How to unit test and deploy AWS Glue jobs using AWS CodePipeline

by Praveen Kumar Jeyarajan and Vaidyanathan Ganesa Sankaran | on 09 MAY 2022 | in Advanced (300), AWS CloudFormation, AWS CodeBuild, AWS CodePipeline, AWS Glue, Learning Levels, Technical How-To | Permalink | Share

This post is intended to assist users in understanding and replicating a method to unit test Python-based ETL Glue Jobs, using the PyTest Framework in AWS CodePipeline. In the current practice, several options exist for unit testing Python scripts for Glue jobs in a local environment. Although a local development environment may be set up to build and unit test Python-based Glue jobs, by following the documentation, replicating the same procedure in a DevOps pipeline is difficult and time consuming.

Unit test scripts are one of the initial quality gates used by developers to provide a high-quality build. One must reuse these scripts during regression testing to make sure that all of the existing functionality is intact, and that new releases don't disrupt key application functionality. The majority of the regression test suites are expected to be integrated with the DevOps Pipeline for its execution. Unit testing an application code is a fundamental task that evaluates whether each (unit) code written by a programmer functions as expected. Unit testing of code provides a mechanism to determine that software quality hasn't been compromised. One of the difficulties in building Python-based Glue ETL tasks is their ability for unit testing to be incorporated within DevOps Pipeline, especially when there are modernization of mainframe ETL process to modern tech stacks in AWS

AWS Glue is a serverless data integration service that makes it easy to discover, prepare, and combine data for analytics, machine learning (ML), and application development. AWS Glue provides all of the capabilities needed for data integration. This means that you can start analyzing your data and putting it to use in minutes rather than months. AWS Glue provides both visual and code-based interfaces to make data integration easier.

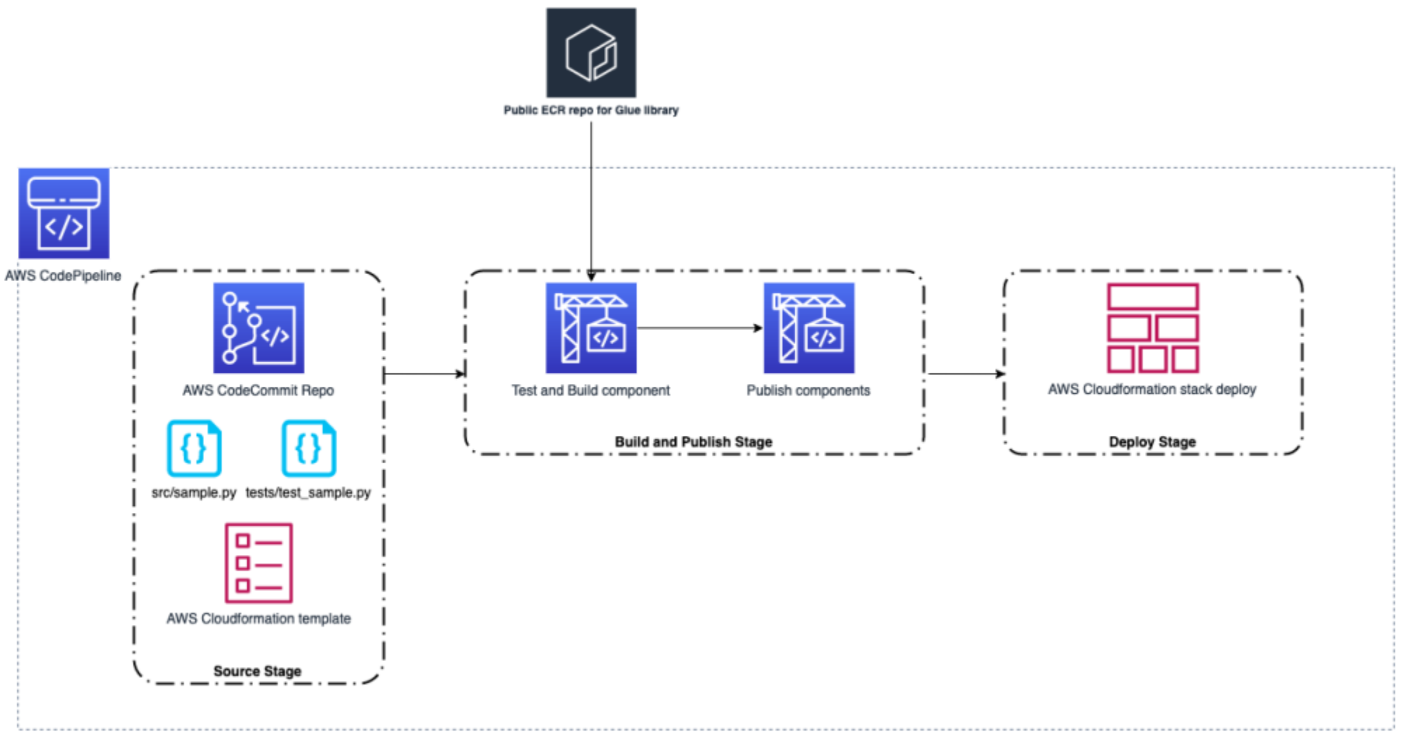
Prerequisites

GitHub Repository

Amazon ECR Image URI for Glue Library

Solution overview

A typical enterprise-scale DevOps pipeline is illustrated in the following diagram. This solution describes how to incorporate the unit testing of Python-based AWS Glue ETL processes into the AWS DevOps Pipeline.

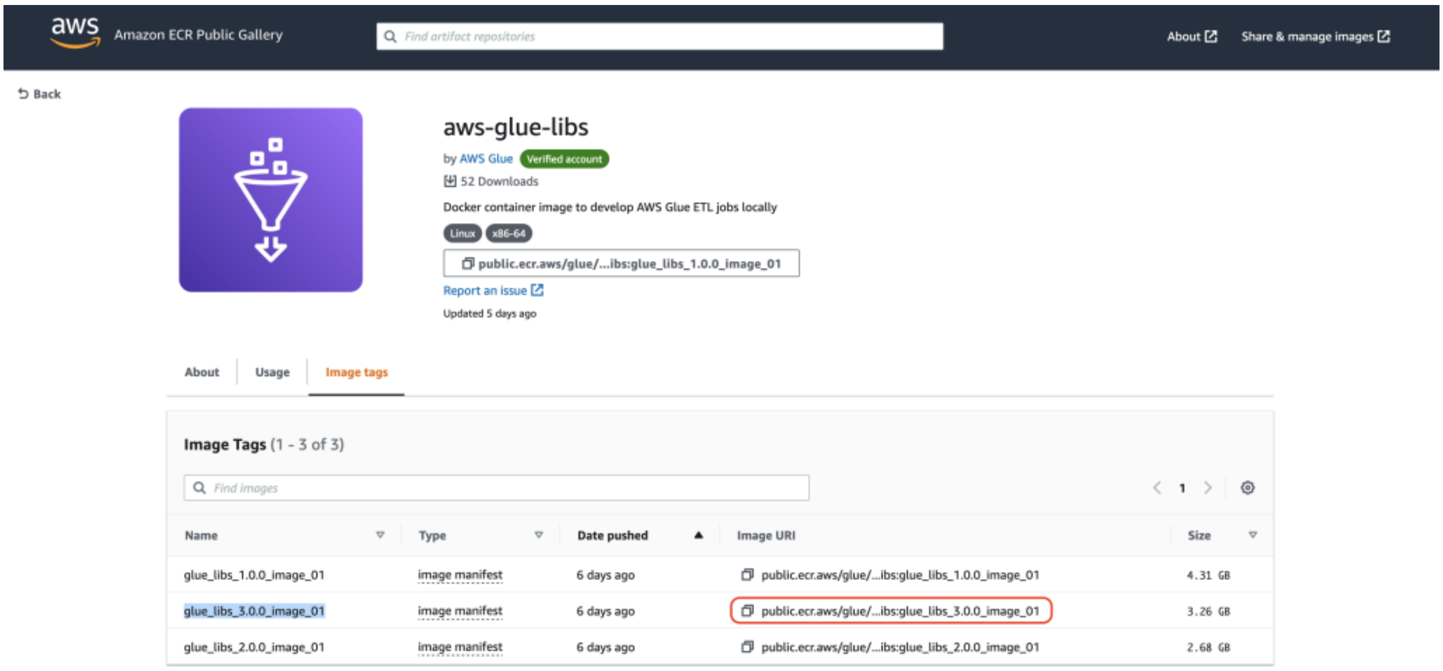


The GitHub repository aws-glue-jobs-unit-testing has a sample Python-based Glue job in the `src` folder. Its associated unit test cases built using the Pytest Framework are accessible in the `tests` folder. An AWS CloudFormation template written in YAML is included in the `deploy` folder. As a runtime environment, AWS CodeBuild utilizes custom container images. This feature is used to build a project utilizing Glue libraries from Public ECR repository, that can run the code package to demonstrate unit testing integration.

Solution walkthrough

Time to read 7 min
Time to complete 15-20 min
Learning level 300
Services used
AWS CodePipeline, AWS CodeCommit, AWS CodeBuild, Amazon Elastic Container Registry (Amazon ECR) Public Repositories, AWS CloudFormation

The container image at the Public ECR repository for AWS Glue libraries includes all of the binaries required to run PySpark-based AWS Glue ETL tasks locally, as well as unit test them. The public container repository has three image tags, one for each AWS Glue version supported by AWS Glue. To demonstrate the solution, we use the image tag `glue_libs_3.0.0_image_01` in this post. To utilize this container image as a runtime image in CodeBuild, copy the Image URI corresponding to the image tag that you intend to use, as shown in the following image.



aws-glue-libs

by AWS Glue Verified account

52 Downloads

Docker container image to develop AWS Glue ETL jobs locally

Linux x86-64

public.ecr.aws/glue/...ibs:glue_libs_1.0.0_image_01

Report an issue

Updated 5 days ago

Image Tags (1 - 3 of 3)

Name	Type	Date pushed	Image URI	Size
glue_libs_1.0.0_image_01	image manifest	6 days ago	public.ecr.aws/glue/...ibs:glue_libs_1.0.0_image_01	4.31 GB
glue_libs_3.0.0_image_01	image manifest	6 days ago	public.ecr.aws/glue/...ibs:glue_libs_3.0.0_image_01	3.26 GB
glue_libs_2.0.0_image_01	image manifest	6 days ago	public.ecr.aws/glue/...ibs:glue_libs_2.0.0_image_01	2.68 GB

The aws-glue-jobs-unit-testing GitHub repository contains a CloudFormation template, `pipeline.yml`, which deploys a CodePipeline with CodeBuild projects to create, test, and publish the AWS Glue job. As illustrated in the following, use the copied image URL from Amazon ECR public to create and test a CodeBuild project.

YAML

```yaml
  TestBuild:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      BadgeEnabled: false
      Environment:
        ComputeType: BUILD_GENERAL1_LARGE
        Image: "public.ecr.aws/glue/aws-glue-libs:glue_libs_3.0.0_image_01"
        ImagePullCredentialsType: CODEBUILD
        PrivilegedMode: false
        Type: LINUX_CONTAINER
      Name: !Sub "${RepositoryName}-${BranchName}-build"
      ServiceRole: !GetAtt CodeBuildRole.Arn
```

The pipeline performs the following operations:

1. It uses the CodeCommit repository as the source and transfers the most recent code from the main branch to the CodeBuild project for further processing.
2. The following stage is build and test, in which the most recent code from the previous phase is unit tested and the test report is published to CodeBuild report groups.
3. If all of the test results are good, then the next CodeBuild project is launched to publish the code to an Amazon Simple Storage Service (Amazon S3) bucket.
4. Following the successful completion of the publish phase, the final step is to deploy the AWS Glue task using the CloudFormation template in the `deploy` folder.

Deploying the solution

Set up

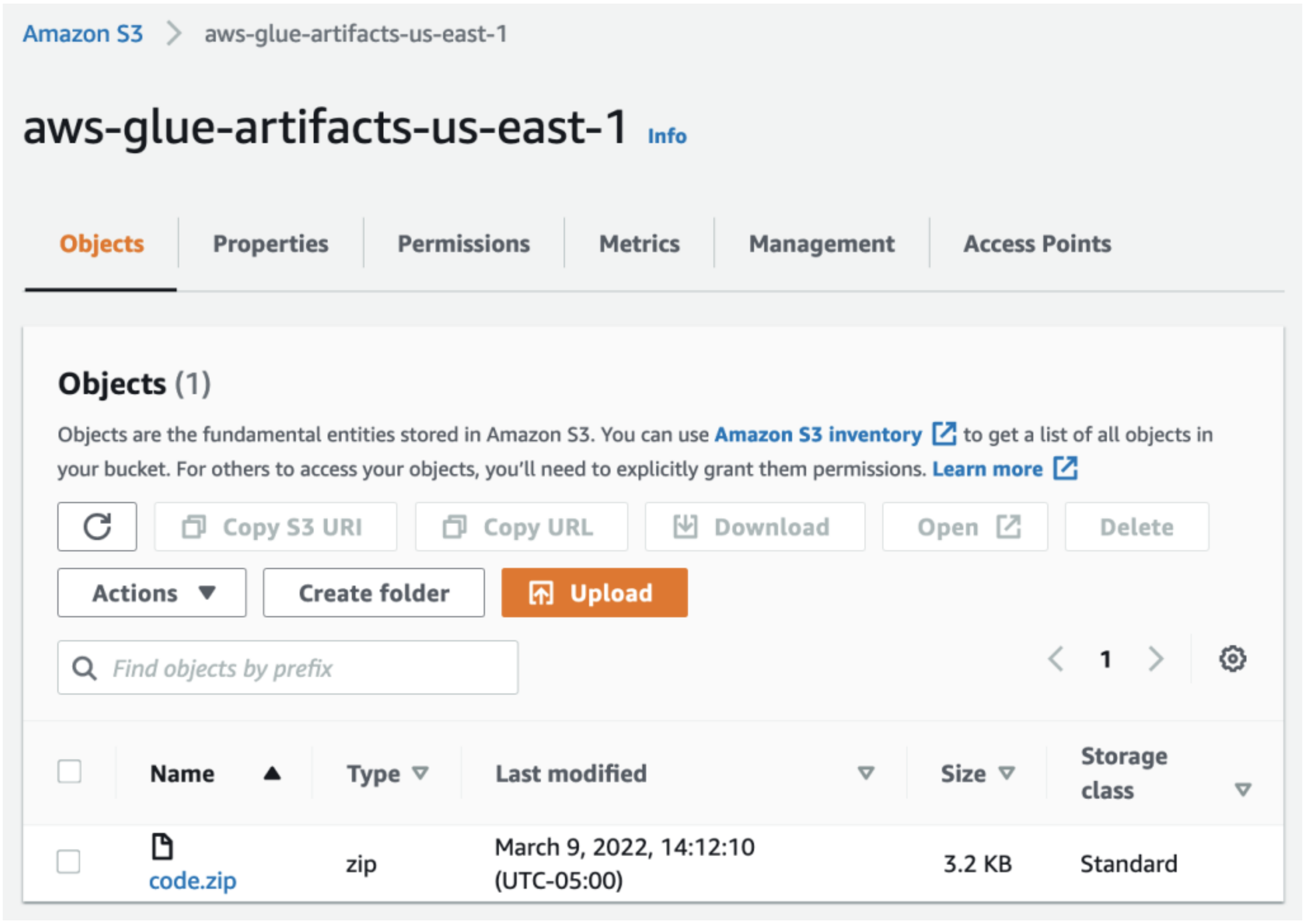
Now we'll deploy the solution using a CloudFormation template.

- Using the GitHub Web, download the code.zip file from the aws-glue-jobs-unit-testing repository. This zip file contains the GitHub repository's src, tests, and deploy folders. You may also create the zip file yourself using command-line tools, such as git and zip. To create the zip file on Linux or Mac, open the terminal and enter the following commands.

Bash
```bash
git clone https://github.com/aws-samples/aws-glue-jobs-unit-testing.git
cd aws-glue-jobs-unit-testing
git checkout master
zip -r code.zip src/ tests/ deploy/
```

- Sign in to the AWS Management Console and choose the AWS Region of your choice.
- Create an Amazon S3 bucket. For more information, see How Do I Create an S3 Bucket? in the AWS documentation.
- Upload the downloaded zip package, code.zip, to the Amazon S3 bucket that you created.

In this example, I created an Amazon S3 bucket named aws-glue-artifacts-us-east-1 in the N. Virginia (us-east-1) Region, and used the console to upload the zip package from the GitHub repository to the Amazon S3 bucket.



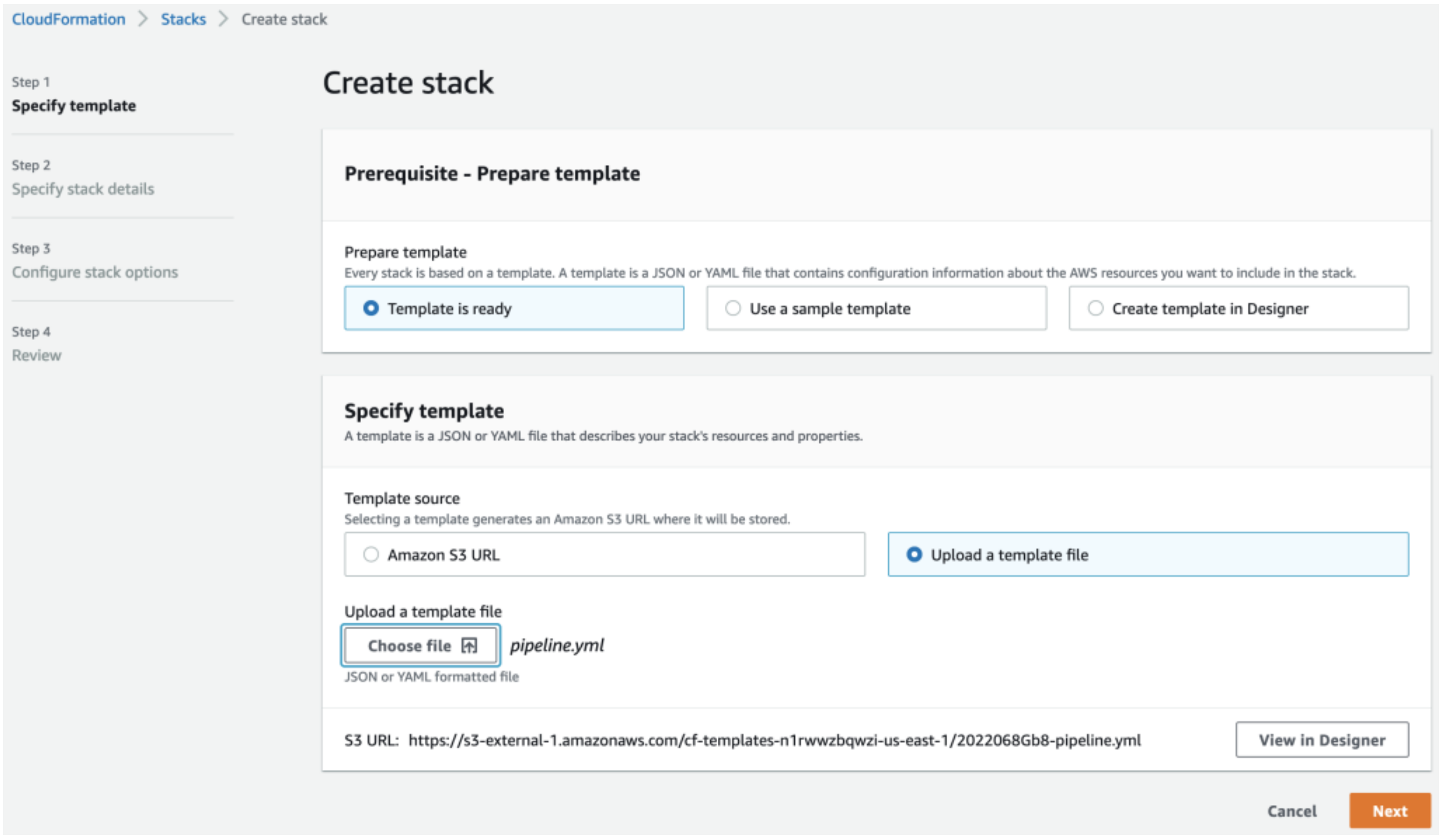
Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
code.zip	zip	March 9, 2022, 14:12:10 (UTC-05:00)	3.2 KB	Standard

Creating the stack

1. In the CloudFormation console, choose **Create stack**.
2. On the **Specify template** page, choose **Upload a template file**, and then choose the **pipeline.yml** template, downloaded from the GitHub repository


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.

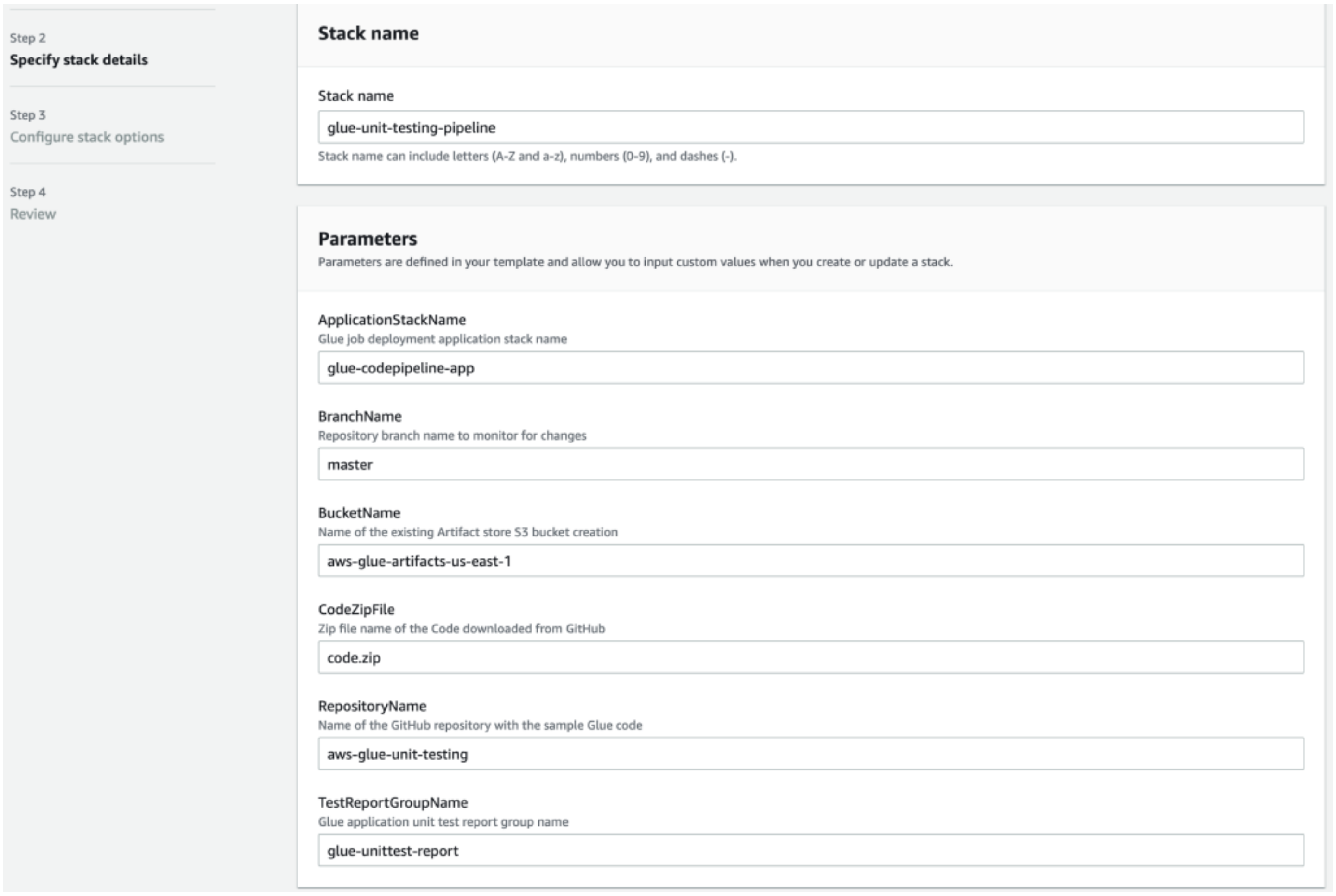
Amazon S3 URL [✓ Upload a template file]

Upload a template file
Choose file *pipeline.yml*
JSON or YAML formatted file

S3 URL: https://s3-external-1.amazonaws.com/cf-templates-n1rwwzbqwzi-us-east-1/2022068Gb8-pipeline.yml

3. Specify the following parameters:.

- **Stack name:** glue-unit-testing-pipeline (Choose a stack name of your choice)
- **ApplicationStackName:** glue-codepipeline-app (This is the name of the CloudFormation stack that will be created by the pipeline)
- **BranchName:** master (This is the name of the branch to be created in the CodeCommit repository to check-in the code from the Amazon S3 bucket zip file)
- **BucketName:** aws-glue-artifacts-us-east-1 (This is the name of the Amazon S3 bucket that contains the zip file. This bucket will also be used by the pipeline for storing code artifacts)
- **CodeZipFile:** lambda.zip (This is the key name of the sample code Amazon S3 object. The object should be a zip file)
- **RepositoryName:** aws-glue-unit-testing (This is the name of the CodeCommit repository that will be created by the stack)
- **TestReportGroupName:** glue-unittest-report (This is the name of the CodeBuild test report group that will be created to store the unit test reports)


Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Stack name

Stack name
glue-unit-testing-pipeline
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.

ApplicationStackName
Glue job deployment application stack name
glue-codepipeline-app

BranchName
Repository branch name to monitor for changes
master

BucketName
Name of the existing Artifact store S3 bucket creation
aws-glue-artifacts-us-east-1

CodeZipFile
Zip file name of the Code downloaded from GitHub
code.zip

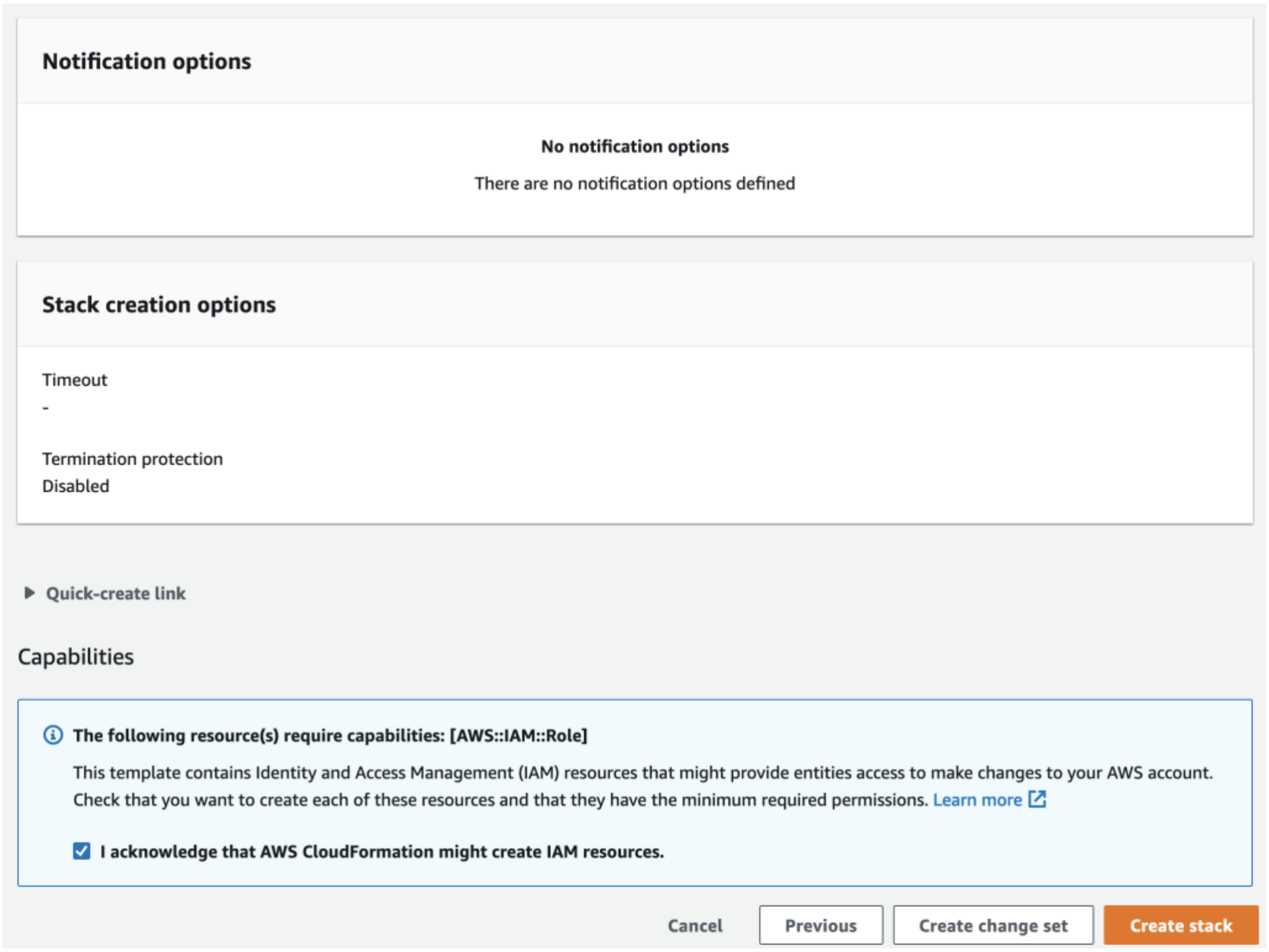
RepositoryName
Name of the GitHub repository with the sample Glue code
aws-glue-unit-testing

TestReportGroupName
Glue application unit test report group name
glue-unittest-report

4. Choose **Next**, and again **Next**.

5. On the **Review** page, under **Capabilities**, choose the following options:

- I acknowledge that CloudFormation might create IAM resources with custom names.


Notification options

No notification options

There are no notification options defined

Stack creation options

Timeout

-

Termination protection

Disabled

Quick-create link

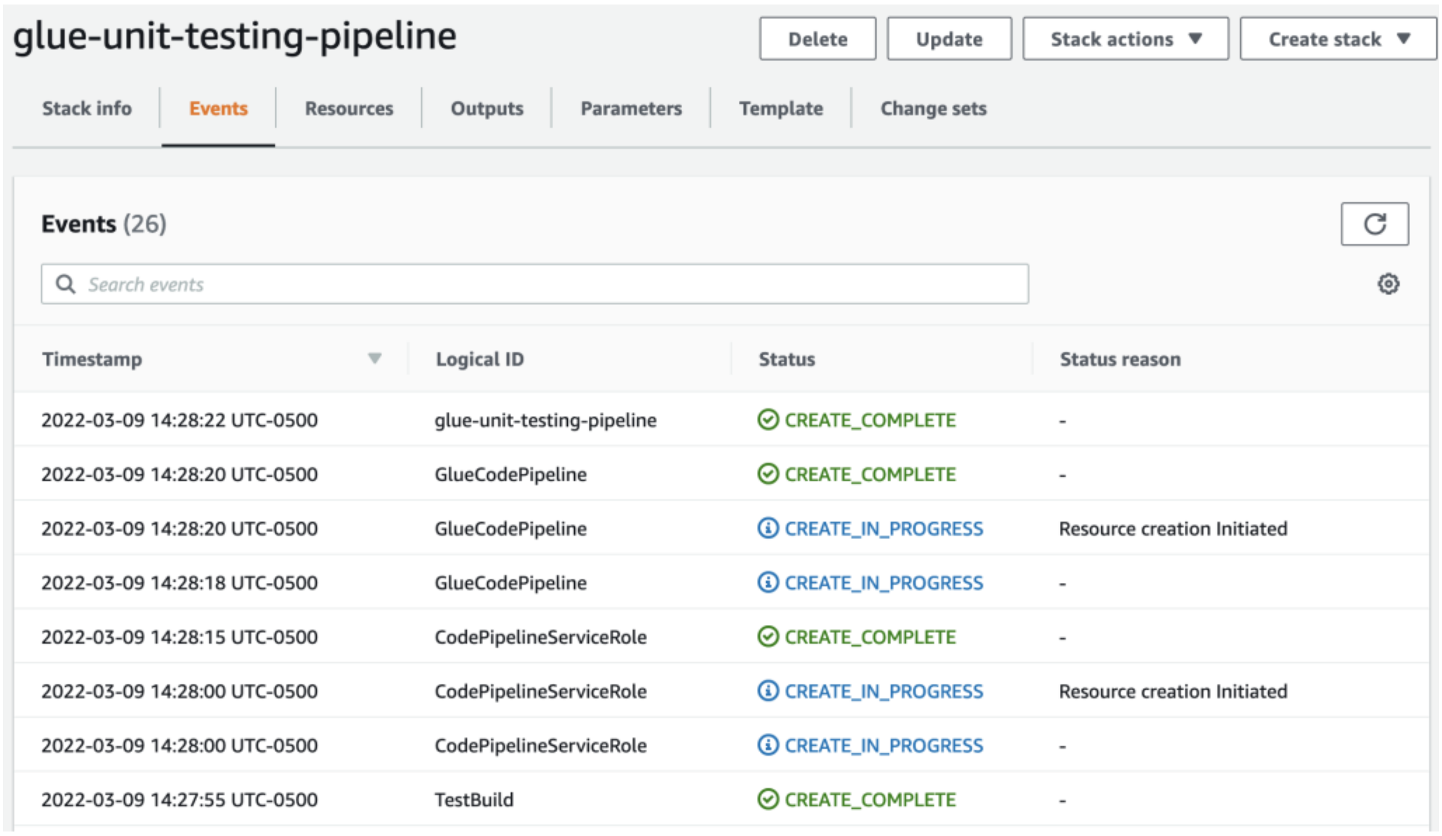
Capabilities

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.

6. Choose **Create stack** to begin the stack creation process. Once the stack creation is complete, the resources that were created are displayed on the **Resources** tab. The stack creation takes approximately 5-7 minutes.

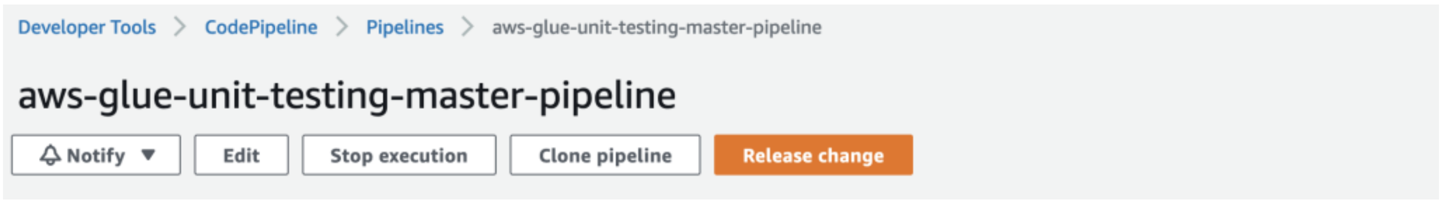


Timestamp	Logical ID	Status	Status reason
2022-03-09 14:28:22 UTC-0500	glue-unit-testing-pipeline	CREATE_COMPLETE	-
2022-03-09 14:28:20 UTC-0500	GlueCodePipeline	CREATE_COMPLETE	-
2022-03-09 14:28:20 UTC-0500	GlueCodePipeline	CREATE_IN_PROGRESS	Resource creation Initiated
2022-03-09 14:28:18 UTC-0500	GlueCodePipeline	CREATE_IN_PROGRESS	-
2022-03-09 14:28:15 UTC-0500	CodePipelineServiceRole	CREATE_COMPLETE	-
2022-03-09 14:28:00 UTC-0500	CodePipelineServiceRole	CREATE_IN_PROGRESS	Resource creation Initiated
2022-03-09 14:28:00 UTC-0500	CodePipelineServiceRole	CREATE_IN_PROGRESS	-
2022-03-09 14:27:55 UTC-0500	TestBuild	CREATE_COMPLETE	-

The stack automatically creates a CodeCommit repository with the initial code checked-in from the zip file uploaded to the Amazon S3 bucket. Furthermore, it creates a CodePipeline view using the CodeCommit repository as the source. In the above example, the CodeCommit repository is aws-glue-unit-test, and the pipeline is aws-glue-unit-test-pipeline.

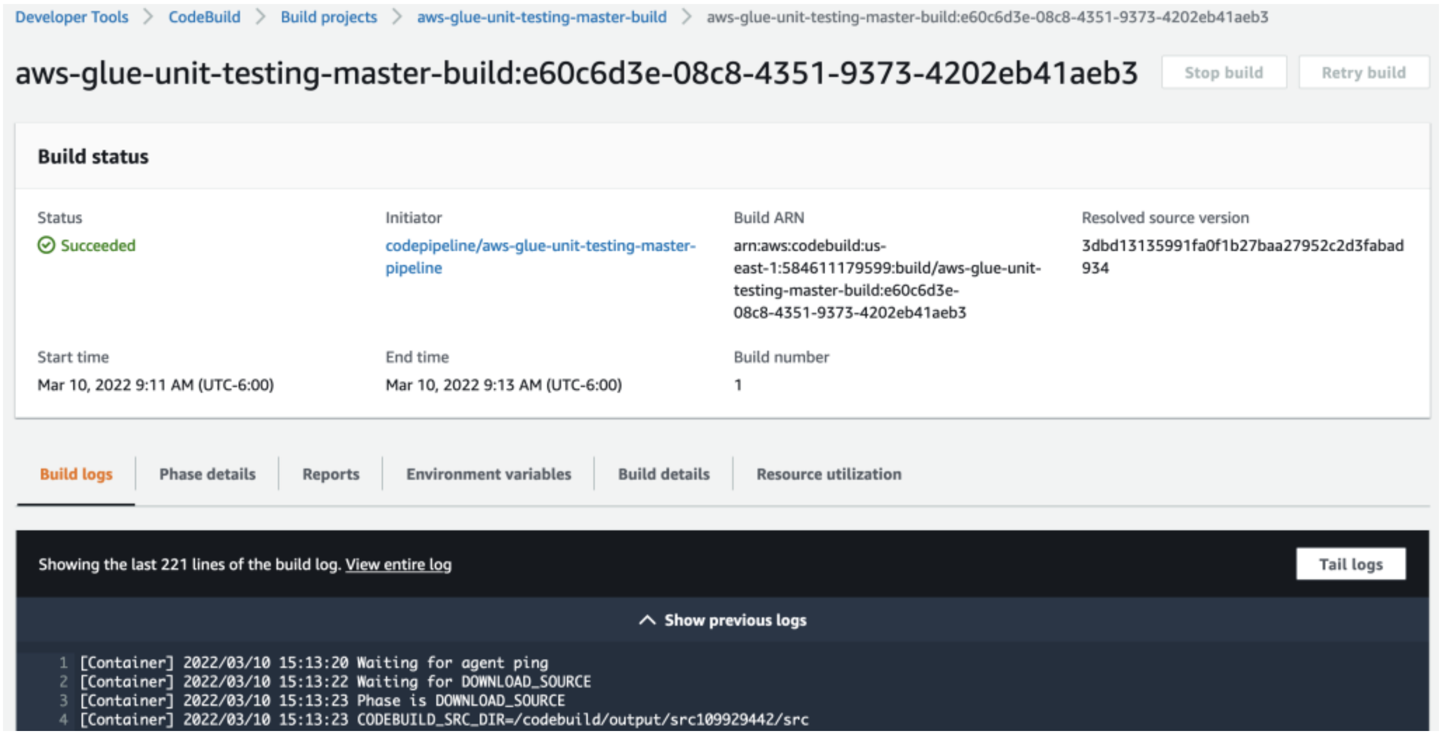
Testing the solution

To test the deployed pipeline, open the CodePipeline console and select the pipeline created by the CloudFormation stack. Select the **Release Change** button on the pipeline page.


Developer Tools > CodePipeline > Pipelines > aws-glue-unit-testing-master-pipeline
aws-glue-unit-testing-master-pipeline
Notify
Edit
Stop execution
Clone pipeline
Release change

The pipeline begins its execution with the most recent code in the CodeCommit repository.

When the **Test_and_Build** phase is finished, select the **Details** link to examine the execution logs.


Developer Tools > CodeBuild > Build projects > aws-glue-unit-testing-master-build > aws-glue-unit-testing-master-build:e60c6d3e-08c8-4351-9373-4202eb41aeb3
aws-glue-unit-testing-master-build:e60c6d3e-08c8-4351-9373-4202eb41aeb3
Stop build
Retry build
Build logs
Phase details
Reports
Environment variables
Build details
Resource utilization
Tail logs
Show previous logs

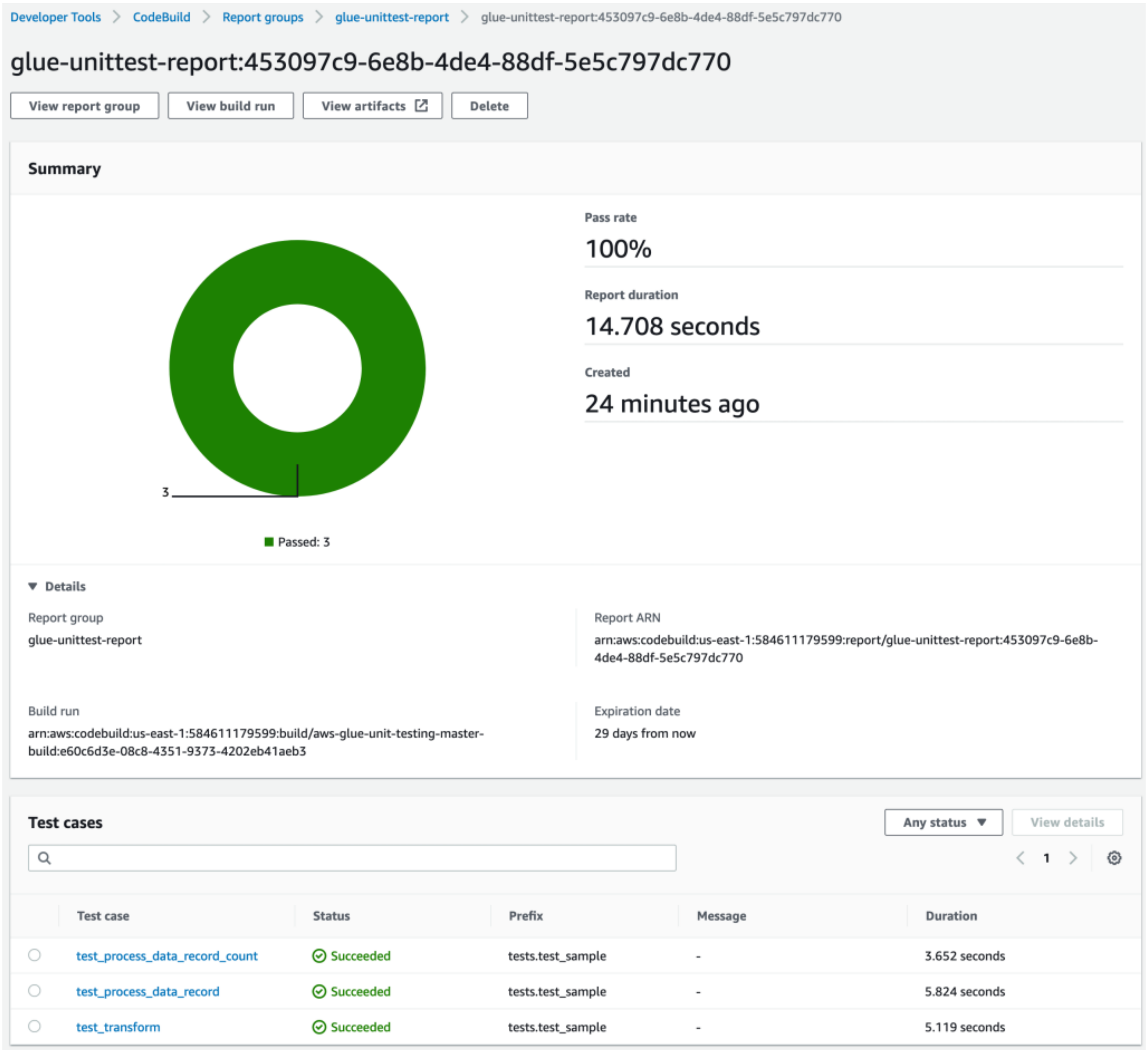
Build status

Status	Initiator	Build ARN	Resolved source version
Succeeded	codepipeline/aws-glue-unit-testing-master-pipeline	arn:aws:codebuild:us-east-1:584611179599:build/aws-glue-unit-testing-master-build:e60c6d3e-08c8-4351-9373-4202eb41aeb3	3dbd13135991fa0f1b27baa27952c2d3fabad934
Start time	End time	Build number	
Mar 10, 2022 9:11 AM (UTC-6:00)	Mar 10, 2022 9:13 AM (UTC-6:00)	1	

Showing the last 221 lines of the build log. View entire log

```
[Container] 2022/03/10 15:13:20 Waiting for agent ping
[Container] 2022/03/10 15:13:22 Waiting for DOWNLOAD_SOURCE
[Container] 2022/03/10 15:13:23 Phase is DOWNLOAD_SOURCE
[Container] 2022/03/10 15:13:23 CODEBUILD_SRC_DIR=/codebuild/output/src109929442/src
```

Select the **Reports** tab, and choose the test report from **Report history** to view the unit execution results.


Developer Tools > CodeBuild > Report groups > glue-unittest-report > glue-unittest-report:453097c9-6e8b-4de4-88df-5e5c797dc770

glue-unittest-report:453097c9-6e8b-4de4-88df-5e5c797dc770

Summary

Pass rate
100%

Report duration
14.708 seconds

Created
24 minutes ago

▼ Details

Report group
glue-unittest-report

Report ARN
arn:aws:codebuild:us-east-1:584611179599:report/glue-unittest-report:453097c9-6e8b-4de4-88df-5e5c797dc770

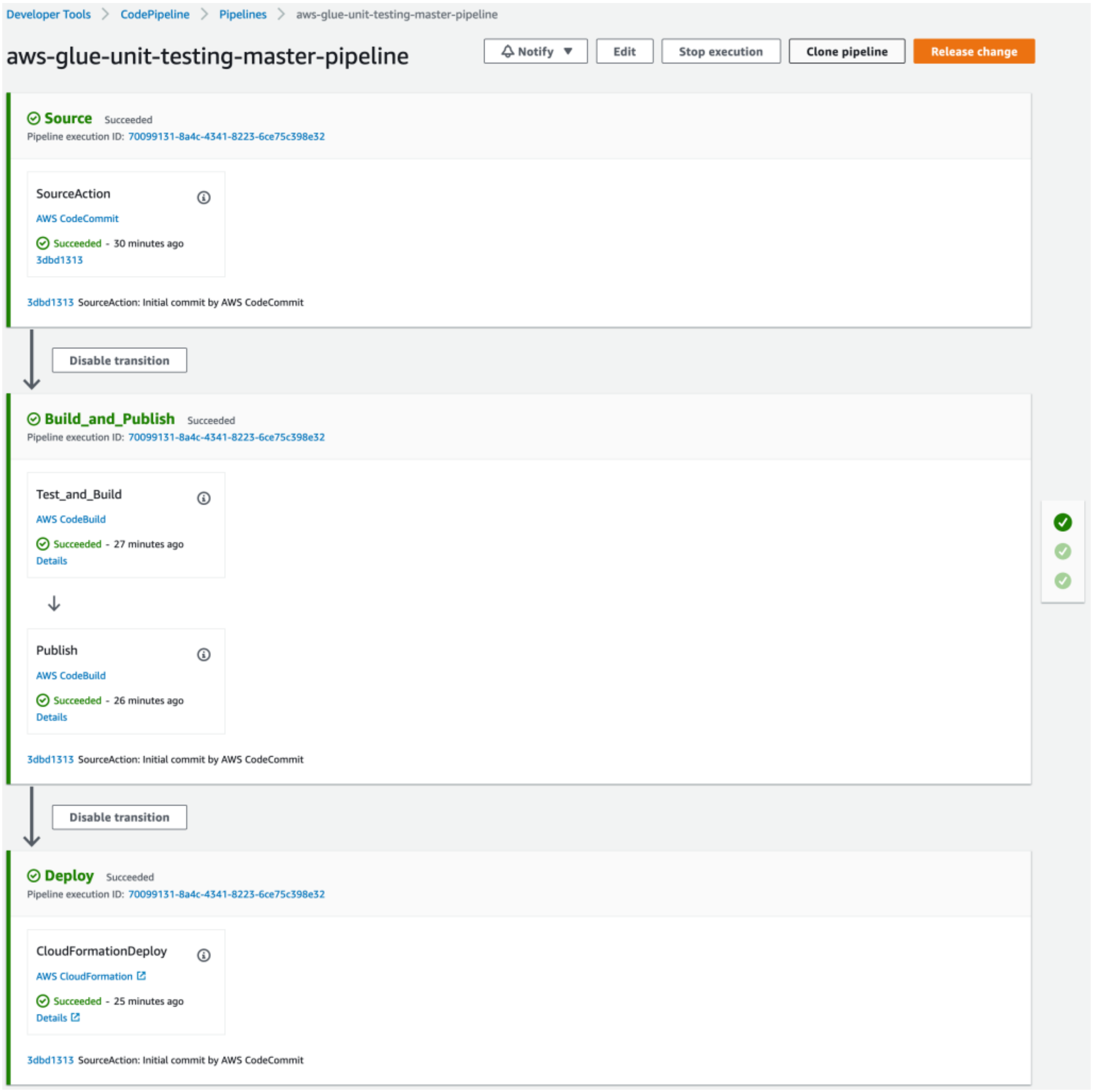
Build run
arn:aws:codebuild:us-east-1:584611179599:build/aws-glue-unit-testing-master-build:e60c6d3e-08c8-4351-9373-4202eb41aeb3

Expiration date
29 days from now

Test cases

Test case	Status	Prefix	Message	Duration
test_process_data_record_count	✓ Succeeded	tests.test_sample	-	3.652 seconds
test_process_data_record	✓ Succeeded	tests.test_sample	-	5.824 seconds
test_transform	✓ Succeeded	tests.test_sample	-	5.119 seconds

Finally, after the deployment stage is complete, you can see, run, and monitor the deployed AWS Glue job on the AWS Glue console page. For more information, refer to the Running and monitoring AWS Glue documentation


Developer Tools > CodePipeline > Pipelines > aws-glue-unit-testing-master-pipeline

aws-glue-unit-testing-master-pipeline

Source Succeeded
Pipeline execution ID: 70099131-8a4c-4341-8223-6ce75c398e32

SourceAction
AWS CodeCommit
Succeeded - 30 minutes ago
3dbd1313

3dbd1313 SourceAction: Initial commit by AWS CodeCommit

Build_and_Publish Succeeded
Pipeline execution ID: 70099131-8a4c-4341-8223-6ce75c398e32

Test_and_Build
AWS CodeBuild
Succeeded - 27 minutes ago
Details

Publish
AWS CodeBuild
Succeeded - 26 minutes ago
Details

3dbd1313 SourceAction: Initial commit by AWS CodeCommit

Deploy Succeeded
Pipeline execution ID: 70099131-8a4c-4341-8223-6ce75c398e32

CloudFormationDeploy
AWS CloudFormation
Succeeded - 25 minutes ago
Details

3dbd1313 SourceAction: Initial commit by AWS CodeCommit

Cleanup

To avoid additional infrastructure costs, make sure that you delete the stack after experimenting with the examples provided in the post. On the CloudFormation console, select the stack that you created, and then choose **Delete**. This will delete all of the resources that it created, including CodeCommit repositories, IAM roles/policies, and CodeBuild projects.

Summary

In this post, we demonstrated how to unit test and deploy Python-based AWS Glue jobs in a pipeline with unit tests written with the PyTest framework. The approach is not limited to CodePipeline, and it can be used to build up a local development environment, as demonstrated in the Big Data blog. The aws-glue-jobs-unit-testing GitHub repository contains the example's CloudFormation template, as well as sample AWS Glue Python code and Pytest code used in this post. If you have any questions or comments regarding this example, please open an issue or submit a pull request.

Authors: