# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>What is AWS Solutions Constructs?</td>
<td>1</td>
</tr>
<tr>
<td>Why use AWS Solutions Constructs?</td>
<td>1</td>
</tr>
<tr>
<td>Getting Started</td>
<td>2</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>2</td>
</tr>
<tr>
<td>Installing the AWS CDK</td>
<td>2</td>
</tr>
<tr>
<td>Working with AWS Solutions Constructs</td>
<td>3</td>
</tr>
<tr>
<td>Walkthrough - Part 1</td>
<td>3</td>
</tr>
<tr>
<td>Hello Constructs</td>
<td>4</td>
</tr>
<tr>
<td>Creating the App Directory and Initializing the AWS CDK</td>
<td>4</td>
</tr>
<tr>
<td>Update project base dependencies</td>
<td>4</td>
</tr>
<tr>
<td>Lambda handler code</td>
<td>6</td>
</tr>
<tr>
<td>Install the AWS CDK and AWS Solutions Constructs dependencies</td>
<td>7</td>
</tr>
<tr>
<td>Add an Amazon API Gateway/AWS Lambda pattern to your stack</td>
<td>8</td>
</tr>
<tr>
<td>cdk deploy</td>
<td>12</td>
</tr>
<tr>
<td>Stack outputs</td>
<td>12</td>
</tr>
<tr>
<td>Testing your app</td>
<td>13</td>
</tr>
<tr>
<td>Walkthrough - Part 2</td>
<td>13</td>
</tr>
<tr>
<td>Hit Counter Lambda code</td>
<td>14</td>
</tr>
<tr>
<td>Install the new dependencies</td>
<td>15</td>
</tr>
<tr>
<td>Define the resources</td>
<td>16</td>
</tr>
<tr>
<td>Review the changes</td>
<td>24</td>
</tr>
<tr>
<td>cdk deploy</td>
<td>26</td>
</tr>
<tr>
<td>Stack outputs</td>
<td>26</td>
</tr>
<tr>
<td>Testing your app</td>
<td>26</td>
</tr>
<tr>
<td>Sample Use Cases</td>
<td>27</td>
</tr>
<tr>
<td>AWS Static S3 Website</td>
<td>28</td>
</tr>
<tr>
<td>AWS Simple Serverless Image Handler</td>
<td>28</td>
</tr>
<tr>
<td>AWS Serverless Web App</td>
<td>28</td>
</tr>
<tr>
<td>API Reference</td>
<td>29</td>
</tr>
<tr>
<td>Modules</td>
<td>29</td>
</tr>
<tr>
<td>Module Contents</td>
<td>29</td>
</tr>
<tr>
<td>aws-apigateway-dynamodb</td>
<td>30</td>
</tr>
<tr>
<td>Overview</td>
<td>30</td>
</tr>
<tr>
<td>Initializer</td>
<td>30</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>30</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>30</td>
</tr>
<tr>
<td>Default settings</td>
<td>30</td>
</tr>
<tr>
<td>Architecture</td>
<td>33</td>
</tr>
<tr>
<td>aws-apigateway-iot</td>
<td>34</td>
</tr>
<tr>
<td>Overview</td>
<td>34</td>
</tr>
<tr>
<td>Initializer</td>
<td>34</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>34</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>34</td>
</tr>
<tr>
<td>Default settings</td>
<td>35</td>
</tr>
<tr>
<td>Architecture</td>
<td>37</td>
</tr>
<tr>
<td>Examples</td>
<td>37</td>
</tr>
<tr>
<td>aws-apigateway-kinesisstreams</td>
<td>38</td>
</tr>
<tr>
<td>Overview</td>
<td>39</td>
</tr>
<tr>
<td>Initializer</td>
<td>39</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>39</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>39</td>
</tr>
<tr>
<td>Sample API Usage</td>
<td>40</td>
</tr>
<tr>
<td>Default settings</td>
<td>41</td>
</tr>
<tr>
<td>Construct</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>aws-apigateway-lambda</td>
<td>42</td>
</tr>
<tr>
<td>Overview</td>
<td>43</td>
</tr>
<tr>
<td>Initializer</td>
<td>43</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>44</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>44</td>
</tr>
<tr>
<td>Default settings</td>
<td>44</td>
</tr>
<tr>
<td>Architecture</td>
<td>45</td>
</tr>
<tr>
<td>aws-apigateway-sagemakerendpoint</td>
<td>45</td>
</tr>
<tr>
<td>Overview</td>
<td>46</td>
</tr>
<tr>
<td>Initializer</td>
<td>46</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>47</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>48</td>
</tr>
<tr>
<td>Sample API Usage</td>
<td>41</td>
</tr>
<tr>
<td>Default settings</td>
<td>48</td>
</tr>
<tr>
<td>Architecture</td>
<td>49</td>
</tr>
<tr>
<td>aws-apigateway-sqs</td>
<td>49</td>
</tr>
<tr>
<td>Overview</td>
<td>50</td>
</tr>
<tr>
<td>Initializer</td>
<td>50</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>50</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>52</td>
</tr>
<tr>
<td>Sample API Usage</td>
<td>41</td>
</tr>
<tr>
<td>Default settings</td>
<td>52</td>
</tr>
<tr>
<td>Architecture</td>
<td>53</td>
</tr>
<tr>
<td>aws-cloudfront-apigateway</td>
<td>53</td>
</tr>
<tr>
<td>Initializer</td>
<td>54</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>55</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>55</td>
</tr>
<tr>
<td>Default settings</td>
<td>55</td>
</tr>
<tr>
<td>Architecture</td>
<td>56</td>
</tr>
<tr>
<td>aws-cloudfront-apigateway-lambda</td>
<td>56</td>
</tr>
<tr>
<td>Initializer</td>
<td>57</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>57</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>58</td>
</tr>
<tr>
<td>Default settings</td>
<td>59</td>
</tr>
<tr>
<td>Architecture</td>
<td>59</td>
</tr>
<tr>
<td>aws-cloudfront-mediastore</td>
<td>59</td>
</tr>
<tr>
<td>Initializer</td>
<td>60</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>61</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>61</td>
</tr>
<tr>
<td>Default settings</td>
<td>62</td>
</tr>
<tr>
<td>Architecture</td>
<td>62</td>
</tr>
<tr>
<td>aws-cloudfront-s3</td>
<td>63</td>
</tr>
<tr>
<td>Initializer</td>
<td>64</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>64</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>65</td>
</tr>
<tr>
<td>Default settings</td>
<td>65</td>
</tr>
<tr>
<td>Architecture</td>
<td>66</td>
</tr>
<tr>
<td>aws-cognito-apigateway-lambda</td>
<td>66</td>
</tr>
<tr>
<td>Initializer</td>
<td>67</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>68</td>
</tr>
<tr>
<td>Pattern Properties</td>
<td>68</td>
</tr>
<tr>
<td>Default settings</td>
<td>69</td>
</tr>
<tr>
<td>Architecture</td>
<td>69</td>
</tr>
<tr>
<td>aws-dynamodb-stream-lambda</td>
<td>70</td>
</tr>
<tr>
<td>Initializer</td>
<td>70</td>
</tr>
<tr>
<td>Pattern Construct Props</td>
<td>71</td>
</tr>
<tr>
<td>Architecture</td>
<td>70</td>
</tr>
</tbody>
</table>
What is AWS Solutions Constructs?

AWS Solutions Constructs (Constructs) is an open-source extension of the AWS Cloud Development Kit (AWS 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 (AWS 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 (AWS 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 (AWS 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>Python</td>
<td>&gt;= 3.6</td>
</tr>
<tr>
<td>TypeScript</td>
<td>&gt;= 2.7</td>
</tr>
<tr>
<td>Java</td>
<td>&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**

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

**Python**

```bash
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:
"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"
},
"dependencies": {
  "@aws-cdk/core": "VERSION_NUMBER",
  "source-map-support": "^0.5.16"
}

Python

Edit the setup.py file with the following information:

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

Install the projects base dependencies.

TypeScript

```
npm install
```

Python

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

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

TypeScript

```
npm run build
cdk synth
```

Python

```
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**

```typescript
Resources:
  CDKMetadata:
    Type: AWS::CDK::Metadata
    Properties:
      Modules: aws-cdk=CDK-VERSION,@aws-cdk/core=VERSION_NUMBER,@aws-cdk/cx-api=VERSION_NUMBER,jsii-runtime=node.js/10.17.0
```

**Python**

```python
Resources:
  CDKMetadata:
    Type: AWS::CDK::Metadata
    Properties:
      Modules: aws-cdk=CDK-VERSION,@aws-cdk/core=VERSION_NUMBER,@aws-cdk/cx-api=VERSION_NUMBER,jsii-runtime=Python/3.7.7
```

## 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:

```typescript
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': {
```
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.

**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**

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

**Python**

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

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

**TypeScript**

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

**Python**

```
    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**

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

**Python**

```
pip install 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',
            deploy_lambda=True,
            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 rebuild our project to reflect the changes we made to the stack above:

```bash
npm run build
```

Now, let's compare our new stack to the original one:

```bash
cdk diff
```

The output should look like this:
Add an Amazon API Gateway/AWS Lambda pattern to your stack

```python
# # * 
# Service: apigateway.amazonaws.com
# #

# # * 
# Allow # sts:AssumeRole
# lambdaArn

# # * 
# Allow # logs:CreateLogGroup
# ${LambdaRestApiCloudWatchRole}.

# # * 
# Allow # logs:CreateLogStream
# (${AWS::Region})

# # * 
# Allow # logs:DescribeLogGroups
# (${LambdaRestApiCloudWatchRole})

# # * 
# Allow # logs:DescribeLogStreams
# (${LambdaRestApiCloudWatchRole})

# # * 
# Allow # logs:FilterLogEvents
# (${LambdaRestApiCloudWatchRole})

# # * 
# Allow # logs:GetLogEvents
# (${LambdaRestApiCloudWatchRole})

# # * 
# Allow # logs:PutLogEvents
# (${LambdaRestApiCloudWatchRole})

# # *
# arn:aws:logs:${AWS::Region} # Allow # logs:CreateLogGroup
# lambdaFunctionService

# # *
# arn:aws:logs:${AWS::Region} # Allow # logs:CreateLogStream
# Role

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

Parameters

[+] Parameter AssetParameters/ba91444ebd644d9419e8cfee417f3aa728507dd428788a2fc40574646c4340a/S3Bucket

AssetParameters/ba91444ebd644d9419e8cfee417f3aa728507dd428788a2fc40574646c4340a/S3Bucket/AWS:

  { "Type" : "String", "Description" : "S3 bucket for asset" }

[+] Parameter AssetParameters/ba91444ebd644d9419e8cfee417f3aa728507dd428788a2fc40574646c4340a/S3VersionKey

AssetParameters/ba91444ebd644d9419e8cfee417f3aa728507dd428788a2fc40574646c4340a/S3VersionKey/AWS:

  { "Type" : "String", "Description" : "S3 key for asset version" }

[+] Parameter AssetParameters/ba91444ebd644d9419e8cfee417f3aa728507dd428788a2fc40574646c4340a/ArtifactHash

AssetParameters/ba91444ebd644d9419e8cfee417f3aa728507dd428788a2fc40574646c4340a/ArtifactHash/AWS:

  { "Type" : "String", "Description" : "Artifact hash for asset" }

Conditions

[+] Condition CDKMetadataAvailable: ("Fn:Or":{"Fn:Or":{"Fn:Equals":

  [{"Ref":"AWS:Region"},"ap-east-1"],
  [{"Ref":"AWS:Region"},"ap-northeast-1"],
  [{"Ref":"AWS:Region"},"ap-northeast-2"],
  [{"Ref":"AWS:Region"},"ap-south-1"],
  [{"Ref":"AWS:Region"},"ap-southeast-1"],
  [{"Ref":"AWS:Region"},"ap-southeast-2"],
  [{"Ref":"AWS:Region"},"ap-north-1"],
  [{"Ref":"AWS:Region"},"au-central-1"],
  [{"Ref":"AWS:Region"},"au-north-1"],
  [{"Ref":"AWS:Region"},"ca-central-1"],
  [{"Ref":"AWS:Region"},"cn-north-1"],
  [{"Ref":"AWS:Region"},"cn-northwest-1"],
  [{"Ref":"AWS:Region"},"eu-central-1"],
  [{"Ref":"AWS:Region"},"eu-north-1"],
  [{"Ref":"AWS:Region"},"eu-south-1"],
  [{"Ref":"AWS:Region"},"me-central-1"],
  [{"Ref":"AWS:Region"},"me-south-1"],
  [{"Ref":"AWS:Region"},"sa-east-1"],
  [{"Ref":"AWS:Region"},"sa-east-2"],
  [{"Ref":"AWS:Region"},"us-east-1"],
  [{"Ref":"AWS:Region"},"us-east-2"],
  [{"Ref":"AWS:Region"},"us-west-1"],
  [{"Ref":"AWS:Region"},"us-west-2"],
  [{"Ref":"AWS:Region"},"us-west-3"],
  [{"Ref":"AWS:Region"},"us-west-4"],
  [{"Ref":"AWS:Region"},"us-west-6"],
  [{"Ref":"AWS:Region"},"us-west-8"]})}}

Resources
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/
```

Output should look like this:

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

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

**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)](http://example.com). 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:

```javascript
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:

```python
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(
```
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**
  ```sh
npm install -s @aws-cdk/aws-dynamodb@VERSION_NUMBER
  ```

- **Python**
  ```sh
  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**
  ```sh
  npm install -s @aws-solutions-constructs/aws-lambda-dynamodb@VERSION_NUMBER
  ```

- **Python**
  ```sh
  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);

    // 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'
    });

    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'),
        )

        apigw_lambda.ApiGatewayToLambda(
            self, 'ApiGatewayToLambda',
            deploy_lambda=True,
            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
                )
            )
        )

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: {
        DOWNSTREAM_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_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'),  
        )
```
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:

```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.PYTHON_3_7,
      code: lambda.Code.asset('lambda'),
      handler: 'hitcounter.handler',
      environment: {
        'DOWNSTREAM_FUNCTION_NAME': self.hello_func.function_name
      },
      deploy_lambda=True,
      lambda_function_props=lambda.FunctionProps(
        runtime=lambda.Runtime.PYTHON_3_7,
        code=lambda.Code.asset('lambda'),
        handler='hitcounter.handler',
        environment={
          'DOWNSTREAM_FUNCTION_NAME': self.hello_func.function_name
        },
      ),
      dynamo_table_props=dynamodb.TableProps(
        table_name='Hits',
        partition_key={
          'name': 'path',
          'type': dynamodb.AttributeType.STRING
        }
      )
    }),
    api_gateway_props=api.RestApiProps(
      default_method_options=api.MethodOptions(
        authorization_type=api.AuthorizationType.NONE
      )
    )
  }
}
```
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.fromAsset('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:

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',  
    deploy_lambda=True,  
    lambda_function_props=_lambda.FunctionProps(  
        runtime=_lambda.Runtime.PYTHON_3_7,  
        code=_lambda.Code.asset('lambda'),  
        handler='hitcounter.handler',  
        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)

api_gateway_props=apigw.RestApiProps(  
    default_method_options=apigw.MethodOptions(  
        authorization_type=apigw.AuthorizationType.NONE
    )
)

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
                }
            }
        };

        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',
            deploy_lambda=True,
            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',
            deploy_lambda=False,
            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
```
review the changes

```bash
cdk diff
```

our output should look like this:

<table>
<thead>
<tr>
<th>Stack</th>
<th>IAM Statement Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelloConstructsStack</td>
<td># # Resource # Effect # Action #</td>
</tr>
<tr>
<td></td>
<td>Principal</td>
</tr>
<tr>
<td># + # ${HelloHandler.Arn}</td>
<td>Allow</td>
</tr>
<tr>
<td># + # ${HelloHandler/ServiceRole.Arn}</td>
<td>Allow</td>
</tr>
<tr>
<td># + # ${LambdaToDynamoDB/DynamoTable.Arn}</td>
<td>Allow</td>
</tr>
<tr>
<td># # n}</td>
<td># dynamodb:BatchWriteItem</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:DeleteItem</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:GetItem</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:GetRecords</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:GetShardIterator</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:GetItem</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:PutItem</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:Query</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:Scan</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:UpdateItem</td>
</tr>
<tr>
<td># #</td>
<td># dynamodb:Scan</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IAM Policy Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td># # Resource # Managed Policy ARN</td>
</tr>
<tr>
<td># + # ${HelloHandler/ServiceRole}</td>
</tr>
</tbody>
</table>

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

Resources

| [-] AWS::IAM::Role HelloHandler/ServiceRole HelloHandler/ServiceRole11EF7C63 |
| [+] AWS::Lambda::Function HelloHandler HelloHandler2E4FBA4D |
| [+] AWS::DynamoDB::Table LambdaToDynamoDB/DynamoTable LambdaToDynamoDB/DynamoTable53C1442D |
| [+] AWS::IAM::Policy LambdaFunctionServiceRole/DefaultPolicy LambdaFunctionServiceRole/DefaultPolicy126C8897 |
| [-] AWS::Lambda::Function LambdaFunction LambdaFunctionBF21E41F |

[+] Environment

```json```
{"Variables":"
"DOWNSTREAM_FUNCTION_NAME":
"Ref":"HelloHandler2E4FBA4D"},
"DDB_TABLE_NAME":
"Ref":"LambdaToDynamoDB/DynamoTable53C1442D"}
```json```

[+] Handler

```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
```json```
```
cdk deploy
Okay, ready to deploy?

cdk deploy

Stack outputs
When deployment is complete, you’ll notice this line:

Outputs:
HelloConstructsStack.RestApiEndpoint0551178A = https://xxxxxxxxxxx.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://xxxxxxxxxxx.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 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
API Reference

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

The patterns defined in Constructs are high level, multi-service abstractions of AWS CDK constructs that have default configurations based on well-architected best practices. The library is organized into logical modules using object-oriented techniques to create each architectural pattern model.

The CDK is available in the following languages:

- JavaScript, TypeScript (Node.js ≥ 10.3.0)
- Python (Python ≥ 3.6)
- Java (Java ≥ 1.8)

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-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.
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. 30)`

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>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 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>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 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>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 pattern that enables access logging from the API Gateway REST API to CloudWatch.</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>
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

**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.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td><code>aws_solutions Constructs.aws_apigateway_iot</code></td>
</tr>
<tr>
<td>Typescript</td>
<td><code>@aws-solutions-constructs/aws-apigateway-iot</code></td>
</tr>
<tr>
<td>Coffee</td>
<td><code>software.amazon.awsconstructs.services.apigateway</code></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. 34)

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 <code>true</code>, an API Key is created and associated to a UsagePlan. User should specify <code>x-api-key</code> header while</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

### 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. 37) 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 thingName 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 Updating device shadows (p. 38) 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 document of a thing, given its thingName and the shadowName using the Named shadow type. The body shall comply with the standard shadow structure comprising a state node</td>
</tr>
</tbody>
</table>
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.

```
```

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.

---

**aws-apigateway-kinesisstreams**

**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.
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. 39)

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>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>putRecordRequestTemplate?</td>
<td>string</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>api.ModelOptions</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>string</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>api.ModelOptions</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>kinesis.Stream</td>
<td>An optional, existing Kinesis stream to be used instead of the default stream. If an existing stream is provided, the kinesisStreamProps property will be ignored.</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>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>
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><code>{ &quot;data&quot;: &quot;Hello World!&quot;, &quot;partitionKey&quot;: &quot;pk001&quot; }</code></td>
<td><code>kinesis:PutRecord</code></td>
<td>Writes a single data record into the stream.</td>
</tr>
<tr>
<td>POST</td>
<td>/records</td>
<td><code>{ &quot;records&quot;: [ { &quot;data&quot;: &quot;abc&quot;, &quot;partitionKey&quot;: &quot;pk001&quot; }, { &quot;data&quot;: &quot;xyz&quot;, &quot;partitionKey&quot;: &quot;pk001&quot; } ] }</code></td>
<td><code>kinesis:PutRecords</code></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.

Architecture

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_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_10_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`${__dirname}/lambda`)  
  }
});
```

**Initializer**

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

**Parameters**

- scope **Construct**
- id **string**
- props **ApiGatewayToLambdaProps** (p. 44)
# 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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the lambdaFunctionProps property will be ignored.</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>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 IAM role created by the pattern that enables access logging from the API Gateway REST API to CloudWatch.</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

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

`aws-apigateway-sagemakerendpoint`
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
    #end
  ]
};

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

Initializer
new ApiGatewayToSageMakerEndpoint(scope: Construct, id: string, props: ApiGatewayToSageMakerEndpointProps);

**Parameters**

- **scope** `Construct`
- **id** `string`
- **props** `ApiGatewayToSageMakerEndpointProps` (p. 47)

### 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 pattern that enables access logging from the API Gateway REST API to CloudWatch.</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:InvokeEndpoint</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

aws-apigateway-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.
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:

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

Initializer

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

Parameters

- `scope` `Construct`
- `id` `string`
- `props` `ApiGatewayToSqsProps` (p. 50)

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.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</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 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.</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 true.</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 true.</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 true.</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 pattern that enables access logging from the API Gateway REST API to CloudWatch.</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?receiptHandle=[value]</td>
<td></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

Architecture

aws-cloudfront-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.
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:

```javascript
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', {
  existingApiGatewayObj: apiGateway
});
```

**Initializer**

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

- scope `Construct`
- id `string`
- props `CloudFrontToApiGatewayProps` (p. 55)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingApiGatewayObj</td>
<td><code>api.RestApi</code></td>
<td>The regional API Gateway that will be fronted with the CloudFront</td>
</tr>
<tr>
<td>cloudFrontDistributionProps?</td>
<td><code>cloudfront.CloudFrontWebDistribution</code></td>
<td>Optional user-provided props to override the default props for 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>
</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>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>
<tr>
<td>cloudFrontWebDistribution</td>
<td><code>cloudfront.CloudFrontWebDistribution</code></td>
<td>Returns an instance of the CloudFront web distribution 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 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

**Architecture**

![Architecture Diagram]

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

**Initializer**

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

**Parameters**

- `scope` `Construct`
- `id` `string`
- `props` `CloudFrontToApiGatewayToLambdaProps` (p. 57)

**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>An optional, existing Lambda function to be used instead</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 pattern that enables access logging from the API Gateway REST API to CloudWatch.</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>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

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

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_cloudfront_mediastore</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-cloudfront-mediastore</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.cloudfrontmediastore</td>
</tr>
</tbody>
</table>
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. 61)

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingMediaStoreContainerObj?</td>
<td><code>mediastore.CfnContainer</code></td>
<td>Optional user-provided MediaStore container to override the default MediaStore container.</td>
</tr>
<tr>
<td>mediaStoreContainerProps?</td>
<td><code>mediastore.CfnContainerProps</code></td>
<td>Optional user-provided props to override the default props for the MediaStore Container.</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>
</tbody>
</table>

**Pattern Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudFrontWebDistribution</td>
<td><code>cloudfront.CloudFrontWebDistribution</code></td>
<td>Returns an instance of the CloudFront web distribution 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

↓

Amazon Simple Storage Service

AWS Elemental MediaStore

aws-cloudfront-s3

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_cloudfront_s3</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-cloudfront-s3</td>
</tr>
</tbody>
</table>

63
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. 64)

**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.Bucket</code></td>
<td>An optional, existing bucket to be used instead of the default bucket. If an existing bucket is provided, the <code>bucketProps</code> property will be ignored.</td>
</tr>
<tr>
<td>bucketProps?</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>cloudFrontDistributionProps?</td>
<td><code>cloudfront.CloudFrontWebDistributionProps</code></td>
<td>Optional user-provided props to override the default props for CloudFront Distribution</td>
</tr>
<tr>
<td>insertHttpSecurityHeaders?</td>
<td><code>boolean</code></td>
<td>Optional user provided props to turn on/off the automatic</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>injection of best practice HTTP security headers in all responses from CloudFront</td>
<td><code>cloudfront.CloudFrontWebDistribution</code></td>
<td>Returns an instance of the CloudFront web distribution created by the pattern.</td>
</tr>
<tr>
<td>cloudFrontWebDistribution</td>
<td><code>cloudfront.CloudFrontWebDistribution</code></td>
<td>Returns an instance of the CloudFront web distribution 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>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 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-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.

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

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: {
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        runtime: lambda.Runtime.NODEJS_12_X,
        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: {
        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. 68)
# 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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the <code>lambdaFunctionProps</code> property will be ignored.</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 API Gateway</td>
</tr>
<tr>
<td>cognitoUserPoolProps?</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>cognitoUserPoolClientProps?</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>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><code>api.RestApi</code></td>
<td>Returns an instance of the API Gateway REST API 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>
<tr>
<td>userPool</td>
<td><code>cognito.UserPool</code></td>
<td>Returns an instance of the Cognito user pool created by the pattern.</td>
</tr>
<tr>
<td>userPoolClient</td>
<td><code>cognito.UserPoolClient</code></td>
<td>Returns an instance of the Cognito user pool client 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 pattern that</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 NodeJs Lambda function
- Enable X-Ray tracing
- Set environment variables:
  - AWS_NODEJS_CONNECTION_REUSE_ENABLED (for Node 10.x and higher functions)
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 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: {
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'index.handler'
    },
});
```

**Initializer**

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

**Parameters**

- `scope` `Construct`
- `id` `string`
- `props` `DynamoDBStreamToLambdaProps` (p. 71)

**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</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)

**Architecture**

```
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.
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:

```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**

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

**Parameters**

- `scope` **Construct**
- `id` **string**
- `props` **DynamoDBStreamToLambdaToElasticSearchAndKibanaProps**

(p. 75)
## 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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the <code>lambdaFunctionProps</code> property will be ignored.</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>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. If this is set, then the <code>dynamoTableProps</code> is ignored.</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 Elasticsearch Service.</td>
</tr>
<tr>
<td>domainName</td>
<td><code>string</code></td>
<td>Domain name for the Cognito and the Amazon Elasticsearch 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>dynamoTable</td>
<td><code>dynamodb.Table</code></td>
<td>Returns an instance of the DynamoDB table 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>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>
<tr>
<td>userPoolClient</td>
<td>cognito.UserPoolClient</td>
<td>Returns an instance of the Cognito user pool client 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 Elasticsearch 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 Amazon ES domain

**Architecture**

```
aws-events-rule-kinesisfirehose-s3
```

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 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:

```typescript
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);
```

Initializer

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

Parameters

- `scope` Construct
- `id` string
- `props` EventsRuleToKinesisFirehoseToS3Props (p. 79)
# Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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.Bucket</td>
<td>An optional, existing S3 bucket to be used instead of the default bucket. If an existing bucket is provided, the bucketProps property will be ignored.</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>
</tbody>
</table>

# Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>kinesisFirehoseLogGroup</td>
<td>logs.LogGroup</td>
<td>Returns an instance of the log group 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**
- 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

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_events_rule_kinesisstreams</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-events-rule-kinesisstreams</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.eventsrulekinesisstream</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-stream', props);
```

Initializer

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

Parameters

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

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>An optional, existing Kinesis stream to be used instead of the default stream. If an existing stream is provided, the kinesisStreamProps property will be ignored.</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>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.
This AWS Solutions Construct implements an AWS Events rule and an AWS Lambda function.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_events_rule_lambda</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-events-rule-lambda</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.event-rule-lambda</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.
Here is a minimal deployable pattern definition in TypeScript:

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

const props: EventsRuleToLambdaProps = {
  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 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. 85)`

**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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the lambdaFunctionProps property will be ignored.</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>eventRuleProps</td>
<td><code>events.RuleProps</code></td>
<td>User provided eventRuleProps to override the defaults</td>
</tr>
</tbody>
</table>

85
Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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

Role

Lambda function

Amazon CloudWatch

aws-events-rule-sns

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_events_rule_sns</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-events-rule-sns</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.eventsrulesns</td>
</tr>
</tbody>
</table>
Overview

This pattern implements an Amazon CloudWatch Events rule connected to an Amazon SNS topic. Here is a minimal deployable pattern definition:

```java
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);

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 EventsRuleToSNS(scope: Construct, id: string, props: EventsRuleToSnsProps);

Parameters

- scope `Construct`
- id `string`
- props `EventsRuleToSnsProps (p. 88)`

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>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingTopicObj?</td>
<td>sns.Topic</td>
<td>An optional, existing SNS topic to be used instead of the default topic. If an existing topic is provided, the topicProps property will be ignored.</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>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 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

aws-events-rule-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.

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

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

Here is a minimal deployable pattern definition:

```typescript
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

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

Parameters

- scope Construct
- id string
- props EventsRuleToSqsProps (p. 92)
# 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>existingQueueObj?</td>
<td><code>sqs.Queue</code></td>
<td>An optional, existing SQS queue to be used instead of the default queue. If an existing queue is provided, the queueProps property will be ignored.</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. Ignored if an existingQueueObj is provided.</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 15.</td>
</tr>
<tr>
<td>enableEncryptionWithCustomerManagedKey?</td>
<td><code>boolean</code></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</td>
</tr>
</tbody>
</table>
## Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 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

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><code>aws_solutions_constructs.aws_events_rule_step_function</code></td>
</tr>
<tr>
<td>Typescript</td>
<td><code>@aws-solutions-constructs/aws-events-rule-step-function</code></td>
</tr>
<tr>
<td>Java</td>
<td><code>software.amazon.awsconstructs.services.eventrulestepfunction</code></td>
</tr>
</tbody>
</table>
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(stack, 'test-events-rule-step-function-stack', props);
```

### Parameters

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

### Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stateMachineProps</td>
<td>sfn.StateMachineProps</td>
<td>Optional user provided props to override the default props for sfn.StateMachine</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>
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>eventsRule</td>
<td><code>events.Rule</code></td>
<td>Returns an instance of the Events rule created by the pattern.</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.LogGroup</code></td>
<td>Returns an instance of the log group 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 practices CloudWatch Alarms for the Step Function
Architecture

Amazon CloudWatch Event Rule

AWS Step Functions

Amazon CloudWatch

aws-iot-kinesisfirehose-s3

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_con structs.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>
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:

```typescript
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);
```

### Parameters

- **scope** `Construct`  
- **id** `string`  
- **props** `IotToKinesisFirehoseToS3Props` (p. 98)

### Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iotTopicRuleProps</code></td>
<td><code>iot.CfnTopicRuleProps</code></td>
<td>User provided CfnTopicRuleProps to override the defaults</td>
</tr>
<tr>
<td><code>kinesisFirehoseProps</code></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><code>existingBucketObj</code></td>
<td><code>s3.Bucket</code></td>
<td>An optional, existing bucket to be used instead of the default bucket. If an existing bucket is provided, the bucketProps property will be ignored.</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</td>
</tr>
</tbody>
</table>
### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

**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**

---

**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.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>aws_solutions_constructs.aws_iot_lambda</td>
</tr>
</tbody>
</table>
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: {
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    runtime: lambda.Runtime.NODEJS_10_X,
    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**

- **scope** `Construct`
- **id** `string`
- **props** `IotToLambdaProps` (p. 102)

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>@aws-solutions-constructs/aws-iot-lambda</td>
</tr>
<tr>
<td>Typescript</td>
<td>software.amazon.awsconstructs.services.iotlambda</td>
</tr>
<tr>
<td>Java</td>
<td></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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the lambdaFunctionProps property will be ignored.</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

Role

IoT rule

Lambda function

Amazon CloudWatch

aws-iot-lambda-dynamodb

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.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_iot_lambda_dynamodb</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-iot-lambda-dynamodb</td>
</tr>
<tr>
<td></td>
<td>software.amazon.awsconstructs.services.iotlambda-dynamodb</td>
</tr>
</tbody>
</table>
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: {
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    runtime: lambda.Runtime.NODEJS_12_X,
    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 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. 104)

**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. If an existing function is provided, the lambdaFunctionProps property will be ignored.</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>iotTopicRuleProps</td>
<td>iot.CfnTopicRuleProps</td>
<td>User provided props to override the default props</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>tablePermissions?</td>
<td>string</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>

### 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**

```
aws-kinesisfirehose-s3
```

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 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` Constructs
- `id` string
- `props KinesisFirehoseToS3Props (p. 107)`

### Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesisFirehoseProps?</td>
<td>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.Bucket</td>
<td>An optional, existing bucket to be used instead of the default bucket. If an existing bucket is</td>
</tr>
</tbody>
</table>
Pattern Properties

<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 properties to override the default properties for the bucket. Ignored if an existingBucketObj is provided.</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>

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

Architecture

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_kinesisfirehose_s3-and-kinesisanalytics</td>
</tr>
</tbody>
</table>
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 TypeScri

```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);
```
Parameter

- scope Construct
- id string
- props KinesisFirehoseToAnalyticsAndS3Props (p. 111)

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesisFirehoseProps?</td>
<td>kinesisFirehose.CfnDeliveryStreamProps</td>
<td>Optional user-provided props to override the default props for the Kinesis Firehose delivery stream.</td>
</tr>
<tr>
<td>kinesisAnalyticsProps?</td>
<td>kinesisAnalytics.CfnApplicationProps</td>
<td>Optional user-provided props to override the default props for the Kinesis Analytics application.</td>
</tr>
<tr>
<td>existingBucketObj?</td>
<td>s3.Bucket</td>
<td>An optional, existing bucket to be used instead of the default bucket. If an existing bucket is provided, the bucketProps property will be ignored.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user-provided properties to override the default properties for the bucket. Ignored if an existingBucketObj is provided.</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>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>
</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

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>
Architecture

aws-kinesisstreams-kinesisfirehose-s3

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_kinesisstreams_kinesisfirehose_s3</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-kinesis-streams-kinesis-firehose-s3</td>
</tr>
<tr>
<td>Java</td>
<td>software.amazon.awsconstructs.services.kinesisstreams_kinesisfirehose_s3</td>
</tr>
</tbody>
</table>

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:
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. 114)

Pattern Construct Props

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesisFirehoseProps?</td>
<td>kinesisFirehose.CfnDeliveryStreamProps</td>
<td>Optional user-provided props to override the default props for the Kinesis Firehose delivery stream.</td>
</tr>
<tr>
<td>existingBucketObj?</td>
<td>s3.IBucket</td>
<td>An optional, existing bucket to be used instead of the default bucket. If an existing bucket is provided, the bucketProps property will be ignored.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td>s3.BucketProps</td>
<td>Optional user-provided properties to override the default properties for the bucket. Ignored if an existingBucketObj 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>An optional, existing Kinesis stream to be used instead of the default stream. If an existing stream is provided, the kinesisStreamProps property will be ignored.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</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>

### 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 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>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>kinesisStream</td>
<td>kinesis.Stream</td>
<td>Returns an instance of the Kinesis stream 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 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**

```python
aws-kinesisstreams-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.
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', {
  eventSourceProps: {
    startingPosition: lambda.StartingPosition.TRIM_HORIZON,
    batchSize: 1
  },
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_10_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`${__dirname}/lambda`)
  }
});
```

**Initializer**

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

**Parameters**

- **scope** `Construct`
- **id** `string`
- **props** `KinesisStreamsToLambdaProps` (p. 118)
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. If an existing function is provided, the lambdaFunctionProps property will be ignored.</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>An optional, existing Kinesis stream to be used instead of the default stream. If an existing stream is provided, the kinesisStreamProps property will be ignored.</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>
</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)

---

<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>
</tbody>
</table>
## Architecture

![Architecture Diagram](https://via.placeholder.com/150)

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><code>aws_solutions_con structs.aws_lambda_dynamodb</code></td>
</tr>
<tr>
<td>Typescript</td>
<td><code>@aws-solutions-constructs/aws-lambda-dynamodb</code></td>
</tr>
<tr>
<td>Java</td>
<td><code>software.amazon.awsconstructs.services.lambdadynamodb</code></td>
</tr>
</tbody>
</table>
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: {
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        runtime: lambda.Runtime.NODEJS_12_X,
        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. 121)`

### 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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the lambdaFunctionProps property will be ignored.</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>
</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>dynamoTableProps?</td>
<td>dynamodb.TableProps</td>
<td>Optional user provided props to override the default props for DynamoDB Table</td>
</tr>
<tr>
<td>existingTableObj?</td>
<td>dynamodb.Table</td>
<td>Existing instance of DynamoDB Table. If this is set, then the dynamoTableProps is ignored.</td>
</tr>
<tr>
<td>tablePermissions?</td>
<td>string</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>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>
</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
  - 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

aws-lambda-elasticsearch-kibana

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.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_lambda_elasticsearch-kibana</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-lambda-elasticsearch-kibana</td>
</tr>
<tr>
<td></td>
<td>software.amazon.awsconstructs.services.lambdaelasticsearchkibana</td>
</tr>
</tbody>
</table>
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 = {
  code: lambda.Code.fromAsset(`${__dirname}/lambda`),
  runtime: lambda.Runtime.NODEJS_12_X,
  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. 124)*

**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. If an existing function is provided, the lambdaFunctionProps property will be ignored.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for 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>esDomainProps?</td>
<td><code>elasticsearch.CfnDomainProps</code></td>
<td>Optional user provided props to override the default props for the Amazon Elasticsearch Service</td>
</tr>
<tr>
<td>domainName</td>
<td><code>string</code></td>
<td>Domain name for the Cognito and the Amazon Elasticsearch Service</td>
</tr>
<tr>
<td>cognitoDomainName?</td>
<td><code>string</code></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><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>elasticsearchDomain</td>
<td><code>elasticsearch.CfnDomain</code></td>
<td>Returns an instance of the Elasticsearch domain created by the pattern.</td>
</tr>
<tr>
<td>elasticsearchDomainRole</td>
<td><code>iam.Role</code></td>
<td>Returns an instance of the IAM role created by the pattern for the Elasticsearch domain.</td>
</tr>
<tr>
<td>identityPool</td>
<td><code>cognito.CfnIdentityPool</code></td>
<td>Returns an instance of the Cognito identity pool 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>
<tr>
<td>userPool</td>
<td><code>cognito.UserPool</code></td>
<td>Returns an instance of the Cognito user pool created by the pattern.</td>
</tr>
<tr>
<td>userPoolClient</td>
<td><code>cognito.UserPoolClient</code></td>
<td>Returns an instance of the Cognito user pool client created by the pattern.</td>
</tr>
</tbody>
</table>

125
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`
  - `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 Elasticsearch 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 Amazon ES domain.
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 AWS Lambda function connected to an Amazon S3 bucket. Here is a minimal deployable pattern definition in TypeScript:

```typescript
import { LambdaToS3 } from '@aws-solutions-constructs/aws-lambda-s3';

new LambdaToS3(this, 'LambdaToS3Pattern', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_10_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`${__dirname}/lambda`)
  }
});
```

**Initializer**

```typescript
new LambdaToS3(scope: Construct, id: string, props: LambdaToS3Props);
```

**Parameters**
- `scope` **Construct**
- `id` **string**
- `props` **LambdaToS3Props** (p. 128)

**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. If an existing function is provided,</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. Ignored if an existingLambdaObj is provided.</td>
</tr>
<tr>
<td>existingBucketObj?</td>
<td><code>{s3.Bucket}</code></td>
<td>An optional, existing bucket to be used instead of the default bucket. If an existing bucket is provided, the bucketProps property will be ignored.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td><code>{s3.BucketProps}</code></td>
<td>Optional user-provided properties to override the default properties for the bucket. Ignored if an existingBucketObj is provided.</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 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>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`. |
| 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 overriden. If `deployVpc` is not `true` then this property will be ignored. |

**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>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>vpc?</td>
<td><code>ec2.IVpc</code></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>
</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
  - 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**

Architecture diagram showing AWS Lambda Function connected to an Amazon S3 Bucket through a Role.

***aws-lambda-sns***

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 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_12_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`${__dirname}/lambda`)
  }
});
```

**Initializer**

```typescript
new LambdaToSns(scope: Construct, id: string, props: LambdaToSnsProps);
```

**Parameters**
- `scope` `Construct`
- `id` `string`
- `props` `LambdaToSnsProps` (p. 132)

**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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided,</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>lambda.FunctionProps</td>
<td>Optional user-provided properties to override the default properties for the Lambda function. Ignored if an existing Lambda object is provided.</td>
</tr>
<tr>
<td>existingTopicObj?</td>
<td>sns.Topic</td>
<td>An optional, existing SNS topic to be used instead of the default topic. If an existing topic is provided, the topicProps property will be ignored.</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>
<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 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 ec2.Vpc.fromLookup() 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.  
|                      |                 | 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>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>
</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
  - SNS_TOPIC_ARN
  - 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.
This AWS Solutions Construct implements an AWS Lambda function connected to an Amazon SQS queue.

Here is a minimal deployable pattern definition in TypeScript:

```javascript
import { LambdaToSqs, LambdaToSqsProps } from '@aws-solutions-constructs/aws-lambda-sqs';
new LambdaToSqs(this, 'LambdaToSqsPattern', {
```
AWS Solutions Constructs AWS Solutions
Initializer

```javascript
lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`${__dirname}/lambda`)
};
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployDeadLetterQueue?</td>
<td>boolean</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>sqs.QueueProps</td>
<td>Optional user-provided props to override the default props for the dead letter queue. Only used if the <code>deployDeadLetterQueue</code> property is set 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. Defaults to <code>15</code>.</td>
</tr>
<tr>
<td>existingVpc?</td>
<td>ec2.IVpc</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 <code>deployVpc</code> 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>
<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 <code>true</code> 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`. |
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>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>
<tr>
<td>vpc?</td>
<td>ec2.IVpc</td>
<td>Returns an instance of the VPC created or used by the pattern (if any). This may be a VPC created by the pattern or a VPC supplied to the pattern constructor.</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.
- 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)

aws-lambda-sqs-lambda

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.

<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>

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**

- `scope` `Construct`
- `id` `string`
- `props` `LambdaToSqsToLambdaProps` (p. 141)
# 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. If an existing function is provided, the producerLambdaFunctionProps property will be ignored.</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. Ignored if an existingProducerLambdaObj 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. If an existing queue is provided, the queueProps property will be ignored.</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>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. If an existing function is provided, the consumerLambdaFunctionProps property will be ignored.</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. Ignored if an existingConsumerLambdaObj is provided.</td>
</tr>
<tr>
<td>consumerLambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the consumer Lambda function 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>producerLambdaFunction</td>
<td>lambda.Function</td>
<td>Returns an instance of the producer 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 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
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_lambda_step_function</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-lambda-step-function</td>
</tr>
</tbody>
</table>
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(stack, 'StartState');

new LambdaToStepFunction(this, 'LambdaToStepFunctionPattern', {
  lambdaFunctionProps: {
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(`#{dirname}/lambda`)
  },
  stateMachineProps: {
    definition: startState
  }
});
```

### Initializer

new LambdaToStepFunction(scope: Construct, id: string, props: LambdaToStepFunctionProps);

### Parameters

- **scope** `Construct`
- **id** `string`
- **props** `LambdaToStepFunctionProps` (p. 144)

### 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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the lambdaFunctionProps property will be ignored.</td>
</tr>
<tr>
<td>lambdaFunctionProps?</td>
<td><code>lambda.FunctionProps</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>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>

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.LogGroup</td>
<td>Returns an instance of the log group 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:

**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`
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED` (for Node 10.x and higher functions)

**AWS Step Functions State Machine**

- Deploy best-practice CloudWatch alarms for the AWS Step Functions State Machine.
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_s3_lambda</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-s3-lambda</td>
</tr>
</tbody>
</table>
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: {
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'index.handler'
    },
});
```

### Initializer

```typescript
new S3ToLambda(scope: Construct, id: string, props: S3ToLambdaProps);
```

### Parameters

- `scope` `Construct`
- `id` `string`
- `props` `S3ToLambdaProps (p. 147)`

### 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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the <code>lambdaFunctionProps</code> property will be ignored.</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>existingBucketObj?</td>
<td><code>s3.Bucket</code></td>
<td>An optional, existing bucket to be used instead of the default bucket.</td>
</tr>
</tbody>
</table>

[Image of the page]
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bucketProps?</td>
<td>Optional user-provided properties to override the default properties for the bucket. Ignored if an existingBucketObj is provided.</td>
</tr>
<tr>
<td></td>
<td>s3Bucket?</td>
<td>Returns an instance of the S3 bucket created by the pattern.</td>
</tr>
<tr>
<td></td>
<td>s3LoggingBucket?</td>
<td>Returns an instance of the logging bucket created by the pattern 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>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

```plaintext
Role → Lambda function → Amazon CloudWatch
```

### `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.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td><code>aws_solutions_constructs.aws_s3_sqs</code></td>
</tr>
<tr>
<td>Typescript</td>
<td><code>@aws-solutions-constructs/aws-s3-sqs</code></td>
</tr>
<tr>
<td></td>
<td><code>software.amazon.awsconstructs.services.s3sqs</code></td>
</tr>
</tbody>
</table>
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. 150)

**Pattern Construct Props**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existingBucketObj?</td>
<td>s3.Bucket</td>
<td>An optional, existing S3 bucket to be used instead of the default bucket. If an existing bucket is provided, the <code>bucketProps</code> property will be ignored.</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>s3EventTypes?</td>
<td>s3.EventType[]</td>
<td>The S3 event types that will trigger the notification. Defaults to <code>s3.EventType.OBJECT_CREATED</code>.</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>An optional, existing SQS queue to be used instead of the default queue. If an existing queue is</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</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>boolean</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, 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>

**Pattern Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
AWS Solutions Constructs AWS Solutions

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
</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.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>aws_solutions_constructs.aws_s3_step_function</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-s3-step-function</td>
</tr>
<tr>
<td></td>
<td>software.amazon.awsconstructs.services.s3stepfunction</td>
</tr>
</tbody>
</table>
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(stack, '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. 154)*

**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.Bucket</code></td>
<td>An optional, existing bucket to be used instead of the default bucket. If an existing bucket is provided, the <code>bucketProps</code> property will be ignored.</td>
</tr>
<tr>
<td>bucketProps?</td>
<td><code>s3.BucketProps</code></td>
<td>Optional user-provided properties to override the default properties for the bucket. Ignored if an</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>
</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.
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><code>aws_solutions_constructs.aws_sns_lambda</code></td>
</tr>
<tr>
<td>Typescript</td>
<td><code>@aws-solutions-constructs/aws-sns-lambda</code></td>
</tr>
<tr>
<td>Java</td>
<td><code>software.amazon.awsconstructs.services.snslambda</code></td>
</tr>
</tbody>
</table>

This AWS Solutions Construct implements an Amazon SNS connected to an AWS Lambda function. Here is a minimal deployable pattern definition in TypeScript:
import { SnsToLambda, SnsToLambdaProps } from "@aws-solutions-constructs/aws-sns-lambda";

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

### Initializer

```js
new SnsToLambda(scope: Construct, id: string, props: SnsToLambdaProps);
```

### Parameters

- **scope** `Construct`
- **id** `string`
- **props** `SnsToLambdaProps (p. 158)`

### 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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the <code>lambdaFunctionProps</code> property will be ignored.</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>existingTopicObj?</td>
<td><code>sns.Topic</code></td>
<td>An optional, existing SNS topic to be used instead of the default topic. If an existing topic is provided, the <code>topicProps</code> property will be ignored.</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>
</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)
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_sns_sqs</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-sns-sqs</td>
</tr>
<tr>
<td></td>
<td>software.amazon.awsconstructs.services.snssqs</td>
</tr>
</tbody>
</table>
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. 161)*

**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>An optional, existing SNS topic to be used instead of the default topic. If an existing topic is provided, the topicProps property will be ignored.</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</td>
</tr>
</tbody>
</table>
### Name | Type | Description
---|---|---
queueProps? | sqs.QueueProps | Optional user-provided properties to override the default properties for the SQS queue. Ignored if an existingQueueObj is provided.
deployDeadLetterQueue? | boolean | Whether to create a secondary queue to be used as a dead letter queue. Defaults to true.
deadLetterQueueProps? | sqs.QueueProps | Optional user-provided props to override the default props for the dead letter queue. Only used if the deployDeadLetterQueue property is set to true.
maxReceiveCount? | number | The number of times a message can be unsuccessfully dequeued before being moved to the dead letter queue. Defaults to 15.
enableEncryptionWithCustomerManagedKey? |  | 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.
enryptionKey? | kms.Key | An optional, existing encryption key to be used instead of the default encryption key.
enryptionKeyProps? | kms.KeyProps | Optional user-provided properties to override the default properties for the encryption key.

### Pattern Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
snsTopic | sns.Topic | Returns an instance of the SNS topic created by the pattern. |
enryptionKey | kms.Key | Returns an instance of the encryption key created by the pattern. |
## 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.

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
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_sqs_lambda</td>
</tr>
<tr>
<td>Typescript</td>
<td>@aws-solutions-constructs/aws-sqs-lambda</td>
</tr>
<tr>
<td></td>
<td>software.amazon.awsconstructs.services.sqslambda</td>
</tr>
</tbody>
</table>
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_10_X,
    handler: 'index.handler',
    code: lambda.Code.fromAsset(path.join(__dirname, '/lambda'))
  }
});
```

### Initializer

```typescript
new SqsToLambda(scope: Construct, id: string, props: SqsToLambdaProps);
```

**Parameters**

- **scope** `Construct`
- **id** `string`
- **props** `SqsToLambdaProps` (p. 165)

### 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>An optional, existing Lambda function to be used instead of the default function. If an existing function is provided, the lambdaFunctionProps property will be ignored.</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>existingQueueObj?</td>
<td><code>sqs.Queue</code></td>
<td>An optional, existing SQS queue to be used instead of the default queue. If an existing queue is provided, the existingQueueObj property will be ignored.</td>
</tr>
</tbody>
</table>
Name | Type | Description
--- | --- | ---
| queueProps? | sqs.QueueProps | Optional user-provided properties to override the default properties for the SQS queue. Ignored if an existingQueueObj is provided.
| deployDeadLetterQueue? | boolean | Whether to create a secondary queue to be used as a dead letter queue. Defaults to true.
| deadLetterQueueProps? | sqs.QueueProps | Optional user-provided props to override the default props for the dead letter queue. Only used if the deployDeadLetterQueue property is set to true.
| maxReceiveCount? | number | The number of times a message can be unsuccessfully dequeued before being moved to the dead letter queue. Defaults to 15.

Pattern Properties

| Name | Type | Description
--- | --- | ---
| deadLetterQueue? | sqs.Queue | Returns an instance of the dead letter queue created by the pattern, if one is deployed.
| lambdaFunction | lambda.Function | Returns an instance of the Lambda function created by the pattern.
| sqsQueue | sqs.Queue | Returns an instance of the SQS queue created by the pattern.

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

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.

```javascript
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. 170)</td>
<td>Updated properties for select patterns.</td>
<td>February 1, 2021</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Updated documentation of properties and default settings for select patterns.</td>
<td>January 4, 2021</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Added new patterns: aws-cloudfront-mediastore and aws-s3-sqs.</td>
<td>December 20, 2020</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Removed aws-lambda-sagemaker pattern.</td>
<td>November 17, 2020</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Added new patterns: aws-events-rule-kinesisstreams, aws-events-rule-kinesisfirehose-s3, and aws-lambda-sagemaker.</td>
<td>October 27, 2020</td>
</tr>
<tr>
<td>Content updated (p. 170)</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>October 22, 2020</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Added aws-apigateway-sagemakerendpoint and aws-kinesisstreams-kinesisfirehose-s3 patterns; other minor updates to existing content.</td>
<td>October 20, 2020</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Added aws-apigateway-iot pattern; other minor updates to existing content.</td>
<td>October 7, 2020</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Updated minimal deployable pattern code snippets and best practice defaults for all patterns.</td>
<td>October 5, 2020</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Updated properties for aws-kinesisstreams-lambda pattern to reflect breaking change.</td>
<td>September 14, 2020</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Minor fix to second part of walkthrough.</td>
<td>September 10, 2020</td>
</tr>
<tr>
<td>Content updated (p. 170)</td>
<td>Added aws-apigateway-kinesisstreams, aws-events-rule-sns, and aws-events-rule-sqs patterns.</td>
<td>September 10, 2020</td>
</tr>
<tr>
<td>Date</td>
<td>Updates</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------</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>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>

Initial release (p. 170) AWS Solutions Constructs made publicly available.
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.