Below this, the 'Role name' field is populated with 'codebuild-Amercold_QA_BillingApp_UI-service-role'. The 'Buildspec' section has two radio buttons: 'Use a buildspec file' (selected) and 'Insert build commands'. Below this, there's a 'Buildspec name - optional' field. The footer contains 'Feedback', 'English (US)', and copyright information.

Always use the latest image for this runtime version

Environment type

Linux

Privileged

☐ Enable this flag if you want to build Docker images or want your builds to get elevated privileges

Service role

☒ New service role
Create a service role in your account

☐ Existing service role
Choose an existing service role from your account

Role name

codebuild-Amercold_QA_BillingApp_UI-service-role

Type your service role name

► Additional configuration
Timeout, certificate, VPC, compute type, environment variables

Buildspec

Build specifications

☒ Use a buildspec file
Store build commands in a YAML-formatted buildspec file

☐ Insert build commands
Store build commands as build project configuration

Buildspec name - optional
By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Under the build spec 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.

The screenshot shows the AWS CodeBuild console interface, specifically the 'Buildspec' section. The 'Insert build commands' radio button is selected and highlighted with a red box. Below this, there's a 'Build commands' section with a text input field and a 'Switch to editor' button. The 'Artifacts' section shows 'Artifact 1 - Primary' with a 'Type' dropdown set to 'No artifacts'. The footer contains 'Feedback', 'English (US)', and copyright information.

► Additional configuration
Timeout, certificate, VPC, compute type, environment variables

Buildspec

Build specifications

☐ Use a buildspec file
Store build commands in a YAML-formatted buildspec file

☒ Insert build commands
Store build commands as build project configuration

Build commands

Enter commands you want to run during the build phase. Separate each build command with "&&." For example, "mvn test && mvn package." Use a buildspec file to run commands in other phases or if you have a long list of commands.

Switch to editor

Artifacts

Add artifact

Artifact 1 - Primary

Type

No artifacts

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

► Additional configuration
Cache, encryption key

Logs

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

BUILDSPEC

```

1 version: 0.2
2
3 #env:
4 #variables:
5   # key: "value"
6   # key: "value"
7 #parameter-store:
8   # key: "value"
9   # key: "value"
10 #git-credential-helper: yes
11
12 phases:
13 install:
14   # If you use the Ubuntu standard image 2.0 or later, you must specify runtime-versions.
15   # If you specify runtime-versions and use an image other than Ubuntu standard image 2.0, the build fails.
16   runtime-versions:
17     nodejs: 10
18   # name: version
19   # name: version
20 #commands:
21 # - command
22 # - command
23 pre_build:
24   commands:
25     - apt install jq -y
26     - unset AWS_SESSION_TOKEN
27     - export aws_credentials=west-2
28     - temp_role=$(aws sts assume-role --role-arn "arn:aws:iam::81737090411:role/Americold_CrossAccount_Role" --role-session-name "qa")
29
30     - export aws_access_key_id=$(echo $temp_role | jq .Credentials.AccessKeyId | xargs)
31     - export aws_secret_access_key=$(echo $temp_role | jq .Credentials.SecretAccessKey | xargs)
32     - export aws_session_token=$(echo $temp_role | jq .Credentials.SessionToken | xargs)
33     - DYNAMODB_ID=$(aws cloudfront create-invalidation --distribution-id EISFQOZEVWVC --paths "/" | jq ".Invalidation[].ID" | cut -d '"' -f 2)
34     - aws cloudfront wait invalidation-completed --id $DYNAMODB_ID --distribution-id EISFQOZEVWVC
35   # - command
36   # - command
37 build:
38   commands:
39     - npm install
40     - npm install -g @angular/cli
41     - ng build --prod --configuration=qa
42     - DYNAMODB_ID=$(aws cloudfront create-invalidation --distribution-id EISFQOZEVWVC --paths "/" | jq ".Invalidation[].ID" | cut -d '"' -f 2)
43   # - command
44   # - command
45 post_build:
46   commands:
47     - aws s3 mv s3://americold.billingapp.us.af/ --recursive
48     - aws s3 cp dist/inventory-consolidation-us s3://americold.billingapp.us.af/ --recursive --sse
49   # - command
50   # - command
51 artifacts:
52 #files:
```

Install stage:

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

- **apt install jq -y:** Install JQ modules using the following command
- **unset AWS_SESSION_TOKEN:** Clear the AWS session token
- **export AWS_REGION=us-east-1:** Set the AWS region
- **temp_role=\$(aws sts assume-role --role-arn "arn:aws:iam::XXXXXXXXXXXX:role/" --role-session-name "QA"):** Create IAM role for cross account in destination account and give the role ARN as temp_role.
- **export AWS_ACCESS_KEY_ID=\$(echo \$temp_role | jq.Credentials.AccessKeyId | xargs)**
export AWS_SECRET_ACCESS_KEY=\$(echo \$temp_role |
jq.Credentials.SecretAccessKey | xargs)
export AWS_SESSION_TOKEN=\$(echo \$temp_role | jq .Credentials.SessionToken |
xargs) :

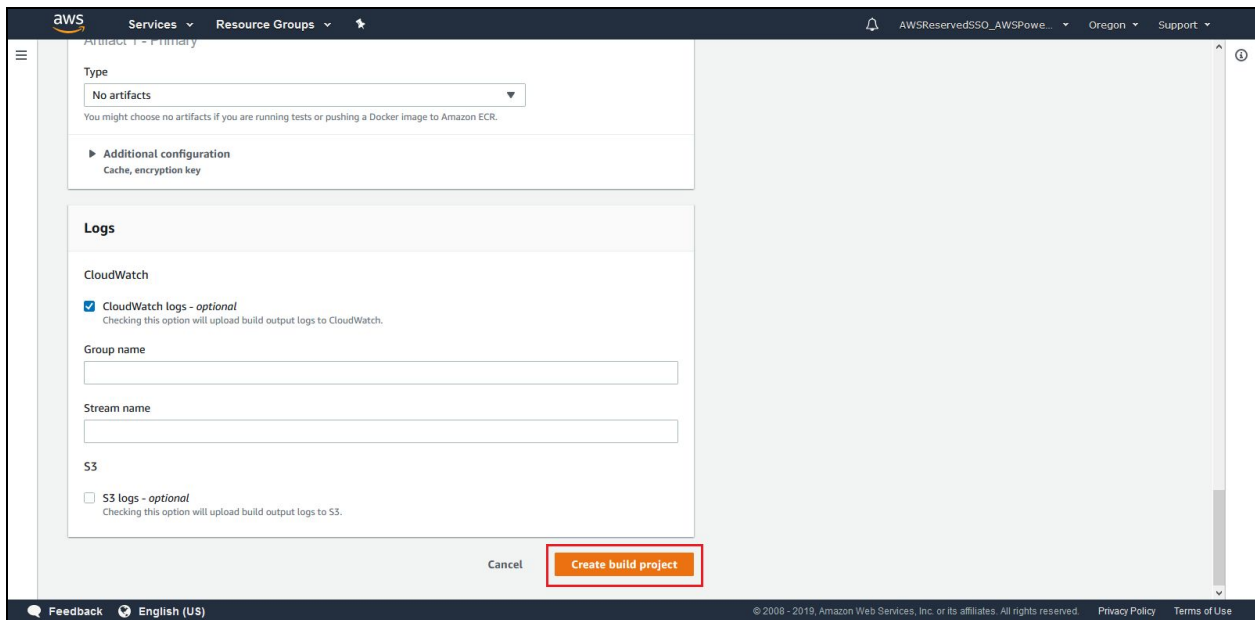
- **INVALIDATION_ID=\$(aws cloudfront create-invalidation --distribution-id E2SBFO2PBYY6MC --paths '/*' | jq '.Invalidation | .Id' | cut -d \" -f2) :**
Create invalidation for cloudfront distribution to clear cache
- **aws cloudfront wait invalidation-completed --id \$INVALIDATION_ID --distribution-id E2SBFO2PBYY6MC:** Wait for invalidation to be completed
- **npm install:** Install node modules using the following command
- **npm install -g @angular/cli:** Install angular cli using the following command

- **ng build --aot --configuration=qa:** Generate build artifact using the following command
- **INVALIDATION_ID=\$(aws cloudfront create-invalidation --distribution-id E3GILM5AHJTX5 --paths '/*' | jq '.Invalidation | .Id' | cut -d \" -f2)**
- **--distribution-id E3GILM5AHJTX5:** Create invalidation for cloudfront distribution to clear cache

Post Build:

- **aws s3 rm s3://americold.billingapp.ui.qa/ --recursive:** To remove the billing app UI artifacts in S3
- **aws s3 cp dist/Inventory-Consolidation-UI s3://americold.billingapp.ui.qa/ --recursive --sse:** To copy dist folder to S3 bucket

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



The screenshot shows the AWS IAM console interface for creating a build project. The 'Type' dropdown is set to 'No artifacts'. Under 'Additional configuration', the 'Cache, encryption key' section is visible. In the 'Logs' section, the 'CloudWatch' checkbox is checked, and the 'S3' checkbox is unchecked. The 'Create build project' button is highlighted with a red box.

aws Services Resource Groups

Artifact Type: Primary

Type: No artifacts

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

Additional configuration: Cache, encryption key

Logs

CloudWatch

☒ CloudWatch logs - optional
Checking this option will upload build output logs to CloudWatch.

Group name:

Stream name:

S3

☐ S3 logs - optional
Checking this option will upload build output logs to S3.

Cancel Create build project

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

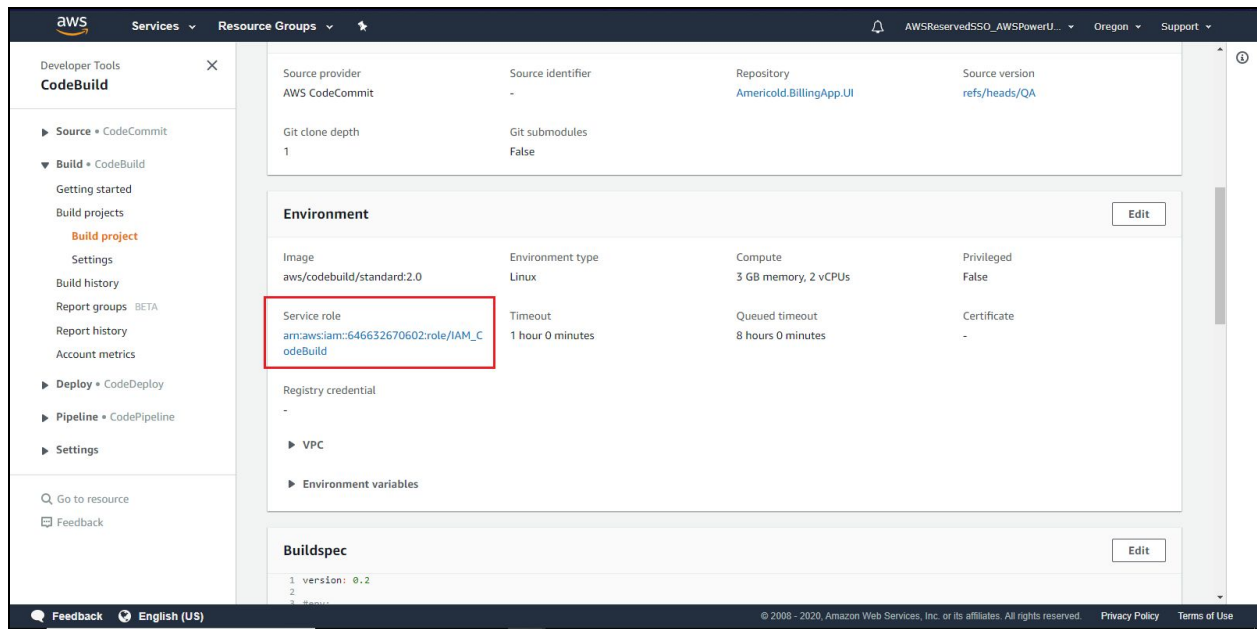
The new build project has been created and the dashboard is shown below.

The screenshot shows the AWS CodeBuild console dashboard for a newly created build project named 'Americold_QA_BillingApp_UI'. The left sidebar contains navigation links for 'Source', 'Build', 'Deploy', 'Pipeline', and 'Settings'. The main content area displays the project's configuration, including the source provider (AWS CodeCommit), primary repository (Americold.BillingApp.UI), artifacts upload location, and build badge status (Disabled). Below the configuration, there are tabs for 'Build history', 'Build details', 'Build triggers', and 'Metrics'. The 'Build details' tab is currently selected, showing the project configuration and source information. The 'Start build' button is visible in the top right corner.

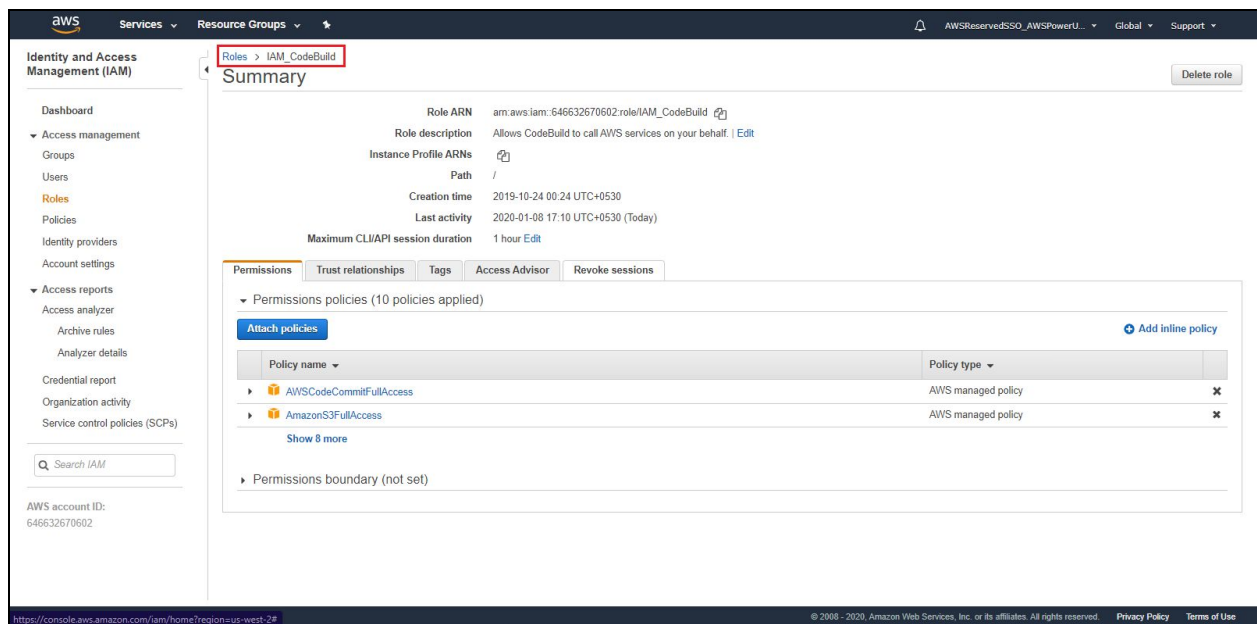
Now goto details and goto **environment** section in build details.

The screenshot shows the AWS CodeBuild console 'Build details' section for the 'Americold_QA_BillingApp_UI' build project. The 'Build details' tab is highlighted with a red box. The main content area displays the project configuration, including the source provider (AWS CodeCommit), primary repository (Americold.BillingApp.UI), artifacts upload location, and build badge status (Disabled). Below the configuration, there are tabs for 'Build history', 'Build details', 'Build triggers', and 'Metrics'. The 'Build details' tab is currently selected, showing the project configuration and source information. The 'Start build' button is visible in the top right corner.

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



Now 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.



Now click on **Attach policies** to add the policies for the IAM Role.

The screenshot shows the AWS IAM console interface. On the left, the 'Identity and Access Management (IAM)' sidebar is visible with various navigation options. The main content area displays the 'Summary' page for the role 'IAM_CodeBuild'. Below the summary, the 'Permissions' tab is active, showing a list of 10 applied policies. The 'Attach policies' button is highlighted with a red box.

Policy name	Policy type
AWSCodeCommitFullAccess	AWS managed policy
AmazonS3FullAccess	AWS managed policy
CloudFrontFullAccess	AWS managed policy
CloudWatchLogsFullAccess	AWS managed policy
AWSCodePipelineFullAccess	AWS managed policy
Americold_CrossAccount_Policy_UAT	Managed policy
Americold_CrossAccount_Policy_QA	Managed policy
CodeBuildBasePolicy-Americold-BillingApp_UI-us-west-2	Managed policy
CodeBuildBasePolicy-Americold-BillingApp-Backend-NodeJs-UAT-us-west-2	Managed policy

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

The screenshot shows the 'Add permissions to IAM_CodeBuild' dialog in the AWS IAM console. The 'Attach Permissions' section is active, displaying a search bar and a list of policies. The search bar is highlighted with a red box. At the bottom, the 'Attach policy' button is also highlighted with a red box.

Policy name	Type	Used as
AdministratorAccess	Job function	Permissions policy (9)
AlexaForBusinessDeviceSetup	AWS managed	None
AlexaForBusinessFullAccess	AWS managed	None
AlexaForBusinessGatewayExecution	AWS managed	None
AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None
AlexaForBusinessReadOnlyAccess	AWS managed	None
AmazonAPIGatewayAdministrator	AWS managed	Permissions policy (5)
AmazonAPIGatewayInvokeFullAccess	AWS managed	None
AmazonAPIGatewayPushToCloudWatchLogs	AWS managed	Permissions policy (1)
AmazonAppStreamFullAccess	AWS managed	None
AmazonAppStreamReadOnlyAccess	AWS managed	None
AmazonAppStreamServiceAccess	AWS managed	None
AmazonAthenaFullAccess	AWS managed	None
AmazonDocumentDBFullAccess	AWS managed	None

Here we have attached the policies for **Code commit, S3, cloud front, cloud watch logs, code pipeline, and Cross account policy.**

The screenshot shows the AWS IAM console for the role 'IAM_CodeBuild'. The role is attached to 10 policies. The policies are listed in a table with columns for Policy name, Policy type, and a delete icon.

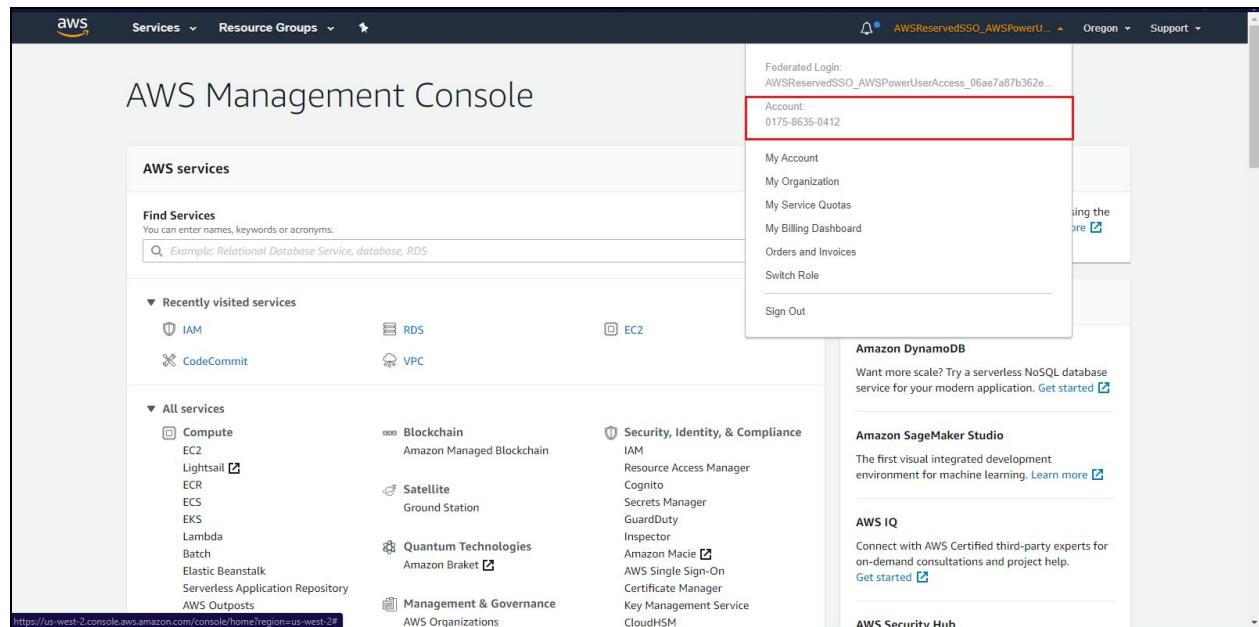
Policy name	Policy type
AWSCodeCommitFullAccess	AWS managed policy
AmazonS3FullAccess	AWS managed policy
CloudFrontFullAccess	AWS managed policy
CloudWatchLogsFullAccess	AWS managed policy
AWSCodePipelineFullAccess	AWS managed policy
Americold_CrossAccount_Policy_UAT	Managed policy
Americold_CrossAccount_Policy_QA	Managed policy
CodeBuildBasePolicy-Americold_BillingApp_UI-us-west-2	Managed policy
CodeBuildBasePolicy-Americold-BillingApp-Backend-NodeJs-UAT-us-west-2	Managed policy

Here we have attached the QA environment account assume role policy to the code build role to provide the necessary permissions to deploy the code into the QA environment.

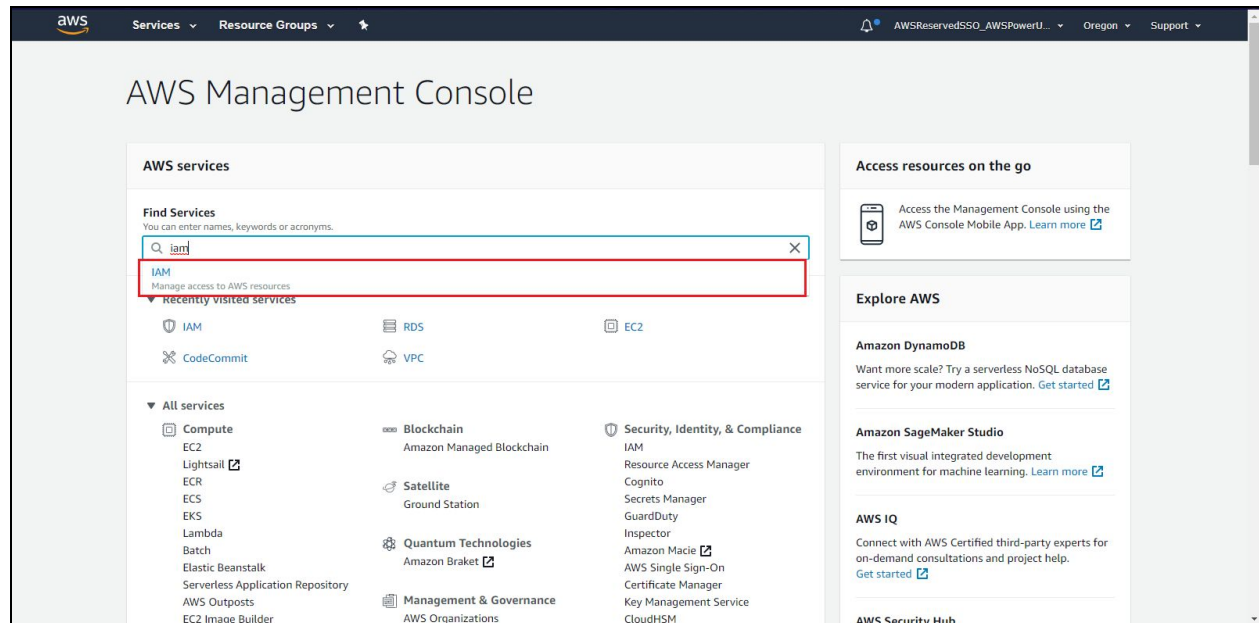
The screenshot shows the AWS IAM console for the policy 'Americold_CrossAccount_Policy_QA'. The policy is a managed policy that allows the role to assume the 'Americold_CrossAccount_Role' in the QA environment. The policy is shown in a JSON format.

```
1 {
2   "Version": "2012-10-17",
3   "Statement": {
4     "Effect": "Allow",
5     "Action": "sts:AssumeRole",
6     "Resource": "arn:aws:iam::017586350412:role/Americold_CrossAccount_Role"
7   }
8 }
```

Login into the AWS QA environment account and we can verify the account number of QA environment with the policy attached in the dev environment.



After login into the QA environment goto AWS IAM service



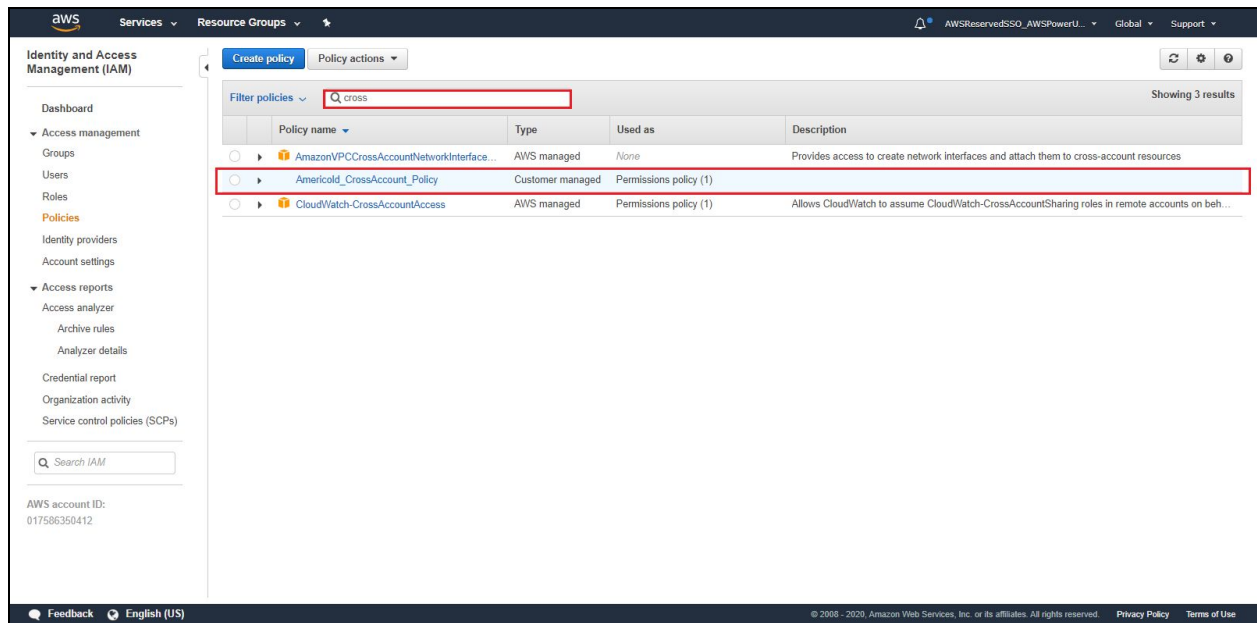
The dashboard of the AWS IAM service will be as shown below

The screenshot shows the AWS IAM dashboard. The left sidebar contains navigation links for Identity and Access Management (IAM), including Dashboard, Access management, Access reports, and Credential report. The main content area displays 'Welcome to Identity and Access Management' with a sign-in link, IAM Resources (Users: 3, Groups: 1, Roles: 26, Identity Providers: 1), and a Security Status section with four items: Activate MFA on your root account, Create individual IAM users, Use groups to assign permissions, and Apply an IAM password policy. A Feature Spotlight video is also visible on the right.

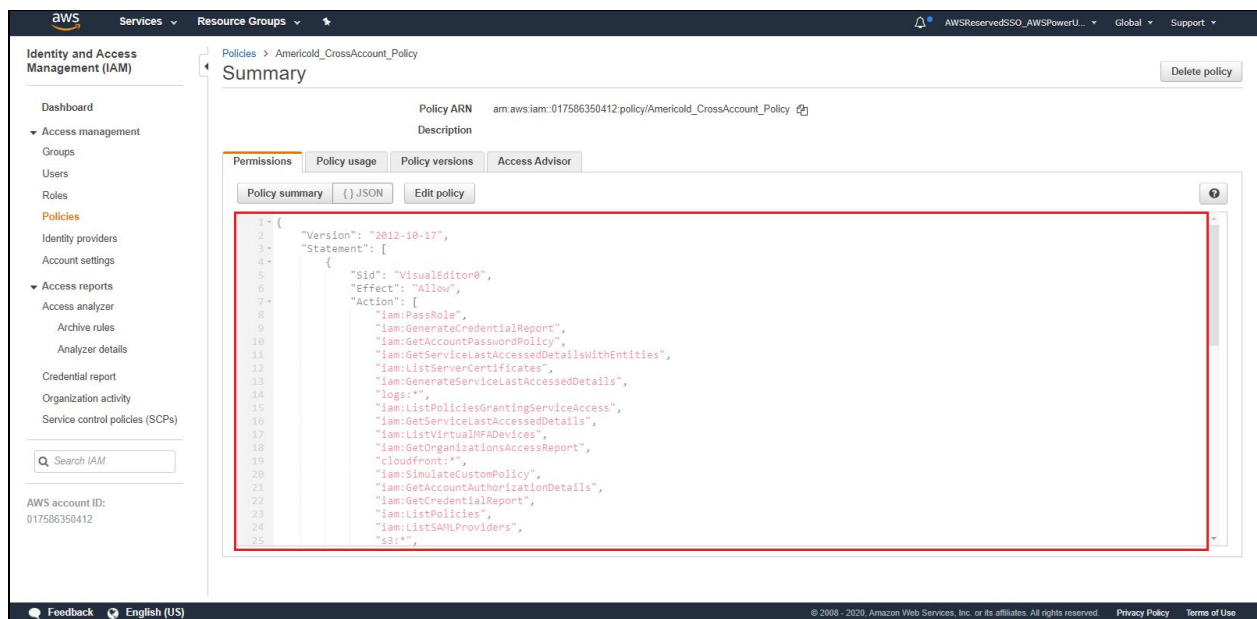
Goto policies section

The screenshot shows the AWS IAM dashboard with the 'Policies' section selected. The left sidebar highlights the 'Policies' link. The main content area displays a table of policies with columns: Policy name, Type, Used as, and Description. The table lists various AWS managed policies, such as AccessAnalyzerServiceRolePolicy, AdministratorAccess, and AmazonAPIGatewayAdministrator. The 'Policies' link in the sidebar is highlighted with a red box.

Here we can see the cross-account access policy which is created previously.

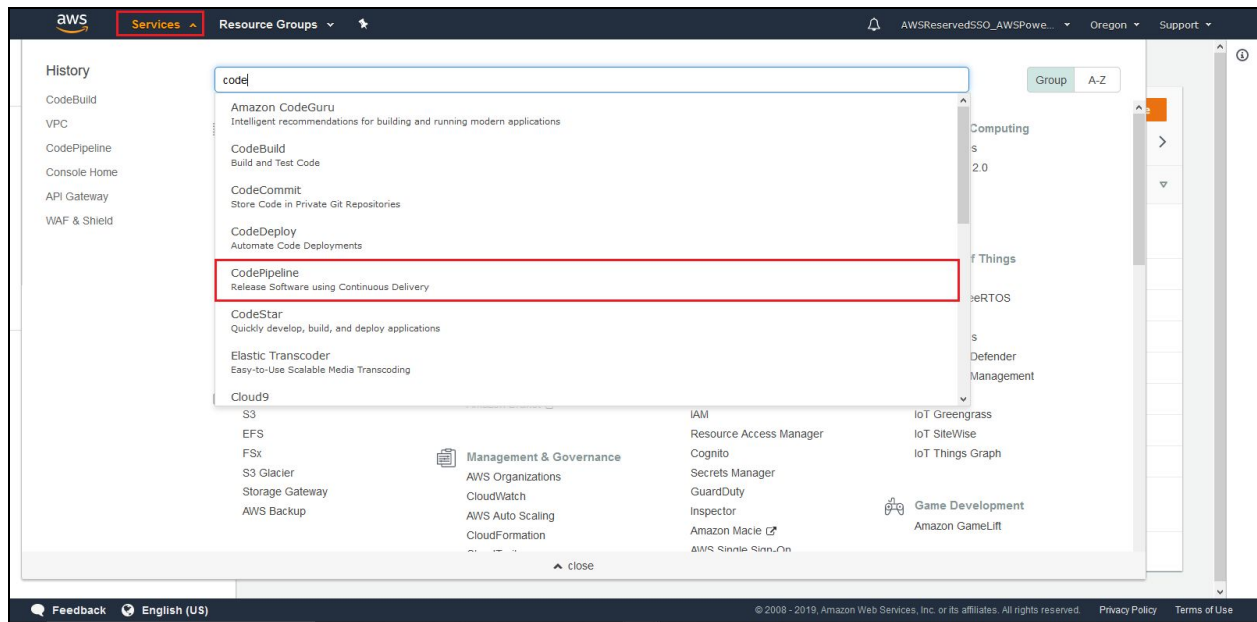


Here we can check the permissions given to the policy which are required for the deployment

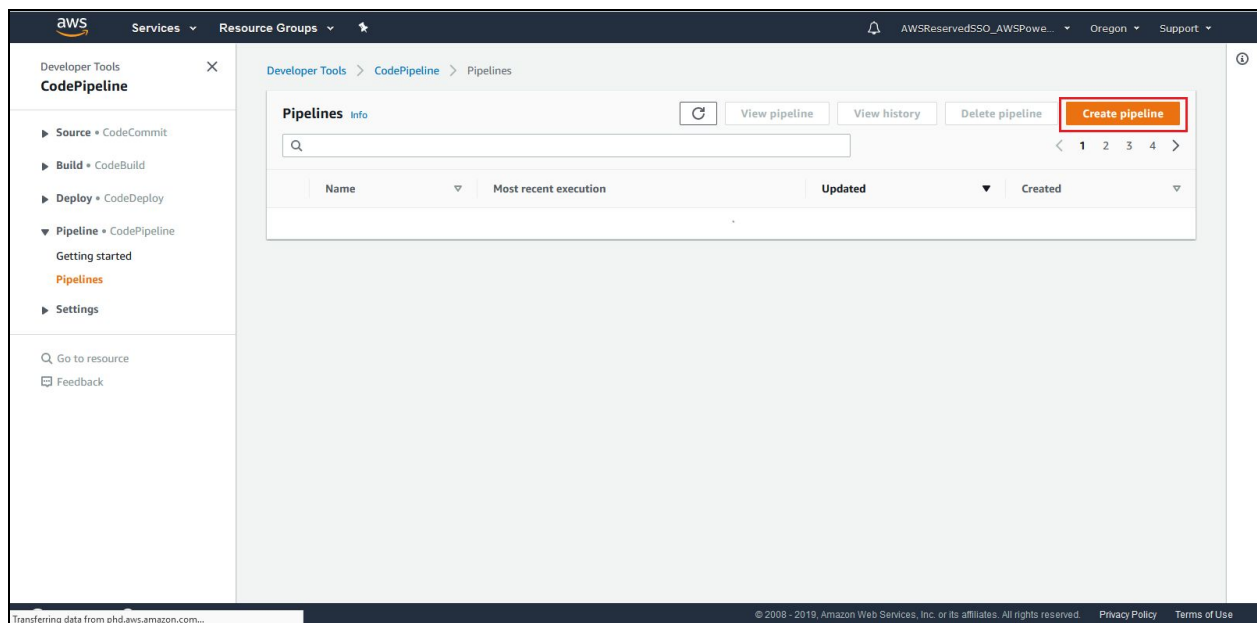


We are going to automate the process for integration and deployment using the code pipeline.

Open AWS console and select code pipeline service.



In the code pipeline dashboard, Goto pipelines and click on **create pipelines**.



Enter the name of the pipeline and choose the service role as a new service role and click on **Next** as shown below.

The screenshot shows the 'Choose pipeline settings' step in the AWS CodePipeline console. The left sidebar lists steps: Step 1 (Choose pipeline settings), Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), and Step 5 (Review). The main content area is titled 'Choose pipeline settings' and contains the following fields:

- Pipeline name:** A text input field containing 'Americold.QA.BillingApp.UI'. Below it, a note says 'Enter the pipeline name. You cannot edit the pipeline name after it is created.' and 'No more than 100 characters'.
- Service role:** Two radio buttons. The first is 'New service role' (selected) with the subtext 'Create a service role in your account'. The second is 'Existing service role' with the subtext 'Choose an existing service role from your account'.
- Role name:** A text input field containing 'AWSCodePipelineServiceRole-us-west-2-Americold.QA.BillingApp.UI'. Below it, a note says 'Type your service role name' and a checked checkbox 'Allow AWS CodePipeline to create a service role so it can be used with this new pipeline'.
- Advanced settings:** A section with a right-pointing arrow.

At the bottom right, there are 'Cancel' and 'Next' buttons. The 'Next' button is highlighted with a red box.

Under add source stage, select the source provider as **AWS CodeCommit**, select the name of the Repository and Branch name. Choose the change detection option as amazon cloudwatch events to enable cloudwatch logs for the pipeline and click on **Next**.

The screenshot shows the 'Add source stage' step in the AWS CodePipeline console. The left sidebar lists steps: Step 1 (Choose pipeline settings), Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), and Step 5 (Review). The main content area is titled 'Add source stage' and contains the following fields:

- Source provider:** A dropdown menu showing 'AWS CodeCommit'.
- Repository name:** A text input field containing 'Americold.QA.BillingApp.UI'.
- Branch name:** A text input field containing 'QA'.
- Change detection options:** Two radio buttons. The first is 'Amazon CloudWatch Events (recommended)' (selected) with the subtext 'Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs'. The second is 'AWS CodePipeline' with the subtext 'Use AWS CodePipeline to check periodically for changes'.

At the bottom, there are 'Cancel', 'Previous', and 'Next' buttons. The 'Next' button is highlighted with a red box.

Under add build stage, select the build provider as **AWS code build**, region as **US West – (Oregon)**, select the build project name that we created in the above steps and click on **Next** as shown below.

The screenshot shows the AWS CodePipeline console interface. On the left, a sidebar lists the steps: Step 1 (Choose pipeline settings), Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), and Step 5 (Review). The 'Add build stage' step is currently selected. The main content area is titled 'Add build stage' and contains a form for configuring a build stage. The form has a section titled 'Build - optional' with the following fields: 'Build provider' (set to 'AWS CodeBuild'), 'Region' (set to 'US West - (Oregon)'), and 'Project name' (set to 'Americold_QA_BillingApp_UI'). There is also an 'Add environment variable' button. At the bottom of the form, there are four buttons: 'Cancel', 'Previous', 'Skip build stage', and 'Next'. The 'Next' button is highlighted with a red box.

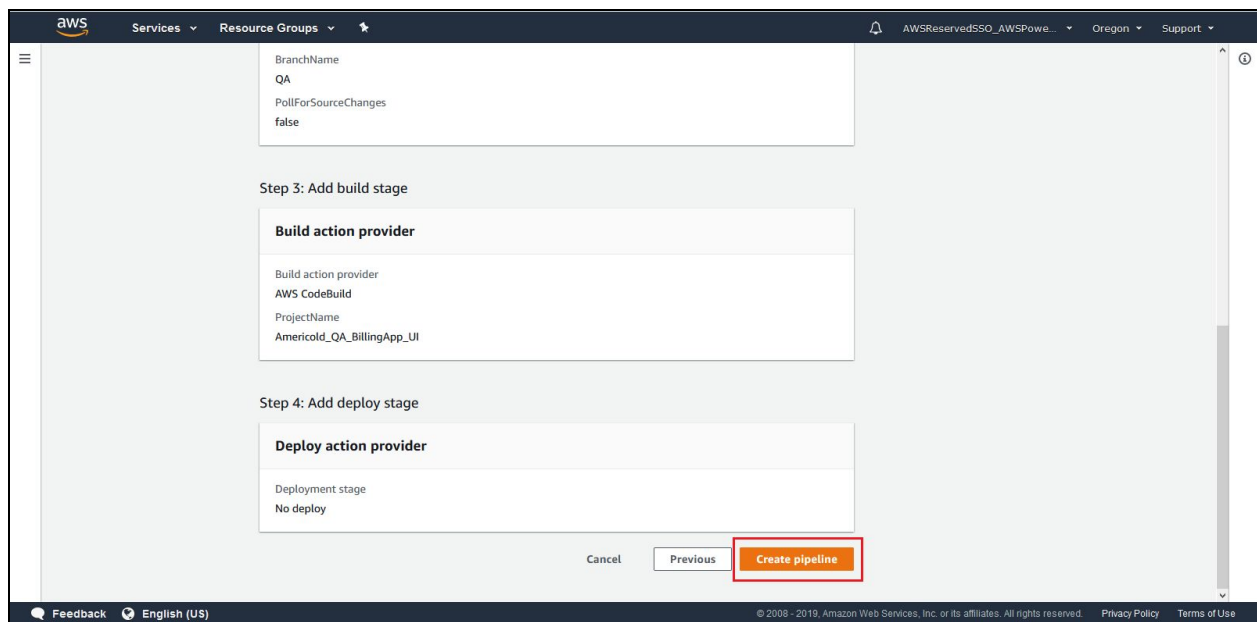
We are deploying through the commands. So, we can skip the deploy stage.

The screenshot shows the AWS CodePipeline console interface. On the left, a sidebar lists the steps: Step 1 (Choose pipeline settings), Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), and Step 5 (Review). The 'Add deploy stage' step is currently selected. The main content area is titled 'Add deploy stage' and contains a form for configuring a deploy stage. The form has a section titled 'Deploy - optional' with the following fields: 'Deploy provider' (set to 'AWS CodeDeploy'). At the bottom of the form, there are four buttons: 'Cancel', 'Previous', 'Skip deploy stage', and 'Next'. The 'Skip deploy stage' button is highlighted with a red box.

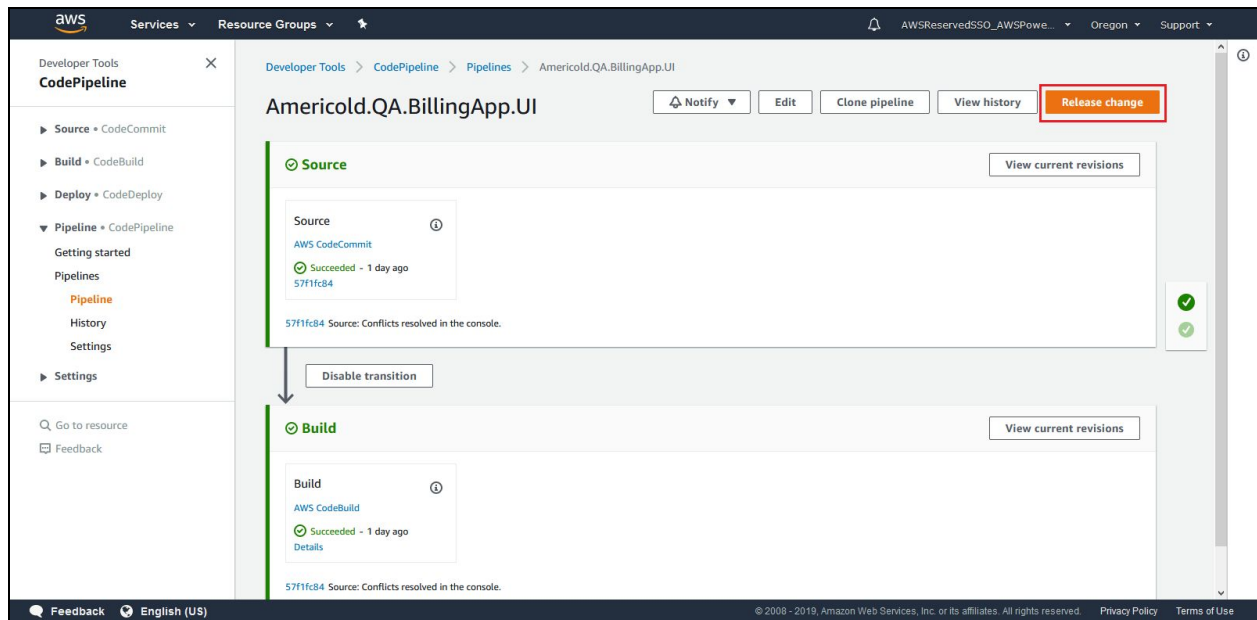
Give the Confirmation as **Skip**



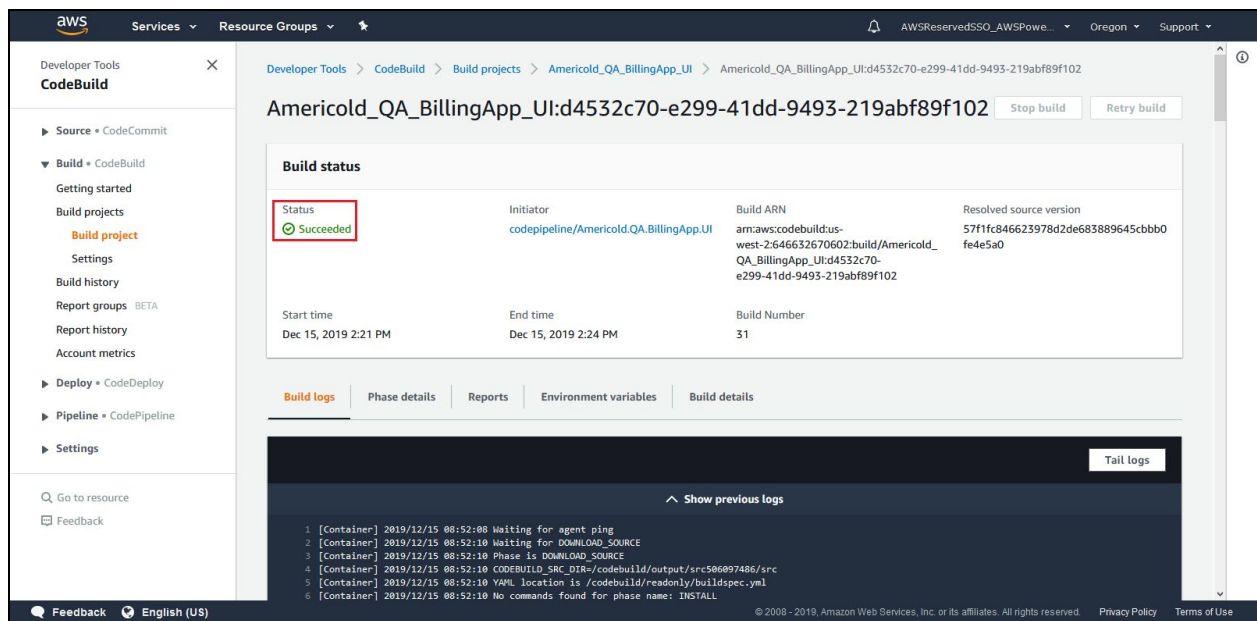
Review the pipeline configuration and click on create pipelines.

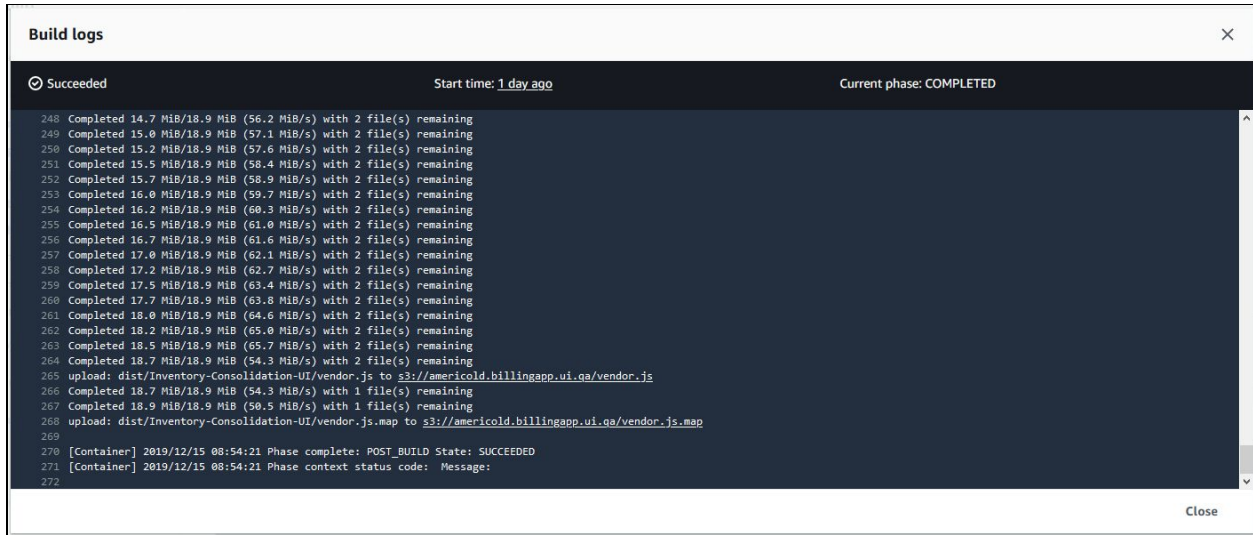


We can see the process is automated for CI/CD. We can check the stage logs by clicking on the details. The pipeline will trigger automatically for the first time to release changes and later it will be triggered when the new changes are committed to the repository.

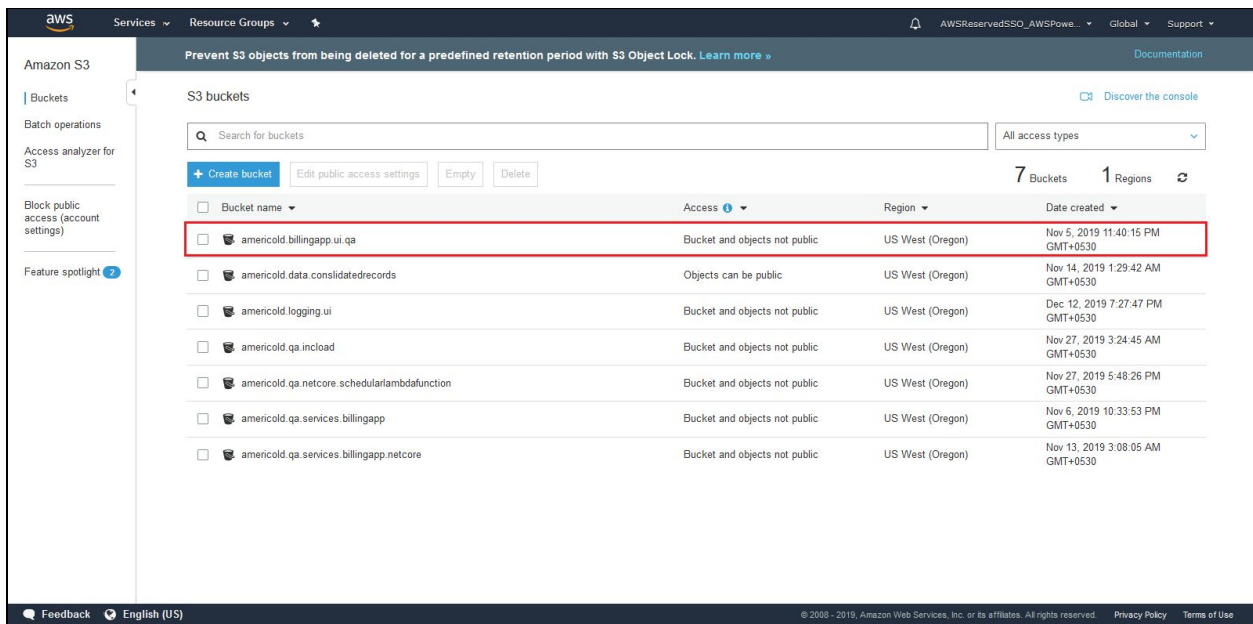


We can check the build logs here.





After completing the CI/CD we can see the updated data in the S3 bucket.



AWS

Services

Resource Groups

AWSReservedSSO_AWSPowe...

Global

Support

Amazon S3

americold.billingapp.ui.qa

americold.billingapp.ui.qa

Overview

Properties

Permissions

Management

Access points

Q

Type a prefix and press Enter to search. Press ESC to clear.

Upload

Create folder

Download

Actions

US West (Oregon)

Viewing 1 to 14

<input type="checkbox"/> Name	Last modified	Size	Storage class
<input type="checkbox"/> assets	--	--	--
<input type="checkbox"/> index.html	Dec 18, 2019 5:06:48 AM GMT+0530	1.2 KB	Standard
<input type="checkbox"/> main.js	Dec 18, 2019 5:06:48 AM GMT+0530	1.0 MB	Standard
<input type="checkbox"/> main.js.map	Dec 18, 2019 5:06:48 AM GMT+0530	259.5 KB	Standard
<input type="checkbox"/> polyfills-es5.js	Dec 18, 2019 5:06:48 AM GMT+0530	434.2 KB	Standard
<input type="checkbox"/> polyfills-es5.js.map	Dec 18, 2019 5:06:48 AM GMT+0530	275.3 KB	Standard
<input type="checkbox"/> polyfills.js	Dec 18, 2019 5:06:48 AM GMT+0530	424.3 KB	Standard
<input type="checkbox"/> polyfills.js.map	Dec 18, 2019 5:06:48 AM GMT+0530	368.0 KB	Standard
<input type="checkbox"/> runtime.js	Dec 18, 2019 5:06:48 AM GMT+0530	6.1 KB	Standard
<input type="checkbox"/> runtime.js.map	Dec 18, 2019 5:06:48 AM GMT+0530	6.1 KB	Standard

Feedback

English (US)

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