AWS Solutions Constructs: AWS Solutions
Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

Overview ................................................................................................................................. 1
  What is AWS Solutions Constructs? .................................................................................... 1
  Why use AWS Solutions Constructs? ............................................................................. 1
Getting Started ....................................................................................................................... 2
  Prerequisites ...................................................................................................................... 2
  Installing the AWS CDK .................................................................................................... 2
Working with AWS Solutions Constructs .................................................................................. 2
  Walkthrough - Part 1 ......................................................................................................... 3
  Hello Constructs ............................................................................................................... 3
    Creating the App Directory and Initializing the AWS CDK ............................................. 4
    Update project base dependencies .................................................................................. 4
    Lambda handler code ...................................................................................................... 6
  Install the AWS CDK and AWS Solutions Constructs dependencies .................................. 7
  Add an Amazon API Gateway/AWS Lambda pattern to your stack .................................... 8
  cdk deploy ......................................................................................................................... 12
  Stack outputs .................................................................................................................... 12
  Testing your app ................................................................................................................. 12
Walkthrough - Part 2 ............................................................................................................. 13
  Hit Counter Lambda code ............................................................................................... 13
  Install the new dependencies ............................................................................................ 15
  Define the resources ......................................................................................................... 15
  Review the changes .......................................................................................................... 24
  cdk deploy ........................................................................................................................ 25
  Stack outputs .................................................................................................................... 25
  Testing your app ................................................................................................................. 25
Sample Use Cases .................................................................................................................. 26
  AWS Static S3 Website ..................................................................................................... 27
  AWS Simple Serverless Image Handler ............................................................................ 27
  AWS Serverless Web App .................................................................................................. 27
  AWS Restaurant Management Demo ................................................................................. 27
API Reference ........................................................................................................................ 28
  Modules .............................................................................................................................. 28
  Module Contents ............................................................................................................... 28
    aws-alb-lambda ................................................................................................................. 28
      Initializer ....................................................................................................................... 29
      Pattern Construct Props .............................................................................................. 29
      Pattern Properties ......................................................................................................... 31
      Default settings ............................................................................................................ 32
      Architecture ................................................................................................................ 33
      GitHub .......................................................................................................................... 33
    aws-apigateway-dynamodb ............................................................................................... 33
      Overview ....................................................................................................................... 34
      Initializer ....................................................................................................................... 34
      Pattern Construct Props .............................................................................................. 34
      Pattern Properties ......................................................................................................... 36
      Default settings ............................................................................................................ 36
      Architecture ................................................................................................................ 37
      GitHub .......................................................................................................................... 37
    aws-apigateway-iot .......................................................................................................... 37
      Overview ....................................................................................................................... 38
      Initializer ....................................................................................................................... 38
      Pattern Construct Props .............................................................................................. 39
      Pattern Properties ......................................................................................................... 39
      Default settings ............................................................................................................ 40
AWS Solutions Constructs AWS Solutions

Architecture ................................................................. 41
Examples ........................................................................ 41
GitHub ........................................................................... 43
aws-apigateway-kinesisstreams .................................................. 43
Overview ........................................................................ 43
Initializer ................................................................. 44
Pattern Construct Props .................................................... 44
Pattern Properties .......................................................... 45
Sample API Usage .......................................................... 45
Default settings ............................................................. 46
Architecture ................................................................. 47
GitHub ........................................................................... 47
aws-apigateway-lambda .......................................................... 48
Overview ........................................................................ 48
Initializer ................................................................. 48
Pattern Construct Props .................................................... 49
Pattern Properties .......................................................... 49
Default settings ............................................................. 49
Architecture ................................................................. 50
GitHub ........................................................................... 50
aws-apigateway-sagemakerendpoint ............................................. 51
Overview ........................................................................ 51
Initializer ................................................................. 52
Pattern Construct Props .................................................... 52
Pattern Properties .......................................................... 53
Sample API Usage .......................................................... 45
Default settings ............................................................. 53
Architecture ................................................................. 54
GitHub ........................................................................... 50
aws-apigateway-sqs ............................................................. 55
Overview ........................................................................ 55
Initializer ................................................................. 55
Pattern Construct Props .................................................... 56
Pattern Properties .......................................................... 57
Sample API Usage .......................................................... 45
Default settings ............................................................. 57
Architecture ................................................................. 58
GitHub ........................................................................... 58
aws-cloudfront-apigateway ....................................................... 59
Overview ........................................................................ 59
Initializer ................................................................. 60
Pattern Construct Props .................................................... 60
Pattern Properties .......................................................... 60
Default settings ............................................................. 61
Architecture ................................................................. 62
GitHub ........................................................................... 62
aws-cloudfront-apigateway-lambda ............................................... 62
Overview ........................................................................ 63
Initializer ................................................................. 63
Pattern Construct Props .................................................... 63
Pattern Properties .......................................................... 64
Default settings ............................................................. 65
Architecture ................................................................. 66
GitHub ........................................................................... 66
aws-cloudfront-mediastore ......................................................... 66
Overview ........................................................................ 67
Initializer ................................................................. 67
<table>
<thead>
<tr>
<th>Construct Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws-cognito-apigateway-lambda</td>
<td>73</td>
</tr>
<tr>
<td>aws-cloudfront-s3</td>
<td>69</td>
</tr>
<tr>
<td>aws-dynamodb-stream-lambda</td>
<td>77</td>
</tr>
<tr>
<td>aws-dynamodbstreams-lambda-elasticsearch-kibana</td>
<td>85</td>
</tr>
<tr>
<td>aws-dynamodbstreams-lambda</td>
<td>81</td>
</tr>
<tr>
<td>aws-dynamodb-streams-lambda-elasticsearch-kibana</td>
<td>89</td>
</tr>
</tbody>
</table>

Pattern Properties .......................................................... 67
Pattern Construct Props ......................................................... 70
Default settings .................................................................. 68
Architecture ......................................................................... 69
GitHub ................................................................................. 69
aws-cloudfront-s3 ............................................................... 69
Overview .............................................................................. 70
Initializer ............................................................................ 70
Pattern Construct Props ......................................................... 70
Pattern Properties .................................................................. 71
Default settings .................................................................. 72
Architecture ......................................................................... 73
GitHub ................................................................................. 73
aws-cognito-apigateway-lambda ............................................ 73
Overview .............................................................................. 59
Initializer ............................................................................ 75
Pattern Construct Props ......................................................... 75
Pattern Properties .................................................................. 75
Default settings .................................................................. 76
Architecture ......................................................................... 77
GitHub ................................................................................. 77
aws-dynamodb-stream-lambda .................................................... 77
Overview .............................................................................. 78
Initializer ............................................................................ 78
Pattern Construct Props ......................................................... 79
Pattern Properties .................................................................. 79
Lambda function .................................................................... 80
Default settings .................................................................. 80
Architecture ......................................................................... 81
GitHub ................................................................................. 81
aws-dynamodbstreams-lambda .................................................... 81
Overview .............................................................................. 82
Initializer ............................................................................ 82
Pattern Construct Props ......................................................... 82
Pattern Properties .................................................................. 83
Lambda function .................................................................... 83
Default settings .................................................................. 83
Architecture ......................................................................... 84
GitHub ................................................................................. 85
aws-dynamodb-stream-lambda-elasticsearch-kibana .................. 85
Overview .............................................................................. 85
Initializer ............................................................................ 86
Pattern Construct Props ......................................................... 86
Pattern Properties .................................................................. 87
Lambda function .................................................................... 88
Default settings .................................................................. 88
Architecture ......................................................................... 89
GitHub ................................................................................. 89
aws-dynamodbstreams-lambda-elasticsearch-kibana .................. 89
Overview .............................................................................. 90
Initializer ............................................................................ 90
Pattern Construct Props ......................................................... 91
Pattern Properties .................................................................. 91
Lambda function .................................................................... 92
Default settings .................................................................. 92
Architecture ......................................................................... 93
GitHub ................................................................................. 94
<table>
<thead>
<tr>
<th>Construct</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws-lambda-dynamodb</td>
<td>Overview 177, Initializer 178, Pattern Construct Props 178, Pattern Properties 178, Default settings 180, Architecture 181, GitHub 181</td>
</tr>
<tr>
<td>aws-lambda-eventbridge</td>
<td>Overview 186, Initializer 186, Pattern Construct Props 187, Pattern Properties 188, Default settings 189, Architecture 189, GitHub 189</td>
</tr>
<tr>
<td>aws-lambda-s3</td>
<td>Overview 190, Initializer 191, Pattern Construct Props 191, Pattern Properties 193, Default settings 193, Architecture 194, GitHub 194</td>
</tr>
<tr>
<td>aws-lambda-sagemakerendpoint</td>
<td>Overview 195, Initializer 195, Pattern Construct Props 196, Pattern Properties 197, Default settings 198, Architecture 199, GitHub 199</td>
</tr>
<tr>
<td>aws-lambda-secretsmanager</td>
<td>Overview 200, Initializer 200, Pattern Construct Props 201, Pattern Properties 202, Default settings 202, Architecture 203, GitHub 203</td>
</tr>
<tr>
<td>aws-lambda-sns</td>
<td>Overview 204</td>
</tr>
</tbody>
</table>
AWS Solutions Constructs

Publication date: December 2021 (Document Revisions (p. 273))

What is AWS Solutions Constructs?

AWS Solutions Constructs (Constructs) is an open-source extension of the AWS Cloud Development Kit (CDK) that provides multi-service, well-architected patterns for quickly defining solutions in code to create predictable and repeatable infrastructure. The goal is to accelerate the experience for developers to build solutions of any size using pattern-based definitions for their architecture.

Use the AWS Solutions Constructs to define your solutions in a familiar programming language. The AWS Solutions Constructs supports TypeScript, JavaScript, Python, and Java at this time.

To browse the full catalog of AWS Solutions Constructs patterns, click here.

Why use AWS Solutions Constructs?

With the rate of innovation of cloud providers, knowing and understanding best practices and ensuring they are implemented correctly across your solution can be daunting. Constructs allows you to combine pre-built, well-architected patterns and use cases that perform common actions using cloud services in a scalable and secure manner. Because Constructs provides a library for modern programming languages, you can apply existing development skills and familiar tools to the task of building well-architected cloud infrastructure for your solutions.

Other advantages of AWS Solutions Constructs include:

• It is built upon the AWS Cloud Development Kit (CDK) open source software development framework.
• Use logic (if statements, for-loops, etc.) when defining your solution infrastructure.
• Use object-oriented techniques to create a model of your system.
• Define high level abstractions, share them, and publish them to your team, company, or community.
• Organize your solutions into logical modules.
• Share and reuse your solution as a library.
• Test your infrastructure code using industry-standard protocols.
• Use your existing code review workflow.

The aim of AWS Solutions Constructs is to reduce the complexity and glue logic required when integrating common well-architected patterns to achieve your solution goals on AWS.
Getting Started with AWS Solutions Constructs

This topic describes how to install and configure AWS Cloud Development Kit (CDK), AWS Solutions Constructs, and create your first AWS CDK app using AWS Solutions Constructs patterns.

Note
AWS Solutions Constructs is supported on AWS CDK versions ≥ 1.46.0.

Tip
Want to dig deeper? Try the CDK Workshop for a more in-depth tour of a real-world project.

Tip
For more information about getting started with the AWS Cloud Development Kit (CDK), refer to the AWS CDK Developer Guide.

Prerequisites

AWS Solutions Constructs is built upon the AWS CDK, so you need to install Node.js (>= 10.3.0), even those working in languages other than TypeScript or JavaScript. This is because the AWS CDK and AWS Solutions Constructs are developed in TypeScript and run on Node.js. The bindings for other supported languages use this backend and toolset.

You must provide your credentials and an AWS Region to use the AWS CDK CLI, as described in Specifying Your Credentials and Region.

Other prerequisites depend on your development language, as follows.

<table>
<thead>
<tr>
<th>Language</th>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="python.png" alt="Python" /></td>
<td>Python &gt;= 3.6</td>
</tr>
<tr>
<td><img src="typescript.png" alt="TypeScript" /></td>
<td>TypeScript &gt;= 2.7</td>
</tr>
<tr>
<td><img src="java.png" alt="Java" /></td>
<td>Java &gt;= 1.8</td>
</tr>
</tbody>
</table>

Installing the AWS CDK

To install and configure the AWS CDK, please refer to the AWS CDK Developer Guide - Installing the AWS CDK.

Working with AWS Solutions Constructs

The typical workflow for creating a new app when working with AWS Solutions Constructs follows the same approach as the AWS CDK.
1. Create the app directory.
2. Initialize the app.
3. Add the AWS Solutions Constructs pattern dependencies.
4. Add additional code to the app.
5. Compile the app, if necessary.
6. Deploy the resources defined in the app.
7. Test the app.

If there are any issues, loop through modify, compile (if necessary), deploy, and test again.

Walkthrough - Part 1

Note
AWS Solutions Constructs is supported on AWS CDK versions ≥ 1.46.0.

This tutorial walks you through how to create and deploy a simple "Hello Constructs" AWS CDK app that uses a pattern from AWS Solutions Constructs, from initializing the project to deploying the resulting AWS CloudFormation template. The Hello Constructs app will create the following simple solution:

Hello Constructs
Let's get started building our first AWS CDK App using pattern based development.

Note
This is a sample modification of Hello CDK! from the CDK Workshop. If this is your first time using the AWS CDK, we recommend starting with this workshop for a hands-on walkthrough and how to leverage the CDK in building a real-world project.
Creating the App Directory and Initializing the AWS CDK

Create a directory for your CDK app, and then create a AWS CDK app in that directory.

**TypeScript**

```
mkdir hello-constructs
cd hello-constructs
cdk init --language typescript
```

**Python**

```
mkdir hello-constructs
cd hello-constructs
cdk init --language python
```

**Tip**

Now's a good time to open the project in your favorite IDE and explore. To learn more about the project structure, select the appropriate link:

- TypeScript
- Python

**Update project base dependencies**

**Warning**

To ensure proper functionality, AWS Solutions Constructs and AWS CDK packages must use the same version number within your project. For example, if you are using AWS Solutions Constructs v.1.52.0, you must also use AWS CDK v.1.52.0.

**Tip**

Take note of the most recent version of AWS Solutions Constructs, and apply that version number to the VERSION_NUMBER placeholders in the steps below (for both AWS Solutions Constructs and AWS CDK packages). To check all the public releases of the Constructs library, click here.

**TypeScript**

Edit the `package.json` file with the following information:

```json
"devDependencies": {
  "@aws-cdk/assert": "VERSION_NUMBER",
  "@types/jest": "^24.0.22",
  "@types/node": "10.17.5",
  "jest": "^24.9.0",
  "ts-jest": "^24.1.0",
  "aws-cdk": "VERSION_NUMBER",
  "ts-node": "^8.1.0",
  "typescript": "^3.7.2"
}
```
Update project base dependencies

Python

Edit the `setup.py` file with the following information:

```python
install_requires=[
    "aws-cdk.core==VERSION_NUMBER",
],
```

Install the projects base dependencies.

TypeScript

```bash
npm install
```

Python

```bash
source .venv/bin/activate
pip install -r requirements.txt
```

Build and run the app and confirm that it creates an empty stack.

TypeScript

```bash
npm run build
cdk synth
```

Python

```bash
cdk synth
```

You should see a stack like the following, where `CDK-VERSION` is the version of the CDK. (Your output may differ slightly from what’s shown here.)

TypeScript

```
Resources:
    CDKMetadata:
        Type: AWS::CDK::Metadata
```
Lambda handler code

We'll start with the AWS Lambda handler code. 

Create a directory `lambda` in the root of your project tree.

**TypeScript**

Add a file called `lambda/hello.js` with the following contents:

```javascript
exports.handler = async function(event) {
    console.log("request: ", JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        headers: { "Content-Type": "text/plain" },
        body: 'Hello, AWS Solutions Constructs! You've hit ${event.path}\n'
    };
};
```

**Python**

Add a file called `lambda/hello.py` with the following contents:

```python
import json

def handler(event, context):
    print('request: {}'.format(json.dumps(event)))
    return {
        'statusCode': 200,
        'headers': {
            'Content-Type': 'text/plain'
        },
        'body': 'Hello, CDK! You have hit {}
'.format(event['path'])
    }
```

This is a simple Lambda function which returns the text "Hello, Constructs! You've hit [url path]". The function's output also includes the HTTP status code and HTTP headers. These are used by API Gateway to formulate the HTTP response to the user.
This Lambda is provided in JavaScript. For more information on writing Lambda functions in your language of choice, refer to the [AWS Lambda documentation](https://docs.aws.amazon.com/lambda/latest/dg/).

## Install the AWS CDK and AWS Solutions Constructs dependencies

The AWS Solutions Constructs is shipped with an extensive library of constructs. The library is divided into modules, one for each well-architected pattern. For example, if you want to define an Amazon API Gateway Rest API to an AWS Lambda function, we will need to use the `aws-apigateway-lambda` pattern library.

We also need to add the AWS Lambda and Amazon API Gateway construct library from the AWS CDK.

Install the AWS Lambda module and all its dependencies into our project:

**Note**
Remember to substitute the correct, matching version to be used for both AWS Solutions Constructs and the AWS CDK into the `VERSION_NUMBER` placeholder fields for each command. Mismatching versions between packages may cause errors.

### TypeScript

```bash
npm install -s @aws-cdk/aws-lambda@VERSION_NUMBER
```

### Python

```bash
pip install aws_cdk.aws_lambda==VERSION_NUMBER
```

Next, install the Amazon API Gateway module and all its dependencies into our project:

### TypeScript

```bash
npm install -s @aws-cdk/aws-apigateway@VERSION_NUMBER
```

### Python

```bash
pip install aws_cdk.aws_apigateway==VERSION_NUMBER
```

Finally, install the AWS Solutions Constructs `aws-apigateway-lambda` module and all its dependencies into our project:

### TypeScript

```bash
npm install -s @aws-solutions-constructs/aws-apigateway-lambda@VERSION_NUMBER
```
Add an Amazon API Gateway/AWS Lambda pattern to your stack

Now, let's define the AWS Solutions Constructs pattern for implementing an Amazon API Gateway with an AWS Lambda proxy.

TypeScript

Edit the file `lib/hello-constructs.ts` with the following:

```typescript
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import * as api from '@aws-cdk/aws-apigateway';
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-constructs/aws-apigateway-lambda';

export class HelloConstructsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // The code that defines your stack goes here
    const api_lambda_props: ApiGatewayToLambdaProps = {
      lambdaFunctionProps: {
        code: lambda.Code.fromAsset('lambda'),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'hello.handler'
      },
      apiGatewayProps: {
        defaultMethodOptions: {
          authorizationType: api.AuthorizationType.NONE
        }
      }
    };

    new ApiGatewayToLambda(this, 'ApiGatewayToLambda', api_lambda_props);
  }
}
```

Python

Edit the file `hello_constructs/hello_constructs_stack.py` with the following:

```python
from aws_cdk import (  
    aws_lambda as _lambda,  
    aws_apigateway as apigw,  
    core,
)

from aws_solutions_constructs import (  
    aws_apigateway_lambda as apigw_lambda
)  
```


class HelloConstructsStack(core.Stack):
    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)
        # The code that defines your stack goes here
        apigw_lambda.ApiGatewayToLambda(
            self, 'ApiGatewayToLambda',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
                code=_lambda.Code.asset('lambda'),
                handler='hello.handler',
            ),
            api_gateway_props=apigw.RestApiProps(
                default_method_options=apigw.MethodOptions(
                    authorization_type=apigw.AuthorizationType.NONE
                )
            )
        )

That's it. This is all you need to do in order to define an API Gateway which proxies all requests to an AWS Lambda function. Let's compare our new stack to the original one:

TypeScript

```bash
npm run build
cdk diff
```

Python

```bash
cdk diff
```

The output should look like this:

```
Stack HelloConstructsStack
IAM Statement Changes

# # Resource # Effect # Action # Principal
# Condition #

# + # ${LambdaFunction.Arn} # Allow # lambda:InvokeFunction #
Service:apigateway.amazonaws.com # "ArnLike": {
# # # # #
# "AWS:SourceArn": "arn:${AWS::Region}:${AWS::AccountId}:${RestApi0C43BF4B}/${RestApi/DeploymentStage.prod}/*" #
```
Add an Amazon API Gateway/AWS Lambda pattern to your stack

```yaml
# + ${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:${RestApi0C43BF4B}/test_invoke-stage/*/{proxy+)
# + ${LambdaFunction.Arn}       # Allow  # lambda:InvokeFunction
Service:apigateway.amazonaws "ArnLike": { #
  
  "AWS:SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:${RestApi0C43BF4B}/test_invoke-stage/*/{proxy+}"
} #
# + # ${LambdaFunction.Arn}      # Allow  # lambda:InvokeFunction
Service:apigateway.amazonaws "ArnLike": { #
  
  "AWS:SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:${RestApi0C43BF4B}/test_invoke-stage/*/
} #
# + ${LambdaFunctionServiceRole # Allow  # sts:AssumeRole
Service:lambda.amazonaws.co #
#   # .Arn}                       #
# + ${LambdaRestApiCloudWatchRole # Allow  # sts:AssumeRole
Service:apigateway.amazonaws #
#   # .Arn}                     #
```
AWS Solutions Constructs AWS Solutions

Add an Amazon API Gateway/AWS Lambda pattern to your stack

# logs:DescribeLogStreams
# logs:FilterLogEvents
# logs:GetLogEvents
# logs:PutLogEvents

# + arn:aws:logs:{AWS::Region} # Allow logs:CreateLogGroup AWS:
# {LambdaFunctionService 
# +:#{AWS::AccountId}:log-grou 
# logs:CreateLogStream # Role}
# p/aws/lambda/* 
# logs:PutLogEvents

(NOTE: There may be security-related changes not in this list. See https://github.com/aws/aws-cdk/issues/1299)

Parameters

[+] Parameter AssetParameters/
ba91444ebd64d9419e8cfee41f7f3aa728507dda28788a2fc40574646c4340a/S3Bucket
AssetParametersba91444ebd64d9419e8cfee41f7f3aa728507dda28788a2fc40574646c4340a/S3Bucket9780A3BC:
{"Type":"String","Description":"S3 bucket for asset "ba91444ebd64d9419e8cfee41f7f3aa728507dda28788a2fc40574646c4340a""}

[+] Parameter AssetParameters/
ba91444ebd64d9419e8cfee41f7f3aa728507dda28788a2fc40574646c4340a/S3VersionKey
AssetParametersba91444ebd64d9419e8cfee41f7f3aa728507dda28788a2fc40574646c4340a/S3VersionKey37F36FFB:
{"Type":"String","Description":"S3 key for asset version "ba91444ebd64d9419e8cfee41f7f3aa728507dda28788a2fc40574646c4340a""}

[+] Parameter AssetParameters/
ba91444ebd64d9419e8cfee41f7f3aa728507dda28788a2fc40574646c4340a/ArtifactHash
AssetParametersba91444ebd64d9419e8cfee41f7f3aa728507dda28788a2fc40574646c4340a/ArtifactHash80199FBC:
{"Type":"String","Description":"Artifact hash for asset "ba91444ebd64d9419e8cfee41f7f3aa728507dda28788a2fc40574646c4340a""}

Conditions

[+] Condition CDKMetadataAvailable: ("Fn::Or":{"Fn::Or":{"Fn::Equals":
{"Ref":"AWS::Region"},"ap-east-1"}],{"Fn::Equals":{"Ref":"AWS::Region"},"ap-northeast-2"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"ap-southeast-1"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"ap-southeast-2"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"ca-central-1"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"cn-north-1"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"cn-northwest-1"}],
{"Fn::Equals":
{"Ref":"AWS::Region"},"eu-central-1"}],{"Fn::Or":
{"Fn::Equals":
{"Ref":"AWS::Region"},"eu-north-1"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"eu-west-1"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"eu-west-2"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"eu-west-3"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"me-south-1"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"sa-east-1"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"us-east-1"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"us-east-2"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"us-west-1"}],{"Fn::Equals":
{"Ref":"AWS::Region"},"us-west-2"}])

Resources

[+] AWS::Logs::LogGroup ApiGatewayToLambda/ApiAccessLogGroup
ApiGatewayToLambda/ApiAccessLogGroupE2B41502
[+] AWS::IAM::Role LambdaFunctionServiceRole LambdaFunctionServiceRole0C4CDE0B
[+] AWS::Lambda::Function LambdaFunctionBF21E41F
[+] AWS::ApiGateway::RestApi RestApi RestApi0C43BF4B
[+] AWS::ApiGateway::Deployment RestApi/Deployment
RestApiDeployment180EC503d2c6f3c8dc8db7193bd98c10a0bf4e677
[+] AWS::ApiGateway::Stage RestApi/DeploymentStage.prod RestApiDeploymentStageprod38S5DE66
[+] AWS::ApiGateway::Resource RestApi/Default/{proxy+} RestApiproxyC95856DD
[+] AWS::Lambda::Permission RestApi/Default/{proxy+}/ANY
ApiPermission.HelloCon structsStackRestApiFDB18C28E.ANY..{proxy+}
RestApiproxyANYApiPermissionHelloCon structsStackRestApiFDB18C2EANYproxyE43D39B3
That's nice. This simple example with one well-architected pattern from the AWS Solutions Constructs added 21 new resources to your stack.

**cdk deploy**

**Tip**
Before you can deploy your first AWS CDK app containing a Lambda function, you must bootstrap your AWS environment. This creates a staging bucket that the AWS CDK uses to deploy stacks containing assets. If this is the first time you are using the AWS CDK to deploy assets, you will need to run the `cdk bootstrap` to deploy the CDK toolkit stack into your AWS environment.

Okay, ready to deploy?

```
cdk deploy
```

**Stack outputs**

When deployment is complete, you'll notice this line:

```
Outputs:
HelloConstructsStack.RestApiEndpoint0551178A = https://xxxxxxxxxx.execute-api.us-east-1.amazonaws.com/prod/
```

This is a stack output that's automatically added by the AWS Solutions Constructs pattern and includes the URL of the API Gateway endpoint.

**Testing your app**

Let's try to hit this endpoint with curl. Copy the URL and execute (your prefix and Region will likely be different).

```
curl https://xxxxxxxxxx.execute-api.us-east-1.amazonaws.com/prod/
```
Walkthrough - Part 2

**Note**
AWS Solutions Constructs is supported on AWS CDK versions ≥ 1.46.0.

This tutorial walks you through how to modify the “Hello Constructs” app created in part 1 (p. 3). Our modification will add a site hit counter using the AWS Lambda to DynamoDB pattern from AWS Solutions Constructs. Modifying the Hello Constructs app will result in the following solution:

Hit Counter Lambda code

Let’s get started by writing the code for the Hit Counter AWS Lambda function. This function will:

- increment a counter related to the API path in a Amazon DynamoDB table,
- invoke the downstream Hello AWS Lambda function,
- and return the response to end user.

TypeScript

Add a file called `lambda/hitcounter.js` with the following contents:
const { DynamoDB, Lambda } = require('aws-sdk');

exports.handler = async function(event) {
  console.log("request:", JSON.stringify(event, undefined, 2));

  // create AWS SDK clients
  const dynamo = new DynamoDB();
  const lambda = new Lambda();

  // update dynamo entry for "path" with hits++
  await dynamo.updateItem({
    TableName: process.env.DDB_TABLE_NAME,
    Key: { path: { S: event.path } },
    UpdateExpression: 'ADD hits :incr',
    ExpressionAttributeValues: { ':incr': { N: '1' } }
  }).promise();

  // call downstream function and capture response
  const resp = await lambda.invoke({
    FunctionName: process.env.DOWNSTREAM_FUNCTION_NAME,
    Payload: JSON.stringify(event)
  }).promise();

  console.log('downstream response:', JSON.stringify(resp, undefined, 2));

  // return response back to upstream caller
  return JSON.parse(resp.Payload);
};

Python

Add a file called lambda/hitcounter.py with the following contents:

import json
import os
import boto3

ddb = boto3.resource('dynamodb')
table = ddb.Table(os.environ['DDB_TABLE_NAME'])
_lambda = boto3.client('lambda')

def handler(event, context):
  print('request: {}'.format(json.dumps(event)))
  table.update_item({
    'Key': {'path': event['path']},
    'UpdateExpression': 'ADD hits :incr',
    'ExpressionAttributeValues': {':incr': 1}
  })

  resp = _lambda.invoke({
    'FunctionName': os.environ['DOWNSTREAM_FUNCTION_NAME'],
    'Payload': json.dumps(event),
  })

  body = resp['Payload'].read()

  print('downstream response: {}'.format(body))
  return json.loads(body)
Install the new dependencies

Note
Remember to substitute the correct, matching version to be used for both AWS Solutions Constructs and the AWS CDK into the VERSION_NUMBER placeholder fields for each command. This should be identical to the version number used for dependencies in the first part of this walkthrough. Mismatching versions between packages may cause errors.

As usual, we first need to install the dependencies we need for our solution update. First, we need to install the DynamoDB construct library:

TypeScript

```typescript
npm install -s @aws-cdk/aws-dynamodb@VERSION_NUMBER
```

Python

```python
pip install aws_cdk.aws_dynamodb==VERSION_NUMBER
```

Finally, install the AWS Solutions Constructs aws-lambda-dynamodb module and all its dependencies into our project:

TypeScript

```typescript
npm install -s @aws-solutions-constructs/aws-lambda-dynamodb@VERSION_NUMBER
```

Python

```python
pip install aws_solutions_constructs.aws_lambda_dynamodb==VERSION_NUMBER
```

Define the resources

Now, let's update our stack code to accommodate our new architecture.

First, we are going to import our new dependencies and move the "Hello" function outside of the aws-apigateway-lambda pattern we created in part 1.

TypeScript

Edit the file `lib/hello-constructs.ts` with the following:

```typescript
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import * as api from '@aws-cdk/aws-apigateway';
```
import * as dynamodb from '@aws-cdk/aws-dynamodb';
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-constructs/aws-apigateway-lambda';
import { LambdaToDynamoDB, LambdaToDynamoDBProps } from '@aws-solutions-constructs/aws-lambda-dynamodb';

export class HelloConstructsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const helloFunc = new lambda.Function(this, 'HelloHandler', {
      runtime: lambda.Runtime.NODEJS_12_X,
      code: lambda.Code.fromAsset('lambda'),
      handler: 'hello.handler'
    });

    const api_lambda_props: ApiGatewayToLambdaProps = {
      lambdaFunctionProps: {
        code: lambda.Code.fromAsset('lambda'),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'hello.handler'
      },
      apiGatewayProps: {
        defaultMethodOptions: {
          authorizationType: api.AuthorizationType.NONE
        }
      }
    };

    new ApiGatewayToLambda(this, 'ApiGatewayToLambda', api_lambda_props);
  }
}

Python

Edit the file hello_constructs/hello_constructs_stack.py with the following:

```python
from aws_cdk import (
    aws_lambda as _lambda,
    aws_apigateway as apigw,
    aws_dynamodb as ddb,
    core,
)

from aws_solutions_constructs import (
    aws_apigateway_lambda as apigw_lambda,
    aws_lambda_dynamodb as lambda_ddb
)

class HelloConstructsStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        # The code that defines your stack goes here

        self._handler = _lambda.Function(
            self, 'HelloHandler',
            runtime=_lambda.Runtime.PYTHON_3_7,
            handler='hello.handler',
            code=_lambda.Code.asset('lambda'),
```
Next, we are going to add the `aws-lambda-dynamodb` pattern to build out the hit counter service for our updated architecture.

The next update below defines the properties for the `aws-lambda-dynamodb` pattern by defining the AWS Lambda function with the Hit Counter handler. Additionally, the Amazon DynamoDB table is defined with a name of `Hits` and a partition key of `path`.

**TypeScript**

Edit the file `lib/hello-constructs.ts` with the following:

```typescript
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
```
import * as api from '@aws-cdk/aws-apigateway';
import * as dynamodb from '@aws-cdk/aws-dynamodb';
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-constructs/aws-apigateway-lambda';
import { LambdaToDynamoDB, LambdaToDynamoDBProps } from '@aws-solutions-constructs/aws-lambda-dynamodb';

export class HelloConstructsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // The code that defines your stack goes here
    const helloFunc = new lambda.Function(this, 'HelloHandler', {
      runtime: lambda.Runtime.NODEJS_12_X,
      code: lambda.Code.fromAsset('lambda'),
      handler: 'hello.handler'
    });

    // hit counter, aws-lambda-dynamodb pattern
    const lambda_ddb_props: LambdaToDynamoDBProps = {
      lambdaFunctionProps: {
        code: lambda.Code.asset(`lambda`),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'hitcounter.handler',
        environment: {
          DOWNTREAM_FUNCTION_NAME: helloFunc.functionName
        }
      },
      dynamoTableProps: {
        tableName: 'Hits',
        partitionKey: { name: 'path', type: dynamodb.AttributeType.STRING }
      }
    };
    const hitcounter = new LambdaToDynamoDB(this, 'LambdaToDynamoDB', lambda_ddb_props);
    const api_lambda_props: ApiGatewayToLambdaProps = {
      lambdaFunctionProps: {
        code: lambda.Code.fromAsset('lambda'),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'hello.handler'
      },
      apiGatewayProps: {
        defaultMethodOptions: {
          authorizationType: api.AuthorizationType.NONE
        }
      }
    };
    new ApiGatewayToLambda(this, 'ApiGatewayToLambda', api_lambda_props);
  }
}

Python

Edit the file hello_con structs/hello_con structs_stack.py with the following:

```python
from aws_cdk import (  
    aws_lambda as _lambda,  
    aws_apigateway as apigw,  
    aws_dynamodb as ddb,
```
Next, we need to grant the Hit Counter function created from the aws-lambda-dynamodb pattern added above permission to invoke our Hello function.

TypeScript

Edit the file `lib/hello-constructs.ts` with the following:
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import * as api from '@aws-cdk/aws-apigateway';
import * as dynamodb from '@aws-cdk/aws-dynamodb';
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-constructs/aws-apigateway-lambda';
import { LambdaToDynamoDB, LambdaToDynamoDBProps } from '@aws-solutions-constructs/aws-lambda-dynamodb';

export class HelloConstructsStack extends cdk.Stack {
    constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
        super(scope, id, props);

        // The code that defines your stack goes here

        // hello function responding to http requests
        const helloFunc = new lambda.Function(this, 'HelloHandler', {
            runtime: lambda.Runtime.NODEJS_12_X,
            code: lambda.Code.fromAsset('lambda'),
            handler: 'hello.handler'
        });

        // hit counter, aws-lambda-dynamodb pattern
        const lambda_ddb_props: LambdaToDynamoDBProps = {
            lambdaFunctionProps: {
                code: lambda.Code.asset(`lambda`),
                runtime: lambda.Runtime.NODEJS_12_X,
                handler: 'hitcounter.handler',
                environment: {
                    DOWNSTREAM_FUNCTION_NAME: helloFunc.functionName
                }
            },
            dynamoTableProps: {
                tableName: 'Hits',
                partitionKey: { name: 'path', type: dynamodb.AttributeType.STRING }
            }
        };

        const hitcounter = new LambdaToDynamoDB(this, 'LambdaToDynamoDB', lambda_ddb_props);

        // grant the hitcounter lambda role invoke permissions to the hello function
        helloFunc.grantInvoke(hitcounter.lambdaFunction);

        const api_lambda_props: ApiGatewayToLambdaProps = {
            lambdaFunctionProps: {
                code: lambda.Code.fromAsset('lambda'),
                runtime: lambda.Runtime.NODEJS_12_X,
                handler: 'hello.handler'
            },
            apiGatewayProps: {
                defaultMethodOptions: {
                    authorizationType: api.AuthorizationType.NONE
                }
            }
        };

        new ApiGatewayToLambda(this, 'ApiGatewayToLambda', api_lambda_props);  
    }
}
Python

Edit the file `hello_constructs/hello_constructs_stack.py` with the following:

```python
from aws_cdk import (    aws_lambda as _lambda,    aws_apigateway as apigw,    aws_dynamodb as ddb,    core,)

from aws_solutions_constructs import (    aws_apigateway_lambda as apigw_lambda,    aws_lambda_dynamodb as lambda_ddb)

class HelloConstructsStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        # The code that defines your stack goes here

        self.hello_func = _lambda.Function(
            self, 'HelloHandler',
            runtime=_lambda.Runtime.PYTHON_3_7,
            handler='hello.handler',
            code=_lambda.Code.asset('lambda'),
        )

        # hit counter, aws-lambda-dynamodb pattern
        self.hit_counter = lambda_ddb.LambdaToDynamoDB(
            self, 'LambdaToDynamoDB',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
                handler='hitcounter.handler',
                code=_lambda.Code.asset('lambda'),
                environment={
                    'DOWNSTREAM_FUNCTION_NAME': self.hello_func.function_name
                },
            ),
            dynamo_table_props=ddb.TableProps(
                table_name='Hits',
                partition_key={'name': 'path', 'type': ddb.AttributeType.STRING}
            )
        )

        # grant the hitcounter lambda role invoke permissions to the hello function
        self.hello_func.grant_invoke(self.hit_counter.lambda_function)

        apigw_lambda.ApiGatewayToLambda(
            self, 'ApiGatewayToLambda',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
                code=_lambda.Code.asset('lambda'),
                handler='hello.handler',
            ),
            api_gateway_props=apigw.RestApiProps(
                default_method_options=apigw.MethodOptions(
                    authorization_type=apigw.AuthorizationType.NONE
                )
            )
        )
```

21
Finally, we need to update our original `aws-apigateway-lambda` pattern to utilize our new Hit Counter function that was provisioned with the `aws-lambda-dynamodb` pattern above.

**TypeScript**

Edit the file `lib/hello-constructs.ts` with the following:

```typescript
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import * as api from '@aws-cdk/aws-apigateway';
import * as dynamodb from '@aws-cdk/aws-dynamodb';
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-constructs/aws-apigateway-lambda';
import { LambdaToDynamoDB, LambdaToDynamoDBProps } from '@aws-solutions-constructs/aws-lambda-dynamodb';

export class HelloConstructsStack extends cdk.Stack {
    constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
        super(scope, id, props);

        // The code that defines your stack goes here

        // hello function responding to http requests
        const helloFunc = new lambda.Function(this, 'HelloHandler', {
            runtime: lambda.Runtime.NODEJS_12_X,
            code: lambda.Code.fromAsset('lambda'),
            handler: 'hello.handler'
        });

        // hit counter, aws-lambda-dynamodb pattern
        const lambda_ddb_props: LambdaToDynamoDBProps = {
            lambdaFunctionProps: {
                code: lambda.Code.asset(`lambda`),
                runtime: lambda.Runtime.NODEJS_12_X,
                handler: 'hitcounter.handler',
                environment: {
                    DOWNSTREAM_FUNCTION_NAME: helloFunc.functionName
                }
            },
            dynamoTableProps: {
                tableName: 'Hits',
                partitionKey: { name: 'path', type: dynamodb.AttributeType.STRING }
            }
        };

        const hitcounter = new LambdaToDynamoDB(this, 'LambdaToDynamoDB', lambda_ddb_props);

        // grant the hitcounter lambda role invoke permissions to the hello function
        helloFunc.grantInvoke(hitcounter.lambdaFunction);

        const api_lambda_props: ApiGatewayToLambdaProps = {
            existingLambdaObj: hitcounter.lambdaFunction,
            apiGatewayProps: {
                defaultMethodOptions: {
                    authorizationType: api.AuthorizationType.NONE
                }
            }
        };
    }
}
```
Define the resources

```python
from aws_cdk import (    aws_lambda as _lambda,    aws_apigateway as apigw,    aws_dynamodb as ddb,    core,) from aws_solutions_constructs import (    aws_apigateway_lambda as apigw_lambda,    aws_lambda_dynamodb as lambda_ddb)
class HelloConstructsStack(core.Stack):
    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)
        # The code that defines your stack goes here
        # hit counter, aws-lambda-dynamodb pattern
        self.hit_counter = lambda_ddb.LambdaToDynamoDB(            self, 'LambdaToDynamoDB',            lambda_function_props=_lambda.FunctionProps(                runtime=_lambda.Runtime.PYTHON_3_7,                handler='hitcounter.handler',                code=_lambda.Code.asset('lambda'),                environment={                    'DOWNSTREAM_FUNCTION_NAME': self.hello_func.function_name                }),            dynamo_table_props=ddb.TableProps(                table_name='Hits',                partition_key={                    'name': 'path',                    'type': ddb.AttributeType.STRING                })        )
        # grant the hitcounter lambda role invoke permissions to the hello function
        self.hello_func.grant_invoke(self.hit_counter.lambda_function)
        apigw_lambda.ApiGatewayToLambda(            self, 'ApiGatewayToLambda',            existing_lambda_obj=self.hit_counter.lambda_function,            api_gateway_props=apigw.RestApiProps(        )
```

Edit the file `hello_constructs/hello_constructs_stack.py` with the following:

```python
from aws_cdk import (    aws_lambda as _lambda,    aws_apigateway as apigw,    aws_dynamodb as ddb,    core,) from aws_solutions_constructs import (    aws_apigateway_lambda as apigw_lambda,    aws_lambda_dynamodb as lambda_ddb)
class HelloConstructsStack(core.Stack):
    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)
        # The code that defines your stack goes here
        # hit counter, aws-lambda-dynamodb pattern
        self.hit_counter = lambda_ddb.LambdaToDynamoDB(            self, 'LambdaToDynamoDB',            lambda_function_props=_lambda.FunctionProps(                runtime=_lambda.Runtime.PYTHON_3_7,                handler='hitcounter.handler',                code=_lambda.Code.asset('lambda'),                environment={                    'DOWNSTREAM_FUNCTION_NAME': self.hello_func.function_name                }),            dynamo_table_props=ddb.TableProps(                table_name='Hits',                partition_key={                    'name': 'path',                    'type': ddb.AttributeType.STRING                })        )
        # grant the hitcounter lambda role invoke permissions to the hello function
        self.hello_func.grant_invoke(self.hit_counter.lambda_function)
        apigw_lambda.ApiGatewayToLambda(            self, 'ApiGatewayToLambda',            existing_lambda_obj=self.hit_counter.lambda_function,            api_gateway_props=apigw.RestApiProps(        )
```
default_method_options=apigw.MethodOptions(
    authorization_type=apigw.AuthorizationType.NONE
)  

Review the changes

Let's build our project and review the changes to our resources that will happen when we deploy this:

```
npm run build
cdk diff
```

Our output should look like this:

```
Stack HelloConstructsStack
IAM Statement Changes
# # Resource                          # Effect # Action                           # Principal                          # Condition #
# # # ${HelloHandler.Arn}               # Allow # lambda:InvokeFunction             # AWS:
#{LambdaFunctionServiceRole}   # #
# # # ${HelloHandler/ServiceRole.Arn}   # Allow # sts:AssumeRole                    # Service:lambda.amazonaws.com   # #
# # # ${LambdaToDynamoDB/DynamoTable.A # Allow # dynamodb:BatchGetItem             # AWS:
#{LambdaFunctionServiceRole}   # # n}
# # # # # dynamodb:BatchWriteItem       #
# # # # # dynamodb:DeleteItem           #
# # # # # dynamodb:GetItem               #
# # # # # dynamodb:GetRecords           #
# # # # # dynamodb:GetShardIterator      #
# # # # # dynamodb:PutItem              #
# # # # # dynamodb:Query                 #
# # # # # dynamodb:Scan                  #
# # # # # dynamodb:UpdateItem           #
# # # # #
IAM Policy Changes
# # Resource                          # Managed Policy ARN
# # ${HelloHandler/ServiceRole} # arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole #
(NOTE: There may be security-related changes not in this list. See https://github.com/aws/aws-cdk/issues/1299)
```
cdk deploy

Okay, ready to deploy?

cdk deploy

Stack outputs

When deployment is complete, you'll notice this line:

```
Outputs:
HelloConstructsStack.RestApiEndpoint0551178A = https://xxxxxxxxxx.execute-api.us-east-1.amazonaws.com/prod/
```

Testing your app

Let's try to hit this endpoint with curl. Copy the URL and execute (your prefix and region will likely be different).

```
curl https://xxxxxxxxxx.execute-api.us-east-1.amazonaws.com/prod/
```

Output should look like this:

```
Hello, AWS Solutions Constructs! You've hit /
```

Now, let's review the Hits Amazon DynamoDB table.

1. Go to the DynamoDB console.
2. Make sure you are in the Region where you created the table.
3. Select Tables in the navigation pane and select the Hits table.
4. Open the table and select “Items”.
5. You should see how many hits you got for each path.

6. Try hitting a new path and refresh the Items view. You should see a new item with a hits count of one.

If this is the output you received, your app works!

Sample Use Cases

This library includes a collection of functional use case implementations to demonstrate the usage of Constructs architectural patterns. These can be used in the same way as architectural patterns, and can be conceptualized as an additional "higher-level" abstraction of those patterns. The following use cases are provided as functional examples.
AWS Solutions Constructs AWS Solutions

AWS Static S3 Website

This use case pattern (aws-s3-static-website) implements an Amazon CloudFront distribution, Amazon S3 bucket, and AWS Lambda-based custom resource to copy the static website content for the Wild Rydes demo website (part of the aws-serverless-web-app implementation).

Source Code (aws-s3-static-website)
https://github.com/awslabs/aws-solutions-constructs/tree/master/source/use_cases/aws-s3-static-website

AWS Simple Serverless Image Handler

This use case pattern (aws-serverless-image-handler) implements an Amazon CloudFront distribution, an Amazon API Gateway REST API, an AWS Lambda function, and necessary permissions/logic to provision a functional image handler API for serving image content from one or more Amazon S3 buckets within the deployment account.

Source Code (aws-serverless-image-handler)

AWS Serverless Web App

This use case pattern (aws-serverless-web-app) implements a simple serverless web application that enables users to request unicorn rides from the Wild Rydes fleet. The application will present users with an HTML based user interface for indicating the location where they would like to be picked up and will interface on the backend with a RESTful web service to submit the request and dispatch a nearby unicorn. The application will also provide facilities for users to register with the service and log in before requesting rides.

Source Code (aws-serverless-web-app)
https://github.com/awslabs/aws-solutions-constructs/tree/master/source/use_cases/aws-serverless-web-app

AWS Restaurant Management Demo

This use case pattern (aws-restaurant-management-demo) implements a complex, multi-stack architecture that models a restaurant management system. This use case will provision a stack for service/wait staff to open/close orders, a stack for kitchen staff to view/complete orders, and a stack for managers to perform various business functions. It will also provision a stack containing a central DynamoDB table for managing orders, as well as a Lambda layer for sharing common database access patterns.

Source Code (aws-restaurant-management-demo)
AWS Solutions Constructs AWS Solutions

Modules

AWS Solutions Constructs is organized into several modules. They are named like this:

- **aws-xxx**: Well-architected pattern package for the indicated services. This package will contain constructs that contain multiple AWS CDK service modules to configure the given pattern.
- **xxx**: Packages that don't start "aws-" are Constructs core modules that are used to configure best practice defaults for services used within the pattern library.

Module Contents

Modules contain the following types:

- **Patterns** - All higher-level, multi-services constructs in this library.
- **Other Types** - All non-construct classes, interfaces, structs and enums that exist to support the patterns.

Patterns take a set of (input) properties in their constructor; the set of properties (and which ones are required) can be seen on a pattern's documentation page.

The pattern's documentation page also lists the available methods to call and the properties which can be used to retrieve information about the pattern after it has been instantiated.

aws-alb-lambda

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>aws_solutions_constructs.aws_alb_lambda</td>
</tr>
</tbody>
</table>
This AWS Solutions Construct implements an Application Load Balancer to an AWS Lambda function.

Here is a minimal deployable pattern definition in Typescript:

```typescript
// Obtain a pre-existing certificate from your account
const certificate = acm.Certificate.fromCertificateArn(
  scope,
  'existing-cert',
  "arn:aws:acm:us-east-1:123456789012:certificate/11112222-3333-1234-1234-123456789012"
);
const props: AlbToLambdaProps = {
  lambdaFunctionProps: {
    code: lambda.Code.fromAsset(${__dirname}/lambda),
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler'
  },
  listenerProps: {
    certificates: [ certificate ]
  },
  publicApi: true
};
new AlbToLambda(stack, 'new-construct', props);
```

### Parameters
- **scope** `Construct`
- **id** `string`
- **props** `AlbToLambdaProps`

### Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loadBalancerProps?</td>
<td><code>elasticloadbalancingv2.ApplicationLoadBalancerProps</code></td>
<td>Optional custom properties for a new loadBalancer. Providing both this and</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>existingLoadBalancerObj?</td>
<td>elasticloadbalancingv2.ApplicationLoadBalancer</td>
<td>Existing Application Load Balancer to incorporate into the construct architecture. Providing both this and loadBalancerProps is an error. The VPC containing this loadBalancer must match the VPC provided in existingVpc.</td>
</tr>
<tr>
<td>listenerProps?</td>
<td>ApplicationListenerProps</td>
<td>Props to define the listener. Must be provided when adding the listener to an ALB (eg - when creating the alb), may not be provided when adding a second target to an already established listener. When provided, must include either a certificate or protocol: HTTP</td>
</tr>
<tr>
<td>targetProps?</td>
<td>ApplicationTargetGroupProps</td>
<td>Optional custom properties for a new target group. While this is a standard attribute of props for ALB constructs, there are few pertinent properties for a Lambda target.</td>
</tr>
<tr>
<td>ruleProps?</td>
<td>AddRuleProps</td>
<td>Rules for directing traffic to the target being created. May not be specified for the first listener added to an ALB, and must be specified for the second target added to a listener. Add a second target by instantiating this construct a second time and providing the existingAlb from the first instantiation.</td>
</tr>
<tr>
<td>vpcProps?</td>
<td>ec2.VpcProps</td>
<td>Optional custom properties for a VPC the construct will create. This VPC will be used by the new ALB and any Private Hosted Zone the construct creates (that's why loadBalancerProps and privateHostedZoneProps can't include a VPC). Providing both this and existingVpc is an error.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user provided props to override the default props for the Lambda function.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td>ec2.IVpc</td>
<td>An existing VPC in which to deploy the construct. Providing both this and vpcProps is an error. If the client provides an existing load balancer and/or existing Private Hosted Zone, those constructs must exist in this VPC.</td>
</tr>
<tr>
<td>logAlbAccessLogs?</td>
<td>boolean</td>
<td>Whether to turn on Access Logs for the Application Load Balancer. Uses an S3 bucket with associated storage costs. Enabling Access Logging is a best practice. default - true</td>
</tr>
<tr>
<td>albLoggingBucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional properties to customize the bucket used to store the ALB Access Logs. Supplying this and setting logAccessLogs to false is an error. @default - none</td>
</tr>
<tr>
<td>publicApi</td>
<td>boolean</td>
<td>Whether the construct is deploying a private or public API. This has implications for the VPC and ALB.</td>
</tr>
</tbody>
</table>

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpc</td>
<td>ec2.IVpc</td>
<td>The VPC used by the construct (whether created by the construct or provided by the client)</td>
</tr>
<tr>
<td>loadBalancer</td>
<td>elasticloadbalancingv2.ApplicationLoadBalancer</td>
<td>The Load Balancer used by the construct (whether created by the construct or provided by the client)</td>
</tr>
</tbody>
</table>
### Default settings

Out of the box implementation of the Construct without any override will set the following defaults:

#### Application Load Balancer

- Creates or configures an Application Load Balancer with:
  - Required listeners
  - New target group with routing rules if appropriate

#### AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for NodeJs Lambda function
- Enable X-Ray Tracing
- Set Environment Variables
  - AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function used in the pattern.</td>
</tr>
<tr>
<td>listener</td>
<td>elb.ApplicationListener</td>
<td>The listener used by this pattern.</td>
</tr>
</tbody>
</table>
Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-alb-lambda

aws-apigateway-dynamodb

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
### Overview

This AWS Solutions Construct implements an Amazon API Gateway REST API connected to an Amazon DynamoDB table.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { ApiGatewayToDynamoDBProps, ApiGatewayToDynamoDB } from '@aws-solutions-constructs/aws-apigateway-dynamodb';

new ApiGatewayToDynamoDB(this, 'test-api-gateway-dynamodb-default', {});
```

### Initializer

```typescript
new ApiGatewayToDynamoDB(scope: Construct, id: string, props: ApiGatewayToDynamoDBProps);
```

**Parameters**

- `scope` `Construct`
- `id` `string`
- `props` `ApiGatewayToDynamoDBProps` (p. 34)

### Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamoTableProps?</td>
<td><code>dynamodb.TableProps</code></td>
<td>Optional user provided props to override the default props for DynamoDB Table</td>
</tr>
<tr>
<td>existingTableObj?</td>
<td><code>dynamodb.Table</code></td>
<td>Existing instance of DynamoDB table object, providing both this</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>apiGatewayProps?</td>
<td>api.RestApiProps</td>
<td>Optional user-provided props to override the default props for the API Gateway.</td>
</tr>
<tr>
<td>allowCreateOperation?</td>
<td>boolean</td>
<td>Whether to deploy API Gateway Method for Create operation on DynamoDB table.</td>
</tr>
<tr>
<td>createRequestTemplate?</td>
<td>string</td>
<td>API Gateway Request template for Create method, required if allowCreateOperation is set to true.</td>
</tr>
<tr>
<td>allowReadOperation?</td>
<td>boolean</td>
<td>Whether to deploy API Gateway Method for Read operation on DynamoDB table.</td>
</tr>
<tr>
<td>readRequestTemplate?</td>
<td>string</td>
<td>Optional API Gateway Request template for Read method, it will use the default template if allowReadOperation is true and readRequestTemplate is not provided. The default template only supports a partition key and not partition + sort keys.</td>
</tr>
<tr>
<td>allowUpdateOperation?</td>
<td>boolean</td>
<td>Whether to deploy API Gateway Method for Update operation on DynamoDB table.</td>
</tr>
<tr>
<td>updateRequestTemplate?</td>
<td>string</td>
<td>API Gateway Request template for Update method, required if allowUpdateOperation is set to true.</td>
</tr>
<tr>
<td>allowDeleteOperation?</td>
<td>boolean</td>
<td>Whether to deploy API Gateway Method for Delete operation on DynamoDB table.</td>
</tr>
<tr>
<td>deleteRequestTemplate?</td>
<td>string</td>
<td>Optional API Gateway Request template for Delete method, it will use the default template if allowDeleteOperation is true and deleteRequestTemplate is not provided. The default template only supports a partition key and not partition + sort keys.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td>logs.LogGroupProps</td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGateway</td>
<td>api.RestApi</td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
<tr>
<td>apiGatewayCloudWatchRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the iam.Role created by the construct for API Gateway for CloudWatch access.</td>
</tr>
<tr>
<td>apiGatewayLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group created by the pattern that API Gateway REST API access logs are sent to.</td>
</tr>
<tr>
<td>apiGatewayRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the API Gateway REST API.</td>
</tr>
<tr>
<td>dynamoTable</td>
<td>dynamodb.Table</td>
<td>Returns an instance of the DynamoDB table created by the pattern.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon API Gateway

- Deploy an edge-optimized API endpoint
- Enable CloudWatch logging for API Gateway
- Configure least privilege access IAM role for API Gateway
- Set the default authorizationType for all API methods to IAM
- Enable X-Ray tracing

Amazon DynamoDB Table

- Set the billing mode for DynamoDB Table to On-Demand (Pay per request)
- Enable server-side encryption for DynamoDB Table using AWS managed KMS Key
- Creates a partition key called ‘id’ for DynamoDB Table
- Retain the Table when deleting the CloudFormation stack
- Enable continuous backups and point-in-time recovery
Architecture

Amazon API Gateway → Amazon DynamoDB

Role → Amazon CloudWatch

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>@aws-solutions-constructs/aws-apigateway-dynamodb</td>
</tr>
</tbody>
</table>

aws-apigateway-iot

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>aws_solutions Constructs.aws_apigateway_iot</td>
</tr>
</tbody>
</table>
Overview

This AWS Solutions Construct implements an Amazon API Gateway REST API connected to AWS IoT pattern.

This construct creates a scalable HTTPS proxy between API Gateway and AWS IoT. This comes in handy when wanting to allow legacy devices that do not support the MQTT or MQTT/Websocket protocol to interact with the AWS IoT platform.

This implementation enables write-only messages to be published on given MQTT topics, and also supports shadow updates of HTTPS devices to allowed things in the device registry. It does not involve Lambda functions for proxying messages, and instead relies on direct API Gateway to AWS IoT integration which supports both JSON messages as well as binary messages.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { ApiGatewayToIot } from '@aws-solutions-constructs/aws-apigateway-iot';

new ApiGatewayToIot(this, 'ApiGatewayToIotPattern', {
  iotEndpoint: 'a1234567890123-ats'
});
```

Initializer

```typescript
new ApiGatewayToIot(scope: Construct, id: string, props: ApiGatewayToIotProps);
```

Parameters

- `scope` `Construct`
- `id` `string`
- `props` `ApiGatewayToIotProps` (p. 39)
### Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iotEndpoint</td>
<td>string</td>
<td>The AWS IoT endpoint subdomain to integrate the API Gateway with (e.g a1234567890123-ats).</td>
</tr>
<tr>
<td>apiGatewayCreateApiKey?</td>
<td>boolean</td>
<td>If set to true, an API Key is created and associated to a UsagePlan. User should specify <code>x-api-key</code> header while accessing RestApi. Default value set to false.</td>
</tr>
<tr>
<td>apiGatewayExecutionRole?</td>
<td>iam.Role</td>
<td>The IAM Role used by API Gateway to access AWS IoT. If not specified, a default role is created with wildcard (*) access to all topics and things.</td>
</tr>
<tr>
<td>apiGatewayProps?</td>
<td>api.restApiProps</td>
<td>Optional user-provided props to override the default props for the API Gateway REST API.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td>logs.LogGroupProps</td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
</tbody>
</table>

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGateway</td>
<td>api.RestApi</td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
<tr>
<td>apiGatewayCloudWatchRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the iam.Role created by the construct for API Gateway for CloudWatch access.</td>
</tr>
<tr>
<td>apiGatewayLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group created by the pattern that API Gateway REST API access logs are sent to.</td>
</tr>
<tr>
<td>apiGatewayRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the API Gateway REST API.</td>
</tr>
</tbody>
</table>
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon API Gateway**

- Deploy an edge-optimized API endpoint
- Creates API Resources with **POST** Method to publish messages to IoT Topics
- Creates API Resources with **POST** Method to publish messages to **ThingShadow** and **NamedShadows**
- Enable CloudWatch logging for API Gateway
- Configure IAM role for API Gateway with access to all topics and things
- Set the default authorizationType for all API methods to IAM
- Enable X-Ray Tracing
- Creates a UsagePlan and associates to `prod` stage

Below is a description of the different resources and methods exposed by the API Gateway after deploying the Construct. See the [Examples (p. 41)](#) section for more information on how to easily test these endpoints using `curl`.

<table>
<thead>
<tr>
<th>Method</th>
<th>Resource</th>
<th>Query Parameter(s)</th>
<th>Return Code(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/message/ &lt;topics&gt;</td>
<td>qos</td>
<td>200/403/500</td>
<td>By calling this endpoint, you need to pass the topics on which you would like to publish (e.g <code>message/device/foo</code>).</td>
</tr>
<tr>
<td>POST</td>
<td>/shadow/ &lt;thingName&gt;</td>
<td>None</td>
<td>200/403/500</td>
<td>This route allows to update the shadow document of a thing, given its <strong>thingName</strong> using Unnamed (classic) shadow type. The body shall comply with the standard shadow structure comprising a state node and associated desired and reported nodes. See the <a href="#">Updating device shadows (p. 42)</a> section for an example.</td>
</tr>
<tr>
<td>POST</td>
<td>/shadow/ &lt;thingName&gt;/ &lt;shadowName&gt;</td>
<td>None</td>
<td>200/403/500</td>
<td>This route allows to update the named shadow</td>
</tr>
</tbody>
</table>
Architecture

The following examples only work with API_KEY authentication types, since IAM authorization requires a SIGv4 token to be specified as well, make sure the apiGatewayCreateApiKey property of your Construct props is set to true while deploying the stack, otherwise the below examples won’t work.
Publishing a message

You can use curl to publish a message on different MQTT topics using the HTTPS API. The below example will post a message on the device/foo topic.

```
curl -XPOST https://<stage-id>.execute-api.<region>.amazonaws.com/prod/message/device/foo -H "x-api-key: <api-key>" -H "Content-Type: application/json" -d '{"Hello": "World"}'
```

Note: Replace the stage-id, region, and api-key parameters with your deployment values.

You can chain topic names in the URL and the API accepts up to 7 sub-topics that you can publish on. For instance, the below example publishes a message on the topic device/foo/bar/abc/xyz.

```
```

Updating device shadows

To update the shadow document associated with a given thing, you can issue a shadow state request using a thing name. See the following example on how to update a thing shadow.

```
curl -XPOST https://<stage-id>.execute-api.<region>.amazonaws.com/prod/shadow/device1 -H "x-api-key: <api-key>" -H "Content-Type: application/json" -d '{"state": {"desired": { "Hello": "World" }}}'
```

Updating named shadows

To update the shadow document associated with a given thing's named shadow, you can issue a shadow state request using a thing name and shadow name. See the following example on how to update a named shadow.

```
curl -XPOST https://<stage-id>.execute-api.<region>.amazonaws.com/prod/shadow/device1/shadow1 -H "x-api-key: <api-key>" -H "Content-Type: application/json" -d '{"state": {"desired": { "Hello": "World" }}}'
```

Sending binary payloads

It is possible to send a binary payload to the proxy API, down to the AWS IoT service. In the following example, we send the content of the README.md file associated with this module (treated as a binary data) to device/foo topic using the application/octet-stream content type.

```
curl -XPOST https://<stage-id>.execute-api.<region>.amazonaws.com/prod/message/device/foo/bar/baz/qux -H "x-api-key: <api-key>" -H "Content-Type: application/octet-stream" --data-binary @README.md
```
Note: Execute this command while in the directory of this project. You can then test sending other type of binary files from your file system.

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-apigateway-iot

aws-apigateway-kinesisstreams

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>🐍 Python</td>
<td>aws_solutions Constructs.aws_apigateway_kinesisstream</td>
</tr>
<tr>
<td>🌐 Typescript</td>
<td>@aws-solutions-constructs/aws-apigateway-kinesisstreams</td>
</tr>
<tr>
<td>☕️ Java</td>
<td>software.amazon.awsconstructs.services.apigatewaykinesisstreams</td>
</tr>
</tbody>
</table>

Overview

This pattern implements an Amazon API Gateway REST API connected to an Amazon Kinesis Data Stream.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { ApiGatewayToKinesisStreams, ApiGatewayToKinesisStreamsProps } from '@aws-solutions-constructs/aws-apigateway-kinesisstreams';
new ApiGatewayToKinesisStreams(this, 'test-apigw-kinesis', {});
```
Initializer

```typescript
new ApiGatewayToKinesisStreams(scope: Construct, id: string, props: ApiGatewayToKinesisStreamsProps);
```

**Parameters**

- `scope` `Construct`
- `id` `string`
- `props ApiGatewayToKinesisStreamsProps (p. 44)`

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGatewayProps?</td>
<td><code>api.RestApiProps</code></td>
<td>Optional user-provided props to override the default props for the API Gateway REST API.</td>
</tr>
<tr>
<td>putRecordRequestTemplate?</td>
<td><code>string</code></td>
<td>API Gateway request template for the PutRecord action. If not provided, a default one will be used.</td>
</tr>
<tr>
<td>putRecordRequestModel?</td>
<td><code>api.ModelOptions</code></td>
<td>API Gateway request model for the PutRecord action. If not provided, a default one will be created.</td>
</tr>
<tr>
<td>putRecordsRequestTemplate?</td>
<td><code>string</code></td>
<td>API Gateway request template for the PutRecords action. If not provided, a default one will be used.</td>
</tr>
<tr>
<td>putRecordRequestModel?</td>
<td><code>api.ModelOptions</code></td>
<td>API Gateway request model for the PutRecords action. If not provided, a default one will be created.</td>
</tr>
<tr>
<td>existingStreamObj?</td>
<td><code>kinesis.Stream</code></td>
<td>Existing instance of Kinesis Stream, providing both this and kinesisStreamProps will cause an error.</td>
</tr>
<tr>
<td>kinesisStreamProps?</td>
<td><code>kinesis.StreamProps</code></td>
<td>Optional user-provided props to override the default props for the Kinesis stream.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td><code>boolean</code></td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGateway</td>
<td>api.RestApi</td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
<tr>
<td>apiGatewayRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the API Gateway REST API.</td>
</tr>
<tr>
<td>apiGatewayCloudWatchRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the iam.Role created by the construct for API Gateway for CloudWatch access.</td>
</tr>
<tr>
<td>apiGatewayLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group created by the pattern that API Gateway REST API access logs are sent to.</td>
</tr>
<tr>
<td>kinesisStream</td>
<td>kinesis.Stream</td>
<td>Returns an instance of the Kinesis stream created by the pattern.</td>
</tr>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns an array of recommended CloudWatch Alarms created by the construct for Kinesis Data stream.</td>
</tr>
</tbody>
</table>

### Sample API Usage

<table>
<thead>
<tr>
<th>Method</th>
<th>Request Path</th>
<th>Request Body</th>
<th>Queue Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/record</td>
<td>{&quot;data&quot;: &quot;Hello World!&quot;, &quot;partitionKey&quot;: &quot;pk001&quot;}</td>
<td>kinesis:PutRecord</td>
<td>Writes a single data record into the stream.</td>
</tr>
<tr>
<td>POST</td>
<td>/records</td>
<td>{&quot;records&quot;: [{ &quot;data&quot;: &quot;abc&quot;, }]}</td>
<td>kinesis:PutRecords</td>
<td>Writes multiple data records into the stream in a single call.</td>
</tr>
</tbody>
</table>
### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

#### Amazon API Gateway

- Deploy an edge-optimized API endpoint.
- Enable CloudWatch logging for API Gateway.
- Configure least privilege access IAM role for API Gateway.
- Set the default authorizationType for all API methods to IAM.
- Enable X-Ray tracing.
- Validate request body before passing data to Kinesis.

#### Amazon Kinesis Data Stream

- Configure least privilege access IAM role for Kinesis stream.
- Enable server-side encryption for Kinesis Stream using AWS Managed KMS Key.

<table>
<thead>
<tr>
<th>Method</th>
<th>Request Path</th>
<th>Request Body</th>
<th>Queue Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>&quot;partitionKey&quot;: &quot;pk001&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{ &quot;data&quot;: &quot;xyz&quot;, &quot;partitionKey&quot;: &quot;pk001&quot; }</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Architecture

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-apigateway-kinesisstreams
aws-apigateway-lambda

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions Constructs.aws_apigateway_lambda</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-apigateway-lambda</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.apigatewaylambda</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an Amazon API Gateway REST API connected to an AWS Lambda function.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { ApiGatewayToLambda } from '@aws-solutions-constructs/aws-apigateway-lambda';

new ApiGatewayToLambda(this, 'ApiGatewayToLambdaPattern', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
});
```

**Initializer**

```typescript
new ApiGatewayToLambda(scope: Construct, id: string, props: ApiGatewayToLambdaProps);
```

**Parameters**

- **scope Construct**
- **id string**
• props `ApiGatewayToLambdaProps (p. 49)`

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and <code>lambdaFunctionProps</code> will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an <code>existingLambdaObj</code> is provided.</td>
</tr>
<tr>
<td>apiGatewayProps?</td>
<td><code>api.LambdaRestApiProps</code></td>
<td>Optional user-provided props to override the default props for the API.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
</tbody>
</table>

## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGatewayCloudWatchRole?</td>
<td><code>iam.Role</code></td>
<td>Returns an instance of the <code>iam.Role</code> created by the construct for API Gateway for CloudWatch access.</td>
</tr>
<tr>
<td>apiGatewayLogGroup</td>
<td><code>logs.LogGroup</code></td>
<td>Returns an instance of the log group created by the pattern that API Gateway REST API access logs are sent to.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td><code>lambda.Function</code></td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>apiGateway</td>
<td><code>api.LambdaRestApi</code></td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
</tbody>
</table>

## Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:
Amazon API Gateway

- Deploy an edge-optimized API endpoint
- Enable CloudWatch logging for API Gateway
- Configure least privilege access IAM role for API Gateway
- Set the default authorizationType for all API methods to IAM
- Enable X-Ray tracing
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for NodeJs Lambda function
- Enable X-Ray tracing

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-apigateway-lambda
aws-apigateway-sagemakerendpoint

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_apigateway_sagemakerendpoint</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-apigateway-sagemakerendpoint</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.apigateway</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an Amazon API Gateway REST API connected to an Amazon SageMaker endpoint.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { ApiGatewayToSageMakerEndpoint, ApiGatewayToSageMakerEndpointProps } from '@aws-solutions-constructs/aws-apigateway-sagemakerendpoint';

// Below is an example VTL (Velocity Template Language) mapping template for mapping the Api GET request to the Sagemaker POST request
const requestTemplate = `{
  "instances": [
    #set( $user_id = $input.params("user_id") )
    #set( $items = $input.params("items") )
    #foreach( $item in $items.split(\",\") )
      {"in0": "$user_id", "in1": "$item"}#if( $foreach.hasNext ),#end
    $esc.newline
  ]
};

// Replace 'my-endpoint' with your Sagemaker Inference Endpoint
new ApiGatewayToSageMakerEndpoint(this, 'test-apigw-sagemakerendpoint', {
  endpointName: 'my-endpoint',
  resourcePath: '{user_id}',
  requestMappingTemplate: requestTemplate
});
```
new ApiGatewayToSageMakerEndpoint(scope: Construct, id: string, props: ApiGatewayToSageMakerEndpointProps);

Parameters

- **scope** `Construct`
- **id** `string`
- **props** `ApiGatewayToSageMakerEndpointProps` (p. 52)

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGatewayProps?</td>
<td><code>api.RestApiProps</code></td>
<td>Optional user-provided props to override the default props for the API Gateway REST API.</td>
</tr>
<tr>
<td>apiGatewayExecutionRole?</td>
<td><code>iam.Role</code></td>
<td>IAM Role used by API Gateway to invoke the SageMaker endpoint. If not specified, a default role is created with access to <code>endpointName</code>.</td>
</tr>
<tr>
<td>endpointName</td>
<td><code>string</code></td>
<td>Name of the deployed SageMaker inference endpoint.</td>
</tr>
<tr>
<td>resourceName?</td>
<td><code>string</code></td>
<td>Optional resource name where the GET method will be available.</td>
</tr>
<tr>
<td>resourcePath</td>
<td><code>string</code></td>
<td>Resource path for the GET method. The variable defined here can be referenced in <code>requestMappingTemplate</code>.</td>
</tr>
<tr>
<td>requestMappingTemplate</td>
<td><code>string</code></td>
<td>Mapping template to convert GET requests received on the REST API to POST requests expected by the SageMaker endpoint.</td>
</tr>
<tr>
<td>responseMappingTemplate?</td>
<td><code>string</code></td>
<td>Optional mapping template to convert responses received from the SageMaker endpoint.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGateway</td>
<td>api.LambdaRestApi</td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
<tr>
<td>apiGatewayRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the API Gateway REST API.</td>
</tr>
<tr>
<td>apiGatewayCloudWatchRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the iam.Role created by the construct for API Gateway for CloudWatch access.</td>
</tr>
<tr>
<td>apiGatewayLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group created by the pattern that API Gateway REST API access logs are sent to.</td>
</tr>
</tbody>
</table>

Sample API Usage

Note: Each SageMaker endpoint is unique, and the response from the API will depend on the deployed model. The example given below assumes the sample from this blog post. For a reference on how that'd be implemented, please refer to integ.apigateway-sagemakerendpoint-overwrite.ts.

<table>
<thead>
<tr>
<th>Method</th>
<th>Request Path</th>
<th>Query String</th>
<th>SageMaker Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/321</td>
<td>items=101,131,162</td>
<td>sagemaker:Invoke</td>
<td>Retrieves the predictions for a specific user and items.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon API Gateway

- Deploy an edge-optimized API endpoint
- Enable CloudWatch logging for API Gateway
- Configure least privilege access IAM role for API Gateway
- Set the default authorizationType for all API methods to IAM
- Enable X-Ray tracing
- Validate request parameters before passing data to SageMaker
Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-apigateway-sagemakerendpoint
aws-apigateway-sqs

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_apigateway_sqs</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-apigateway-sqs</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.apigateway_sqs</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an Amazon API Gateway REST API connected to an Amazon SQS queue.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { ApiGatewayToSqs, ApiGatewayToSqsProps } from "@aws-solutions-constructs/aws-apigateway-sqs";
new ApiGatewayToSqs(this, 'ApiGatewayToSqsPattern', {});
```

**Initializer**

```typescript
new ApiGatewayToSqs(scope: Construct, id: string, props: ApiGatewayToSqsProps);
```

**Parameters**

- `scope` [Construct](#)
- `id` string
- `props` [ApiGatewayToSqsProps](p. 56)
# Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGatewayProps?</td>
<td>api.RestApiProps</td>
<td>Optional user-provided props to override the default props for the API Gateway.</td>
</tr>
<tr>
<td>queueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user-provided props to override the default props for the queue.</td>
</tr>
<tr>
<td>deployDeadLetterQueue?</td>
<td>boolean</td>
<td>Whether to deploy a secondary queue to be used as a dead letter queue. Defaults to <code>true</code>.</td>
</tr>
<tr>
<td>maxReceiveCount</td>
<td>number</td>
<td>The number of times a message can be unsuccessfully dequeued before being moved to the dead letter queue.</td>
</tr>
<tr>
<td>allowCreateOperation?</td>
<td>boolean</td>
<td>Whether to deploy an API Gateway Method for Create operations on the queue (i.e. sqs:SendMessage).</td>
</tr>
<tr>
<td>createRequestTemplate?</td>
<td>string</td>
<td>Override the default API Gateway request template for the Create method, if allowCreateOperation is set to <code>true</code>.</td>
</tr>
<tr>
<td>allowReadOperation?</td>
<td>boolean</td>
<td>Whether to deploy an API Gateway Method for Read operations on the queue (i.e. sqs:ReceiveMessage).</td>
</tr>
<tr>
<td>readRequestTemplate?</td>
<td>string</td>
<td>Override the default API Gateway request template for the Read method, if allowReadOperation is set to <code>true</code>.</td>
</tr>
<tr>
<td>allowDeleteOperation?</td>
<td>boolean</td>
<td>Whether to deploy an API Gateway Method for Delete operations on the queue (i.e. sqs:DeleteMessage).</td>
</tr>
<tr>
<td>deleteRequestTemplate?</td>
<td>string</td>
<td>Override the default API Gateway request template for the Delete method, if allowDeleteOperation is set to <code>true</code>.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td>logs.LogGroupProps</td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGateway</td>
<td>api.RestApi</td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
<tr>
<td>apiGatewayCloudWatchRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the iam.Role created by the construct for API Gateway for CloudWatch access.</td>
</tr>
<tr>
<td>apiGatewayLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group created by the pattern that API Gateway REST API access logs are sent to.</td>
</tr>
<tr>
<td>apiGatewayRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the API Gateway REST API.</td>
</tr>
<tr>
<td>deadLetterQueue?</td>
<td>sqs.Queue</td>
<td>Returns an instance of the dead letter queue created by the pattern, if one is deployed.</td>
</tr>
<tr>
<td>sqsQueue</td>
<td>sqs.Queue</td>
<td>Returns an instance of the SQS queue created by the pattern.</td>
</tr>
</tbody>
</table>

Sample API Usage

<table>
<thead>
<tr>
<th>Method</th>
<th>Request Path</th>
<th>Request Body</th>
<th>Queue Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/</td>
<td></td>
<td>sqs::ReceiveMessage</td>
<td>Retrieves a message from the queue.</td>
</tr>
<tr>
<td>POST</td>
<td>/</td>
<td>{ &quot;data&quot;: &quot;Hello World!&quot; }</td>
<td>sqs::SendMessage</td>
<td>Delivers a message to the queue.</td>
</tr>
<tr>
<td>DELETE</td>
<td>/message?</td>
<td>receiptHandle=[value]</td>
<td>sqs::DeleteMessage</td>
<td>Deletes a specified message from the queue.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon API Gateway

- Deploy an edge-optimized API endpoint
• Enable CloudWatch logging for API Gateway
• Configure least privilege access IAM role for API Gateway
• Set the default authorizationType for all API methods to IAM
• Enable X-Ray tracing

Amazon SQS Queue

• Deploy SQS dead-letter queue for the source SQS Queue
• Enable server-side encryption for source SQS Queue using AWS managed KMS Key
• Enforce encryption of data in transit

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-apigateway-sqs
Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_cloudfront_apigateway</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-cloudfront-apigateway</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.cloudfrontapigateway</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an Amazon CloudFront distribution in front of an Amazon API Gateway REST API.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import * as api from '@aws-cdk/aws-apigateway';
import * as lambda from '@aws-cdk/aws-lambda';
import { CloudFrontToApiGateway } from '@aws-solutions-constructs/aws-cloudfront-apigateway';

const lambdaProps: lambda.FunctionProps = {
  code: lambda.Code.fromAsset(`{__dirname}/lambda`),
  runtime: lambda.Runtime.NODEJS_12_X,
  handler: 'index.handler'
};

const lambdafunction = new lambda.Function(this, 'LambdaFunction', lambdaProps);

const apiGatewayProps: api.LambdaRestApiProps = {
  handler: lambdafunction,
  endpointConfiguration: {
    types: [api.EndpointType.REGIONAL]
  },
  defaultMethodOptions: {
    authorizationType: api.AuthorizationType.NONE
  }
};

const apiGateway = new api.LambdaRestApi(this, 'LambdaRestApi', apiGatewayProps);

new CloudFrontToApiGateway(this, 'test-cloudfront-apigateway', {
```
Initializer

new CloudFrontToApiGateway(scope: Construct, id: string, props: CloudFrontToApiGatewayProps);

Parameters
- scope Construct
- id string
- props CloudFrontToApiGatewayProps (p. 60)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingApiGatewayObj</td>
<td>api.RestApi</td>
<td>The regional API Gateway that will be fronted with the CloudFront</td>
</tr>
<tr>
<td>cloudFrontDistributionProps?</td>
<td>cloudfront.DistributionProps</td>
<td>Optional user provided props to override the default props for the CloudFront Distribution.</td>
</tr>
<tr>
<td>insertHttpSecurityHeaders?</td>
<td>boolean</td>
<td>Optional user provided props to turn on/off the automatic injection of best practice HTTP security headers in all responses from CloudFront</td>
</tr>
<tr>
<td>cloudFrontLoggingBucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user provided props to override the default props for the CloudFront Logging Bucket.</td>
</tr>
</tbody>
</table>

Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGateway</td>
<td>api.RestApi</td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
<tr>
<td>cloudFrontLoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the CloudFront web distribution.</td>
</tr>
</tbody>
</table>
AWS Solutions Constructs AWS Solutions

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon CloudFront**

- Configure Access logging for CloudFront WebDistribution
- Enable automatic injection of best practice HTTP security headers in all responses from CloudFront WebDistribution

**Amazon API Gateway**

- User provided API Gateway object is used as-is
- Enable X-Ray tracing

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudFrontWebDistribution</td>
<td>cloudfront.CloudFrontWebDistribution</td>
<td>Returns an instance of the CloudFront web distribution created by the pattern.</td>
</tr>
<tr>
<td>cloudFrontFunction?</td>
<td>cloudfront.Function</td>
<td>Returns an instance of the CloudFront function created by the pattern.</td>
</tr>
</tbody>
</table>
Architecture

Amazon CloudFront → Amazon API Gateway

Amazon Simple Storage Service → Role → Amazon CloudWatch

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-cloudfront-apigateway

aws-cloudfront-apigateway-lambda

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an Amazon CloudFront distribution in front of an Amazon API Gateway Lambda-backed REST API.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { CloudFrontToApiGatewayToLambda } from '@aws-solutions-constructs/aws-cloudfront-apigateway-lambda';

new CloudFrontToApiGatewayToLambda(this, 'test-cloudfront-apigateway-lambda', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
});
```

Initializer

```typescript
new CloudFrontToApiGatewayToLambda(scope: Construct, id: string, props: CloudFrontToApiGatewayToLambdaProps);
```

Parameters

- `scope` Construct
- `id` string
- `props` CloudFrontToApiGatewayToLambdaProps (p. 63)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object,</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user provided props to override the default props for the Lambda function.</td>
</tr>
<tr>
<td>apiGatewayProps?</td>
<td><code>api.LambdaRestApiProps</code></td>
<td>Optional user provided props to override the default props for API Gateway</td>
</tr>
<tr>
<td>cloudFrontDistributionProps?</td>
<td><code>cloudfront.DistributionProps</code></td>
<td>Optional user provided props to override the default props for the CloudFront Distribution.</td>
</tr>
<tr>
<td>insertHttpSecurityHeaders?</td>
<td><code>boolean</code></td>
<td>Optional user provided props to turn on/off the automatic injection of best practice HTTP security headers in all responses from CloudFront</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user provided props to override the default props for the CloudWatchLogs LogGroup.</td>
</tr>
<tr>
<td>cloudFrontLoggingBucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the CloudFront Logging Bucket.</td>
</tr>
</tbody>
</table>

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiGateway</td>
<td><code>api.RestApi</code></td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
<tr>
<td>apiGatewayCloudWatchRole?</td>
<td><code>iam.Role</code></td>
<td>Returns an instance of the iam.Role created by the construct for API Gateway for CloudWatch access.</td>
</tr>
<tr>
<td>apiGatewayLogGroup</td>
<td><code>logs.LogGroup</code></td>
<td>Returns an instance of the log group created by the pattern that API Gateway REST API access logs are sent to.</td>
</tr>
<tr>
<td>cloudFrontLoggingBucket?</td>
<td><code>s3.Bucket</code></td>
<td>Returns an instance of the logging bucket created by the pattern for the CloudFront web distribution.</td>
</tr>
</tbody>
</table>
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon CloudFront**
- Configure Access logging for CloudFront WebDistribution
- Enable automatic injection of best practice HTTP security headers in all responses from CloudFront WebDistribution

**Amazon API Gateway**
- Deploy a regional API endpoint
- Enable CloudWatch logging for API Gateway
- Configure least privilege access IAM role for API Gateway
- Set the default authorizationType for all API methods to IAM
- Enable X-Ray tracing

**AWS Lambda Function**
- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for NodeJs Lambda function
- Enable X-Ray tracing
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)
AWS Solutions Constructs AWS Solutions Architecture

**Architecture**

Amazon CloudFront → Amazon API Gateway → AWS Lambda

Amazon Simple Storage Service

Role

Amazon CloudWatch

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-cloudfront-apigateway-lambda

**aws-cloudfront-mediastore**

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_cloudfront_mediastore</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-cloudfront-mediastore</td>
</tr>
<tr>
<td></td>
<td>software.amazon.awsconstructs.services.cloudfront</td>
</tr>
</tbody>
</table>

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an Amazon CloudFront distribution connected to an AWS Elemental MediaStore container.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { CloudFrontToMediaStore } from '@aws-solutions-constructs/aws-cloudfront-mediastore';
new CloudFrontToMediaStore(this, 'test-cloudfront-mediastore-default', {});
```

Initializer

```typescript
new CloudFrontToMediaStore(scope: Construct, id: string, props: CloudFrontToMediaStoreProps);
```

Parameters

- `scope` **Construct**
- `id` **string**
- `props` **CloudFrontToMediaStoreProps** (p. 67)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingMediaStoreContainerObj?</td>
<td>mediastore.CfnContainer</td>
<td>Optional user-provided MediaStore container to override the default MediaStore container.</td>
</tr>
<tr>
<td>mediaStoreContainerProps?</td>
<td>mediastore.CfnContainerProps</td>
<td>Optional user-provided props to override the default props for the MediaStore Container.</td>
</tr>
<tr>
<td>cloudFrontDistributionProps?</td>
<td>cloudfront.DistributionProps</td>
<td>any</td>
</tr>
<tr>
<td>insertHttpSecurityHeaders?</td>
<td>boolean</td>
<td>Optional user-provided props to turn on/off the automatic injection of best practice HTTP security headers in all responses from CloudFront.</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudFrontWebDistribution</td>
<td>cloudfront.CloudFrontWebDistribution</td>
<td>Returns an instance of the CloudFront web distribution created by the pattern.</td>
</tr>
<tr>
<td>mediaStoreContainer</td>
<td>mediastore.CfnContainer</td>
<td>Returns an instance of the MediaStore container created by the pattern.</td>
</tr>
<tr>
<td>cloudFrontLoggingBucket</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the CloudFront web distribution.</td>
</tr>
<tr>
<td>cloudFrontOriginRequestPolicy</td>
<td>cloudfront.OriginRequestPolicy</td>
<td>Returns an instance of the CloudFront origin request policy created by the pattern for the CloudFront web distribution.</td>
</tr>
<tr>
<td>cloudFrontOriginAccessIdentity?</td>
<td>cloudfront.OriginAccessIdentity</td>
<td>Returns an instance of the CloudFront origin access identity created by the pattern for the CloudFront web distribution.</td>
</tr>
<tr>
<td>cloudFrontFunction?</td>
<td>cloudfront.Function</td>
<td>Returns an instance of the CloudFront function created by the pattern.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon CloudFront**

- Configure access logging for CloudFront web distribution
- Enable CloudFront origin request policy for AWS Elemental MediaStore container
- Set `User-Agent` custom header with CloudFront origin access identity
- Enable automatic injection of best practice HTTP security headers in all responses from CloudFront web distribution

**AWS Elemental MediaStore**

- Set the deletion policy to retain the resource
- Set the container name with the CloudFormation stack name
• Set the default container Cross-origin resource sharing (CORS) policy
• Set the default object lifecycle policy
• Set the default container policy to allow only `aws:UserAgent` with CloudFront origin access identity
• Set the default metric policy
• Enable access logging

**Architecture**

Amazon CloudFront  
AWS Elemental MediaStore

Amazon Simple Storage Service

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:
[@aws-solutions-constructs/aws-cloudfront-mediastore](https://github.com/aws-solutions-constructs/aws-cloudfront-mediastore)

**aws-cloudfront-s3**

*Note:* To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an Amazon CloudFront distribution in front of an Amazon S3 bucket.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { CloudFrontToS3 } from '@aws-solutions-constructs/aws-cloudfront-s3';
new CloudFrontToS3(this, 'test-cloudfront-s3', {});
```

Initializer

```typescript
new CloudFrontToS3(scope: Construct, id: string, props: CloudFrontToS3Props);
```

Parameters

- scope `Construct`
- id `string`
- props `CloudFrontToS3Props` (p. 70)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingBucketObj?</td>
<td>s3.IBucket</td>
<td>Existing instance of S3 Bucket object or interface. If this is provided, then also providing bucketProps will cause an error.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user provided props to override the default props for the S3 Bucket.</td>
</tr>
<tr>
<td>cloudFrontDistributionProps?</td>
<td>cloudfront.DistributionProps</td>
<td>Optional user provided props to override the default props for the CloudFront Distribution.</td>
</tr>
<tr>
<td>insertHttpSecurityHeaders?</td>
<td>boolean</td>
<td>Optional user provided props to turn on/off the automatic injection of best practice HTTP security headers in all responses from CloudFront</td>
</tr>
</tbody>
</table>
| originPath?                 | string                              | Optional user provided props to provide an originPath that CloudFront appends to the origin domain name when CloudFront requests content from the origin. The string should start with a /, for example: /production. Default value is /.
| loggingBucketProps?         | s3.BucketProps                      | Optional user provided props to override the default props for the S3 Logging Bucket.                                                        |
| cloudFrontLoggingBucketProps?| s3.BucketProps                      | Optional user provided props to override the default props for the CloudFront Logging Bucket.                                              |
| logS3AccessLogs?            | boolean                             | Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is true. |

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudFrontWebDistribution</td>
<td>cloudfront.CloudFrontWebDistribution</td>
<td>Returns an instance of the CloudFront web distribution created by the pattern.</td>
</tr>
<tr>
<td>s3BucketInterface</td>
<td>s3.IBucket</td>
<td>Returns an instance of s3.IBucket created by the construct.</td>
</tr>
<tr>
<td>s3Bucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of s3.Bucket created by the construct. <strong>Important:</strong></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
<tr>
<td>cloudFrontFunction?</td>
<td>cloudfront.Function</td>
<td>Returns an instance of the CloudFront function created by the pattern.</td>
</tr>
<tr>
<td>cloudFrontLoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the CloudFront web distribution.</td>
</tr>
</tbody>
</table>

**Default settings**

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon CloudFront**

- Configure Access logging for CloudFront WebDistribution
- Enable automatic injection of best practice HTTP security headers in all responses from CloudFront WebDistribution.
- CloudFront originPath set to /.

**Amazon S3 Bucket**

- Configure Access logging for S3 Bucket
- Enable server-side encryption for S3 Bucket using AWS managed KMS Key
- Turn on the versioning for S3 Bucket
- Don't allow public access for S3 Bucket
- Retain the S3 Bucket when deleting the CloudFormation stack
- Enforce encryption of data in transit
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days
Architecture

Amazon CloudFront

Amazon Simple Storage Service

Amazon Simple Storage Service (Access Logs)

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-cloudfront-s3

aws-cognito-apigateway-lambda

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_construction.aws_cognito_apigateway_lambda</td>
</tr>
<tr>
<td>TS</td>
<td>@aws-solutions-constructs/aws-cognito-apigateway-lambda</td>
</tr>
</tbody>
</table>
Overview

This AWS Solutions Construct implements Amazon Cognito securing an Amazon API Gateway Lambda-backed REST API.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { CognitoToApiGatewayToLambda } from '@aws-solutions-constructs/aws-cognito-apigateway-lambda';

new CognitoToApiGatewayToLambda(this, 'test-cognito-apigateway-lambda', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler'
  }
});
```

If you are defining resources and methods on your API (e.g. `proxy = false`), you must call the `addAuthorizers()` method after the API is fully defined. This ensures that every method in your API is protected.

Here is an example in TypeScript:

```typescript
import { CognitoToApiGatewayToLambda } from '@aws-solutions-constructs/aws-cognito-apigateway-lambda';

const construct = new CognitoToApiGatewayToLambda(this, 'test-cognito-apigateway-lambda', {
  lambdaFunctionProps: {
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler'
  },
  apiGatewayProps: {
    proxy: false
  }
});

const resource = construct.apiGateway.root.addResource('foobar');
resource.addMethod('POST');

// Mandatory to call this method to Apply the Cognito Authorizers on all API methods
construct.addAuthorizers();
```
Initializer

```typescript
new CognitoToApiGatewayToLambda(scope: Construct, id: string, props: CognitoToApiGatewayToLambdaProps);
```

**Parameters**

- `scope` *Construct*
- `id` *string*
- `props` *CognitoToApiGatewayToLambdaProps (p. 75)*

### Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>existingLambdaObj?</code></td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td><code>lambdaFunctionProps?</code></td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td><code>apiGatewayProps?</code></td>
<td><code>api.LambdaRestApiProps</code></td>
<td>Optional user provided props to override the default props for API Gateway</td>
</tr>
<tr>
<td><code>cognitoUserPoolProps?</code></td>
<td><code>cognito.UserPoolProps</code></td>
<td>Optional user provided props to override the default props for Cognito User Pool</td>
</tr>
<tr>
<td><code>cognitoUserPoolClientProps?</code></td>
<td><code>cognito.UserPoolClientProps</code></td>
<td>Optional user provided props to override the default props for Cognito User Pool Client</td>
</tr>
<tr>
<td><code>logGroupProps?</code></td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
</tbody>
</table>

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>apiGateway</code></td>
<td><code>api.RestApi</code></td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
</tbody>
</table>
## Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

### Amazon Cognito
- Set password policy for User Pools
- Enforce the advanced security mode for User Pools

### Amazon API Gateway
- Deploy an edge-optimized API endpoint
- Enable CloudWatch logging for API Gateway
- Configure least privilege access IAM role for API Gateway
- Set the default authorizationType for all API methods to IAM
- Enable X-Ray tracing

### AWS Lambda Function
- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for Node.js Lambda function
- Enable X-Ray tracing
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>userPool</td>
<td>cognito.UserPool</td>
<td>Returns an instance of the Cognito user pool created by the pattern.</td>
</tr>
<tr>
<td>userPoolClient</td>
<td>cognito.UserPoolClient</td>
<td>Returns an instance of the Cognito user pool client created by the pattern.</td>
</tr>
<tr>
<td>apiGatewayCloudWatchRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the iam.Role created by the pattern for API Gateway for CloudWatch access.</td>
</tr>
<tr>
<td>apiGatewayLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group created by the pattern for API Gateway REST API access logs are sent to.</td>
</tr>
<tr>
<td>apiGatewayAuthorizer</td>
<td>api.CfnAuthorizer</td>
<td>Returns an instance of the API Gateway authorizer created by the pattern.</td>
</tr>
</tbody>
</table>
Architecture

Amazon Cognito

Amazon API Gateway

AWS Lambda

Role

Role

Amazon CloudWatch

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-cognito-apigateway-lambda

aws-dynamodb-stream-lambda

STABILITY

DEPRECATED
Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.

**Note:** This construct has been deprecated and is superseded by the `aws-dynamodbstreams-lambda` construct.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_dynamodb_stream_lambda</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-dynamodb-stream-lambda</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.dynamodbstreamlambda</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements a pattern Amazon DynamoDB table with stream to invoke the AWS Lambda function with the least privileged permissions.

Here is a minimal deployable pattern definition:

```javascript
import { DynamoDBStreamToLambdaProps, DynamoDBStreamToLambda } from '@aws-solutions-constructs/aws-dynamodb-stream-lambda';

new DynamoDBStreamToLambda(this, 'test-dynamodb-stream-lambda', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`#{$dirname}/lambda`),
    handler: 'index.handler'
  }
});
```

**Initializer**

```javascript
new DynamoDBStreamToLambda(scope: Construct, id: string, props: DynamoDBStreamToLambdaProps);
```

**Parameters**
Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>dynamoTableProps?</td>
<td><code>dynamodb.TableProps</code></td>
<td>Optional user provided props to override the default props for DynamoDB Table</td>
</tr>
<tr>
<td>existingTableInterface?</td>
<td><code>dynamodb.ITable</code></td>
<td>Existing instance of DynamoDB table object or interface, providing both this and dynamoTableProps will cause an error.</td>
</tr>
<tr>
<td>dynamoEventSourceProps?</td>
<td><code>aws-lambda-event-sources.DynamoEventSourceProps</code></td>
<td>Optional user provided props to override the default props for DynamoDB Event Source</td>
</tr>
</tbody>
</table>

Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamoTableInterface</td>
<td><code>dynamodb.ITable</code></td>
<td>Returns an instance of dynamodb.ITable created by the construct.</td>
</tr>
<tr>
<td>dynamoTable?</td>
<td><code>dynamodb.Table</code></td>
<td>Returns an instance of dynamodb.Table created by the construct. IMPORTANT: If existingTableInterface was provided in Pattern Construct Props, this property will be undefined.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td><code>lambda.Function</code></td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
</tbody>
</table>
Lambda function

This pattern requires a Lambda function that can post data into the Elasticsearch service from the DynamoDB stream. A sample function is provided here.

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon DynamoDB Table

- Set the billing mode for DynamoDB Table to On-Demand (Pay per request)
- Enable server-side encryption for DynamoDB Table using AWS managed KMS Key
- Creates a partition key called 'id' for DynamoDB Table
- Retain the Table when deleting the CloudFormation stack
- Enable continuous backups and point-in-time recovery

AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for NodeJs Lambda function
- Enable X-Ray tracing
- Enable Failure-Handling features: enable bisect on function Error; set default Maximum Record Age (24 hours); set default Maximum Retry Attempts (500); and deploy SQS dead-letter queue as destination on failure
- Set environment variables:
  - AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)

80
Architecture

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-dynamodb-stream-lambda

aws-dynamodbstreams-lambda

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an Amazon DynamoDB table with stream to invoke an AWS Lambda function with the least privileged permissions.

Here is a minimal deployable pattern definition:

```javascript
import { DynamoDBStreamsToLambdaProps, DynamoDBStreamsToLambda } from '@aws-solutions-constructs/aws-dynamodbstreams-lambda';

new DynamoDBStreamsToLambda(this, 'test-dynamodbstreams-lambda', {
  lambdaFunctionProps: {
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler'
  },
});
```

Initializer

```javascript
new DynamoDBStreamsToLambda(scope: Construct, id: string, props: DynamoDBStreamsToLambdaProps);
```

Parameters

- `scope` *Construct*
- `id` *string*
- `props` *DynamoDBStreamsToLambdaProps* (p. 82)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and</td>
</tr>
</tbody>
</table>
### Lambda function

This pattern requires a Lambda function that can post data into the Elasticsearch service from the DynamoDB stream. A sample function is provided [here](#).

### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

---

**Pattern Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamoTableInterface</td>
<td>dynamodb.ITable</td>
<td>Returns an instance of dynamodb.ITable created by the construct.</td>
</tr>
<tr>
<td>dynamoTable?</td>
<td>dynamodb.Table</td>
<td>Returns an instance of dynamodb.Table created by the construct. IMPORTANT: If existingTableInterface was provided in Pattern Construct Props, this property will be undefined.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
</tbody>
</table>

---

**Pattern Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>dynamoTableProps?</td>
<td>dynamodb.TableProps</td>
<td>Optional user provided props to override the default props for DynamoDB Table</td>
</tr>
<tr>
<td>existingTableInterface?</td>
<td>dynamodb.ITable</td>
<td>Existing instance of DynamoDB table object or interface, providing both this and dynamoTableProps will cause an error.</td>
</tr>
<tr>
<td>dynamoEventSourceProps?</td>
<td>aws-lambda-event-sources.DynamoEventSourceProps</td>
<td>Optional user provided props to override the default props for DynamoDB Event Source</td>
</tr>
</tbody>
</table>

---

**Lambda function**

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:
Amazon DynamoDB Table
- Set the billing mode for DynamoDB Table to On-Demand (Pay per request)
- Enable server-side encryption for DynamoDB Table using AWS managed KMS Key
- Creates a partition key called ‘id’ for DynamoDB Table
- Retain the Table when deleting the CloudFormation stack
- Enable continuous backups and point-in-time recovery

AWS Lambda Function
- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for NodeJs Lambda function
- Enable X-Ray tracing
- Enable Failure-Handling features: enable bisect on function Error; set default Maximum Record Age (24 hours); set default Maximum Retry Attempts (500); and deploy SQS dead-letter queue as destination on failure
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

Architecture
Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.

**Note:** This construct has been deprecated and is superseded by the `aws-dynamodbstreams-lambda-elasticsearch-kibana` construct.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
<th>Python</th>
<th>Typescript</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>aws_solutions_constructs.aws_dynamodb_stream_lambda</td>
<td>@aws-solutions-constructs/aws-dynamodb-stream-lambda-elasticsearch-kibana</td>
<td>software.amazon.awsconstructs.services.dynamodbstreamlambdaelasticsearchkibana</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements Amazon DynamoDB table with stream, an AWS Lambda function, and an Amazon Elasticsearch Service with the least privileged permissions.

Here is a minimal deployable pattern definition in TypeScript:
import { DynamoDBStreamToLambdaToElasticSearchAndKibana, DynamoDBStreamToLambdaToElasticSearchAndKibanaProps } from '@aws-solutions-constructs/aws-dynamodb-stream-lambda-elasticsearch-kibana';
import { Aws } from '@aws-cdk/core';

const props: DynamoDBStreamToLambdaToElasticSearchAndKibanaProps = {
  lambdaFunctionProps: {
    code: lambda.Code.fromAsset(`@{__dirname}/lambda`),
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler'
  },
  domainName: 'test-domain',
  // TODO: Ensure the Cognito domain name is globally unique
  cognitoDomainName: 'globallyuniquedomain' + Aws.ACCOUNT_ID;
}

new DynamoDBStreamToLambdaToElasticSearchAndKibana(this, 'test-dynamodb-stream-lambda-elasticsearch-kibana', props);

Initializer

new DynamoDBStreamToLambdaToElasticSearchAndKibana(scope: Construct, id: string, props: DynamoDBStreamToLambdaToElasticSearchAndKibanaProps);

Parameters

- scope Construct
- id string
- props DynamoDBStreamToLambdaToElasticSearchAndKibanaProps (p. 86)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>dynamoTableProps?</td>
<td>dynamodb.TableProps</td>
<td>Optional user provided props to override the default props for DynamoDB Table</td>
</tr>
<tr>
<td>existingTableInterface?</td>
<td>dynamodb.ITable</td>
<td>Existing instance of DynamoDB table object or interface,</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns a list of one or more CloudWatch alarms created by the pattern.</td>
</tr>
<tr>
<td>dynamoTableInterface</td>
<td>dynamodb.ITable</td>
<td>Returns an instance of dynamodb.ITable created by the construct.</td>
</tr>
<tr>
<td>dynamoTable?</td>
<td>dynamodb.Table</td>
<td>Returns an instance of dynamodb.Table created by the construct. IMPORTANT: If existingTableInterface was provided in Pattern Construct Props, this property will be undefined.</td>
</tr>
<tr>
<td>elasticsearchDomain</td>
<td>elasticsearch.CfnDomain</td>
<td>Returns an instance of the Elasticsearch domain created by the pattern.</td>
</tr>
<tr>
<td>identityPool</td>
<td>cognito.CfnIdentityPool</td>
<td>Returns an instance of the Cognito identity pool created by the pattern.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>userPool</td>
<td>cognito.UserPool</td>
<td>Returns an instance of the Cognito user pool created by the pattern.</td>
</tr>
</tbody>
</table>
Lambda function

This pattern requires a Lambda function that can post data into the Elasticsearch service from the DynamoDB stream. A sample function is provided here.

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon DynamoDB Table

- Set the billing mode for DynamoDB Table to On-Demand (Pay per request)
- Enable server-side encryption for DynamoDB Table using AWS managed KMS Key
- Creates a partition key called ‘id’ for DynamoDB Table
- Retain the Table when deleting the CloudFormation stack
- Enable continuous backups and point-in-time recovery

AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for NodeJs Lambda function
- Enable X-Ray tracing
- Enable Failure-Handling features: enable bisect on function Error; set default Maximum Record Age (24 hours); set default Maximum Retry Attempts (500); and deploy SQS dead-letter queue as destination on failure
- Set environment variables:
  - AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)

Amazon Cognito

- Set password policy for User Pools
- Enforce the advanced security mode for User Pools

Amazon OpenSearch Service

- Deploy best practices CloudWatch Alarms for the Elasticsearch Domain
- Secure the Kibana dashboard access with Cognito User Pools
- Enable server-side encryption for Elasticsearch Domain using AWS managed KMS Key
- Enable node-to-node encryption for Elasticsearch Domain
- Configure the cluster for the OpenSearch Service domain
To view the code for this pattern, create/view issues and pull requests, and more:

`@aws-solutions-constructs/aws-dynamodb-stream-lambda-elasticsearch-kibana`

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_dynamodbstreams_lambda_elasticsearch_kibana</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-dynamodbstreams-lambda-elasticsearch-kibana</td>
</tr>
<tr>
<td>Java</td>
<td>See GitHub for Java package reference</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an Amazon DynamoDB table with stream, an AWS Lambda function and Amazon Elasticsearch Service with least privileged permissions.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { DynamoDBStreamsToLambdaToElasticSearchAndKibana, DynamoDBStreamsToLambdaToElasticSearchAndKibanaProps } from '@aws-solutions-constructs/aws-dynamodbstreams-lambda-elasticsearch-kibana';
import { Aws } from '@aws-cdk/core';

const props: DynamoDBStreamsToLambdaToElasticSearchAndKibanaProps = {
    lambdaFunctionProps: {
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'index.handler'
    },
    domainName: 'test-domain',
    // TODO: Ensure the Cognito domain name is globally unique
cognitoDomainName: 'globallyuniquedomain' + Aws.ACCOUNT_ID;
};

new DynamoDBStreamsToLambdaToElasticSearchAndKibana(this, 'test-dynamodbstreams-lambda-elasticsearch-kibana', props);
```

**Initializer**

```typescript
new DynamoDBStreamsToLambdaToElasticSearchAndKibana(scope: Construct, id: string, props: DynamoDBStreamsToLambdaToElasticSearchAndKibanaProps);
```

**Parameters**

- `scope` *Construct*
- `id` string
- `props` *DynamoDBStreamsToLambdaToElasticSearchAndKibanaProps* (p. 91)
# Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>dynamoTableProps?</td>
<td><code>dynamodb.TableProps</code></td>
<td>Optional user provided props to override the default props for DynamoDB Table</td>
</tr>
<tr>
<td>existingTableInterface?</td>
<td><code>dynamodb.ITable</code></td>
<td>Existing instance of DynamoDB table object or interface, providing both this and dynamoTableProps will cause an error.</td>
</tr>
<tr>
<td>dynamoEventSourceProps?</td>
<td><code>aws-lambda-event-sources.DynamoEventSourceProps</code></td>
<td>Optional user provided props to override the default props for DynamoDB Event Source</td>
</tr>
<tr>
<td>esDomainProps?</td>
<td><code>elasticsearch.CfnDomainProps</code></td>
<td>Optional user provided props to override the default props for the Amazon OpenSearch Service</td>
</tr>
<tr>
<td>domainName</td>
<td><code>string</code></td>
<td>Domain name for the Cognito and the Amazon OpenSearch Service</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td><code>boolean</code></td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
</tbody>
</table>

## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudwatchAlarms?</td>
<td><code>cloudwatch.Alarm[]</code></td>
<td>Returns a list of one or more CloudWatch alarms created by the pattern.</td>
</tr>
<tr>
<td>dynamoTableInterface</td>
<td><code>dynamodb.ITable</code></td>
<td>Returns an instance of dynamodb.ITable created by the construct.</td>
</tr>
</tbody>
</table>
AWS Solutions Constructs AWS Solutions
Lambda function

### Name | Type | Description
--- | --- | ---
| dynamoTable? | dynamodb.Table | Returns an instance of dynamodb.Table created by the construct. IMPORTANT: If existingTableInterface was provided in Pattern Construct Props, this property will be undefined.
| elasticsearchDomain | elasticsearch.CfnDomain | Returns an instance of the Elasticsearch domain created by the pattern.
| identityPool | cognito.CfnIdentityPool | Returns an instance of the Cognito identity pool created by the pattern.
| lambdaFunction | lambda.Function | Returns an instance of the Lambda function created by the pattern.
| userPool | cognito.UserPool | Returns an instance of the Cognito user pool created by the pattern.
| userPoolClient | cognito.UserPoolClient | Returns an instance of the Cognito user pool client created by the pattern.

### Lambda function

This pattern requires a Lambda function that can post data into the Elasticsearch service from the DynamoDB stream. A sample function is provided here.

### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon DynamoDB Table**

- Set the billing mode for DynamoDB Table to On-Demand (Pay per request)
- Enable server-side encryption for DynamoDB Table using AWS managed KMS Key
- Creates a partition key called 'id' for DynamoDB Table
- Retain the Table when deleting the CloudFormation stack
- Enable continuous backups and point-in-time recovery

**AWS Lambda Function**

- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for NodeJs Lambda function
- Enable X-Ray tracing
- Enable Failure-Handling features: enable bisect on function Error; set default Maximum Record Age (24 hours); set default Maximum Retry Attempts (500); and deploy SQS dead-letter queue as destination on failure
• Set environment variables:
  • `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**Amazon Cognito**

• Set password policy for User Pools
• Enforce the advanced security mode for User Pools

**Amazon OpenSearch Service**

• Deploy best practices CloudWatch Alarms for the Elasticsearch Domain
• Secure the Kibana dashboard access with Cognito User Pools
• Enable server-side encryption for Elasticsearch Domain using AWS managed KMS Key
• Enable node-to-node encryption for Elasticsearch Domain
• Configure the cluster for the OpenSearch Service domain

**Architecture**

```
AWS Lambda
  ▼
  Amazon CloudWatch
  ▲
  Amazon Elasticsearch Service
  ▼
  Amazon Cognito
```

```
Role
  ▲
  Amazon DynamoDB
```

---

93
GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-dynamodbstreams-lambda-elasticsearch-kibana

aws-events-rule-kinesisfirehose-s3

Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.

Note: This construct has been deprecated and is superseded by the aws-eventbridge-kinesisfirehose-s3 construct.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>@aws-solutions-constructs/aws-events-rule-kinesisfirehose-s3</td>
</tr>
<tr>
<td>Typescript</td>
<td>software.amazon.awsconstructs.services.eventsrulekinesisfirehose.s3</td>
</tr>
<tr>
<td>Java</td>
<td></td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements an Amazon CloudWatch Events rule to send data to an Amazon Kinesis Data Firehose delivery stream connected to an Amazon S3 bucket.

Here is a minimal deployable pattern definition in TypeScript:

```javascript
import * as cdk from '@aws-cdk/core';
import { EventsRuleToKinesisFirehoseToS3, EventsRuleToKinesisFirehoseToS3Props } from '@aws-solutions-constructs/aws-events-rule-kinesisfirehose-s3';
```
const eventsRuleToKinesisFirehoseToS3Props: EventsRuleToKinesisFirehoseToS3Props = {
    eventRuleProps: {
        schedule: events.Schedule.rate(cdk.Duration.minutes(5))
    }
};

new EventsRuleToKinesisFirehoseToS3(this, 'test-events-rule-firehose-s3',
    eventsRuleToKinesisFirehoseToS3Props);

new EventsRuleToKinesisFirehoseToS3(scope: Construct, id: string, props:
    EventsRuleToKinesisFirehoseToS3Props);

**Parameters**

- **scope** `Construct`
- **id** `string`
- **props** `EventsRuleToKinesisFirehoseToS3Props (p. 95)`

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingEventBusInterface?</td>
<td><code>events.IEventBus</code></td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and <code>eventBusProps</code> results in an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td><code>events.EventBusProps</code></td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to <code>{}</code> will create a custom EventBus using all default properties. If neither this nor <code>existingEventBusInterface</code> is provided, the construct will use the default EventBus. Providing both this and <code>existingEventBusInterface</code> results in an error.</td>
</tr>
<tr>
<td>eventRuleProps</td>
<td><code>events.RuleProps</code></td>
<td>User-provided properties to override the default properties for the CloudWatch Events rule.</td>
</tr>
<tr>
<td>kinesisFirehoseProps?</td>
<td><code>aws-kinesisfirehose.CfnDeliveryStreamProps</code></td>
<td>Optional user-provided props to override the default props for Kinesis Firehose Delivery Stream.</td>
</tr>
</tbody>
</table>
# AWS Solutions Constructs AWS Solutions

## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingBucketObj?</td>
<td>s3.IBucket</td>
<td>Existing instance of S3 Bucket object. If this is provided, then also providing bucketProps is an error.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user-provided props to override the default props for the S3 bucket.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td>logs.LogGroupProps</td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
<tr>
<td>loggingBucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td>logS3AccessLogs?</td>
<td>boolean</td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is true.</td>
</tr>
</tbody>
</table>

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the Event bus created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>kinesisFirehose</td>
<td>kinesisfirehose.CfnDeliveryStream</td>
<td>Returns an instance of the Kinesis Firehose delivery stream created by the pattern.</td>
</tr>
<tr>
<td>s3Bucket</td>
<td>s3.Bucket</td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
<tr>
<td>eventsRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the role created by the construct for the CloudWatch Events rule.</td>
</tr>
<tr>
<td>kinesisFirehoseRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the Kinesis Firehose delivery stream.</td>
</tr>
</tbody>
</table>
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon CloudWatch Events rule**
- Configure least privilege access IAM role for Events Rule to publish to the Kinesis Firehose Delivery Stream.

**Amazon Kinesis Firehose**
- Enable CloudWatch logging for Kinesis Firehose.
- Configure least privilege access IAM role for Amazon Kinesis Firehose.

**Amazon S3 bucket**
- Configure access logging for bucket.
- Enable server-side encryption for bucket using AWS managed KMS Key.
- Turn on the versioning for the bucket.
- Don’t allow public access for the bucket.
- Retain the bucket when deleting the CloudFormation stack.
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days.
**Architecture**

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-events-rule-kinesisfirehose-s3

**aws-eventbridge-kinesisfirehose-s3**

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>@aws-solutions-constructs/aws-eventbridge-kinesisfirehose-s3</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-eventbridge-kinesisfirehose-s3</td>
</tr>
</tbody>
</table>

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an Amazon EventBridge Rule to send data to an Amazon Kinesis Data Firehose delivery stream connected to an Amazon S3 bucket.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import * as cdk from '@aws-cdk/core';
import { EventbridgeToKinesisFirehoseToS3, EventbridgeToKinesisFirehoseToS3Props } from
'@aws-solutions-constructs/aws-eventbridge-kinesisfirehose-s3';

const EventbridgeToKinesisFirehoseToS3Props: EventbridgeToKinesisFirehoseToS3Props = {
  eventRuleProps: {
    schedule: events.Schedule.rate(cdk.Duration.minutes(5))
  }
};

new EventbridgeToKinesisFirehoseToS3(this, 'test-eventbridge-firehose-s3',
  EventbridgeToKinesisFirehoseToS3Props);
```

Initializer

```typescript
new EventbridgeToKinesisFirehoseToS3(scope: Construct, id: string, props:
  EventbridgeToKinesisFirehoseToS3Props);
```

Parameters

- **scope** Construct
- **id** string
- **props** `EventbridgeToKinesisFirehoseToS3Props` (p. 99)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingEventBusInterface?</td>
<td><code>events.IEventBus</code></td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and eventBusProps results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td><code>events.EventBusProps</code></td>
<td>Optional user-provided properties to override the</td>
</tr>
</tbody>
</table>
### AWS Solutions Constructs AWS Solutions

**Pattern Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default properties</td>
<td></td>
<td>default properties when creating a custom EventBus. Setting this value to {} will create a custom EventBus using all default properties. If neither this nor existingEventBusInterface is provided the construct will use the default EventBus. Providing both this and existingEventBusInterface results in an error.</td>
</tr>
<tr>
<td>eventRuleProps</td>
<td>events.RuleProps</td>
<td>User-provided properties to override the default properties for the CloudWatch Events rule.</td>
</tr>
<tr>
<td>kinesisFirehoseProps?</td>
<td>aws-kinesisfirehose.CfnDeliveryStreamProps</td>
<td>Optional user provided props to override the default props for Kinesis Firehose Delivery Stream.</td>
</tr>
<tr>
<td>existingBucketObj?</td>
<td>s3.IBucket</td>
<td>Existing instance of S3 Bucket object. If this is provided, then also providing bucketProps is an error.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user-provided props to override the default props for the S3 bucket.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td>logs.LogGroupProps</td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
<tr>
<td>loggingBucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td>logS3AccessLogs?</td>
<td>boolean</td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is true.</td>
</tr>
</tbody>
</table>

**Pattern Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the Event bus created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>kinesisFirehose</td>
<td>kinesisfirehose.CfnDeliveryStream</td>
<td>Returns an instance of the Kinesis Firehose delivery stream created by the pattern.</td>
</tr>
<tr>
<td>s3Bucket</td>
<td>s3.Bucket</td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
<tr>
<td>eventsRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the role created by the construct for the CloudWatch Events rule.</td>
</tr>
<tr>
<td>kinesisFirehoseRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the Kinesis Firehose delivery stream.</td>
</tr>
<tr>
<td>kinesisFirehoseLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group created by the pattern that Kinesis Firehose access logs are sent to.</td>
</tr>
<tr>
<td>s3BucketInterface</td>
<td>s3.IBucket</td>
<td>Returns an instance of s3.IBucket created by the construct.</td>
</tr>
</tbody>
</table>

### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

#### Amazon EventBridge Rule

- Configure least privilege access IAM role for Amazon EventBridge Rule to publish to the Kinesis Firehose Delivery Stream.

#### Amazon Kinesis Firehose

- Enable CloudWatch logging for Kinesis Firehose.
- Configure least privilege access IAM role for Amazon Kinesis Firehose.

#### Amazon S3 bucket

- Configure access logging for bucket.
- Enable server-side encryption for bucket using AWS managed KMS Key.
- Turn on the versioning for the bucket.
- Don't allow public access for the bucket.
- Retain the bucket when deleting the CloudFormation stack.
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days.
Architecture

Amazon EventBridge Rule → Amazon Kinesis Data Firehose → Amazon Simple Storage Service

Amazon CloudWatch → Amazon Simple Storage Service (Access Log)

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-eventbridge-kinesisfirehose-s3

aws-events-rule-kinesisstreams

Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.

Note: This construct has been deprecated and is superseded by the aws-eventbridge-kinesisstreams construct.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_events_rule_kinesisstreams</td>
</tr>
</tbody>
</table>
Overview

This AWS Solutions Construct implements an Amazon CloudWatch Events rule to send data to an Amazon Kinesis Data Stream.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import * as cdk from '@aws-cdk/core';
import {EventsRuleToKinesisStreams, EventsRuleToKinesisStreamsProps} from '@aws-solutions-constructs/aws-events-rule-kinesisstreams';

const props: EventsRuleToKinesisStreamsProps = {
  eventRuleProps: {
    schedule: events.Schedule.rate(Duration.minutes(5)),
  }
};

new EventsRuleToKinesisStreams(this, 'test-events-rule-kinesis-streams', props);
```

Initializer

```typescript
new EventsRuleToKinesisStreams(scope: Construct, id: string, props: EventsRuleToKinesisStreamsProps);
```

Parameters

- `scope` Construct
- `id` string
- `props` EventsRuleToKinesisStreamsProps (p. 103)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingEventBusInterface?</td>
<td>events.IEventBus</td>
<td>Optional user-provided custom EventBus for construct to</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the EventBus created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>kinesisStream</td>
<td>kinesis.Stream</td>
<td>Returns an instance of the Kinesis stream created by the pattern.</td>
</tr>
<tr>
<td>eventsRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the role created by the construct for the CloudWatch Events rule.</td>
</tr>
</tbody>
</table>
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon CloudWatch Events rule**

- Configure least privilege access IAM role for Events Rule to publish to the Kinesis Data Stream.

**Amazon Kinesis Stream**

- Enable server-side encryption for Kinesis Data Stream using AWS Managed KMS Key.

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:
aws-eventbridge-kinesisstreams

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

### Language | Package
--- | ---
Python | aws_solutions_constructs.aws_eventbridge_kinesisstreams
Typescript | @aws-solutions-constructs/aws-eventbridge-kinesisstreams
Java | software.amazon.awsconstructs.services.eventbridgekinesisstreams

### Overview

This AWS Solutions Construct implements an Amazon EventBridge rule to send data to an Amazon Kinesis Data Stream.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import * as cdk from '@aws-cdk/core';
import {EventbridgeToKinesisStreams, EventbridgeToKinesisStreamsProps} from "@aws-solutions-constructs/aws-eventbridge-kinesisstreams";

const props: EventbridgeToKinesisStreamsProps = {
  eventRuleProps: {
    schedule: events.Schedule.rate(Duration.minutes(5)),
  }
};

new EventbridgeToKinesisStreams(this, 'test-eventbridge-kinesis-streams', props);
```

### Initializer

```typescript
new EventbridgeToKinesisStreams(scope: Construct, id: string, props: EventbridgeToKinesisStreamsProps);
```
Parameters

- scope Construct
- id string
- props EventbridgeToKinesisStreamsProps (p. 107)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingEventBusInterface?</td>
<td>events.IEventBus</td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and eventBusProps results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td>events.EventBusProps</td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to {} will create a custom EventBus using all default properties. If neither this nor existingEventBusInterface is provided the construct will use the default EventBus. Providing both this and existingEventBusInterface results in an error.</td>
</tr>
<tr>
<td>eventRuleProps</td>
<td>events.RuleProps</td>
<td>User-provided properties to override the default properties for the CloudWatch Events rule.</td>
</tr>
<tr>
<td>existingStreamObj?</td>
<td>kinesis.Stream</td>
<td>Existing instance of Kinesis Stream, providing both this and kinesisStreamProps will cause an error.</td>
</tr>
<tr>
<td>kinesisStreamProps?</td>
<td>kinesis.StreamProps</td>
<td>Optional user-provided props to override the default props for the Kinesis stream.</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td>boolean</td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
</tbody>
</table>

Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the EventBus created by the pattern.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>kinesisStream</td>
<td>kinesis.Stream</td>
<td>Returns an instance of the Kinesis stream created by the pattern.</td>
</tr>
<tr>
<td>eventsRole?</td>
<td>iam.Role</td>
<td>Returns an instance of the role created by the construct for the CloudWatch Events rule.</td>
</tr>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns an instance of the cloudwatch.Alarm[] created by the construct.</td>
</tr>
</tbody>
</table>

**Default settings**

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon EventBridge Rule**

- Configure least privilege access IAM role for the EventBridge Rule to publish to the Kinesis Data Stream.

**Amazon Kinesis Stream**

- Enable server-side encryption for Kinesis Data Stream using AWS Managed KMS Key.
Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.
Note: This construct has been deprecated and is superseded by the `aws-eventbridge-lambda` construct.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td><code>aws_solutions_constructs.aws_events_rule_lambda</code></td>
</tr>
<tr>
<td>Typescript</td>
<td><code>@aws-solutions-constructs/aws-events-rule-lambda</code></td>
</tr>
<tr>
<td>Java</td>
<td><code>software.amazon.awsconstructs.services.eventsrulelambda</code></td>
</tr>
</tbody>
</table>

### Overview

This AWS Solutions Construct implements an AWS Events rule and an AWS Lambda function.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { EventsRuleToLambdaProps, EventsRuleToLambda } from '@aws-solutions-constructs/aws-events-rule-lambda';

const props: EventsRuleToLambdaProps = {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler'
  },
  eventRuleProps: {
    schedule: events.Schedule.rate(Duration.minutes(5))
  }
};

new EventsRuleToLambda(this, 'test-events-rule-lambda', props);
```

### Initializer

```typescript
new EventsRuleToLambda(scope: Construct, id: string, props: EventsRuleToLambdaProps);
```

### Parameters

- **scope** `Construct`
- **id** `string`
- **props** `EventsRuleToLambdaProps` (p. 111)
## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>existingEventBusInterface?</td>
<td>events.IEventBus</td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and eventBusProps results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td>events.EventBusProps</td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to {} will create a custom EventBus using all default properties. If neither this nor existingEventBusInterface is provided the construct will use the default EventBus. Providing both this and existingEventBusInterface results in an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>eventRuleProps</td>
<td>events.RuleProps</td>
<td>User provided eventRuleProps to override the defaults</td>
</tr>
</tbody>
</table>

## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the EventBus created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
</tbody>
</table>
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon CloudWatch Events Rule**
- Grant least privilege permissions to CloudWatch Events to trigger the Lambda Function

**AWS Lambda Function**
- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for NodeJs Lambda function
- Enable X-Ray tracing
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**Architecture**

![Architecture Diagram]

Role

Events Rule

Lambda function

Amazon CloudWatch
**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td><code>aws_solutions_constructs.aws_eventbridge_lambda</code></td>
</tr>
<tr>
<td>TypeScript</td>
<td><code>@aws-solutions-constructs/aws-eventbridge-lambda</code></td>
</tr>
<tr>
<td>Java</td>
<td><code>software.amazon.awsconstructs.services.eventbridge.lambda</code></td>
</tr>
</tbody>
</table>

**aws-eventbridge-lambda**

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

**Overview**

This AWS Solutions Construct implements an AWS EventBridge rule and an AWS Lambda function.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
const { EventbridgeToLambdaProps, EventbridgeToLambda } from '@aws-solutions-constructs/aws-eventbridge-lambda';

const props: EventbridgeToLambdaProps = {
    lambdaFunctionProps: {
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'index.handler'
    },
    eventRuleProps: {
        schedule: events.Schedule.rate(Duration.minutes(5))
    }
};
```
new EventbridgeToLambda(this, 'test-eventbridge-lambda', props);

## Initializer

new EventbridgeToLambda(scope: Construct, id: string, props: EventbridgeToLambdaProps);

### Parameters

- **scope** `Construct`
- **id** `string`
- **props** `EventbridgeToLambdaProps (p. 114)`

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>existingEventBusInterface?</td>
<td><code>events.IEventBus</code></td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and eventBusProps results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td><code>events.EventBusProps</code></td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to {} will create a custom EventBus using all default properties. If neither this nor existingEventBusInterface is provided the construct will use the default EventBus. Providing both this and existingEventBusInterface results in an error.</td>
</tr>
<tr>
<td>eventRuleProps</td>
<td><code>events.RuleProps</code></td>
<td>User provided eventRuleProps to override the defaults</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the Event bus created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon EventBridge Rule

- Grant least privilege permissions to EventBridge to trigger the Lambda Function

AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function
- Enable reusing connections with Keep-Alive for NodeJs Lambda function
- Enable X-Ray tracing
- Set environment variables:
  - AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)
Architecture

Amazon EventBridge Rule → Lambda Function → Amazon CloudWatch

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-eventbridge-lambda
Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.

Note: This construct has been deprecated and is superseded by the `aws-eventbridge-sns` construct.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

### Overview

This pattern implements an Amazon CloudWatch Events rule connected to an Amazon SNS topic.

Here is a minimal deployable pattern definition:

```javascript
import { Duration } from '@aws-cdk/core';
import * as events from '@aws-cdk/aws-events';
import * as iam from '@aws-cdk/aws-iam';
import { EventsRuleToSnsProps, EventsRuleToSns } from '@aws-solutions-constructs/aws-events-rule-sns';

const props: EventsRuleToSnsProps = {
  eventRuleProps: {
    schedule: events.Schedule.rate(Duration.minutes(5)),
  }
};

const constructStack = new EventsRuleToSns(this, 'test-construct', props);

// Grant yourself permissions to use the Customer Managed KMS Key
const policyStatement = new iam.PolicyStatement({
  actions: ['kms:Encrypt', 'kms:Decrypt'],
  effect: iam.Effect.ALLOW,
});
```
```
principals: [ new iam.AccountRootPrincipal() ],
            resources: [ "*" ]
});

constructStack.encryptionKey?.addToResourcePolicy(policyStatement);
```

### New EventsRuleToSNS

```
new EventsRuleToSNS(scope: Construct, id: string, props: EventsRuleToSNSProps);
```

#### Parameters

- `scope` - `Construct`
- `id` - `string`
- `props` - `EventsRuleToSnsProps (p. 118)`

### Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventRuleProps</td>
<td><code>events.RuleProps</code></td>
<td>User-provided properties to override the default properties for the CloudWatch Events rule.</td>
</tr>
<tr>
<td>existingTopicObj?</td>
<td><code>sns.Topic</code></td>
<td>Existing instance of SNS Topic object, providing both this and topicProps will cause an error.</td>
</tr>
<tr>
<td>topicProps?</td>
<td><code>sns.TopicProps</code></td>
<td>Optional user-provided properties to override the default properties for the SNS topic. Ignored if an existingTopicObj is provided.</td>
</tr>
<tr>
<td>existingEventBusInterface?</td>
<td><code>events.IEventBus</code></td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and eventBusProps results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td><code>events.EventBusProps</code></td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to {} will create a custom EventBus using all default properties. If neither this nor existingEventBusInterface is provided the construct will use the default EventBus. Providing both this and</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the Event bus created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>snsTopic</td>
<td>sns.Topic</td>
<td>Returns an instance of the SNS topic created by the pattern.</td>
</tr>
<tr>
<td>encryptionKey</td>
<td>kms.Key</td>
<td>Returns an instance of the encryption key created by the pattern.</td>
</tr>
<tr>
<td>enableEncryptionWithCustomerManagedKey?</td>
<td></td>
<td>Whether to use a customer-managed encryption key, either managed by this CDK app or imported. If importing an encryption key, it must be specified in the encryptionKey property for this construct.</td>
</tr>
<tr>
<td>encryptionKey?</td>
<td>kms.Key</td>
<td>An optional, existing encryption key to be used instead of the default encryption key.</td>
</tr>
<tr>
<td>encryptionKeyProps?</td>
<td>kms.KeyProps</td>
<td>Optional user-provided properties to override the default properties for the encryption key.</td>
</tr>
</tbody>
</table>

### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

#### Amazon CloudWatch Events rule
- Grant least privilege permissions to CloudWatch Events to publish to the SNS topic.

#### Amazon SNS topic
- Configure least privilege access permissions for SNS topic.
- Enable server-side encryption for SNS topic using customer-managed AWS KMS key.
- Enforce encryption of data in transit.
Architecture

To view the code for this pattern, create/view issues and pull requests, and more:

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_eventbridge_sns</td>
</tr>
</tbody>
</table>

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This pattern implements an Amazon EventBridge rule connected to an Amazon SNS topic.

Here is a minimal deployable pattern definition:

```javascript
import { Duration } from '@aws-cdk/core';
import * as events from '@aws-cdk/aws-events';
import * as iam from '@aws-cdk/aws-iam';
import { EventbridgeToSnsProps, EventbridgeToSns } from '@aws-solutions-constructs/aws-eventbridge-sns';

const props: EventbridgeToSnsProps = {
  eventRuleProps: {
    schedule: events.Schedule.rate(Duration.minutes(5)),
  }
};

const constructStack = new EventbridgeToSns(this, 'test-construct', props);

// Grant yourself permissions to use the Customer Managed KMS Key
const policyStatement = new iam.PolicyStatement({
  actions: ['kms:Encrypt', 'kms:Decrypt'],
  effect: iam.Effect.ALLOW,
  principals: [ new iam.AccountRootPrincipal() ],
  resources: ['*']
});

constructStack.encryptionKey?.addToResourcePolicy(policyStatement);
```

Initializer

new EventbridgeToSns(scope: Construct, id: string, props: EventbridgeToSnsProps);

Parameters

- **scope** `Construct`
- **id** `string`
- **props** `EventbridgeToSnsProps` [p. 122]
## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventRuleProps</td>
<td><code>events.RuleProps</code></td>
<td>User-provided properties to override the default properties for the CloudWatch Events rule.</td>
</tr>
<tr>
<td>existingTopicObj?</td>
<td><code>sns.Topic</code></td>
<td>Existing instance of SNS Topic object, providing both this and topicProps will cause an error.</td>
</tr>
<tr>
<td>topicProps?</td>
<td><code>sns.TopicProps</code></td>
<td>Optional user-provided properties to override the default properties for the SNS topic. Ignored if an existingTopicObj is provided.</td>
</tr>
<tr>
<td>existingEventBusInterface?</td>
<td><code>events.IEventBus</code></td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and eventBusProps results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td><code>events.EventBusProps</code></td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to <code>{}</code> will create a custom EventBus using all default properties. If neither this nor existingEventBusInterface is provided the construct will use the default EventBus. Providing both this and existingEventBusInterface results in an error.</td>
</tr>
<tr>
<td>enableEncryptionWithCustomerManagedKey?</td>
<td></td>
<td>Whether to use a customer-managed encryption key, either managed by this CDK app or imported. If importing an encryption key, it must be specified in the encryptionKey property for this construct.</td>
</tr>
<tr>
<td>encryptionKey?</td>
<td><code>kms.Key</code></td>
<td>An optional, existing encryption key to be used instead of the default encryption key.</td>
</tr>
<tr>
<td>encryptionKeyProps?</td>
<td><code>kms.KeyProps</code></td>
<td>Optional user-provided properties to override the default properties for the encryption key.</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the Event bus created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>snsTopic</td>
<td>sns.Topic</td>
<td>Returns an instance of the SNS topic created by the pattern.</td>
</tr>
<tr>
<td>encryptionKey</td>
<td>kms.Key</td>
<td>Returns an instance of the encryption key created by the pattern.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon EventBridge Rule

- Grant least privilege permissions to EventBridge Rule to publish to the SNS topic.

Amazon SNS topic

- Configure least privilege access permissions for SNS topic.
- Enable server-side encryption for SNS topic using customer-managed AWS KMS key.
- Enforce encryption of data in transit.
Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.

Note: This construct has been deprecated and is superseded by the `aws-eventbridge-sqs` construct.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
### Overview

This pattern implements an Amazon CloudWatch Events rule connected to an Amazon SQS queue.

Here is a minimal deployable pattern definition:

```javascript
import { Duration } from '@aws-cdk/core';
import * as events from '@aws-cdk/aws-events';
import * as iam from '@aws-cdk/aws-iam';
import { EventsRuleToSqsProps, EventsRuleToSqs } from '@aws-solutions-constructs/aws-events-rule-sqs';

const props: EventsRuleToSqsProps = {
  eventRuleProps: {
    schedule: events.Schedule.rate(Duration.minutes(5))
  }
};

const constructStack = new EventsRuleToSqs(this, 'test-construct', props);

// Grant yourself permissions to use the Customer Managed KMS Key
const policyStatement = new iam.PolicyStatement(
  actions: ['kms:Encrypt', 'kms:Decrypt'],
  effect: iam.Effect.ALLOW,
  principals: [ new iam.AccountRootPrincipal() ],
  resources: [ '*' ]
);

constructStack.encryptionKey?.addToResourcePolicy(policyStatement);
```

### Initializer

```javascript
new EventsRuleToSqs(scope: Construct, id: string, props: EventsRuleToSqsProps);
```

### Parameters

- **scope** `Construct`
- `id` string
- `props` `EventsRuleToSqsProps` (p. 126)

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>existingEventBusInterface</code></td>
<td><code>events.IEventBus</code></td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and <code>eventBusProps</code> results an error.</td>
</tr>
<tr>
<td><code>eventBusProps</code></td>
<td><code>events.EventBusProps</code></td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to <code>{}</code> will create a custom EventBus using all default properties. If neither this nor <code>existingEventBusInterface</code> is provided the construct will use the default EventBus. Providing both this and <code>existingEventBusInterface</code> results in an error.</td>
</tr>
<tr>
<td><code>eventRuleProps</code></td>
<td><code>events.RuleProps</code></td>
<td>User-provided properties to override the default properties for the CloudWatch Events rule.</td>
</tr>
<tr>
<td><code>existingQueueObj</code></td>
<td><code>sqs.Queue</code></td>
<td>An optional, existing SQS queue to be used instead of the default queue. Providing both this and <code>queueProps</code> will cause an error.</td>
</tr>
<tr>
<td><code>queueProps</code></td>
<td><code>sqs.QueueProps</code></td>
<td>Optional user-provided props to override the default props for the SQS queue. Ignored if an <code>existingQueueObj</code> is provided.</td>
</tr>
<tr>
<td><code>enableQueuePurging</code></td>
<td><code>boolean</code></td>
<td>Whether to grant additional permissions to the Lambda function enabling it to purge the SQS queue. Defaults to false.</td>
</tr>
<tr>
<td><code>deployDeadLetterQueue</code></td>
<td><code>boolean</code></td>
<td>Whether to create a secondary queue to be used as a dead letter queue. Defaults to true.</td>
</tr>
<tr>
<td><code>deadLetterQueueProps</code></td>
<td><code>sqs.QueueProps</code></td>
<td>Optional user-provided props to override the default props for the dead letter queue. Only used if the</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td><code>events.IEventBus</code></td>
<td>Returns an instance of the Event bus created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td><code>events.Rule</code></td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>sqsQueue</td>
<td><code>sqs.Queue</code></td>
<td>Returns an instance of the SQS queue created by the pattern.</td>
</tr>
<tr>
<td>encryptionKey</td>
<td><code>kms.Key</code></td>
<td>Returns an instance of the encryption key created by the pattern.</td>
</tr>
<tr>
<td>deadLetterQueue?</td>
<td><code>sqs.Queue</code></td>
<td>Returns an instance of the dead letter queue created by the pattern, if one is deployed.</td>
</tr>
</tbody>
</table>

**Default settings**

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:
Amazon CloudWatch Events rule

- Grant least privilege permissions to CloudWatch Events to publish to the SQS Queue.

Amazon SQS queue

- Deploy a dead-letter queue for the source queue.
- Enable server-side encryption for the source queue using a customer-managed AWS KMS key.
- Enforce encryption of data in transit.

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-events-rule-sqs

aws-eventbridge-sqs
**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_eventbridge_sqs</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-eventbridge-sqs</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.eventbridgesqs</td>
</tr>
</tbody>
</table>

**Overview**

This pattern implements an Amazon EventBridge rule connected to an Amazon SQS queue.

Here is a minimal deployable pattern definition:

```javascript
import { Duration } from '@aws-cdk/core';
import * as events from '@aws-cdk/aws-events';
import * as iam from '@aws-cdk/aws-iam';
import { EventbridgeToSqsProps, EventbridgeToSqs } from '@aws-solutions-constructs/aws-eventbridge-sqs';

const props: EventbridgeToSqsProps = {
  eventRuleProps: {
    schedule: events.Schedule.rate(Duration.minutes(5))
  }
};

const constructStack = new EventbridgeToSqs(this, 'test-construct', props);

// Grant yourself permissions to use the Customer Managed KMS Key
const policyStatement = new iam.PolicyStatement({
  actions: ['kms:Encrypt', 'kms:Decrypt'],
  effect: iam.Effect.ALLOW,
  principals: [ new iam.AccountRootPrincipal() ],
  resources: [ '*' ]
});

constructStack.encryptionKey?.addToResourcePolicy(policyStatement);
```

**Initializer**

```javascript
new EventbridgeToSqs(scope: Construct, id: string, props: EventbridgeToSqsProps);
```
Parameters

- scope **Construct**
- id **string**
- props **EventbridgeToSqsProps** (p. 130)

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingEventBusInterface?</td>
<td>events.IEventBus</td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and eventBusProps results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td>events.EventBusProps</td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to {} will create a custom EventBus using all default properties. If neither this nor existingEventBusInterface is provided the construct will use the default EventBus. Providing both this and existingEventBusInterface results in an error.</td>
</tr>
<tr>
<td>eventRuleProps</td>
<td>events.RuleProps</td>
<td>User-provided properties to override the default properties for the CloudWatch Events rule.</td>
</tr>
<tr>
<td>existingQueueObj?</td>
<td>sqs.Queue</td>
<td>An optional, existing SQS queue to be used instead of the default queue. Providing both this and queueProps will cause an error.</td>
</tr>
<tr>
<td>queueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user-provided properties to override the default properties for the SQS queue. Ignored if an existingQueueObj is provided.</td>
</tr>
<tr>
<td>enableQueuePurging?</td>
<td>boolean</td>
<td>Whether to grant additional permissions to the Lambda function enabling it to purge the SQS queue. Defaults to false.</td>
</tr>
<tr>
<td>deployDeadLetterQueue?</td>
<td>boolean</td>
<td>Whether to create a secondary queue to be used as a dead letter queue. Defaults to true.</td>
</tr>
<tr>
<td>deadLetterQueueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user-provided props to override the default</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the Event bus created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>sqsQueue</td>
<td>sqs.Queue</td>
<td>Returns an instance of the SQS queue created by the pattern.</td>
</tr>
<tr>
<td>encryptionKey</td>
<td>kms.Key</td>
<td>Returns an instance of the encryption key created by the pattern.</td>
</tr>
<tr>
<td>deadLetterQueue?</td>
<td>sqs.Queue</td>
<td>Returns an instance of the dead letter queue created by the pattern, if one is deployed.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:
Amazon EventBridge Rule

- Grant least privilege permissions to the EventBridge rule to publish to the SQS Queue.

Amazon SQS queue

- Deploy a dead-letter queue for the source queue.
- Enable server-side encryption for the source queue using a customer-managed AWS KMS key.
- Enforce encryption of data in transit.

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-eventbridge-sqs
aws-events-rule-step-function

Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.

Note: This construct has been deprecated and is superseded by the `aws-eventbridge-stepfunctions` construct.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_events_rule_step_function</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-events-rule-step-function</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.eventsrulestepfunction</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an AWS Events rule and an AWS Step function.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { EventsRuleToStepFunction, EventsRuleToStepFunctionProps } from '@aws-solutions-constructs/aws-events-rule-step-function';

const startState = new stepfunctions.Pass(this, 'StartState');

const props: EventsRuleToStepFunctionProps = {
  stateMachineProps: {
    definition: startState
  },
  eventRuleProps: {
    schedule: events.Schedule.rate(Duration.minutes(5))
  }
};

new EventsRuleToStepFunction(this, 'test-events-rule-step-function-stack', props);
```
Initializer

```javascript
new EventsRuleToStepFunction(scope: Construct, id: string, props: EventsRuleToStepFunctionProps);
```

**Parameters**
- `scope` *Construct*
- `id` *string*
- `props` *EventsRuleToStepFunctionProps* (p. 134)

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stateMachineProps</td>
<td><code>sfn.StateMachineProps</code></td>
<td>Optional user provided props to override the default props for <code>sfn.StateMachine</code></td>
</tr>
<tr>
<td>existingEventBusInterface?</td>
<td><code>events.IEventBus</code></td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and <code>eventBusProps</code> results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td><code>events.EventBusProps</code></td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to <code>{}</code> will create a custom EventBus using all default properties. If neither this nor <code>existingEventBusInterface</code> is provided the construct will use the default EventBus. Providing both this and <code>existingEventBusInterface</code> results in an error.</td>
</tr>
<tr>
<td>eventRuleProps</td>
<td><code>events.RuleProps</code></td>
<td>User provided <code>eventRuleProps</code> to override the defaults</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td><code>boolean</code></td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
</tbody>
</table>
# Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the Event bus created by the pattern.</td>
</tr>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns a list of one or more CloudWatch alarms created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>stateMachine</td>
<td>sfn.StateMachine</td>
<td>Returns an instance of the state machine created by the pattern.</td>
</tr>
<tr>
<td>stateMachineLogGroup</td>
<td>logs.ILogGroup</td>
<td>Returns an instance of the ILogGroup created by the pattern for the state machine.</td>
</tr>
</tbody>
</table>

## Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

### Amazon CloudWatch Events Rule
- Grant least privilege permissions to CloudWatch Events to trigger the Lambda Function

### AWS Step Function
- Enable CloudWatch logging for API Gateway
- Deploy best practice CloudWatch Alarms for the Step Function
**Architecture**

Amazon CloudWatch Event Rule → Role → AWS Step Functions → Amazon CloudWatch

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:


**aws-eventbridge-stepfunctions**

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_eventbridge_stepfunctions</td>
</tr>
<tr>
<td>JS</td>
<td>@aws-solutions-constructs/aws-eventbridge-stepfunctions</td>
</tr>
</tbody>
</table>
Overview

This AWS Solutions Construct implements an AWS EventBridge rule connected to an AWS Step Functions State Machine.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
const { EventbridgeToStepfunctions, EventbridgeToStepfunctionsProps } from '@aws-solutions-constructs/aws-eventbridge-stepfunctions';
const startState = new stepfunctions.Pass(this, 'StartState');
const props: EventbridgeToStepfunctionsProps = {
  stateMachineProps: {
    definition: startState
  },
  eventRuleProps: {
    schedule: events.Schedule.rate(Duration.minutes(5))
  }
};
new EventbridgeToStepfunctions(stack, 'test-eventbridge-stepfunctions-stack', props);
```

Initializer

```typescript
new EventbridgeToStepfunctions(scope: Construct, id: string, props: EventbridgeToStepfunctionsProps);
```

Parameters

- `scope` `Construct`
- `id` `string`
- `props` `EventbridgeToStepfunctionsProps` (p. 137)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stateMachineProps</td>
<td><code>sfn.StateMachineProps</code></td>
<td>Optional user provided props to override the default props for <code>sfnStateMachine</code></td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingEventBusInterface?</td>
<td>events.IEventBus</td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and eventBusProps results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td>events.EventBusProps</td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to {} will create a custom EventBus using all default properties. If neither this nor existingEventBusInterface is provided the construct will use the default EventBus. Providing both this and existingEventBusInterface results in an error.</td>
</tr>
<tr>
<td>eventRuleProps</td>
<td>events.RuleProps</td>
<td>User provided eventRuleProps to override the defaults.</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td>boolean</td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td>logs.LogGroupProps</td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventBus?</td>
<td>events.IEventBus</td>
<td>Returns an instance of the EventBus created by the pattern.</td>
</tr>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns a list of one or more CloudWatch alarms created by the pattern.</td>
</tr>
<tr>
<td>eventsRule</td>
<td>events.Rule</td>
<td>Returns an instance of the Events rule created by the pattern.</td>
</tr>
<tr>
<td>stateMachine</td>
<td>sfn.StateMachine</td>
<td>Returns an instance of the state machine created by the pattern.</td>
</tr>
<tr>
<td>stateMachineLogGroup</td>
<td>logs.ILogGroup</td>
<td>Returns an instance of the ILogGroup created by the pattern.</td>
</tr>
</tbody>
</table>
**Default settings**

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon EventBridge Rule**
- Grant least privilege permissions to the EventBridge Rule to trigger the State Machine.

**AWS Step Function**
- Enable CloudWatch logging for API Gateway
- Deploy best practice CloudWatch Alarms for the Step Function

**Architecture**

![Architecture Diagram]

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-eventbridge-stepfunctions
AWS Solutions Constructs AWS Solutions
aws-iot-kinesisfirehose-s3

aws-iot-kinesisfirehose-s3

**STABILITY EXPERIMENTAL**

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions Constructs.aws_iot_kinesisfirehose-s3</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-iot-kinesisfirehose-s3</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.iotkinesisfirehose.s3</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an AWS IoT MQTT topic rule to send data to an Amazon Kinesis Data Firehose delivery stream connected to an Amazon S3 bucket.

Here is a minimal deployable pattern definition in TypeScript:

```javascript
import { IotToKinesisFirehoseToS3Props, IotToKinesisFirehoseToS3 } from '@aws-solutions-constructs/aws-iot-kinesisfirehose-s3';

const props: IotToKinesisFirehoseToS3Props = {
  iotTopicRuleProps: {
    topicRulePayload: {
      ruleDisabled: false,
      description: "Persistent storage of connected vehicle telematics data",
      sql: "SELECT * FROM 'connectedcar/telemetry/#'",
      actions: []
    }
  }
};

new IotToKinesisFirehoseToS3(this, 'test-iot-firehose-s3', props);
```

**Initializer**
```typescript
new IotToKinesisFirehoseToS3(scope: Construct, id: string, props: IotToKinesisFirehoseToS3Props);
```

### Parameters

- **scope** `Construct`
- **id** `string`
- **props** `IotToKinesisFirehoseToS3Props (p. 141)`

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iotTopicRuleProps</td>
<td><code>iot.CfnTopicRuleProps</code></td>
<td>User provided <code>CfnTopicRuleProps</code> to override the defaults</td>
</tr>
<tr>
<td>kinesisFirehoseProps?</td>
<td><code>kinesisfirehose.CfnDeliveryStreamProps</code></td>
<td>Optional user-provided props to override the default props for Kinesis Firehose Delivery Stream</td>
</tr>
<tr>
<td>existingBucketObj?</td>
<td><code>s3.Bucket</code></td>
<td>Existing instance of S3 Bucket object, providing both this and <code>bucketProps</code> will cause an error.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>User provided props to override the default props for the S3 Bucket. If this is provided, then also providing <code>bucketProps</code> is an error.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
<tr>
<td>loggingBucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user-provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td>logS3AccessLogs?</td>
<td><code>boolean</code></td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is <code>true</code>.</td>
</tr>
</tbody>
</table>
# Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iotActionsRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the IoT rule.</td>
</tr>
<tr>
<td>iotTopicRule</td>
<td>iot.CfnTopicRule</td>
<td>Returns an instance of the IoT topic rule created by the pattern.</td>
</tr>
<tr>
<td>kinesisFirehose</td>
<td>kinesisfirehose.CfnDeliveryStream</td>
<td>Returns an instance of the Kinesis Firehose delivery stream created by the pattern.</td>
</tr>
<tr>
<td>kinesisFirehoseLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group created by the pattern that Kinesis Firehose access logs are sent to.</td>
</tr>
<tr>
<td>kinesisFirehoseRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the Kinesis Firehose delivery stream.</td>
</tr>
<tr>
<td>s3Bucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
<tr>
<td>s3BucketInterface</td>
<td>s3.IBucket</td>
<td>Returns an instance of s3.IBucket created by the construct.</td>
</tr>
</tbody>
</table>

## Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

### Amazon IoT Rule

- Configure least privilege access IAM role for Amazon IoT

### Amazon Kinesis Firehose

- Enable CloudWatch logging for Kinesis Firehose
- Configure least privilege access IAM role for Amazon Kinesis Firehose

### Amazon S3 Bucket

- Configure Access logging for S3 Bucket
- Enable server-side encryption for S3 Bucket using AWS managed KMS Key
- Turn on the versioning for S3 Bucket
- Don't allow public access for S3 Bucket
- Retain the S3 Bucket when deleting the CloudFormation stack
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-iot-kinesisfirehose-s3

aws-iot-kinesisstreams

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an AWS IoT MQTT topic rule to send data to an Amazon Kinesis Data Stream.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
const { IotToKinesisStreamsProps, IotToKinesisStreams } from '@aws-solutions-constructs/aws-iot-kinesisstreams';

const props: IotToKinesisStreamsProps = {
  iotTopicRuleProps: {
    topicRulePayload: {
      ruleDisabled: false,
      description: "Sends data to kinesis data stream",
      sql: "SELECT * FROM 'solutions/construct'",
      actions: []
    }
  }
};

new IotToKinesisStreams(this, 'test-iot-kinesisstream', props);
```

Initializer

```typescript
new IotToKinesisStreams(scope: Construct, id: string, props: IotToKinesisStreamsProps);
```

Parameters

- `scope` `Construct`
- `id` `string`
- `props` `IotToKinesisStreamsProps (p. 145)`
## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iotTopicRuleProps</td>
<td>iot.CfnTopicRuleProps</td>
<td>User provided CfnTopicRuleProps to override the defaults.</td>
</tr>
<tr>
<td>existingStreamObj?</td>
<td>kinesis.Stream</td>
<td>Existing instance of Kinesis Stream, providing both this and kinesisStreamProps will cause an error.</td>
</tr>
<tr>
<td>kinesisStreamProps?</td>
<td>kinesis.StreamProps</td>
<td>Optional user-provided props to override the default props for the Kinesis data stream, providing both this and existingStreamObj will cause an error.</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td>boolean</td>
<td>Whether to create recommended CloudWatch alarms for Kinesis Data Stream. Default value is set to true.</td>
</tr>
</tbody>
</table>

## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iotActionsRole</td>
<td>iam.Role</td>
<td>Returns an instance of the iam.Role created by the construct for IoT Rule.</td>
</tr>
<tr>
<td>kinesisStream</td>
<td>kinesis.Stream</td>
<td>Returns an instance of the Kinesis stream created by the construct.</td>
</tr>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns an array of recommended CloudWatch Alarms created by the construct for Kinesis Data stream.</td>
</tr>
</tbody>
</table>

## Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

### Amazon IoT Rule

- Configure least privilege access IAM role for Amazon IoT Rule.
Amazon Kinesis Data Stream

- Configure recommended CloudWatch Alarms for Amazon Kinesis Data Stream.
- Configure least privilege access IAM role for Amazon Kinesis Data Stream.

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-iot-kinesisstreams

aws-iot-lambda

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.
**Overview**

This AWS Solutions Constructs pattern implements an AWS IoT MQTT topic rule and an AWS Lambda function pattern.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { IotToLambdaProps, IotToLambda } from '@aws-solutions-constructs/aws-iot-lambda';

const props: IotToLambdaProps = {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler'
  },
  iotTopicRuleProps: {
    topicRulePayload: {
      ruleDisabled: false,
      description: "Processing of DTC messages from the AWS Connected Vehicle Solution.",
      sql: "SELECT * FROM 'connectedcar/dtc/#'",
      actions: []
    }
  }
};

new IotToLambda(this, 'test-iot-lambda-integration', props);
```

**Initializer**

```typescript
new IotToLambda(scope: Construct, id: string, props: IotToLambdaProps);
```

**Parameters**

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
• scope Construct
• id string
• props IotToLambdaProps (p. 148)

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>iotTopicRuleProps?</td>
<td>iot.CfnTopicRuleProps</td>
<td>User provided CfnTopicRuleProps to override the defaults</td>
</tr>
</tbody>
</table>

## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iotTopicRule</td>
<td>iot.CfnTopicRule</td>
<td>Returns an instance of the IoT topic rule created by the pattern.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
</tbody>
</table>

## Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

### Amazon IoT Rule

* Configure least privilege access IAM role for Amazon IoT.

### AWS Lambda Function

* Configure limited privilege access IAM role for Lambda function.
* Enable reusing connections with Keep-Alive for NodeJs Lambda function.
* Enable X-Ray tracing.
• Set environment variables:
  • AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)

**Architecture**

![Architecture diagram](image)

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-iot-lambda

**aws-iot-lambda-dynamodb**

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
### Overview

This AWS Solutions Constructs pattern implements an AWS IoT topic rule, an AWS Lambda function and Amazon DynamoDB table with the least privileged permissions.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { IotToLambdaToDynamoDBProps, IotToLambdaToDynamoDB } from '@aws-solutions-constructs/aws-iot-lambda-dynamodb';

const props: IotToLambdaToDynamoDBProps = {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler'
  },
  iotTopicRuleProps: {
    topicRulePayload: {
      ruleDisabled: false,
      description: "Processing of DTC messages from the AWS Connected Vehicle Solution."
    },
    actions: []
  }
};

new IotToLambdaToDynamoDB(this, 'test-iot-lambda-dynamodb-stack', props);
```

### Initializer

```typescript
new IotToLambdaToDynamoDB(scope: Construct, id: string, props: IotToLambdaToDynamoDBProps);
```

### Parameters

- **scope** *Construct*
• id string
• props `IotToLambdaToDynamoDBProps` (p. 151)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>iotTopicRuleProps</td>
<td><code>iot.CfnTopicRuleProps</code></td>
<td>User provided props to override the default props</td>
</tr>
<tr>
<td>dynamoTableProps?</td>
<td><code>dynamodb.TableProps</code></td>
<td>Optional user provided props to override the default props for DynamoDB Table</td>
</tr>
<tr>
<td>tablePermissions?</td>
<td><code>string</code></td>
<td>Optional table permissions to be granted to the Lambda function. One of the following options may be specified: All, Read, ReadWrite, or Write.</td>
</tr>
</tbody>
</table>

Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamoTable</td>
<td><code>dynamodb.Table</code></td>
<td>Returns an instance of the DynamoDB table created by the pattern.</td>
</tr>
<tr>
<td>iotTopicRule</td>
<td><code>iot.CfnTopicRule</code></td>
<td>Returns an instance of the IoT topic rule created by the pattern.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td><code>lambda.Function</code></td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:
**Amazon IoT Rule**
- Configure least privilege access IAM role for Amazon IoT.

**AWS Lambda Function**
- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**Amazon DynamoDB Table**
- Set the billing mode for DynamoDB Table to On-Demand (Pay per request).
- Enable server-side encryption for DynamoDB Table using AWS managed KMS Key.
- Creates a partition key called 'id' for DynamoDB Table.
- Retain the Table when deleting the CloudFormation stack.
- Enable continuous backups and point-in-time recovery.

**Architecture**

[Diagram showing the flow from IoT rule to Lambda function, then to DynamoDB, and finally to CloudWatch.]
GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-iot-lambda-dynamodb

aws-iot-sqs

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Python]</td>
<td>aws_solutions_constructs.aws_iot_sqs</td>
</tr>
<tr>
<td>![Typescript]</td>
<td>@aws-solutions-constructs/aws-iot-sqs</td>
</tr>
<tr>
<td>![Java]</td>
<td>software.amazon.awsconstructs.services.iotsqs</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements an AWS IoT MQTT topic rule and an AWS SQS Queue pattern.

Here is a minimal deployable pattern definition in TypeScript:

```javascript
const { IotToSqsProps, IotToSqs } from '@aws-solutions-constructs/aws-iot-sqs';

const props: IotToSqsProps = {
  iotTopicRuleProps: {
    topicRulePayload: {
      ruleDisabled: false,
      description: "Testing the IotToSqs Pattern",
      sql: "SELECT * FROM 'iot/sqs/#'",
      actions: []
    }
  }
};
```
new IotToSqs(this, 'test-iot-sqs-integration', props);

**Initializer**

new IotToSqs(scope: Construct, id: string, props: IotToSqsProps);

**Parameters**

- `scope` *Construct*
- `id` *string*
- `props` *IotToSqsProps* (p. 154)

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingQueueObj?</td>
<td>sqs.Queue</td>
<td>Existing instance of SQS queue object, providing both this and queueProps will cause an error.</td>
</tr>
<tr>
<td>queueProps?</td>
<td>sqs.QueueProps</td>
<td>User provided props to override the default props for the SQS queue.</td>
</tr>
<tr>
<td>deadLetterQueueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user provided properties for the dead letter queue.</td>
</tr>
<tr>
<td>deployDeadLetterQueue?</td>
<td>boolean</td>
<td>Whether to deploy a secondary queue to be used as a dead letter queue. Defaults to true.</td>
</tr>
<tr>
<td>maxReceiveCount?</td>
<td>number</td>
<td>The number of times a message can be unsuccessfully dequeued before being moved to the dead letter queue. Required field if deployDeadLetterQueue is set to true.</td>
</tr>
<tr>
<td>enableEncryptionWithCustomerManagedKey?</td>
<td>[kms.Key]</td>
<td>Whether to use a KMS Key, either managed by this CDK app, or imported. If importing an encryption key, it must be specified in the encryptionKey property for this construct.</td>
</tr>
<tr>
<td>encryptionKey?</td>
<td>[kms.Key]</td>
<td>An optional, imported encryption key to encrypt the SQS queue.</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>encryptionKeyProps?</td>
<td>kms.KeyProps</td>
<td>Optional user-provided props to override the default props for the encryption key.</td>
</tr>
<tr>
<td>iotTopicRuleProps?</td>
<td>iot.CfnTopicRuleProps</td>
<td>User provided CfnTopicRuleProps to override the defaults</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon IoT Rule**

- Configure an IoT Rule to send messages to the SQS Queue.

**Amazon IAM Role**

- Configure least privilege access IAM role for Amazon IoT to be able to publish messages to the SQS Queue.

**Amazon SQS Queue**

- Deploy a dead-letter queue for the source queue.
- Enable server-side encryption for the source queue using a customer-managed AWS KMS key.
- Enforce encryption of data in transit.

**Architecture**

![Architecture Diagram]

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:

[@aws-solutions-constructs/aws-iot-sqs](https://github.com/aws-solutions-constructs/aws-iot-sqs)

**aws-kinesisfirehose-s3**

[CFN-RESOURCES STABLE]
**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws-kinesis-firehose-s3</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-kinesisfirehose-s3</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.kinesisfirehose.s3</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an Amazon Kinesis Data Firehose delivery stream connected to an Amazon S3 bucket.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { KinesisFirehoseToS3 } from '@aws-solutions-constructs/aws-kinesisfirehose-s3';
new KinesisFirehoseToS3(this, 'test-firehose-s3', {});
```

**Initializer**

```typescript
new KinesisFirehoseToS3(scope: Construct, id: string, props: KinesisFirehoseToS3Props);
```

**Parameters**

- **scope** `Construct`
- **id** `string`
- **props** `KinesisFirehoseToS3Props` (p. 157)

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user provided props to override the default props for the S3 Bucket.</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingBucketObj?</td>
<td>s3.IBucket</td>
<td>Optional existing instance of S3 Bucket. If this is provided, then also providing bucketProps is an error.</td>
</tr>
<tr>
<td>existingLoggingBucketObj?</td>
<td>s3.IBucket</td>
<td>Optional existing instance of logging S3 Bucket for the S3 Bucket created by the pattern.</td>
</tr>
<tr>
<td>kinesisFirehoseProps?</td>
<td>kinesisfirehose.CfnDeliveryStream</td>
<td>Optional user provided props to override the default props for Kinesis Firehose Delivery Stream.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td>logs.LogGroupProps</td>
<td>Optional user provided props to override the default props for the CloudWatchLogs LogGroup.</td>
</tr>
<tr>
<td>loggingBucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td>logS3AccessLogs?</td>
<td>boolean</td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is true.</td>
</tr>
</tbody>
</table>

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesisFirehose</td>
<td>kinesisfirehose.CfnDeliveryStream</td>
<td>Returns an instance of kinesisfirehose.CfnDeliveryStream created by the construct.</td>
</tr>
<tr>
<td>kinesisFirehoseLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the logs.LogGroup created by the construct for Kinesis Data Firehose delivery stream.</td>
</tr>
<tr>
<td>kinesisFirehoseRole</td>
<td>iam.Role</td>
<td>Returns an instance of the iam.Role created by the construct for Kinesis Data Firehose delivery stream.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of s3.Bucket created by the construct as the logging bucket for the primary bucket.</td>
</tr>
</tbody>
</table>
### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

#### Amazon Kinesis Firehose

- Enable CloudWatch logging for Kinesis Firehose
- Configure least privilege access IAM role for Amazon Kinesis Firehose

#### Amazon S3 Bucket

- Configure Access logging for S3 Bucket
- Enable server-side encryption for S3 Bucket using AWS managed KMS Key
- Turn on the versioning for S3 Bucket
- Don't allow public access for S3 Bucket
- Retain the S3 Bucket when deleting the CloudFormation stack
- Enforce encryption of data in transit
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s3BucketInterface</td>
<td>s3.IBucket</td>
<td>Returns an instance of s3.IBucket created by the construct.</td>
</tr>
</tbody>
</table>
Architecture

Amazon Kinesis Data Firehose

Amazon CloudWatch

Amazon Simple Storage Service

Amazon Simple Storage Service (Access Logs)

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-kinesisfirehose-s3

aws-kinesisfirehose-s3-and-kinesisanalytics

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an Amazon Kinesis Firehose delivery stream connected to an Amazon S3 bucket, and an Amazon Kinesis Analytics application.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { KinesisFirehoseToAnalyticsAndS3 } from '@aws-solutions-constructs/aws-kinesisfirehose-s3-and-kinesisanalytics';

new KinesisFirehoseToAnalyticsAndS3(this, 'FirehoseToS3AndAnalyticsPattern', {
  kinesisAnalyticsProps: {
    inputs: [{
      inputSchema: {
        recordColumns: [{
          name: 'ticker_symbol',
          sqlType: 'VARCHAR(4)',
          mapping: '$.ticker_symbol'
        }, {
          name: 'sector',
          sqlType: 'VARCHAR(16)',
          mapping: '$.sector'
        }, {
          name: 'change',
          sqlType: 'REAL',
          mapping: '$.change'
        }, {
          name: 'price',
          sqlType: 'REAL',
          mapping: '$.price'
        }],
        recordFormat: {
          recordFormatType: 'JSON'
        },
        recordEncoding: 'UTF-8'
      },
      namePrefix: 'SOURCE_SQL_STREAM'
    }]
  }
});
```
**Initializer**

```typescript
new KinesisFirehoseToAnalyticsAndS3(scope: Construct, id: string, props: KinesisFirehoseToAnalyticsAndS3Props);
```

**Parameters**
- `scope` *Construct*
- `id` *string*
- `props` *KinesisFirehoseToAnalyticsAndS3Props* (p. 162)

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>kinesisFirehoseProps?</code></td>
<td><code>kinesisFirehose.CfnDeliveryStreamProps</code></td>
<td>Optional user-provided props to override the default props for the Kinesis Firehose delivery stream.</td>
</tr>
<tr>
<td><code>kinesisAnalyticsProps?</code></td>
<td><code>kinesisAnalytics.CfnApplicationProps</code></td>
<td>Optional user-provided props to override the default props for the Kinesis Analytics application.</td>
</tr>
<tr>
<td><code>existingBucketObj?</code></td>
<td><code>s3.IBucket</code></td>
<td>Existing instance of S3 Bucket object. If this is provided, then also providing <code>bucketProps</code> is an error.</td>
</tr>
<tr>
<td><code>bucketProps?</code></td>
<td><code>s3.BucketProps</code></td>
<td>Optional user-provided properties to override the default properties for the bucket. Ignored if an <code>existingBucketObj</code> is provided.</td>
</tr>
<tr>
<td><code>logGroupProps?</code></td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
<tr>
<td><code>loggingBucketProps?</code></td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td><code>logS3AccessLogs?</code></td>
<td><code>boolean</code></td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is true.</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesisAnalytics</td>
<td>kinesisAnalytics.CfnApplication</td>
<td>Returns an instance of the Kinesis Analytics application created by the pattern.</td>
</tr>
<tr>
<td>kinesisFirehose</td>
<td>kinesisfirehose.CfnDeliveryStream</td>
<td>Returns an instance of the Kinesis Firehose delivery stream created by the pattern.</td>
</tr>
<tr>
<td>kinesisFirehoseLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group created by the pattern that Kinesis Firehose access logs are sent to.</td>
</tr>
<tr>
<td>kinesisFirehoseRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the Kinesis Firehose delivery stream.</td>
</tr>
<tr>
<td>s3Bucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
<tr>
<td>s3BucketInterface</td>
<td>s3.IBucket</td>
<td>Returns an instance of s3.IBucket created by the construct.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon Kinesis Firehose**
- Enable CloudWatch logging for Kinesis Firehose
- Configure least privilege access IAM role for Amazon Kinesis Firehose

**Amazon S3 Bucket**
- Configure Access logging for S3 Bucket
- Enable server-side encryption for S3 Bucket using AWS managed KMS Key
- Turn on the versioning for S3 Bucket
- Don't allow public access for S3 Bucket
- Retain the S3 Bucket when deleting the CloudFormation stack
- Enforce encryption of data in transit
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days
Amazon Kinesis Data Analytics

- Configure least privilege access IAM role for Amazon Kinesis Analytics

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-kinesisfirehose-s3-and-kinesisanalytics

aws-kinesisstreams-gluejob

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct deploys an Amazon Kinesis Data Stream, and configures an AWS Glue Job to perform custom ETL transformation with the appropriate resources/properties for interaction and security. It also creates an Amazon S3 bucket where the Python script for the AWS Glue Job can be uploaded.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import * as glue from '@aws-cdk/aws-glue';
import * as s3assets from '@aws-cdk/aws-s3-assets';
import { KinesisstreamsToGluejob } from '@aws-solutions-constructs/aws-kinesisstreams-gluejob';

const fieldSchema: glue.CfnTable.ColumnProperty[] = [
  
  {
    name: 'id',
    type: 'int',
    comment: 'Identifier for the record',
  },

  {
    name: 'name',
    type: 'string',
    comment: 'Name for the record',
  },

  {
    name: 'address',
    type: 'string',
    comment: 'Address for the record',
  },

  {
    name: 'value',
    type: 'int',
    comment: 'Value for the record',
  },

];

const customEtlJob = new KinesisstreamsToGluejob(this, 'CustomETL', {
  glueJobProps: {
    command: {
      name: 'gluestreaming',
      pythonVersion: '3',
      scriptLocation: new s3assets.Asset(this, 'ScriptLocation', {
        path: `${__dirname}/../etl/transform.py`,
      }).s3ObjectUrl,
    },
  }
});
```
new KinesisstreamsToGluejob(scope: Construct, id: string, props: KinesisstreamsToGluejobProps);

**Parameters**
- **scope** `Construct`
- **id** `string`
- **props** `KinesisstreamsToGluejobProps (p. 166)`

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesisStreamProps?</td>
<td><code>kinesis.StreamProps</code></td>
<td>Optional user-provided props to override the default props for the Amazon Kinesis Data Stream.</td>
</tr>
<tr>
<td>existingStreamObj?</td>
<td><code>kinesis.Stream</code></td>
<td>Existing instance of Kinesis Stream, providing both this and kinesisStreamProps will cause an error.</td>
</tr>
<tr>
<td>glueJobProps?</td>
<td><code>cfnJob.CfnJobProps</code></td>
<td>User-provided props to override the default props for the AWS Glue job.</td>
</tr>
<tr>
<td>existingGlueJob?</td>
<td><code>cfnJob.CfnJob</code></td>
<td>Existing instance of AWS Glue Job, providing both this and glueJobProps will cause an error.</td>
</tr>
<tr>
<td>existingDatabase?</td>
<td><code>CfnDatabase</code></td>
<td>Existing AWS Glue database to be used with this construct. If this is set, then databaseProps is ignored.</td>
</tr>
<tr>
<td>databaseProps?</td>
<td><code>CfnDatabaseProps</code></td>
<td>User-provided props to override the default props used to create the AWS Glue database.</td>
</tr>
<tr>
<td>existingTable?</td>
<td><code>CfnTable</code></td>
<td>Existing instance of AWS Glue table. If this is set, then tableProps and fieldSchema are ignored.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tableProps?</td>
<td>CfnTableProps</td>
<td>User-provided props to override default props used to create an AWS Glue table.</td>
</tr>
<tr>
<td>fieldSchema?</td>
<td>CfnTable.ColumnProperty[]</td>
<td>User-provided schema structure to create an AWS Glue table.</td>
</tr>
<tr>
<td>outputDataStore?</td>
<td>SinkDataStoreProps</td>
<td>User-provided props for an Amazon S3 bucket that stores output from the AWS Glue job. Currently only supports Amazon S3 as the output datastore type.</td>
</tr>
<tr>
<td>createCloudWatchAlarms?</td>
<td>boolean</td>
<td>Whether to create recommended CloudWatch alarms for Kinesis Data Stream. Default value is set to true.</td>
</tr>
</tbody>
</table>

**SinkDataStoreProps**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingS3OutputBucket?</td>
<td>Bucket</td>
<td>Existing instance of S3 bucket where the data should be written. Providing both this and outputBucketProps will cause an error.</td>
</tr>
<tr>
<td>outputBucketProps</td>
<td>BucketProps</td>
<td>User-provided bucket properties to create the Amazon S3 bucket used to store the output from the AWS Glue job.</td>
</tr>
<tr>
<td>datastoreType</td>
<td>SinkStoreType</td>
<td>Sink data store type.</td>
</tr>
</tbody>
</table>

**SinkStoreType**

Enumeration of data store types that could include S3, DynamoDB, DocumentDB, RDS or Redshift. Current construct implementation only supports S3, but potential to add other output types in the future.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3</td>
<td>string</td>
<td>S3 storage type</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns an array of recommended CloudWatch Alarms created by the construct for Kinesis Data stream</td>
</tr>
<tr>
<td>glueJob</td>
<td>CfnJob</td>
<td>Returns an instance of AWS Glue Job created by the construct</td>
</tr>
<tr>
<td>glueJobRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM Role created by the construct for the Glue Job</td>
</tr>
<tr>
<td>database</td>
<td>CfnDatabase</td>
<td>Returns an instance of AWS Glue Database created by the construct</td>
</tr>
<tr>
<td>table</td>
<td>CfnTable</td>
<td>Returns an instance of the AWS Glue Table created by the construct</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon Kinesis Stream

- Configure least privilege access IAM role for the Amazon Kinesis Data Stream.
- Enable server-side encryption for the Amazon Kinesis Stream using an AWS Managed KMS Key.
- Deploy best-practice Amazon CloudWatch Alarms for the Amazon Kinesis Stream.

Glue Job

- Create an AWS Glue security configuration that configures encryption for CloudWatch, Job Bookmarks, and S3. CloudWatch and Job Bookmarks are encrypted using AWS Managed KMS Key created for AWS Glue Service. The S3 bucket is configured with SSE-S3 encryption mode.
- Configure service role policies that allow AWS Glue to read from Amazon Kinesis Data Streams.

Glue Database

- Create an AWS Glue database. An AWS Glue table will be added to the database. This table defines the schema for the records buffered in the Amazon Kinesis Data Stream.

Glue Table

- Create an AWS Glue table. The table schema definition is based on the JSON structure of the records buffered in the Amazon Kinesis Data Stream.
IAM Role

- A job execution role that has privileges to 1) read the ETL script from the Amazon S3 bucket location, 2) read records from the Amazon Kinesis Data Stream, and 3) execute the Amazon Glue job.

Output S3 Bucket

- An Amazon S3 bucket to store the output of the ETL transformation. This bucket will be passed as an argument to the created AWS Glue job so that it can be used in the ETL script to write data into it.

Cloudwatch Alarms

- A CloudWatch Alarm to report when consumer application is reading data slower than expected.

  A CloudWatch Alarm to report when consumer record processing is falling behind (to avoid risk of data loss due to record expiration).

Architecture

A sample use case which uses this pattern is available under use_cases/aws-custom-glue-etl.
GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-kinesisstreams-gluejob

aws-kinesisstreams-kinesisfirehose-s3

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_kinesisstreams_kinesisfirehose_s3</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-kinesisstreams-kinesis-firehose-s3</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.kinesisstreams.kinesisfirehose_s3</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements an Amazon Kinesis Data Stream (KDS) connected to Amazon Kinesis Data Firehose (KDF) delivery stream connected to an Amazon S3 bucket.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { KinesisStreamsToKinesisFirehoseToS3 } from '@aws-solutions-constructs/aws-kinesisstreams-kinesisfirehose-s3';
new KinesisStreamsToKinesisFirehoseToS3(this, 'test-stream-firehose-s3', {});
```

Initializer
new KinesisStreamsToKinesisFirehoseToS3(scope: Construct, id: string, props: KinesisStreams...ToS3Props);

**Parameters**

- **scope** `Construct`
- **id** `string`
- **props** `KinesisStreams...ToS3Props` (p. 171)

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the S3 Bucket.</td>
</tr>
<tr>
<td>createCloudWatchAlarms?</td>
<td><code>boolean</code></td>
<td>Optional whether to create recommended CloudWatch alarms.</td>
</tr>
<tr>
<td>existingBucketObj?</td>
<td><code>s3.IBucket</code></td>
<td>Optional existing instance of S3 Bucket object. If this is provided, then also providing bucketProps is an error.</td>
</tr>
<tr>
<td>existingLoggingBucketObj?</td>
<td><code>s3.IBucket</code></td>
<td>Optional existing instance of logging S3 Bucket object for the S3 Bucket created by the pattern.</td>
</tr>
<tr>
<td>existingStreamObj?</td>
<td><code>kinesis.Stream</code></td>
<td>Existing instance of Kinesis Stream, providing both this and kinesisStreamProps will cause an error.</td>
</tr>
<tr>
<td>kinesisFirehoseProps?</td>
<td>`aws-kinesisfirehose.CfnDeliveryStreamProps</td>
<td>Optional user provided props to override the default props for Kinesis Firehose Delivery Stream.</td>
</tr>
<tr>
<td>kinesisStreamProps?</td>
<td><code>kinesis.StreamProps</code></td>
<td>Optional user provided props to override the default props for the Kinesis stream.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user provided props to override the default props for the CloudWatchLogs Log Group.</td>
</tr>
<tr>
<td>loggingBucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td>logS3AccessLogs?</td>
<td><code>boolean</code></td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudwatchAlarms?</td>
<td><code>cloudwatch.Alarm[]</code></td>
<td>Returns a list of cloudwatch.Alarm instances created by the construct.</td>
</tr>
<tr>
<td>kinesisFirehose</td>
<td><code>kinesisfirehose.CfnDeliveryStream</code></td>
<td>Returns an instance of kinesisfirehose.CfnDeliveryStream created by the construct.</td>
</tr>
<tr>
<td>kinesisFirehoseLogGroup</td>
<td><code>logs.LogGroup</code></td>
<td>Returns an instance of the logs.LogGroup created by the construct for Kinesis Data Firehose delivery stream.</td>
</tr>
<tr>
<td>kinesisFirehoseRole</td>
<td><code>iam.Role</code></td>
<td>Returns an instance of the iam.Role created by the construct for Kinesis Data Firehose delivery stream.</td>
</tr>
<tr>
<td>kinesisStreamRole</td>
<td><code>iam.Role</code></td>
<td>Returns an instance of the iam.Role created by the construct for Kinesis stream.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td><code>s3.Bucket</code></td>
<td>Returns an instance of s3.Bucket created by the construct as the logging bucket for the primary bucket.</td>
</tr>
<tr>
<td>s3BucketInterface</td>
<td><code>s3.IBucket</code></td>
<td>Returns an instance of s3.IBucket created by the construct.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon Kinesis Stream

- Configure least privilege access IAM role for Kinesis Stream
- Enable server-side encryption for Kinesis Stream using AWS Managed KMS Key
- Deploy best practices CloudWatch Alarms for the Kinesis Stream
Amazon Kinesis Firehose

- Enable CloudWatch logging for Kinesis Firehose
- Configure least privilege access IAM role for Amazon Kinesis Firehose

Amazon S3 Bucket

- Configure access logging for S3 bucket
- Enable server-side encryption for S3 bucket using AWS managed KMS Key
- Enforce encryption of data in transit
- Enable bucket versioning
- Don't allow public access for S3 bucket
- Retain the S3 Bucket when deleting the CloudFormation stack
- Apply lifecycle rule to move noncurrent object versions to Glacier storage after 90 days

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-kinesisstreams-kinesisfirehose-s3

aws-kinesisstreams-lambda
**Overview**

This AWS Solutions Construct deploys a Kinesis Stream and Lambda function with the appropriate resources/properties for interaction and security.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { KinesisStreamsToLambda } from '@aws-solutions-constructs/aws-kinesis-streams-lambda';

new KinesisStreamsToLambda(this, 'KinesisToLambdaPattern', {
  kinesisEventSourceProps: {
    startingPosition: lambda.StartingPosition.TRIM_HORIZON,
    batchSize: 1
  },
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`#{__dirname}/lambda`),
    handler: 'index.handler'
  }
});
```

**Initializer**

```typescript
new KinesisStreamsToLambda(scope: Construct, id: string, props: KinesisStreamsToLambdaProps);
```

**Parameters**

- `scope` *Construct*
- `id` *string*
- `props` *KinesisStreamsToLambdaProps (p. 175)*

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws-kinesis-streams-lambda</td>
</tr>
<tr>
<td>TypeScript</td>
<td>@aws-solutions-constructs/aws-kinesisstreams-lambda</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsservices.kinesisstreamsslambdas</td>
</tr>
</tbody>
</table>
# Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>kinesisStreamProps?</td>
<td>kinesis.StreamProps</td>
<td>Optional user-provided props to override the default props for the Kinesis stream.</td>
</tr>
<tr>
<td>existingStreamObj?</td>
<td>kinesis.Stream</td>
<td>Existing instance of Kinesis Stream, providing both this and kinesisStreamProps will cause an error.</td>
</tr>
<tr>
<td>kinesisEventSourceProps?</td>
<td>aws-lambda-event-sources.KinesisEventSourceProps</td>
<td>Optional user-provided props to override the default props for the Lambda event source mapping.</td>
</tr>
<tr>
<td>createCloudWatchAlarms?</td>
<td>boolean</td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
</tbody>
</table>

# Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesisStream</td>
<td>kinesis.Stream</td>
<td>Returns an instance of the Kinesis stream created by the pattern.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>kinesisStreamRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the Kinesis stream.</td>
</tr>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns a list of one or more CloudWatch alarms created by the pattern.</td>
</tr>
</tbody>
</table>
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon Kinesis Stream

- Configure least privilege access IAM role for Kinesis Stream.
- Enable server-side encryption for Kinesis Stream using AWS managed KMS Key.
- Deploy best-practice CloudWatch Alarms for the Kinesis Stream.

AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Enable Failure-Handling features: enable bisect on function Error; set default Maximum Record Age (24 hours); set default Maximum Retry Attempts (500); and deploy SQS dead-letter queue as destination on failure.
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

Architecture

![Architecture Diagram]

- Amazon CloudWatch
- Amazon Kinesis
- AWS Lambda
- Role
GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-kinesisstreams-lambda

aws-lambda-dynamodb

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_lambda_dynamodb</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-lambda-dynamodb</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.lambdadynamodb</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements the AWS Lambda function and Amazon DynamoDB table with least-privilege permissions.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { LambdaToDynamoDBProps, LambdaToDynamoDB } from '@aws-solutions-constructs/aws-lambda-dynamodb';

const props: LambdaToDynamoDBProps = {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
};

new LambdaToDynamoDB(this, 'test-lambda-dynamodb-stack', props);
```
**Initializer**

```typescript
new LambdaToDynamoDB(scope: Construct, id: string, props: LambdaToDynamoDBProps);
```

**Parameters**

- **scope** *Construct*
- **id** *string*
- **props** *LambdaToDynamoDBProps (p. 178)*

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and <code>lambdaFunctionProps</code> will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>dynamoTableProps?</td>
<td><code>dynamodb.TableProps</code></td>
<td>Optional user provided props to override the default props for DynamoDB Table.</td>
</tr>
<tr>
<td>existingTableObj?</td>
<td><code>dynamodb.Table</code></td>
<td>Existing instance of DynamoDB table object, providing both this and <code>dynamoTableProps</code> will cause an error.</td>
</tr>
<tr>
<td>tablePermissions?</td>
<td><code>string</code></td>
<td>Optional table permissions to be granted to the Lambda function. One of the following options may be specified: All, Read, ReadWrite, or Write.</td>
</tr>
<tr>
<td>tableEnvironmentVariableName?</td>
<td><code>string</code></td>
<td>Optional name for the DynamoDB table environment variable set for the Lambda function.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td><code>ec2.IVpc</code></td>
<td>An optional, existing VPC into which this pattern should be deployed. When deployed in a VPC, the Lambda function will use ENIs in the VPC to access network resources and a Gateway Endpoint will be</td>
</tr>
</tbody>
</table>
### AWS Solutions Constructs AWS Solutions

#### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>created in the VPC for Amazon DynamoDB. If an existing VPC is provided, the deployVpc property cannot be true. This uses ec2.IVpc to allow clients to supply VPCs that exist outside the stack using the ec2.Vpc.fromLookup() method.</td>
</tr>
<tr>
<td>vpcProps?</td>
<td>ec2.VpcProps</td>
<td>Optional user-provided properties to override the default properties for the new VPC. enableDnsHostnames, enableDnsSupport, natGateways, and subnetConfiguration are set by the pattern, so any values for those properties supplied here will be overrriden. If deployVpc is not true then this property will be ignored.</td>
</tr>
</tbody>
</table>
| deployVpc?    | boolean            | Whether to create a new VPC based on vpcProps into which to deploy this pattern. Setting this to true will deploy the minimal, most private VPC to run the pattern:  
• One isolated subnet in each Availability Zone used by the CDK program  
• enableDnsHostnames and enableDnsSupport will both be set to true  
If this property is true, then existingVpc cannot be specified. Defaults to false. |

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamoTable</td>
<td>dynamodb.Table</td>
<td>Returns an instance of the DynamoDB table created by the pattern.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>vpc?</td>
<td>ec2.IVpc</td>
<td>Returns an interface on the VPC used by the pattern (if any). This may be a VPC created by the pattern or the VPC supplied to the pattern constructor.</td>
</tr>
</tbody>
</table>

**Default settings**

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**AWS Lambda Function**
- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Set environment variables:
  - `DDB_TABLE_NAME` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**Amazon DynamoDB Table**
- Set the billing mode for DynamoDB Table to On-Demand (Pay per request).
- Enable server-side encryption for DynamoDB Table using AWS managed KMS Key.
- Creates a partition key called ‘id’ for DynamoDB Table.
- Retain the Table when deleting the CloudFormation stack.
- Enable continuous backups and point-in-time recovery.
Architecture

Role

Lambda function

Amazon DynamoDB

Amazon CloudWatch

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-lambda-dynamodb

aws-lambda-elasticsearch-kibana

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an AWS Lambda function and an Amazon Elasticsearch Service domain with least-privileged permissions.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { LambdaToElasticSearchAndKibana } from '@aws-solutions-constructs/aws-lambda-elasticsearch-kibana';
import { Aws } from '@aws-cdk/core';

const lambdaProps: lambda.FunctionProps = {
  runtime: lambda.Runtime.NODEJS_14_X,
  // This assumes a handler function in lib/lambda/index.js
  code: lambda.Code.fromAsset('${__dirname}/lambda'),
  handler: 'index.handler'
};

new LambdaToElasticSearchAndKibana(this, 'test-lambda-elasticsearch-kibana', {
  lambdaFunctionProps: lambdaProps,
  domainName: 'test-domain',
  // TODO: Ensure the Cognito domain name is globally unique
  cognitoDomainName: 'globallyuniquedomain' + Aws.ACCOUNT_ID;
});
```

Initializer

```typescript
new LambdaToElasticSearchAndKibana(scope: Construct, id: string, props: LambdaToElasticSearchAndKibanaProps);
```

Parameters

- **scope** `Construct`
- **id** `string`
- **props** `LambdaToElasticSearchAndKibanaProps (p. 183)`
## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>esDomainProps?</td>
<td>elasticsearch.CfnDomainProps</td>
<td>Optional user provided props to override the default props for the Amazon OpenSearch Service</td>
</tr>
<tr>
<td>domainName</td>
<td>string</td>
<td>Domain name for the Cognito and the Amazon OpenSearch Service</td>
</tr>
<tr>
<td>cognitoDomainName?</td>
<td>string</td>
<td>Optional Cognito domain name. If provided, it will be used for the Cognito domain, and domainName will be used for the Elasticsearch domain.</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td>boolean</td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
<tr>
<td>domainEndpointEnvironmentVariableName?</td>
<td>string</td>
<td>Optional name for the ElasticSearch domain endpoint environment variable set for the Lambda function.</td>
</tr>
</tbody>
</table>

## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns a list of one or more CloudWatch alarms created by the pattern.</td>
</tr>
<tr>
<td>elasticsearchDomain</td>
<td>elasticsearch.CfnDomain</td>
<td>Returns an instance of the Elasticsearch domain created by the pattern.</td>
</tr>
<tr>
<td>elasticsearchDomainRole</td>
<td>iam.Role</td>
<td>Returns an instance of the IAM role created by the pattern for the Elasticsearch domain.</td>
</tr>
</tbody>
</table>
### Lambda function

This pattern requires a Lambda function that can post data into the Elasticsearch service from the DynamoDB stream. A sample function is provided here.

### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

#### AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Set environment variables:
  - `DOMAIN_ENDPOINT` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

#### Amazon Cognito

- Set password policy for User Pools.
- Enforce the advanced security mode for User Pools.

#### Amazon OpenSearch Service

- Deploy best practices CloudWatch Alarms for the Elasticsearch domain.
- Secure the Kibana dashboard access with Cognito User Pools.
- Enable server-side encryption for Elasticsearch domain using AWS managed KMS Key.
- Enable node-to-node encryption for Elasticsearch domain.
- Configure the cluster for the OpenSearch Service domain.
Architecture

Amazon Cognito

Amazon Elasticsearch Service

AWS Lambda

Role

Amazon CloudWatch

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-lambda-elasticsearch-kibana

aws-lambda-eventbridge

STABILITY EXPERIMENTAL
All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_lambda_eventbridge</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-lambda-eventbridge</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.lambdaeventbridge</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an AWS Lambda function connected to Amazon EventBridge.

Here is a minimal deployable pattern definition:

```javascript
import { LambdaToEventbridge, LambdaToEventbridgeProps } from "@aws-solutions-constructs/aws-lambda-eventbridge";

new LambdaToEventbridge(this, 'LambdaToEventbridgePattern', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`${__dirname}/lambda`)
  }
});
```

**Initializer**

```javascript
new LambdaToEventbridge(scope: Construct, id: string, props: LambdaToEventbridgeProps);
```

**Parameters**

- `scope` **Construct**
- `id` **string**
- `props` **LambdaToEventbridgeProps** (p. 187)
## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>existingEventBusInterface?</td>
<td>events.IEventBus</td>
<td>Optional user-provided custom EventBus for construct to use. Providing both this and eventBusProps results an error.</td>
</tr>
<tr>
<td>eventBusProps?</td>
<td>events.EventBusProps</td>
<td>Optional user-provided properties to override the default properties when creating a custom EventBus. Setting this value to {} will create a custom EventBus using all default properties. If neither this nor existingEventBusInterface is provided the construct will use the default EventBus. Providing both this and existingEventBusInterface results in an error.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td>ec2.IVpc</td>
<td>An optional, existing VPC into which this pattern should be deployed. When deployed in a VPC, the Lambda function will use ENIs in the VPC to access network resources and a Gateway Endpoint will be created in the VPC for Amazon DynamoDB. If an existing VPC is provided, the deployVpc property cannot be true. This uses <code>ec2.IVpc</code> to allow clients to supply VPCs that exist outside the stack using the <code>ec2.Vpc.fromLookup()</code> method.</td>
</tr>
<tr>
<td>vpcProps?</td>
<td>ec2.VpcProps</td>
<td>Optional user-provided properties to override the default properties for the new</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdaFunction</td>
<td><code>lambda.Function</code></td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>eventBus?</td>
<td><code>events.IEventBus</code></td>
<td>Returns an instance of the Event bus created by the pattern.</td>
</tr>
<tr>
<td>vpc?</td>
<td><code>ec2.IVpc</code></td>
<td>Returns an interface on the VPC used by the pattern (if any). This may be a VPC created by the pattern or the VPC supplied to the pattern constructor.</td>
</tr>
</tbody>
</table>
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**AWS Lambda Function**

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Allow the function to put events to EventBus (custom EventBus can be used by specifying `existingEventBusInterface` or `eventBusProps` property).
- Enable X-Ray tracing.
- Set environment variables:
  - `EVENTBUS_NAME`
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**Architecture**
GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-lambda-eventbridge

aws-lambda-s3

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_lambda_s3</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-lambda-s3</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.lambda3</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements an AWS Lambda function connected to an Amazon S3 bucket.

Here is a minimal deployable pattern definition in TypeScript:

```javascript
import { LambdaToS3 } from '@aws-solutions-constructs/aws-lambda-s3';

new LambdaToS3(this, 'LambdaToS3Pattern', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
code: lambda.Code.fromAsset(`#{dirname}/lambda`),
    handler: 'index.handler'
  }
});
```
**Initializer**

```typescript
new LambdaToS3(scope: Construct, id: string, props: LambdaToS3Props);
```

**Parameters**

- `scope` *Construct*
- `id` *string*
- `props` *LambdaToS3Props* (p. 191)

---

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and <code>lambdaFunctionProps</code> will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user provided props to override the default props for the Lambda function.</td>
</tr>
<tr>
<td>existingBucketObj?</td>
<td><code>s3.IBucket</code></td>
<td>Existing instance of S3 Bucket object. If this is provided, then also providing <code>bucketProps</code> is an error.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the S3 Bucket.</td>
</tr>
<tr>
<td>bucketPermissions?</td>
<td><code>string[]</code></td>
<td>Optional bucket permissions to grant to the Lambda function. One or more of the following may be specified: Delete, Put, Read, ReadWrite, Write.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td><code>ec2.IVpc</code></td>
<td>An optional, existing VPC into which this pattern should be deployed. When deployed in a VPC, the Lambda function will use ENIs in the VPC to access network resources and an Interface Endpoint will be created in the VPC for Amazon SQS. If an existing VPC is provided, the <code>deployVpc</code> property cannot be true. This uses <code>ec2.IVpc</code> to allow clients to supply VPCs that exist outside the stack using</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| deployVpc?           | boolean             | Whether to create a new VPC based on vpcProps into which to deploy this pattern. Setting this to true will deploy the minimal, most private VPC to run the pattern:  
  - One isolated subnet in each Availability Zone used by the CDK program.  
  - enableDnsHostnames and enableDnsSupport will both be set to true.  
  If this property is true, then existingVpc cannot be specified. Defaults to false. |
| vpcProps?            | ec2.VpcProps        | Optional user provided properties to override the default properties for the new VPC. enableDnsHostnames, enableDnsSupport, natGateways and subnetConfiguration are set by the pattern, so any values for those properties supplied here will be overridden. If deployVpc is not true then this property will be ignored. |
| bucketEnvironmentVariableName? | string            | Optional name for the S3 bucket environment variable set for the Lambda function.                                                               |
| loggingBucketProps?  | s3.BucketProps      | Optional user provided props to override the default props for the S3 Logging Bucket.                                                        |
| logS3AccessLogs?     | boolean             | Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is true. |
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>s3Bucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
<tr>
<td>vpc?</td>
<td>ec2.IVpc</td>
<td>Returns an instance of the VPC used by the pattern (if any). This may be a VPC created by the pattern or the VPC supplied to the pattern constructor.</td>
</tr>
<tr>
<td>s3BucketInterface</td>
<td>s3.IBucket</td>
<td>Returns an instance of s3.IBucket created by the construct.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**AWS Lambda Function**

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing
- Set environment variables:
  - S3_BUCKET_NAME (default)
  - AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)

**Amazon S3 Bucket**

- Configure Access logging for S3 Bucket.
- Enable server-side encryption for S3 Bucket using AWS managed KMS Key.
- Turn on the versioning for S3 Bucket.
- Don't allow public access for S3 Bucket.
- Retain the S3 Bucket when deleting the CloudFormation stack.
- Enforce encryption of data in transit.
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days.
Architecture

AWS Lambda Function

Role

Amazon S3 Bucket

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-lambda-s3

aws-lambda-sagemakerendpoint

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_lambda_sagemakerendpoint</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-lambda-sagemakerendpoint</td>
</tr>
<tr>
<td></td>
<td>software.amazon.awsconstructs.services.lambdasagemakerendpoint</td>
</tr>
</tbody>
</table>

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an AWS Lambda function connected to an Amazon Sagemaker Endpoint.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { Duration } from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import {
  LambdaToSagemakerEndpoint,
  LambdaToSagemakerEndpointProps,
} from '@aws-solutions-constructs/aws-lambda-sagemakerendpoint';

const constructProps: LambdaToSagemakerEndpointProps = {
  modelProps: {
    primaryContainer: {
      image: '{{AccountId}}.dkr.ecr.{{region}}.amazonaws.com/linear-learner:latest',
      modelDataUrl: 's3://{{bucket-name}}/{{prefix}}/model.tar.gz',
    },
  },
  lambdaFunctionProps: {
    runtime: lambda.Runtime.PYTHON_3_8,
    // This assumes a handler function in lib/lambda/index.py
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler',
    timeout: Duration.minutes(5),
    memorySize: 128,
  },
};

new LambdaToSagemakerEndpoint(this, 'LambdaToSagemakerEndpointPattern', constructProps);
```

Initializer

new LambdaToSagemakerEndpoint(scope: Construct, id: string, props: LambdaToSagemakerEndpointProps);

Parameters

- **scope** Construct
- **id** string
- **props** LambdaToSagemakerEndpointProps (p. 196)
## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function.</td>
</tr>
<tr>
<td>existingSagemakerEndpointObj?</td>
<td>sagemaker.CfnEndpoint</td>
<td>An optional, existing Sagemaker Endpoint to be used. Providing both this and endpointProps will cause an error.</td>
</tr>
<tr>
<td>modelProps?</td>
<td>sagemaker.CfnModelProps</td>
<td>any                                                                                                                                          User-provided properties to override the default properties for the Sagemaker Model. At least modelProps.primaryContainer must be provided to create a model. By default, the pattern will create a role with the minimum required permissions, but the client can provide a custom role with additional capabilities using modelProps.executionRoleArn.</td>
</tr>
<tr>
<td>endpointConfigProps?</td>
<td>sagemaker.CfnEndpointConfigProps</td>
<td>Optional user-provided properties to override the default properties for the Sagemaker Endpoint configuration.</td>
</tr>
<tr>
<td>endpointProps?</td>
<td>sagemaker.CfnEndpointProps</td>
<td>Optional user-provided properties to override the default properties for the Sagemaker Endpoint.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td>ec2.IVpc</td>
<td>An optional, existing VPC into which this construct should be deployed. When deployed in a VPC, the Lambda function and Sagemaker Endpoint will use ENIs in the VPC to access network resources. An Interface Endpoint will be created in the VPC for Amazon Sagemaker Runtime, and Amazon S3 VPC Endpoint. If an existing VPC</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpcProps?</td>
<td><code>ec2.VpcProps</code></td>
<td>Optional user-provided properties to override the default properties for the new VPC. enableDnsHostnames, enableDnsSupport, natGateways and subnetConfiguration are set by the construct, so any values for those properties supplied here will be overridden. If deployVpc is not true, then this property will be ignored.</td>
</tr>
</tbody>
</table>
| deployVpc?                        | `boolean`                     | Whether to create a new VPC based on vpcProps into which to deploy this pattern. Setting this to true will deploy the minimal, most private VPC to run the pattern:  
  > • One isolated subnet in each Availability Zone used by the CDK program.  
  > • enableDnsHostnames and enableDnsSupport will both be set to true.  
  
  If this property is set to true, then existingVpc cannot be specified. Defaults to false. |
| sagemakerEnvironmentVariableName? | `string`                      | Optional name for the SageMaker endpoint environment variable set for the Lambda function.                                                                                                                    |
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Allow the function to invoke the Sagemaker endpoint for Inferences.
- Configure the function to access resources in the VPC, where the Sagemaker endpoint is deployed.
- Enable X-Ray Tracing.
- Set environment variables:
  - SAGEMAKER_ENDPOINT_NAME (default)
  - AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)

Amazon Sagemaker Endpoint

- Configure limited privilege to create Sagemaker resources.
- Deploy Sagemaker model, endpointConfig, and endpoint.
- Configure the Sagemaker endpoint to be deployed in a VPC.
- Deploy S3 VPC Endpoint and Sagemaker Runtime VPC Interface.
### Architecture

![Architecture Diagram](attachment://architecture-diagram.png)

Amazon CloudWatch → AWS Lambda → Amazon SageMaker Endpoint → IAM Role

### GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

[@aws-solutions-constructs/aws-lambda-secretsmanager](https://github.com/aws-solutions-constructs/aws-lambda-secretsmanager)

### `aws-lambda-secretsmanager`

<table>
<thead>
<tr>
<th>Stability</th>
<th>EXPERIMENTAL</th>
</tr>
</thead>
</table>

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the [Semantic Versioning](https://semver.org/) model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.
Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_lambda_secretsmanager</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-lambda-secretsmanager</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.lambda.secretsmanager</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements the AWS Lambda function and AWS Secrets Manager secret with the least privileged permissions.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
const { LambdaToSecretsmanagerProps, LambdaToSecretsmanager } from '@aws-solutions-constructs/aws-lambda-secretsmanager';

const props: LambdaToSecretsmanagerProps = {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler',
    },
};

new LambdaToSecretsmanager(this, 'test-lambda-secretsmanager-stack', props);
```

Initializer

```typescript
new LambdaToSecretsmanager(scope: Construct, id: string, props: LambdaToSecretsmanagerProps);
```

Parameters

- `scope` Construct
- `id` string
- `props` LambdaToSecretsmanagerProps (p. 201)
# Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>User provided props to override the default props for the Lambda function.</td>
</tr>
<tr>
<td>secretProps?</td>
<td>secretsmanager.SecretProps</td>
<td>Optional user provided props to override the default props for Secrets Manager.</td>
</tr>
<tr>
<td>existingSecretObj?</td>
<td>secretsmanager.Secret</td>
<td>Existing instance of Secrets Manager secret object, If this is set then the secretProps is ignored.</td>
</tr>
<tr>
<td>grantWriteAccess?</td>
<td>boolean</td>
<td>Optional write access to the secret for the Lambda function (Read-Only by default).</td>
</tr>
<tr>
<td>secretEnvironmentVariableName?</td>
<td>string</td>
<td>Optional name for the Secrets Manager secret environment variable set for the Lambda function.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td>ec2.IVpc</td>
<td>An optional, existing VPC into which this pattern should be deployed. When deployed in a VPC, the Lambda function will use ENIs in the VPC to access network resources and an Interface Endpoint will be created in the VPC for AWS Secrets Manager. If an existing VPC is provided, the deployVpc property cannot be true. This uses ec2.IVpc to allow clients to supply VPCs that exist outside the stack using the ec2.Vpc.fromLookup() method.</td>
</tr>
<tr>
<td>vpcProps?</td>
<td>ec2.VpcProps</td>
<td>Optional user-provided properties to override the default properties for the new VPC. enableDnsHostnames, enableDnsSupport, natGateways, and subnetConfiguration are set by the pattern, so any values for</td>
</tr>
</tbody>
</table>
AWS Solutions Constructs AWS Solutions
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>those properties supplied here will be overridden. If deployVpc is not true then this property will be ignored.</td>
</tr>
</tbody>
</table>
| deployVpc?   | boolean          | Whether to create a new VPC based on vpcProps into which to deploy this pattern. Setting this to true will deploy the minimal, most private VPC to run the pattern:
|              |                  | • One isolated subnet in each Availability Zone used by the CDK program     |
|              |                  | • enableDnsHostnames and enableDnsSupport will both be set to true         |
|              |                  | If this property is true, then existingVpc cannot be specified. Defaults to false. |

**Pattern Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of lambda.Function created by the construct.</td>
</tr>
<tr>
<td>secret</td>
<td>secretsmanager.Secret</td>
<td>Returns an instance of secretsmanager.Secret created by the construct.</td>
</tr>
<tr>
<td>vpc?</td>
<td>ec2.IVpc</td>
<td>Returns an interface on the VPC used by the pattern (if any). This may be a VPC created by the pattern or the VPC supplied to the pattern constructor.</td>
</tr>
</tbody>
</table>

**Default settings**

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**AWS Lambda Function**

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Set environment variables:
- (default) SECRET ARN containing the ARN of the secret as return by CDK secretArn property
- AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)

**Amazon Secrets Manager Secret**

- Enable read-only access for the associated AWS Lambda Function
- Enable server-side encryption using a default KMS key for the account and region
- Creates a new secret:
  - (default) random name
  - (default) random value
- Retain the secret when deleting the CloudFormation stack

**Architecture**

```
Role

Lambda function

AWS Secrets Manager

Amazon CloudWatch
```

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-lambda-secretsmanager
aws-lambda-sns

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions Constructs.aws_lambda_sns</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-lambda-sns</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.lambdasns</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements an AWS Lambda function connected to an Amazon SNS topic.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { LambdaToSns, LambdaToSnsProps } from '@aws-solutions-constructs/aws-lambda-sns';

new LambdaToSns(this, 'test-lambda-sns', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`#{__dirname}/lambda`),
    handler: 'index.handler'
  }
});
```

Initializer

```typescript
new LambdaToSns(scope: Construct, id: string, props: LambdaToSnsProps);
```

Parameters

- `scope` Construct
- `id` string
- `props` LambdaToSnsProps (p. 205)
## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>existingTopicObj?</td>
<td><code>sns.Topic</code></td>
<td>Existing instance of SNS Topic object, providing both this and topicProps will cause an error.</td>
</tr>
<tr>
<td>topicProps?</td>
<td><code>sns.TopicProps</code></td>
<td>Optional user provided properties to override the default properties for the SNS topic.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td><code>ec2.IVpc</code></td>
<td>An optional, existing VPC into which this pattern should be deployed. When deployed in a VPC, the Lambda function will use ENIs in the VPC to access network resources and an Interface Endpoint will be created in the VPC for Amazon SQS. If an existing VPC is provided, the deployVpc property cannot be true. This uses ec2.IVpc to allow clients to supply VPCs that exist outside the stack using the <code>ec2.Vpc.fromLookup()</code> method.</td>
</tr>
</tbody>
</table>
| deployVpc?            | `boolean`           | Whether to create a new VPC based on vpcProps into which to deploy this pattern. Setting this to true will deploy the minimal, most private VPC to run the pattern:  
  - One isolated subnet in each Availability Zone used by the CDK program.  
  - enableDnsHostnames and enableDnsSupport will both be set to true. |
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpcProps?</td>
<td>ec2.VpcProps</td>
<td>Optional user-provided properties to override the default properties for the new VPC. enableDnsHostnames, enableDnsSupport, natGateways and subnetConfiguration are set by the pattern, so any values for those properties supplied here will be overridden. If deployVpc is not true then this property will be ignored.</td>
</tr>
<tr>
<td>topicArnEnvironmentVariableName?</td>
<td>string</td>
<td>Optional name for the SNS topic ARN environment variable set for the Lambda function.</td>
</tr>
<tr>
<td>topicNameEnvironmentVariableName?</td>
<td>string</td>
<td>Optional name for the SNS topic name environment variable set for the Lambda function.</td>
</tr>
</tbody>
</table>

### Default settings

Out of the box implementation of the Construct without any override will set the following defaults:

**AWS Lambda Function**

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Set environment variables:
- SNS_TOPIC_NAME (default)
- SNS_TOPIC_ARN (default)
- AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)

### Amazon SNS Topic

- Configure least privilege access permissions for SNS topic.
- Enable server-side encryption using AWS managed KMS key.
- Enforce encryption of data in transit.

### Architecture

![Architecture Diagram]

- AWS Lambda
- Role
- Amazon Simple Notification Service

### GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-lambda-sns

### aws-lambda-sqs

**CFN-RESOURCES**

**STABLE**

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_lambda_sqs</td>
</tr>
</tbody>
</table>

207
Overview

This AWS Solutions Construct implements an AWS Lambda function connected to an Amazon SQS queue.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { LambdaToSqs, LambdaToSqsProps } from '@aws-solutions-constructs/aws-lambda-sqs';

new LambdaToSqs(this, 'LambdaToSqsPattern', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler'
  }
});
```

Initializer

```typescript
new LambdaToSqs(scope: Construct, id: string, props: LambdaToSqsProps);
```

Parameters

- `scope` `Construct`
- `id` `string`
- `props` `LambdaToSqsProps` (p. 208)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>An optional, existing Lambda function to be used instead of the default function. Providing both this and</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the default properties for the Lambda function.</td>
</tr>
<tr>
<td>existingQueueObj?</td>
<td><code>sqs.Queue</code></td>
<td>An optional, existing SQS queue to be used instead of the default queue. Providing both this and queueProps will cause an error.</td>
</tr>
<tr>
<td>queueProps?</td>
<td><code>sqs.QueueProps</code></td>
<td>Optional user-provided properties to override the default properties for the SQS queue.</td>
</tr>
<tr>
<td>enableQueuePurging?</td>
<td><code>boolean</code></td>
<td>Whether to grant additional permissions to the Lambda function enabling it to purge the SQS queue. Defaults to <code>false</code>.</td>
</tr>
<tr>
<td>deployDeadLetterQueue?</td>
<td><code>boolean</code></td>
<td>Whether to create a secondary queue to be used as a dead letter queue. Defaults to <code>true</code>.</td>
</tr>
<tr>
<td>deadLetterQueueProps?</td>
<td><code>sqs.QueueProps</code></td>
<td>Optional user-provided props to override the default props for the dead letter queue. Only used if the deployDeadLetterQueue property is set to <code>true</code>.</td>
</tr>
<tr>
<td>maxReceiveCount?</td>
<td><code>number</code></td>
<td>The number of times a message can be unsuccessfully dequeued before being moved to the dead letter queue. Defaults to <code>15</code>.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td><code>ec2.IVpc</code></td>
<td>An optional, existing VPC into which this pattern should be deployed. When deployed into a VPC, the Lambda function will use ENIs in the VPC to access network resources and an Interface Endpoint will be created in the VPC for Amazon SQS. If an existing VPC is provided, the deployVpc property cannot be <code>true</code>. An <code>ec2.IVpc</code> is used to allow clients to supply VPCs that exist outside the stack using the <code>ec2.Vpc.fromLookup()</code> method.</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| deployVpc?                | boolean               | Whether to create a new VPC based on vpcProps into which to deploy this pattern. Setting this to `true` will deploy the minimal, most private VPC to run the pattern:  
  - One isolated subnet in each Availability Zone used by the CDK program  
  - `enableDnsHostnames` and `enableDnsSupport` will both be set to `true`  
  If this property is `true`, then `existingVpc` cannot be specified. Defaults to `false`. |
| vpcProps?                 | ec2.VpcProps          | Optional user-provided properties to override the default properties for the new VPC. `enableDnsHostnames`, `enableDnsSupport`, `natGateways`, and `subnetConfiguration` are set by the pattern, so any values for those properties supplied here will be overridden. If `deployVpc` is not `true`, then this property will be ignored. |
| queueEnvironmentVariableName? | string               | Optional name for the SQS queue URL environment variable set for the Lambda function.                                                                                                                      |
Default settings

Out of the box implementation of the Construct without any override will set the following defaults:

**AWS Lambda Function**

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Allow the function to only send messages to the queue (purging can be enabled using the `enableQueuePurge` property).
- Enable X-Ray tracing
- Set environment variables:
  - `SQS_QUEUE_URL`
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**Amazon SQS Queue**

- Deploy SQS dead-letter queue for the source SQS Queue.
- Enable server-side encryption for source SQS Queue using AWS Managed KMS Key.
- Enforce encryption of data in transit.
Architecture

Amazon CloudWatch

AWS Lambda
   Function

Role

Amazon Simple Queue Service
   Queue

Amazon Simple Queue Service
   DLQ
   (optional, enabled by default)

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-lambda-sqs
All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_lambda_sqs_lambda</td>
</tr>
<tr>
<td>TypeScript</td>
<td>@aws-solutions-constructs/aws-lambda-sqs-lambda</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.lambdasqslambda</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Constructs pattern implements (1) an AWS Lambda function that is configured to send messages to a queue; (2) an Amazon SQS queue; and (3) an AWS Lambda function configured to consume messages from the queue.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { LambdaToSqsToLambda, LambdaToSqsToLambdaProps } from '@aws-solutions-constructs/aws-lambda-sqs-lambda';

new LambdaToSqsToLambda(this, 'LambdaToSqsToLambdaPattern', {
  producerLambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`${__dirname}/lambda/producer-function`)
  },
  consumerLambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`${__dirname}/lambda/consumer-function`)
  }
});
```

**Initializer**

```typescript
new LambdaToSqsToLambda(scope: Construct, id: string, props: LambdaToSqsToLambdaProps);
```

**Parameters**
**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingProducerLambdaObj?</td>
<td>lambda.Function</td>
<td>An optional, existing Lambda function to be used instead of the default function for sending messages to the queue. Providing both this and producerLambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>producerLambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the producer Lambda function.</td>
</tr>
<tr>
<td>existingQueueObj?</td>
<td>sqs.Queue</td>
<td>An optional, existing SQS queue to be used instead of the default queue. Providing both this and queueProps will cause an error.</td>
</tr>
<tr>
<td>queueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user-provided properties to override the default properties for the SQS queue. Providing both this and existingQueueObj will cause an error.</td>
</tr>
<tr>
<td>deployDeadLetterQueue?</td>
<td>boolean</td>
<td>Whether to create a secondary queue to be used as a dead letter queue. Defaults to true.</td>
</tr>
<tr>
<td>deadLetterQueueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user-provided props to override the default props for the dead letter queue. Only used if the deployDeadLetterQueue property is set to true.</td>
</tr>
<tr>
<td>maxReceiveCount?</td>
<td>number</td>
<td>The number of times a message can be unsuccessfully dequeued before being moved to the dead letter queue. Defaults to 15.</td>
</tr>
<tr>
<td>existingConsumerLambdaObj?</td>
<td>lambda.Function</td>
<td>An optional, existing Lambda function to be used instead of the default function for receiving/consuming messages from the queue. Providing both this and</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>consumerLambdaFunctionProps</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the consumer Lambda function.</td>
</tr>
<tr>
<td>queueEnvironmentVariableName</td>
<td>string</td>
<td>Optional name for the SQS queue URL environment variable set for the producer Lambda function.</td>
</tr>
<tr>
<td>sqsEventSourceProps</td>
<td>SqsEventSourceProps</td>
<td>Optional user provided properties for the queue event source.</td>
</tr>
</tbody>
</table>

### Default settings

Out-of-the-box implementation of this Construct (without any overridden properties) will adhere to the following defaults:

#### AWS Lambda Functions

- Configure limited privilege access IAM role for Lambda functions.
- Enable reusing connections with Keep-Alive for NodeJs Lambda functions.
- Enable X-Ray tracing
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)
Amazon SQS Queue

- Deploy a dead letter queue for the primary queue.
- Enable server-side encryption for the primary queue using an AWS Managed KMS Key.
- Enforce encryption of data in transit

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

`@aws-solutions-constructs/aws-lambda-sqs-lambda`

aws-lambda-ssmstringparameter

STABILITY EXPERIMENTAL

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while
you may use them, you may need to update your source code when upgrading to a newer version of this package.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td><code>aws_solutions_constructs.aws_lambda_ssm_string_parameter</code></td>
</tr>
<tr>
<td>TypeScript</td>
<td><code>@aws-solutions-constructs/aws-lambda-ssmstringparameter</code></td>
</tr>
<tr>
<td>Java</td>
<td><code>software.amazon.awsconstructs.services.lambdassmstringparameter</code></td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements the AWS Lambda function and AWS Systems Manager Parameter Store String parameter with the least privileged permissions.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
const { LambdaToSsmstringparameterProps, LambdaToSsmstringparameter } from '@aws-solutions-constructs/aws-lambda-ssmstringparameter';

const props: LambdaToSsmstringparameterProps = {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler',
  },
  stringParameterProps: { stringValue: "test-string-value" }
};

new LambdaToSsmstringparameter(this, 'test-lambda-ssmstringparameter-stack', props);
```

**Initializer**

```typescript
new LambdaToSsmstringparameter(scope: Construct, id: string, props: LambdaToSsmstringparameterProps);
```

**Parameters**

- **scope** `Construct`
id: string
props: LambdaToSsmStringParameterProps (p. 218)

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>existingStringParameterObj?</td>
<td>ssm.StringParameter</td>
<td>Existing instance of SSM String parameter object, providing both this and stringParameterProps will cause an error.</td>
</tr>
<tr>
<td>stringParameterProps?</td>
<td>ssm.StringParameterProps</td>
<td>Optional user provided props to override the default props for SSM String parameter. If existingStringParameterObj is not set,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stringParameterProps is required. The only supported ssm.StringParameterProps.type is STRING if a different value is provided it will be overridden.</td>
</tr>
<tr>
<td>stringParameterEnvironmentVariableName?</td>
<td></td>
<td>Optional name for the SSM String parameter environment variable set for the Lambda function.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td>ec2.IVpc</td>
<td>An optional, existing VPC into which this pattern should be deployed. When deployed in a VPC, the Lambda function will use ENIs in the VPC to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>access network resources and an Interface Endpoint will be created in the VPC for AWS Systems Manager Parameter. If an existing VPC is provided,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the deployVpc property cannot be true. This uses ec2.IVpc to allow clients to supply VPCs that exist outside the stack using</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpcProps?</td>
<td><code>ec2.VpcProps</code></td>
<td>Optional user-provided properties to override the default properties for the new VPC. <code>enableDnsHostnames</code>, <code>enableDnsSupport</code>, <code>natGateways</code> and <code>subnetConfiguration</code> are set by the pattern, so any values for those properties supplied here will be overridden. If <code>deployVpc</code> is not <code>true</code> then this property will be ignored.</td>
</tr>
</tbody>
</table>
| deployVpc?                  | `boolean`                  | Whether to create a new VPC based on `vpcProps` into which to deploy this pattern. Setting this to `true` will deploy the minimal, most private VPC to run the pattern:  
- One isolated subnet in each Availability Zone used by the CDK program.  
- `enableDnsHostnames` and `enableDnsSupport` will both be set to `true`. |
| stringParameterPermissions? | `string`                    | Optional SSM String parameter permissions to grant to the Lambda function. One of the following may be specified: `Read`, `ReadWrite`. |

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdaFunction</td>
<td><code>lambda.Function</code></td>
<td>Returns an instance of <code>lambda.Function</code> created by the construct.</td>
</tr>
<tr>
<td>stringParameter</td>
<td><code>ssm.StringParameter</code></td>
<td>Returns an instance of <code>ssm.StringParameter</code> created by the construct.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>vpc?</td>
<td>ec2.IVpc</td>
<td>Returns an interface on the VPC used by the pattern (if any). This may be a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC created by the pattern or the VPC supplied to the pattern constructor.</td>
</tr>
</tbody>
</table>

**Default settings**

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**AWS Lambda Function**

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Set environment variables:
  - `SSM_STRING_PARAMETER_NAME` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**Amazon AWS Systems Manager Parameter Store String**

- Enable read-only access for the associated AWS Lambda Function.
- Creates a new SSM String parameter with the values provided.
- Retain the SSM String parameter when deleting the CloudFormation stack.
**Architecture**

- **Role**
- **Lambda function**
- **AWS Systems Manager Parameter store**
- **Amazon CloudWatch**

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:

[@aws-solutions-constructs/aws-lambda-ssmstringparameter](https://github.com/aws-solutions-constructs/aws-lambda-ssmstringparameter)

**aws-lambda-step-function**

Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.

**Note:** This construct has been deprecated and is superseded by the `aws-lambda-stepfunctions` construct.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an AWS Lambda function connected to an AWS Step Function.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { LambdaToStepFunction } from '@aws-solutions-constructs/aws-lambda-step-function';
import * as stepfunctions from '@aws-cdk/aws-stepfunctions';

const startState = new stepfunctions.Pass(this, 'StartState');

new LambdaToStepFunction(this, 'LambdaToStepFunctionPattern', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler'
  },
  stateMachineProps: {
    definition: startState
  }
});
```

Initializer

```typescript
new LambdaToStepFunction(scope: Construct, id: string, props: LambdaToStepFunctionProps);
```

Parameters

- **scope** `Construct`
- **id** `string`
- **props** `LambdaToStepFunctionProps (p. 223)`
## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td>boolean</td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td>logs.LogGroupProps</td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
<tr>
<td>stateMachineEnvironmentVariableName</td>
<td></td>
<td>Optional name for the Step Functions state machine environment variable set for the producer Lambda function.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td>ec2.IVpc</td>
<td>An optional, existing VPC into which this pattern should be deployed. When deployed in a VPC, the Lambda function will use ENIs in the VPC to access network resources and a Gateway Endpoint will be created in the VPC for Amazon DynamoDB. If an existing VPC is provided, the deployVpc property cannot be true. This uses ec2.IVpc to allow clients to supply VPCs that exist outside the stack using the ec2.Vpc.fromLookup() method.</td>
</tr>
<tr>
<td>vpcProps?</td>
<td>ec2.VpcProps</td>
<td>Optional user-provided properties to override the default properties for the new VPC. enableDnsHostnames, enableDnsSupport, natGateways, and subnetConfiguration are set.</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployVpc?</td>
<td>boolean</td>
<td>Whether to create a new VPC based on <code>vpcProps</code> into which to deploy this pattern. Setting this to true will deploy the minimal, most private VPC to run the pattern:</td>
</tr>
</tbody>
</table>
|                   |                       | • One isolated subnet in each Availability Zone used by the CDK program  
|                   |                       | • `enableDnsHostnames` and `enableDnsSupport` will both be set to true  
|                   |                       | If this property is true, then `existingVpc` cannot be specified. Defaults to false.                                                                                                                        |
| cloudwatchAlarms? | cloudwatch.Alarm[]    | Returns a list of one or more CloudWatch alarms created by the pattern.                                                                                                                                       |
| lambdaFunction    | lambda.Function       | Returns an instance of the Lambda function created by the pattern.                                                                                                                                              |
| stateMachine      | sfn.StateMachine      | Returns an instance of the state machine created by the pattern.                                                                                                                                               |
| stateMachineLogGroup | logs.ILogGroup        | Returns an instance of the ILogGroup created by the pattern for the state machine.                                                                                                                             |
| vpc?              | ec2.IVpc              | Returns an interface on the VPC used by the pattern (if any). This may be a VPC created by the pattern or the VPC supplied to the pattern constructor.                                                             |

### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:
AWS Lambda Function

- Configure a limited privilege access IAM role for the Lambda function.
- Enable reusing connections with Keep-Alive for NodeJS Lambda functions.
- Enable X-Ray tracing.
- Set environment variables:
  - `STATE_MACHINE_ARN` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

AWS Step Functions State Machine

- Enable CloudWatch logging for API Gateway.
- Deploy best-practice CloudWatch alarms for the AWS Step Functions State Machine.

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-lambda-step-function
aws-lambda-stepfunctions

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_lambda_stepfunctions</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-lambda-stepfunctions</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.lambdastepfunctions</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an AWS Lambda function connected to an AWS Step Functions State Machine.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { LambdaToStepfunctions } from '@aws-solutions-constructs/aws-lambda-stepfunctions';
import * as stepfunctions from '@aws-cdk/aws-stepfunctions';

const startState = new stepfunctions.Pass(stack, 'StartState');

new LambdaToStepfunctions(this, 'LambdaToStepfunctionsPattern', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`${__dirname}/lambda`)
  },
  stateMachineProps: {
    definition: startState
  }
});
```

**Initializer**

```typescript
new LambdaToStepfunctions(scope: Construct, id: string, props: LambdaToStepfunctionsProps);
```

**Parameters**
• scope Construct
• id string
• props LambdaToStepfunctionsProps (p. 227)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td>boolean</td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td>logs.LogGroupProps</td>
<td>Optional user-provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
<tr>
<td>stateMachineEnvironmentVariableName</td>
<td></td>
<td>Optional name for the Step Functions state machine environment variable set for the producer Lambda function.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td>ec2.IVpc</td>
<td>An optional, existing VPC into which this pattern should be deployed. When deployed in a VPC, the Lambda function will use ENIs in the VPC to access network resources and a Gateway Endpoint will be created in the VPC for Amazon DynamoDB. If an existing VPC is provided, the deployVpc property cannot be true. This uses ec2.IVpc to allow clients to supply VPCs that exist outside the stack using the ec2.Vpc.fromLookup() method.</td>
</tr>
<tr>
<td>vpcProps?</td>
<td>ec2.VpcProps</td>
<td>Optional user-provided properties to override the default properties for the new</td>
</tr>
</tbody>
</table>
# AWS Solutions Constructs AWS Solutions

**Pattern Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>VPC. enableDnsHostnames, enableDnsSupport, natGateways, and subnetConfiguration are set by the pattern, so any values for those properties supplied here will be overridden. If deployVpc is not true then this property will be ignored.</td>
</tr>
</tbody>
</table>
| deployVpc?            | boolean                   | Whether to create a new VPC based on vpcProps into which to deploy this pattern. Setting this to true will deploy the minimal, most private VPC to run the pattern:  
  - One isolated subnet in each Availability Zone used by the CDK program  
  - enableDnsHostnames and enableDnsSupport will both be set to true  
  If this property is true, then existingVpc cannot be specified. Defaults to false. |

## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns a list of one or more CloudWatch alarms created by the pattern.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>stateMachine</td>
<td>sfn.StateMachine</td>
<td>Returns an instance of the state machine created by the pattern.</td>
</tr>
<tr>
<td>stateMachineLogGroup</td>
<td>logs.ILogGroup</td>
<td>Returns an instance of the ILogGroup created by the pattern for the state machine.</td>
</tr>
<tr>
<td>vpc?</td>
<td>ec2.IVpc</td>
<td>Returns an interface on the VPC used by the pattern (if any). This may be a VPC created by the pattern or the VPC supplied to the pattern constructor.</td>
</tr>
</tbody>
</table>
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**AWS Lambda Function**

- Configure a limited privilege access IAM role for the Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda functions.
- Enable X-Ray tracing.
- Set environment variables:
  - `STATE_MACHINE_ARN` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**AWS Step Functions State Machine**

- Enable CloudWatch logging for API Gateway.
- Deploy best-practice CloudWatch alarms for the AWS Step Functions State Machine.

**Architecture**

![Architecture diagram showing AWS Lambda Function, AWS Step Functions, and Amazon CloudWatch Alarm]
aws-route53-alb

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

This AWS Solutions Construct implements an Amazon Route53 Hosted Zone routing to an Application Load Balancer. Here is a minimal deployable pattern definition in Typescript:

```typescript
import { Route53ToAlb } from '@aws-solutions-constructs/aws-route53-alb';

new Route53ToAlb(this, 'Route53ToAlbPattern', {
  privateHostedZoneProps: {
    zoneName: 'www.example.com',
  }
  publicApi: false,
});
```

### Initializer

```typescript
new Route53ToAlb(scope: Construct, id: string, props: Route53ToAlbProps);
```

### Parameters

- `scope` *Construct*
- `id` *string*
- `props` *Route53ToAlbProps*
Pattern Construct Props

This construct cannot create a new Public Hosted Zone, if you are creating a public API you must supply an existing Public Hosted Zone that will be reconfigured with a new Alias record. Public Hosted Zones are configured with public domain names and are not well suited to be launched and torn down dynamically, so this construct will only reconfigure existing Public Hosted Zones. This construct can create Private Hosted Zones. If you want a Private Hosted Zone, then you can either provide an existing Private Hosted Zone or a privateHostedZoneProps value with at least the Domain Name defined.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>privateHostedZoneProps?</td>
<td>route53.PrivateHostedZoneProps</td>
<td>Optional custom properties for a new Private Hosted Zone. Cannot be specified for a public API. Cannot specify a VPC, it will use the VPC in existingVpc or the VPC created by the construct. Providing both this and existingHostedZoneInterface is an error.</td>
</tr>
<tr>
<td>existingHostedZoneInterface?</td>
<td>route53.IHostedZone</td>
<td>Existing Public or Private Hosted Zone (type must match publicApi setting). Specifying both this and privateHostedZoneProps is an error. If this is a Private Hosted Zone, the associated VPC must be provided as the existingVpc property.</td>
</tr>
<tr>
<td>loadBalancerProps?</td>
<td>elasticloadbalancingv2.ApplicationLoadBalancerProps</td>
<td>Optional custom properties for a new loadBalancer. Providing both this and existingLoadBalancer is an error. This cannot specify a VPC, it will use the VPC in existingVpc or the VPC created by the construct.</td>
</tr>
<tr>
<td>existingLoadBalancerObj?</td>
<td>elasticloadbalancingv2.ApplicationLoadBalancer</td>
<td>Existing Application Load Balancer to incorporate into the construct architecture. Providing both this and loadBalancerProps is an error. The VPC containing this loadBalancer must match the VPC provided in existingVpc.</td>
</tr>
<tr>
<td>vpcProps?</td>
<td>ec2.VpcProps</td>
<td>Optional custom properties for a VPC the construct will create. This VPC will be used by the new ALB and any Private Hosted Zone the construct creates (that's why loadBalancerProps and privateHostedZoneProps can't include a VPC). Providing both this and existingVpc is an error.</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostedZone</td>
<td>route53.IHostedZone</td>
<td>The hosted zone used by the construct (whether created by the construct or provided by the client)</td>
</tr>
<tr>
<td>vpc</td>
<td>ec2.IVpc</td>
<td>The VPC used by the construct (whether created by the construct or provided by the client)</td>
</tr>
<tr>
<td>loadBalancer</td>
<td>elasticloadbalancingv2.ApplicationLoadBalancer</td>
<td>The Load Balancer used by the construct (whether created by the construct or provided by the client)</td>
</tr>
</tbody>
</table>

Default settings

Out of the box implementation of the Construct without any override will set the following defaults:
Amazon Route53

- Adds an ALIAS record to the new or provided Hosted Zone that routes to the construct's ALB

Application Load Balancer

- Creates an Application Load Balancer with no Listener or target. The construct can incorporate an existing, fully configured ALB if provided.

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-route53-alb

aws-s3-lambda
**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_s3_lambda</td>
</tr>
<tr>
<td>TypeScript</td>
<td>@aws-solutions-constructs/aws-s3-lambda</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.s3lambda</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an Amazon S3 bucket connected to an AWS Lambda function.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { S3ToLambdaProps, S3ToLambda } from '@aws-solutions-constructs/aws-s3-lambda';

new S3ToLambda(this, 'test-s3-lambda', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset("${dirname}/lambda"),
    handler: 'index.handler'
  },
});
```

**Initializer**

```typescript
new S3ToLambda(scope: Construct, id: string, props: S3ToLambdaProps);
```

**Parameters**

- scope `Construct`
- id `string`
- props `S3ToLambdaProps (p. 235)`
## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user provided props to override the default props for the Lambda function.</td>
</tr>
<tr>
<td>existingBucketObj?</td>
<td>s3.Bucket</td>
<td>Existing instance of S3 Bucket object. If this is provided, then also providing bucketProps is an error.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user provided props to override the default props for the S3 Bucket.</td>
</tr>
<tr>
<td>s3EventSourceProps?</td>
<td>S3EventSourceProps</td>
<td>Optional user provided props to override the default props for S3EventSourceProps</td>
</tr>
<tr>
<td>loggingBucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td>logS3AccessLogs?</td>
<td>boolean</td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is true.</td>
</tr>
</tbody>
</table>

## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>s3Bucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
</tbody>
</table>
### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

#### Amazon S3 Bucket

- Configure Access logging for S3 Bucket.
- Enable server-side encryption for S3 Bucket using AWS managed KMS Key.
- Turn on the versioning for S3 Bucket.
- Don't allow public access for S3 Bucket.
- Retain the S3 Bucket when deleting the CloudFormation stack.
- Enforce encryption of data in transit.
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days.

#### AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)
Architecture

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-s3-lambda

aws-s3-sqs

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.
Overview

This AWS Solutions Construct implements an Amazon S3 Bucket that is configured to send notifications to an Amazon SQS queue.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { S3ToSqs } from '@aws-solutions-constructs/aws-s3-sqs';
new S3ToSqs(stack, 'S3ToSQSPattern', {});
```

Initializer

```typescript
new S3ToSqs(scope: Construct, id: string, props: S3ToSqsProps);
```

Parameters

- **scope** `Construct`
- **id** `string`
- **props** `S3ToSqsProps` (p. 238)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>existingBucketObj?</code></td>
<td><code>s3.Bucket</code></td>
<td>Existing instance of S3 Bucket object. If this is provided, then also providing <code>bucketProps</code> is an error.</td>
</tr>
<tr>
<td><code>bucketProps?</code></td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the S3 Bucket.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>s3EventTypes?</td>
<td>s3.EventType[]</td>
<td>The S3 event types that will trigger the notification. Defaults to s3.EventType.OBJECT_CREATED.</td>
</tr>
<tr>
<td>s3EventFilters?</td>
<td>s3.NotificationKeyFilter[]</td>
<td>The S3 object key filter rules to determine which objects trigger this event. If not specified, no filter rules will be applied.</td>
</tr>
<tr>
<td>existingQueueObj?</td>
<td>sqs.Queue</td>
<td>Existing SQS queue to be used instead of the default queue. Providing both this and queueProps will cause an error. If the SQS queue is encrypted, the KMS key utilized for encryption must be a customer managed CMK.</td>
</tr>
<tr>
<td>queueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user provided properties to override the default properties for the SQS queue. Ignored if an existingQueueObj is provided.</td>
</tr>
<tr>
<td>deadLetterQueueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user provided props to override the default props for the dead letter queue. Only used if the deployDeadLetterQueue property is set to true.</td>
</tr>
<tr>
<td>deployDeadLetterQueue?</td>
<td>boolean</td>
<td>Whether to create a secondary queue to be used as a dead letter queue. Defaults to true.</td>
</tr>
<tr>
<td>maxReceiveCount?</td>
<td>number</td>
<td>The number of times a message can be unsuccessfully dequeued before being moved to the dead letter queue. Defaults to 15.</td>
</tr>
<tr>
<td>enableEncryptionWithCustomerManagedKey?</td>
<td></td>
<td>Whether to use a KMS Key, either managed by this CDK app, or imported. If importing an encryption key, it must be specified in the encryptionKey property for this construct.</td>
</tr>
<tr>
<td>encryptionKey?</td>
<td>kms.Key</td>
<td>Optional imported encryption key to encrypt the SQS queue.</td>
</tr>
<tr>
<td>encryptionKeyProps?</td>
<td>kms.KeyProps</td>
<td>Optional user provided properties to override the default properties for the KMS encryption key.</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loggingBucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td>logS3AccessLogs?</td>
<td>boolean</td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is true.</td>
</tr>
<tr>
<td>sqsQueue</td>
<td>sqs.Queue</td>
<td>Returns an instance of the SQS queue created by the pattern.</td>
</tr>
<tr>
<td>deadLetterQueue?</td>
<td>sqs.Queue</td>
<td>Returns an instance of the dead letter queue created by the pattern, if one is deployed.</td>
</tr>
<tr>
<td>encryptionKey</td>
<td>kms.IKey</td>
<td>Returns an instance of the encryption key created by the pattern.</td>
</tr>
<tr>
<td>s3Bucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
<tr>
<td>s3BucketInterface</td>
<td>s3.IBucket</td>
<td>Returns an instance of s3.IBucket created by the construct.</td>
</tr>
</tbody>
</table>

### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon S3 Bucket**

- Configure Access logging for S3 Bucket
- Enable server-side encryption for S3 Bucket using AWS managed KMS Key
- Turn on the versioning for S3 Bucket
- Don't allow public access for S3 Bucket
- Retain the S3 Bucket when deleting the CloudFormation stack
- Enforce encryption of data in transit
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days
Amazon SQS Queue

- Configure least privilege access permissions for SQS Queue
- Deploy SQS dead-letter queue for the source SQS Queue
- Enable server-side encryption for SQS Queue using Customer managed KMS Key
- Enforce encryption of data in transit

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-s3-sqs

aws-s3-step-function

Some of our early constructs don’t meet the naming standards that evolved for the library. We are releasing completely feature compatible versions with corrected names. The underlying implementation
code is the same regardless of whether you deploy the construct using the old or new name. We will support both names for all 1.x releases, but in 2.x we will only publish the correctly named constructs.

**Note:** This construct has been deprecated and is superseded by the `aws-s3-stepfunctions` construct.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td><code>aws_solutions_constructs.aws_s3_step_function</code></td>
</tr>
<tr>
<td>Typescript</td>
<td><code>@aws-solutions-constructs/aws-s3-step-function</code></td>
</tr>
<tr>
<td>Java</td>
<td><code>software.amazon.awsconstructs.services.s3stepfunction</code></td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an Amazon S3 bucket connected to an AWS Step Function.

**Note**

This construct uses Amazon EventBridge (Amazon CloudWatch Events) to trigger AWS Step Functions. EventBridge is more flexible, but triggering Step Functions with S3 Event Notifications has less latency and is more cost effective. If cost and/or latency is an issue, you should consider deploying `aws-s3-lambda` and `aws-lambda-stepfunctions` in place of this construct.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { S3ToStepFunction, S3ToStepFunctionProps } from '@aws-solutions-constructs/aws-s3-step-function';
import * as stepfunctions from '@aws-cdk/aws-stepfunctions';

const startState = new stepfunctions.Pass(this, 'StartState');
new S3ToStepFunction(this, 'test-s3-step-function-stack', {
    stateMachineProps: {
        definition: startState
    }
});
```

**Initializer**

```typescript
new S3ToStepFunction(scope: Construct, id: string, props: S3ToStepFunctionProps);
```
## Parameters

- **scope** `Construct`
- **id** `string`
- **props** `S3ToStepFunctionProps (p. 243)`

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingBucketObj?</td>
<td><code>s3.IBucket</code></td>
<td>Existing instance of S3 Bucket object. If this is provided, then also providing bucketProps is an error.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the S3 Bucket.</td>
</tr>
<tr>
<td>stateMachineProps</td>
<td><code>sfn.StateMachineProps</code></td>
<td>User provided props to override the default props for sfn.StateMachine.</td>
</tr>
<tr>
<td>eventRuleProps?</td>
<td><code>events.RuleProps</code></td>
<td>Optional user provided eventRuleProps to override the defaults.</td>
</tr>
<tr>
<td>deployCloudTrail?</td>
<td><code>boolean</code></td>
<td>Whether to deploy a Trail in AWS CloudTrail to log API events in Amazon S3. Defaults to <code>true</code>.</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td><code>boolean</code></td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user provided props to override the default props for the CloudWatch Logs log group.</td>
</tr>
<tr>
<td>loggingBucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td>logS3AccessLogs?</td>
<td><code>boolean</code></td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Defaults is <code>true</code>.</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudtrail?</td>
<td><code>cloudtrail.Trail</code></td>
<td>Returns an instance of the Cloudtrail trail created by the pattern.</td>
</tr>
<tr>
<td>cloudtrailBucket?</td>
<td><code>s3.Bucket</code></td>
<td>Returns an instance of the bucket created by the pattern for storing Cloudtrail trail data.</td>
</tr>
<tr>
<td>cloudtrailLoggingBucket?</td>
<td><code>s3.Bucket</code></td>
<td>Returns an instance of the logging bucket created by the pattern for the primary bucket used by the Cloudtrail trail.</td>
</tr>
<tr>
<td>cloudwatchAlarms?</td>
<td><code>cloudwatch.Alarm[]</code></td>
<td>Returns a list of one or more CloudWatch alarms created by the pattern.</td>
</tr>
<tr>
<td>s3Bucket?</td>
<td><code>s3.Bucket</code></td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td><code>s3.Bucket</code></td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
<tr>
<td>stateMachine</td>
<td><code>sfn.StateMachine</code></td>
<td>Returns an instance of the state machine created by the pattern.</td>
</tr>
<tr>
<td>stateMachineLogGroup</td>
<td><code>logs.ILogGroup</code></td>
<td>Returns an instance of the ILogGroup created by the pattern for the state machine.</td>
</tr>
<tr>
<td>s3BucketInterface</td>
<td><code>s3.IBucket</code></td>
<td>Returns an instance of <code>s3.IBucket</code> created by the construct.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon S3 Bucket**

- Configure Access logging for S3 Bucket.
- Enable server-side encryption for S3 Bucket using AWS managed KMS Key.
- Turn on the versioning for S3 Bucket.
- Don't allow public access for S3 Bucket.
- Retain the S3 Bucket when deleting the CloudFormation stack.
- Enforce encryption of data in transit.
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days.
**AWS CloudTrail**

- Configure a Trail in AWS CloudTrail to log API events in Amazon S3 related to the Bucket created by the Construct.

**Amazon CloudWatch Events Rule**

- Grant least privilege permissions to CloudWatch Events to trigger the Lambda Function.

**AWS Step Function**

- Enable CloudWatch logging for API Gateway.
- Deploy best practices CloudWatch Alarms for the Step Function.

**Architecture**

Diagram showing the flow from Amazon S3 Bucket, through AWS CloudTrail Event Rule, to AWS Step Functions and Amazon CloudWatch.

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-s3-step-function
All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

**Note:** To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌐 Python</td>
<td>aws_solutions_constructs.aws_s3_stepfunctions</td>
</tr>
<tr>
<td>🇺🇸 TypeScript</td>
<td>@aws-solutions-constructs/aws-s3-stepfunctions</td>
</tr>
<tr>
<td>☕ Java</td>
<td>software.amazon.awsconstructs.services.s3stepfunctions</td>
</tr>
</tbody>
</table>

**Overview**

This AWS Solutions Construct implements an Amazon S3 bucket connected to an AWS Step Functions state machine.

**Note**

This construct uses Amazon EventBridge (Amazon CloudWatch Events) to trigger AWS Step Functions. EventBridge is more flexible, but triggering Step Functions with S3 Event Notifications has less latency and is more cost effective. If cost and/or latency is an issue, you should consider deploying `aws-s3-lambda` and `aws-lambda-stepfunctions` in place of this construct.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { S3ToStepfunctions, S3ToStepfunctionsProps } from '@aws-solutions-constructs/aws-s3-stepfunctions';
import * as stepfunctions from '@aws-cdk/aws-stepfunctions';

const startState = new stepfunctions.Pass(stack, 'StartState');

new S3ToStepfunctions(this, 'test-s3-stepfunctions-stack', {
  stateMachineProps: {
    definition: startState
  }
});
```

**Initializer**

---

246
new S3ToStepfunctions(scope: Construct, id: string, props: S3ToStepfunctionsProps);

Parameters

- **scope** Construct
- **id** string
- **props** `S3ToStepfunctionsProps (p. 247)`

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingBucketObj?</td>
<td><code>s3.IBucket</code></td>
<td>Existing instance of S3 Bucket object. If this is provided, then also providing bucketProps is an error.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the S3 Bucket.</td>
</tr>
<tr>
<td>stateMachineProps</td>
<td><code>sfn.StateMachineProps</code></td>
<td>User provided props to override the default props for sfn.StateMachine.</td>
</tr>
<tr>
<td>eventRuleProps?</td>
<td><code>events.RuleProps</code></td>
<td>Optional user provided eventRuleProps to override the defaults.</td>
</tr>
<tr>
<td>deployCloudTrail?</td>
<td>boolean</td>
<td>Whether to deploy a Trail in AWS CloudTrail to log API events in Amazon S3. Defaults to <code>true</code>.</td>
</tr>
<tr>
<td>createCloudWatchAlarms</td>
<td>boolean</td>
<td>Whether to create recommended CloudWatch alarms.</td>
</tr>
<tr>
<td>logGroupProps?</td>
<td><code>logs.LogGroupProps</code></td>
<td>Optional user provided props to override the default props for the CloudWatchLogs LogGroup.</td>
</tr>
<tr>
<td>loggingBucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user provided props to override the default props for the S3 Logging Bucket.</td>
</tr>
<tr>
<td>logS3AccessLogs?</td>
<td>boolean</td>
<td>Whether to turn on Access Logging for the S3 bucket. Creates an S3 bucket with associated storage costs for the logs. Enabling Access Logging is a best practice. Default is <code>true</code>.</td>
</tr>
</tbody>
</table>
## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudtrail?</td>
<td>cloudtrail.Trail</td>
<td>Returns an instance of the Cloudtrail trail created by the pattern.</td>
</tr>
<tr>
<td>cloudtrailBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the bucket created by the pattern for storing Cloudtrail trail data.</td>
</tr>
<tr>
<td>cloudtrailLoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the primary bucket used by the Cloudtrail trail.</td>
</tr>
<tr>
<td>cloudwatchAlarms?</td>
<td>cloudwatch.Alarm[]</td>
<td>Returns a list of one or more CloudWatch alarms created by the pattern.</td>
</tr>
<tr>
<td>s3Bucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td>s3LoggingBucket?</td>
<td>s3.Bucket</td>
<td>Returns an instance of the logging bucket created by the pattern for the S3 bucket.</td>
</tr>
<tr>
<td>stateMachine</td>
<td>sfn.StateMachine</td>
<td>Returns an instance of the state machine created by the pattern.</td>
</tr>
<tr>
<td>stateMachineLogGroup</td>
<td>logs.ILogGroup</td>
<td>Returns an instance of the ILogGroup created by the pattern for the state machine.</td>
</tr>
<tr>
<td>s3BucketInterface</td>
<td>s3.IBucket</td>
<td>Returns an instance of s3.IBucket created by the construct.</td>
</tr>
</tbody>
</table>

### Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

### Amazon S3 Bucket

- Configure Access logging for S3 Bucket.
- Enable server-side encryption for S3 Bucket using AWS managed KMS Key.
- Turn on the versioning for S3 Bucket.
- Don't allow public access for S3 Bucket.
- Retain the S3 Bucket when deleting the CloudFormation stack.
- Enforce encryption of data in transit.
- Applies lifecycle rule to move noncurrent object versions to Glacier storage after 90 days.
AWS CloudTrail

- Configure a Trail in AWS CloudTrail to log API events in Amazon S3 related to the Bucket created by the Construct.

Amazon CloudWatch Events Rule

- Grant least privilege permissions to CloudWatch Events to trigger the Lambda Function.

AWS Step Function

- Enable CloudWatch logging for API Gateway.
- Deploy best practices CloudWatch Alarms for the Step Function.

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-s3-stepfunctions

aws-sns-lambda
Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_con structs.aws_sns_lambda</td>
</tr>
<tr>
<td>TypeScript</td>
<td>@aws-solutions-constructs/aws-sns-lambda</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.snslambda</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements an Amazon SNS connected to an AWS Lambda function.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { SnsToLambda, SnsToLambdaProps } from '@aws-solutions-constructs/aws-sns-lambda';
new SnsToLambda(this, 'test-sns-lambda', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
});
```

Initializer

```typescript
new SnsToLambda(scope: Construct, id: string, props: SnsToLambdaProps);
```

Parameters

- **scope** [Construct]
- **id** string
- **props** [SnsToLambdaProps](p. 251)
Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td>lambda.Function</td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>existingTopicObj?</td>
<td>sns.Topic</td>
<td>Existing instance of SNS Topic object, providing both this and topicProps will cause an error.</td>
</tr>
<tr>
<td>topicProps?</td>
<td>sns.TopicProps</td>
<td>Optional user provided properties to override the default properties for the SNS topic.</td>
</tr>
</tbody>
</table>

Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>snsTopic</td>
<td>sns.Topic</td>
<td>Returns an instance of the SNS topic created by the pattern.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

**Amazon SNS Topic**
- Configure least privilege access permissions for SNS topic.
- Enable server-side encryption using AWS managed KMS key.
- Enforce encryption of data in transit.

**AWS Lambda Function**
- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**Architecture**

```
Role

Lambda function

Amazon CloudWatch
```

**GitHub**

To view the code for this pattern, create/view issues and pull requests, and more:

```
@aws-solutions-constructs/aws-sns-lambda
```

**aws-sns-sqs**

<table>
<thead>
<tr>
<th>STABILITY</th>
<th>EXPERIMENTAL</th>
</tr>
</thead>
</table>

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.
Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_sns_sqs</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-sns-sqs</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.snssqs</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements an Amazon SNS topic connected to an Amazon SQS queue.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { SnsToSqs, SnsToSqsProps } from "@aws-solutions-constructs/aws-sns-sqs";
import * as iam from '@aws-cdk/aws-iam';

const snsToSqsStack = new SnsToSqs(this, 'SnsToSqsPattern', {});

// Grant yourself permissions to use the Customer Managed KMS Key
const policyStatement = new iam.PolicyStatement({
  actions: ["kms:Encrypt", "kms:Decrypt"],
  effect: iam.Effect.ALLOW,
  principals: [ new iam.AccountRootPrincipal() ],
  resources: [ "*" ]
});

snsToSqsStack.encryptionKey?.addToResourcePolicy(policyStatement);
```

Initializer

```typescript
new SnsToSqs(scope: Construct, id: string, props: SnsToSqsProps);
```

Parameters

- **scope** *Construct*
- **id** *string*
- **props** *SnsToSqsProps* (p. 254)
# Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingTopicObj?</td>
<td>sns.Topic</td>
<td>Existing instance of SNS Topic object, providing both this and topicProps will cause an error.</td>
</tr>
<tr>
<td>topicProps?</td>
<td>sns.TopicProps</td>
<td>Optional user-provided properties to override the default properties for the SNS topic. Ignored if an existingTopicObj is provided.</td>
</tr>
<tr>
<td>existingQueueObj?</td>
<td>sqs.Queue</td>
<td>An optional, existing SQS queue to be used instead of the default queue. Providing both this and queueProps will cause an error.</td>
</tr>
<tr>
<td>queueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user-provided properties to override the default properties for the SQS queue. Ignored if an existingQueueObj is provided.</td>
</tr>
<tr>
<td>deployDeadLetterQueue?</td>
<td>boolean</td>
<td>Whether to create a secondary queue to be used as a dead letter queue. Defaults to true.</td>
</tr>
<tr>
<td>deadLetterQueueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user-provided props to override the default props for the dead letter queue. Only used if the deployDeadLetterQueue property is set to true.</td>
</tr>
<tr>
<td>maxReceiveCount?</td>
<td>number</td>
<td>The number of times a message can be unsuccessfully dequeued before being moved to the dead letter queue. Defaults to 15.</td>
</tr>
<tr>
<td>enableEncryptionWithCustomerManagedKey?</td>
<td>boolean</td>
<td>Whether to use a customer-managed encryption key, either managed by this CDK app or imported. If importing an encryption key, it must be specified in the encryptionKey property for this construct.</td>
</tr>
<tr>
<td>encryptionKey?</td>
<td>kms.Key</td>
<td>An optional, existing encryption key to be used instead of the default encryption key.</td>
</tr>
<tr>
<td>encryptionKeyProps?</td>
<td>kms.KeyProps</td>
<td>Optional user-provided properties to override the</td>
</tr>
</tbody>
</table>
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>snsTopic</td>
<td>sns.Topic</td>
<td>Returns an instance of the SNS topic created by the pattern.</td>
</tr>
<tr>
<td>encryptionKey</td>
<td>kms.Key</td>
<td>Returns an instance of the encryption key created by the pattern.</td>
</tr>
<tr>
<td>sqsQueue</td>
<td>sqs.Queue</td>
<td>Returns an instance of the SQS queue created by the pattern.</td>
</tr>
<tr>
<td>deadLetterQueue?</td>
<td>sqs.Queue</td>
<td>Returns an instance of the dead letter queue created by the pattern, if one is deployed.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon SNS topic

- Configure least privilege access permissions for SNS topic.
- Enable server-side encryption using AWS managed KMS key.
- Enforce encryption of data in transit.

Amazon SQS queue

- Configure least privilege access permissions for SQS queue.
- Deploy dead-letter queue for the source SQS queue.
- Enable server-side encryption for SQS queue using customer-managed KMS key.
- Enforce encryption of data in transit.
GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-sns-sqs

aws-sqs-lambda

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_sqs_lambda</td>
</tr>
<tr>
<td>TypeScript</td>
<td>@aws-solutions-constructs/aws-sqs-lambda</td>
</tr>
</tbody>
</table>
Overview

This AWS Solutions Construct implements an Amazon SQS queue connected to an AWS Lambda function.

Here is a minimal deployable pattern definition in TypeScript:

```typescript
const { SqsToLambda } = require('@aws-solutions-constructs/aws-sqs-lambda');

new SqsToLambda(stack, 'SqsToLambdaPattern', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`#(__dirname)/lambda`),
    handler: 'index.handler'
  }
});
```

Initializer

```typescript
new SqsToLambda(scope: Construct, id: string, props: SqsToLambdaProps);
```

Parameters

- **scope** `Construct`
- **id** `string`
- **props** `SqsToLambdaProps` (p. 257)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingLambdaObj?</td>
<td><code>lambda.Function</code></td>
<td>Existing instance of Lambda Function object, providing both this and lambdaFunctionProps will cause an error.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</code></td>
<td>Optional user-provided properties to override the</td>
</tr>
</tbody>
</table>
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default properties</td>
<td>sqs.Queue</td>
<td>for the Lambda function. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>existingQueueObj?</td>
<td>sqs.Queue</td>
<td>An optional, existing SQS queue to be used instead of the default queue. Providing both this and queueProps will cause an error.</td>
</tr>
<tr>
<td>queueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user-provided properties to override the default properties for the SQS queue. Ignored if an existingQueueObj is provided.</td>
</tr>
<tr>
<td>deployDeadLetterQueue?</td>
<td>boolean</td>
<td>Whether to create a secondary queue to be used as a dead letter queue. Defaults to true.</td>
</tr>
<tr>
<td>deadLetterQueueProps?</td>
<td>sqs.QueueProps</td>
<td>Optional user-provided props to override the default props for the dead letter queue. Only used if the deployDeadLetterQueue property is set to true.</td>
</tr>
<tr>
<td>maxReceiveCount?</td>
<td>number</td>
<td>The number of times a message can be unsuccessfully dequeued before being moved to the dead letter queue. Defaults to 15.</td>
</tr>
<tr>
<td>sqsEventSourceProps?</td>
<td>SqsEventSourceProps</td>
<td>Optional user provided properties for the queue event source.</td>
</tr>
</tbody>
</table>

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deadLetterQueue?</td>
<td>sqs.Queue</td>
<td>Returns an instance of the dead letter queue created by the pattern, if one is deployed.</td>
</tr>
<tr>
<td>lambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the Lambda function created by the pattern.</td>
</tr>
<tr>
<td>sqsQueue</td>
<td>sqs.Queue</td>
<td>Returns an instance of the SQS queue created by the pattern.</td>
</tr>
</tbody>
</table>
Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:

Amazon SQS Queue

- Deploy SQS dead-letter queue for the source SQS Queue.
- Enable server-side encryption for source SQS Queue using AWS Managed KMS Key.
- Enforce encryption of data in transit.

AWS Lambda Function

- Configure limited privilege access IAM role for Lambda function.
- Enable reusing connections with Keep-Alive for NodeJs Lambda function.
- Enable X-Ray tracing.
- Set environment variables:
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

Architecture
GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

[@aws-solutions-constructs/aws-sqs-lambda](https://github.com/aws-solutions-constructs/aws-sqs-lambda)

aws-wafwebacl-alb

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_wafwebacl_alb</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-wafwebacl-alb</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.wafwebaclalb</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements an AWS WAF web ACL connected to an Application Load Balancer.

Here is a minimal deployable pattern definition in Typescript:

```typescript
import { Route53ToAlb } from '@aws-solutions-constructs/aws-route53-alb';
import { WafwebaclToAlbProps, WafwebaclToAlb } from '@aws-solutions-constructs/aws-wafwebacl-alb';

// A constructed ALB is required to be attached to the WAF Web ACL.
// In this case, we are using this construct to create one.
const r53ToAlb = new Route53ToAlb(this, 'Route53ToAlbPattern', {
  privateHostedZoneProps: {
    zoneName: 'www.example.com',
  },
  publicApi: false,
  logAccessLogs: false
});

// This construct can only be attached to a configured Application Load Balancer.
new WafwebaclToAlb(this, 'test-wafwebacl-alb', {
  existingLoadBalancerObj: r53ToAlb.loadBalancer
});
```
### Initializer

```typescript
new WafwebaclToAlb(scope: Construct, id: string, props: WafwebaclToAlbProps);
```

### Parameters

- `scope` - `Construct`
- `id` - `string`
- `props` - `WafwebaclToAlbProps`

### Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>existingLoadBalancerObj</code></td>
<td><code>elbv2.ApplicationLoadBalancer</code></td>
<td>The existing Application Load Balancer Object that will be protected with the WAF web ACL. <em>Note that a WAF web ACL can only be added to a configured Application Load Balancer, so this construct only accepts an existing ApplicationLoadBalancer and does not accept applicationLoadBalancerProps.</em></td>
</tr>
<tr>
<td><code>existingWebaclObj</code></td>
<td><code>waf.CfnWebACL</code></td>
<td>Existing instance of a WAF web ACL, an error will occur if this and props is set.</td>
</tr>
<tr>
<td><code>webaclProps</code></td>
<td><code>waf.CfnWebACLProps</code></td>
<td>Optional user-provided props to override the default props for the AWS WAF web ACL. To use a different collection of managed rule sets, specify a new rules property. Use our <code>wrapManagedRuleSet(managedGroupName: string, vendorName: string, priority: number)</code> function from core to create an array entry from each desired managed rule set.</td>
</tr>
</tbody>
</table>

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>webacl</code></td>
<td><code>waf.CfnWebACL</code></td>
<td>Returns an instance of the <code>waf.CfnWebACL</code> created by the construct.</td>
</tr>
</tbody>
</table>
### Default settings

Out of the box implementation of the Construct without any override will set the following defaults:

#### AWS WAF

- Deploy a WAF web ACL with 7 AWS managed rule groups:
  - AWSManagedRulesBotControlRuleSet
  - AWSManagedRulesKnownBadInputsRuleSet
  - AWSManagedRulesCommonRuleSet
  - AWSManagedRulesAnonymousIpList
  - AWSManagedRulesAmazonIpReputationList
  - AWSManagedRulesAdminProtectionRuleSet
  - AWSManagedRulesSQLiRuleSet

  *Note that the default rules can be replaced by specifying the rules property of CfnWebACLProps.*

- Send metrics to Amazon CloudWatch.

#### Application Load Balancer

- User provided Application Load Balancer object is used as-is.
To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-wafwebacl-alb

aws-wafwebacl-apigateway

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while...
you may use them, you may need to update your source code when upgrading to a newer version of this package.

Note: To ensure proper functionality, the AWS Solutions Constructs packages and AWS CDK packages in your project must be the same version.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_wafwacl_apigateway</td>
</tr>
<tr>
<td>TypeScript</td>
<td>@aws-solutions-constructs/aws-wafwacl-apigateway</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.wafwaclapigateway</td>
</tr>
</tbody>
</table>

Overview

This AWS Solutions Construct implements an AWS WAF web ACL connected to Amazon API Gateway REST API. Here is a minimal deployable pattern definition:

```typescript
import * as api from '@aws-cdk/aws-apigateway';
import * as lambda from '@aws-cdk/aws-lambda';
import { ApiGatewayToLambda } from '@aws-solutions-constructs/aws-apigateway-lambda';
import { WafwebaclToApiGatewayProps, WafwebaclToApiGateway } from '@aws-solutions-constructs/aws-wafwacl-apigateway';

const apiGatewayToLambda = new ApiGatewayToLambda(this, 'ApiGatewayToLambdaPattern', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        handler: 'index.handler',
        code: lambda.Code.fromAsset("lambda")
    }
});

// This construct can only be attached to a configured API Gateway.
new WafwebaclToApiGateway(this, 'test-wafwacl-apigateway', {
    existingApiGatewayInterface: apiGatewayToLambda.apiGateway
});
```

Initializer

```typescript
new WafwebaclToApiGateway(scope: Construct, id: string, props: WafwebaclToApiGatewayProps);
```

Parameters
• scope Construct
• id string
• props WafwebaclToApiGatewayProps (p. 265)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingApiGatewayInterface</td>
<td>apigateway.IRestApi</td>
<td>The existing API Gateway instance that will be protected with the WAF web ACL. Note that a WAF web ACL can only be added to a configured API Gateway, so this construct only accepts an existing IRestApi and does not accept apiGatewayProps.</td>
</tr>
<tr>
<td>existingWebaclObj?</td>
<td>waf.CfnWebACL</td>
<td>Existing instance of a WAF web ACL. An error will occur if this and webaclProps is set.</td>
</tr>
<tr>
<td>webaclProps?</td>
<td>waf.CfnWebACLProps</td>
<td>Optional user-provided props to override the default props for the AWS WAF web ACL. To use a different collection of managed rule sets, specify a new rules property. Use our wrapManagedRuleSet(managedGroupName: string, vendorName: string, priority: number) function from the core directory to create an array entry from each desired managed rule set.</td>
</tr>
</tbody>
</table>

Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>webacl</td>
<td>waf.CfnWebACL</td>
<td>Returns an instance of the waf.CfnWebACL created by the construct.</td>
</tr>
<tr>
<td>apiGateway</td>
<td>apigateway.IRestApi</td>
<td>Returns an instance of the API Gateway REST API created by the pattern.</td>
</tr>
</tbody>
</table>

Default settings

Out-of-the-box implementation of this pattern without any overrides will set the following defaults:
AWS WAF

- Deploy a WAF web ACL with 7 AWS managed rule groups:
  - AWSManagedRulesBotControlRuleSet
  - AWSManagedRulesKnownBadInputsRuleSet
  - AWSManagedRulesCommonRuleSet
  - AWSManagedRulesAnonymousIpList
  - AWSManagedRulesAmazonIpReputationList
  - AWSManagedRulesAdminProtectionRuleSet
  - AWSManagedRulesSQLiRuleSet

  Note that the default rules can be replaced by specifying the rules property of CfnWebACLProps.
- Send metrics to Amazon CloudWatch.

Amazon API Gateway

- User provided API Gateway object is used as-is.
GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

```
@aws-solutions-constructs/aws-wafwebacl-apigateway
```

aws-wafwebacl-cloudfront

All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while...
you may use them, you may need to update your source code when upgrading to a newer version of this package.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_wafwebacl_cloudfront</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-wafwebacl-cloudfront</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.wafwebaclcloudfront</td>
</tr>
</tbody>
</table>

## Overview

This AWS Solutions Construct implements an AWS WAF web ACL connected to Amazon CloudFront.

Here is a minimal deployable pattern definition in Typescript:

```typescript
import { CloudFrontToS3 } from '@aws-solutions-constructs/aws-cloudfront-s3';
import { WafwebaclToCloudFront } from '@aws-solutions-constructs/aws-wafwebacl-cloudfront';

const cloudfrontToS3 = new CloudFrontToS3(this, 'test-cloudfront-s3', {});

// This construct can only be attached to a configured CloudFront.
new WafwebaclToCloudFront(this, 'test-wafwebacl-cloudfront', {
  existingCloudFrontWebDistribution: cloudfrontToS3.cloudFrontWebDistribution
});
```

## Initializer

```typescript
new WafwebaclToCloudFront(scope: Construct, id: string, props: WafwebaclToCloudFrontProps);
```

## Parameters

- **scope** `Construct`
- **id** `string`
- **props** `WafwebaclToCloudFrontProps`

## Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingCloudFrontWebDistribution</td>
<td>cloudfront.Distribution</td>
<td>The existing CloudFront instance that will be protected with the WAF web ACL. Note that a WAF web ACL can only be added to a configured</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingWebaclObj?</td>
<td>waf.CfnWebACL</td>
<td>Existing instance of a WAF web ACL, an error will occur if this and props is set.</td>
</tr>
<tr>
<td>webaclProps?</td>
<td>waf.CfnWebACLProps</td>
<td>Optional user-provided props to override the default props for the AWS WAF web ACL. To use a different collection of managed rule sets, specify a new rules property. Use our wrapManagedRuleSet(managedGroupName: string, vendorName: string, priority: number) function from core to create an array entry from each desired managed rule set.</td>
</tr>
</tbody>
</table>

### Default settings

Out of the box implementation of the Construct without any override will set the following defaults:

**AWS WAF**

- Deploy a WAF web ACL with 7 AWS managed rule groups:
  - AWSManagedRulesBotControlRuleSet
  - AWSManagedRulesKnownBadInputsRuleSet
  - AWSManagedRulesCommonRuleSet
  - AWSManagedRulesAnonymousIpList
  - AWSManagedRulesAmazonIpReputationList
  - AWSManagedRulesAdminProtectionRuleSet
  - AWSManagedRulesSQLiRuleSet

  *Note that the default rules can be replaced by specifying the rules property of CfnWebACLProps.*

- Send metrics to Amazon CloudWatch.
Amazon CloudFront

- User provided CloudFront object is used as-is

Architecture

GitHub

To view the code for this pattern, create/view issues and pull requests, and more:

@aws-solutions-constructs/aws-wafwebacl-cloudfront
All classes are under active development and subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model. This means that while you may use them, you may need to update your source code when upgrading to a newer version of this package.

The core library includes the basic building blocks of AWS Solutions Constructs. It defines the core classes that are used in the rest of AWS Solutions Constructs.

### Default Properties for AWS CDK Constructs

Core library sets the default properties for the AWS CDK Constructs used by AWS Solutions Constructs constructs.

For example, the following is the snippet of default properties for S3 Bucket construct created by AWS Solutions Constructs construct. By default, it will turn on the server-side encryption, bucket versioning, block all public access and setup the S3 access logging.

```json
{
  encryption: s3.BucketEncryption.S3_MANAGED,
  versioned: true,
  blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
  removalPolicy: RemovalPolicy.RETAIN,
  serverAccessLogsBucket: loggingBucket
}
```

### Override the default properties

The default properties set by the Core library can be overridden by user provided properties. For example, the user can override the Amazon S3 Block Public Access property to meet specific requirements.

```typescript
const stack = new cdk.Stack();
const props: CloudFrontToS3Props = {
  bucketProps: {
    blockPublicAccess: {
      blockPublicAcls: false,
      blockPublicPolicy: true,
      ignorePublicAcls: false,
      restrictPublicBuckets: true
    }
  }
};
new CloudFrontToS3(stack, 'test-cloudfront-s3', props);

expect(stack).toHaveResource("AWS::S3::Bucket", {
  PublicAccessBlockConfiguration: {
    BlockPublicAcls: false,
    BlockPublicPolicy: true,
    IgnorePublicAcls: false,
    RestrictPublicBuckets: true
  }
});
```
Property override warnings

When a default property from the Core library is overridden by a user-provided property, Constructs will emit one or more warning messages to the console highlighting the change(s). These messages are intended to provide situational awareness to the user and prevent unintentional overrides that could create security risks. These messages will appear whenever deployment/build-related commands are executed, including `cdk deploy`, `cdk synth`, `npm test`, etc.

Example message: `AWS_Constructs_WARNING: An override has been provided for the property: BillingMode. Default value: 'PAY_PER_REQUEST'. You provided: 'PROVISIONED'.`

Toggling override warnings

Override warning messages are enabled by default, but can be explicitly turned on/off using the `overrideWarningsEnabled` shell variable.

- To explicitly turn off override warnings, run `export overrideWarningsEnabled=false`.
- To explicitly turn on override warnings, run `export overrideWarningsEnabled=true`.
- To revert to the default, run `unset overrideWarningsEnabled`. 
## Document Revisions

To be notified about updates to AWS Solutions Constructs, subscribe to the RSS feed.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content updated (p. 273)</td>
<td>Added logs3AccessLogs and loggingBucketProps input properties, as well as an S3BucketInterface output property, to applicable patterns that deploy an Amazon S3 bucket. Other minor updates. (v2.0.0)</td>
<td>December 3, 2021</td>
</tr>
<tr>
<td>Content updated (p. 273)</td>
<td>Added new aws-alb-lambda, aws-route53-alb, and aws-wafwebacl-alb constructs. Promoted additional constructs to stable from experimental. Other minor updates. (v1.126.0)</td>
<td>October 27, 2021</td>
</tr>
<tr>
<td>Content updated (p. 273)</td>
<td>Added new loggingBucketProps property to the following patterns: aws-cloudfront-s3, aws-lambda-s3, aws-s3-lambda, aws-s3-sqs, aws-s3-step-function, and aws-s3-stepfunctions. (v1.125.0)</td>
<td>October 11, 2021</td>
</tr>
<tr>
<td>Content updated (p. 273)</td>
<td>Added new aws-wafwebacl-cloudfront pattern. Updated properties for aws-apigateway-kinesisstreams pattern. (v1.124.0)</td>
<td>October 3, 2021</td>
</tr>
<tr>
<td>Content updated (p. 273)</td>
<td>Added new aws-iot-kinesisstreams pattern. Other minor content updates. (v1.123.0)</td>
<td>September 21, 2021</td>
</tr>
<tr>
<td>Content updated (p. 273)</td>
<td>Added new aws-lambda-eventbridge and aws-wafwebacl-apigateway pattern. Other minor content updates. (v1.122.0)</td>
<td>September 20, 2021</td>
</tr>
<tr>
<td>Content updated (p. 273)</td>
<td>Added new aws-iot-sqs pattern. Other minor content updates. (v1.117.0)</td>
<td>August 17, 2021</td>
</tr>
<tr>
<td>Content updated (p. 273)</td>
<td>Multiple patterns deprecated and replaced with new patterns based on updated naming convention. Multiple patterns</td>
<td>August 17, 2021</td>
</tr>
<tr>
<td>Date</td>
<td>Content Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>July 26, 2021</td>
<td>Updated properties for the <code>aws-cloudfront-s3</code> pattern. Other minor updates.</td>
<td></td>
</tr>
<tr>
<td>July 23, 2021</td>
<td>Updated properties for the <code>aws-cloudfront-s3</code> pattern. Other minor content updates.</td>
<td></td>
</tr>
<tr>
<td>June 16, 2021</td>
<td>Updated properties for select patterns and added new use case.</td>
<td></td>
</tr>
<tr>
<td>May 27, 2021</td>
<td>Added <code>aws-lambda-ssmstringparameter</code> pattern. Other minor content updates.</td>
<td></td>
</tr>
<tr>
<td>May 12, 2021</td>
<td>Added <code>aws-lambda-secretsmanager</code> pattern. Other minor content updates.</td>
<td></td>
</tr>
<tr>
<td>April 17, 2021</td>
<td>Property updates to select <code>-lambda</code> patterns. Other minor content updates.</td>
<td></td>
</tr>
<tr>
<td>March 30, 2021</td>
<td>Fixed an issue in the Walkthrough for Python users and updated property examples for constructs containing Lambda functions.</td>
<td></td>
</tr>
<tr>
<td>March 8, 2021</td>
<td>Minor fixes/updates to pattern props and default settings for select patterns.</td>
<td></td>
</tr>
<tr>
<td>March 4, 2021</td>
<td>Minor fixes/updates to walkthrough content.</td>
<td></td>
</tr>
<tr>
<td>February 24, 2021</td>
<td>Added <code>aws-lambda-sagemakerendpoint</code> pattern and updated properties for select Kinesis Firehose patterns.</td>
<td></td>
</tr>
<tr>
<td>February 17, 2021</td>
<td>Added <code>aws-kinesisstreams-gluejob</code> pattern and updated walkthrough steps for Python users.</td>
<td></td>
</tr>
<tr>
<td>February 9, 2021</td>
<td>Updated properties for <code>aws-cloudfront-*</code> patterns.</td>
<td></td>
</tr>
<tr>
<td>February 5, 2021</td>
<td>Added link to GitHub for each pattern.</td>
<td></td>
</tr>
<tr>
<td>February 1, 2021</td>
<td>Updated properties for select patterns.</td>
<td></td>
</tr>
<tr>
<td>January 4, 2021</td>
<td>Updated documentation of properties and default settings for select patterns.</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Update Details</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>December 20, 2020</td>
<td>Added new patterns: aws-cloudfront-mediastore and aws-s3-sqs.</td>
<td></td>
</tr>
<tr>
<td>November 17, 2020</td>
<td>Removed aws-lambda-sagemaker pattern.</td>
<td></td>
</tr>
<tr>
<td>October 27, 2020</td>
<td>Added new patterns: aws-events-rule-kinesisstreams, aws-events-rule-kinesisfirehose-s3, and aws-lambda-sagemaker.</td>
<td></td>
</tr>
<tr>
<td>October 22, 2020</td>
<td>Updated to reflect breaking change in aws-events-rule-sns and aws-events-rule-sqs patterns: class and interface names changed to pascal case.</td>
<td></td>
</tr>
<tr>
<td>October 20, 2020</td>
<td>Added aws-apigateway-sagemakerendpoint and aws-kinesisstreams-kinesisfirehose-s3 patterns; other minor updates to existing content.</td>
<td></td>
</tr>
<tr>
<td>October 7, 2020</td>
<td>Added aws-apigateway-iot pattern; other minor updates to existing content.</td>
<td></td>
</tr>
<tr>
<td>October 5, 2020</td>
<td>Updated minimal deployable pattern code snippets and best practice defaults for all patterns.</td>
<td></td>
</tr>
<tr>
<td>September 14, 2020</td>
<td>Updated properties for aws-kinesisstreams-lambda pattern to reflect breaking change.</td>
<td></td>
</tr>
<tr>
<td>September 10, 2020</td>
<td>Minor fix to second part of walkthrough.</td>
<td></td>
</tr>
<tr>
<td>September 2, 2020</td>
<td>Added aws-sns-sqs pattern; updates to all SNS patterns; minor typographical corrections.</td>
<td></td>
</tr>
<tr>
<td>August 31, 2020</td>
<td>Fixed module names for aws-sqs-lambda pattern.</td>
<td></td>
</tr>
<tr>
<td>August 27, 2020</td>
<td>Updated defaults for Lambda patterns; other minor updates.</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>August 10, 2020</td>
<td>Updated public properties for S3 patterns; updated defaults for DynamoDB patterns.</td>
<td></td>
</tr>
<tr>
<td>August 4, 2020</td>
<td>Updated multiple patterns to highlight default enforcement of encryption in transit.</td>
<td></td>
</tr>
<tr>
<td>July 27, 2020</td>
<td>Added aws-lambda-sqs-lambda pattern; improved configuration instructions in Getting Started guide; updated all patterns to make additional resources available through public properties.</td>
<td></td>
</tr>
<tr>
<td>July 20, 2020</td>
<td>Added aws-lambda-sqs pattern; other minor updates.</td>
<td></td>
</tr>
<tr>
<td>July 9, 2020</td>
<td>Removed deployLambda and deployBucket properties from relevant patterns; other minor updates.</td>
<td></td>
</tr>
<tr>
<td>July 7, 2020</td>
<td>Added aws-lambda-step-function pattern and corrected minor typographical errors.</td>
<td></td>
</tr>
<tr>
<td>June 25, 2020</td>
<td>Added existingTableObj? property to select DynamoDB patterns.</td>
<td></td>
</tr>
<tr>
<td>June 23, 2020</td>
<td>Several text corrections and fixes for broken links.</td>
<td></td>
</tr>
<tr>
<td>June 22, 2020</td>
<td>AWS Solutions Constructs made publicly available.</td>
<td></td>
</tr>
</tbody>
</table>
Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2020 Amazon Web Services, Inc. or its affiliates. All rights reserved.