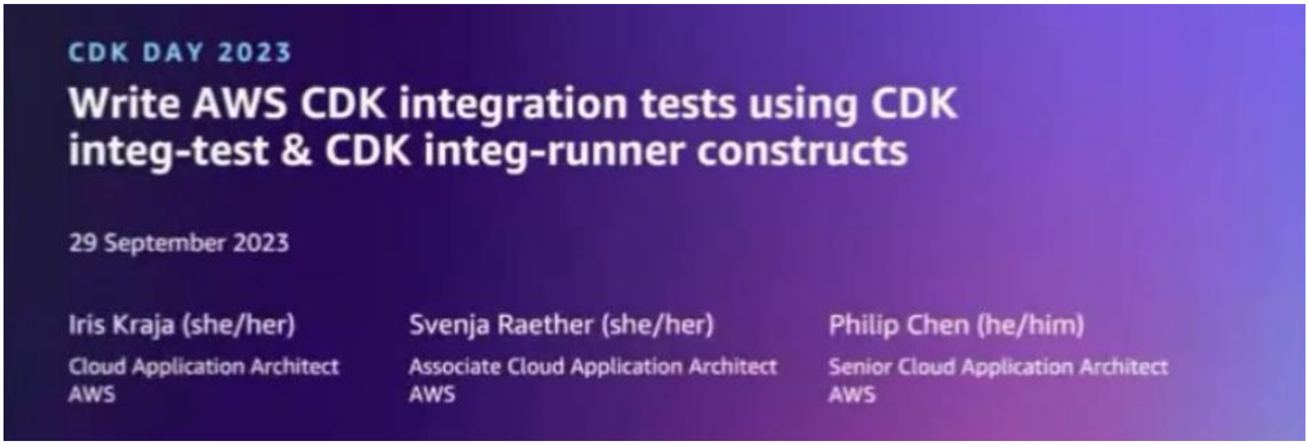


CDK DAY 2023

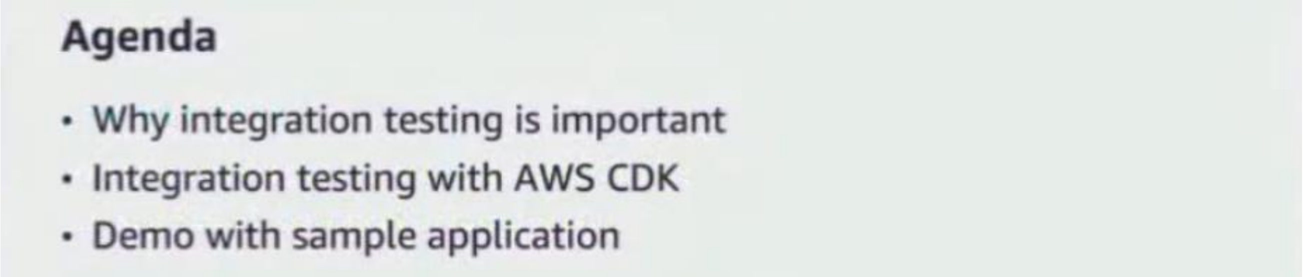
Write AWS CDK integration tests using CDK integ-test & CDK integ-runner constructs

29 September 2023

Iris Kraja (she/her)
Cloud Application Architect
AWS

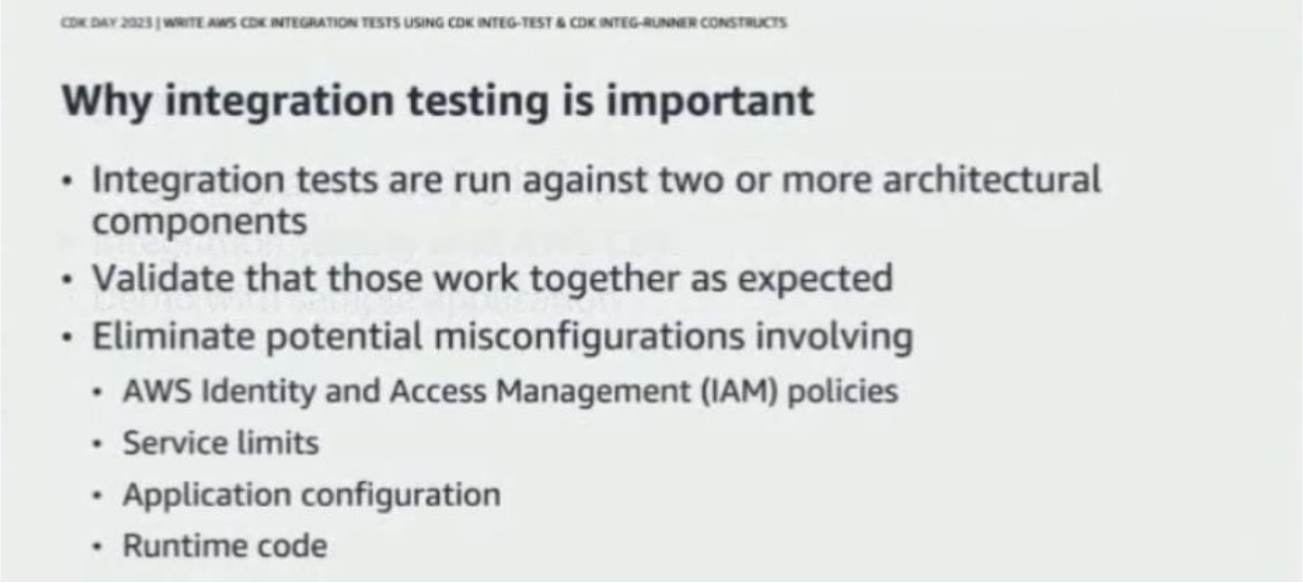
Svenja Raether (she/her)
Associate Cloud Application Architect
AWS

Philip Chen (he/him)
Senior Cloud Application Architect
AWS



Agenda

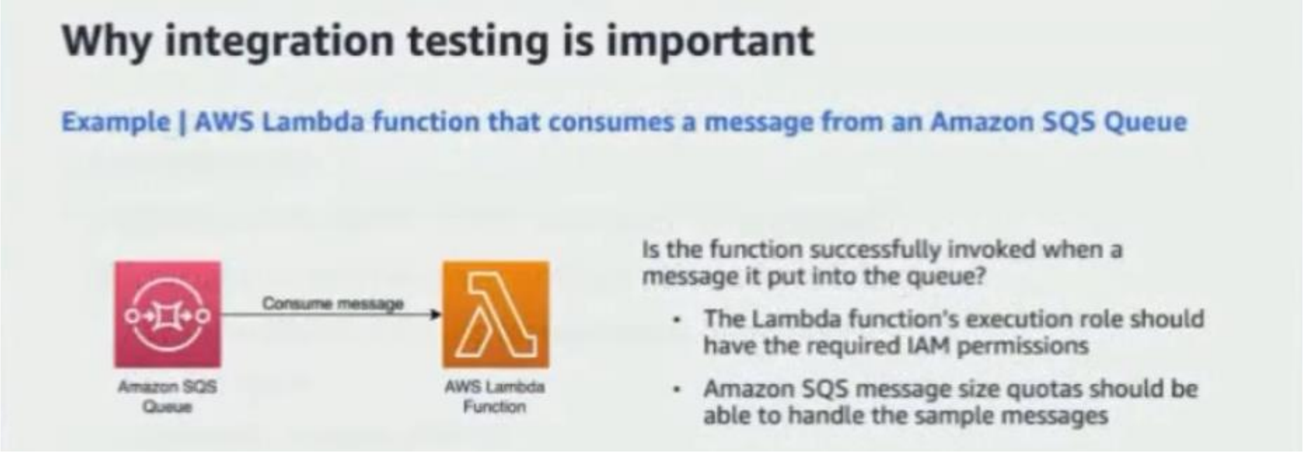
- Why integration testing is important
- Integration testing with AWS CDK
- Demo with sample application



CDK DAY 2023 | WRITE AWS CDK INTEGRATION TESTS USING CDK INTEG-TEST & CDK INTEG-RUNNER CONSTRUCTS

Why integration testing is important

- Integration tests are run against two or more architectural components
- Validate that those work together as expected
- Eliminate potential misconfigurations involving
  - AWS Identity and Access Management (IAM) policies
  - Service limits
  - Application configuration
  - Runtime code

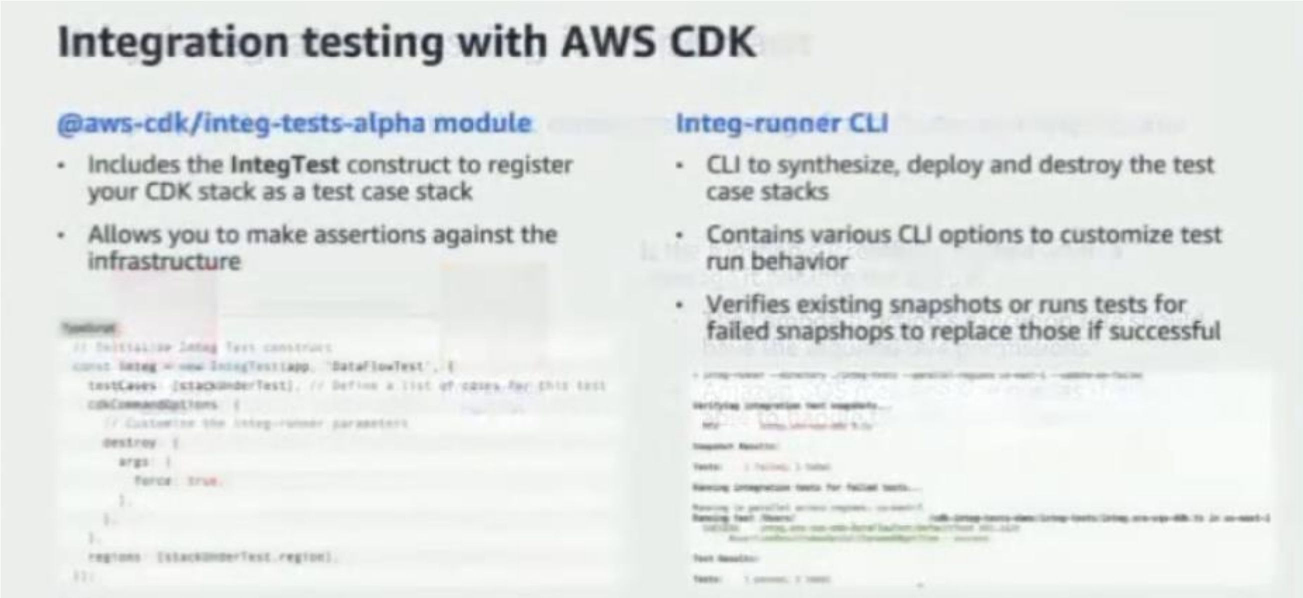


Why integration testing is important

Example | AWS Lambda function that consumes a message from an Amazon SQS Queue

Is the function successfully invoked when a message it put into the queue?

- The Lambda function's execution role should have the required IAM permissions
- Amazon SQS message size quotas should be able to handle the sample messages



Integration testing with AWS CDK

@aws-cdk/integ-tests-alpha module

- Includes the **IntegTest** construct to register your CDK stack as a test case stack
- Allows you to make assertions against the infrastructure

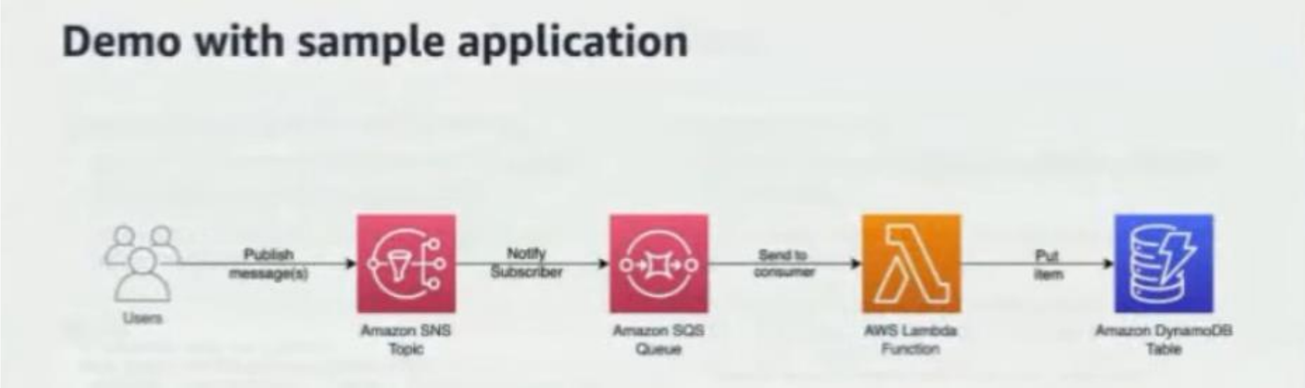
TypeScript

```
// Initialize Integ Test construct
const integ = new IntegTest(app, 'DataFlowTest', {
  testCases: [stackUnderTest], // Define a list of cases for this test
  cdkCommandOptions: {
    // Customize the integ-runner parameters
    destroy: {
      args: {
        force: true,
      },
    },
  },
  regions: [stackUnderTest.region],
});
```

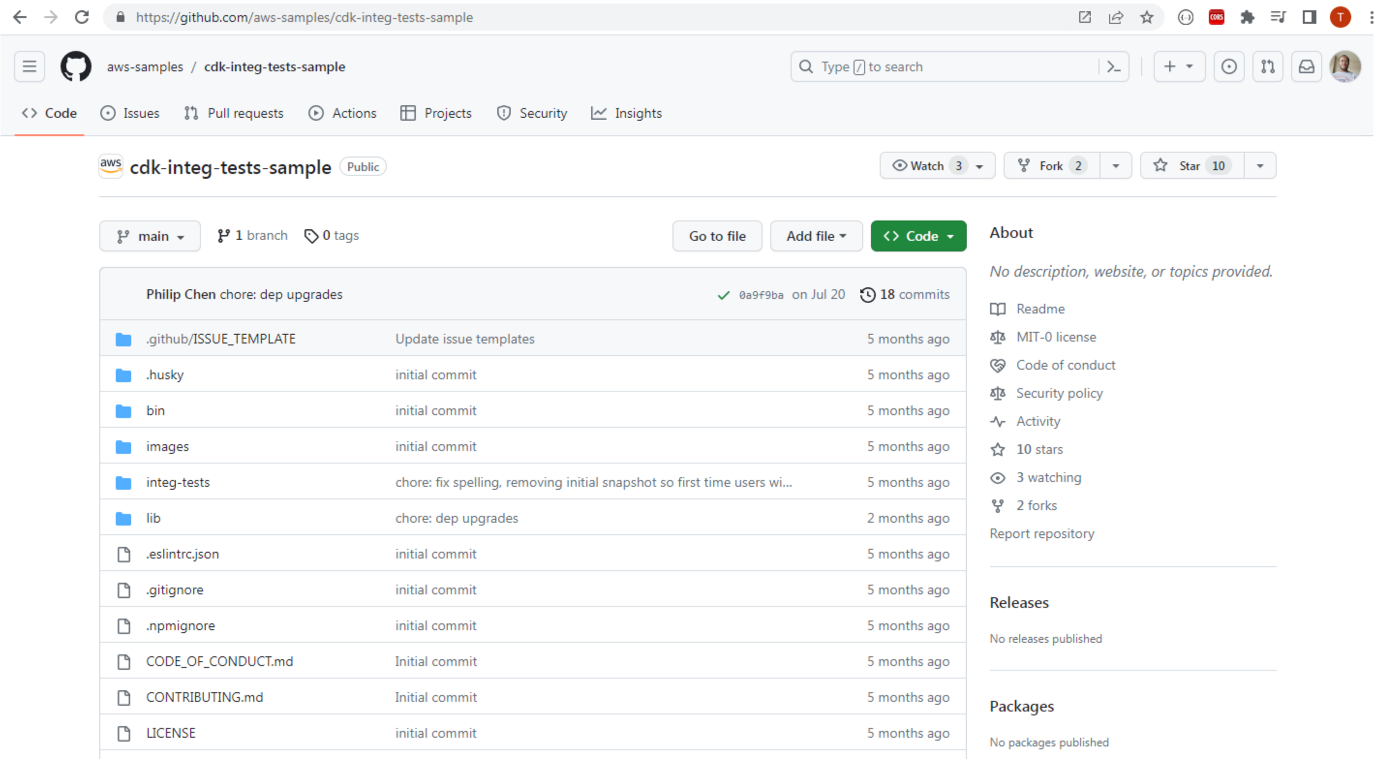
Integ-runner CLI

- CLI to synthesize, deploy and destroy the test case stacks
- Contains various CLI options to customize test run behavior
- Verifies existing snapshots or runs tests for failed snapshops to replace those if successful

```
> integ-runner --directory ./integ-tests --parallel-regions us-east-1 --update-on-failed

Verifying integration test snapshots...
  NEW       integ.sns-sqs-ddb.ts 0.1s

Snapshot Results:

Tests:    1 failed, 1 total

Running integration tests for failed tests...

Running in parallel across regions: us-east-1
Running test /Users/[illegible]/cdk-integ-tests-demo/integ-tests/integ.sns-sqs-ddb.ts in us-east-1
  SUCCESS    integ.sns-sqs-ddb-[illegible]
      [illegible] - success

Test Results:

Tests:    1 passed, 1 total
```



Demo with sample application



https://github.com/aws-samples/cdk-integ-tests-sample

aws cdk-integ-tests-sample Public

1 branch 0 tags

Philip Chen chore: dep upgrades ✓ 0a9f9ba on Jul 20 18 commits

.github/ISSUE_TEMPLATE	Update issue templates	5 months ago
.husky	initial commit	5 months ago
bin	initial commit	5 months ago
images	initial commit	5 months ago
integ-tests	chore: fix spelling, removing initial snapshot so first time users wi...	5 months ago
lib	chore: dep upgrades	2 months ago
.eslintrc.json	initial commit	5 months ago
.gitignore	initial commit	5 months ago
.npmignore	initial commit	5 months ago
CODE_OF_CONDUCT.md	Initial commit	5 months ago
CONTRIBUTING.md	Initial commit	5 months ago
LICENSE	initial commit	5 months ago

About

No description, website, or topics provided.

- Readme
- MIT-0 license
- Code of conduct
- Security policy
- Activity
- 10 stars
- 3 watching
- 2 forks

Report repository

Releases

No releases published

Packages

No packages published

https://github.com/aws-samples/cdk-integ-tests-sample

AWS DevOps Blog

How to write and execute integration tests for AWS CDK applications

by Svenja Raether, Ahmed Bakry, Iris Kraja, and Philip Chen | on 20 JUL 2023 | in AWS Cloud Development Kit, Integration & Automation, Intermediate (200), Technical How-To | Permalink | Share

Automated integration testing validates system components and boosts confidence for new software releases. Performing integration tests on resources deployed to the AWS cloud enables the validation of AWS Identity and Access Management (IAM) policies, service limits, application configuration, and runtime code. For developers that are currently leveraging AWS Cloud Development Kit (AWS CDK) as their Infrastructure as Code tool, there is a testing framework available that makes integration testing easier to implement in the software release.

AWS CDK is an open-source framework for defining and provisioning AWS cloud infrastructure using supported programming languages. The framework includes constructs for writing and running unit and integration tests. The assertions construct can be used to write unit tests and assert against the generated CloudFormation templates. CDK integ-tests construct can be used for defining integration test cases and can be combined with CDK integ-runner for executing these tests. The integ-runner handles automatic resource provisioning and removal and supports several customization options. Unit tests using assertion functions are used to test configurations in the CloudFormation templates before deploying these templates, while integration tests run assertions in the deployed resources. This blog post demonstrates writing automated integration tests for an example application using AWS CDK.

Solution Overview

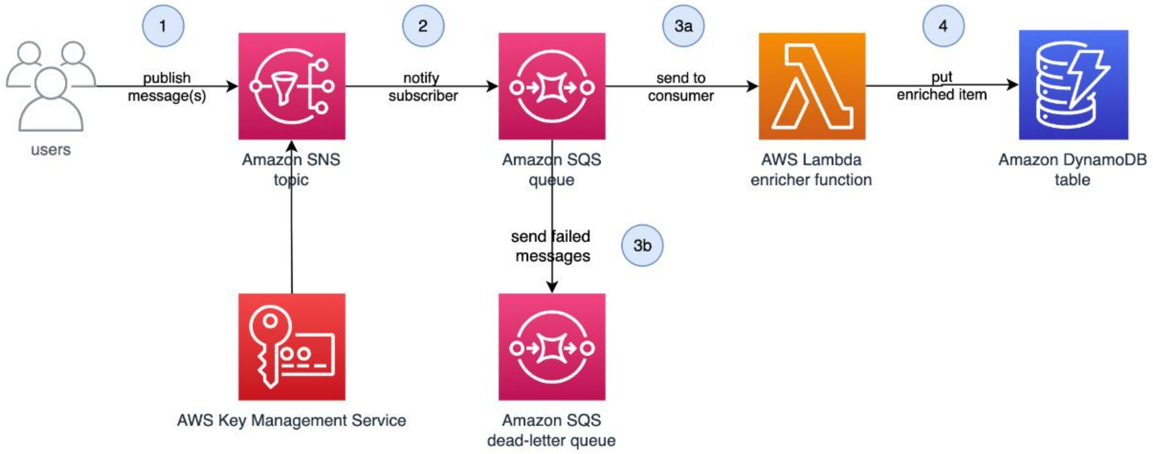


Figure 1: Serverless data enrichment application

The example application shown in Figure 1 is a sample serverless data enrichment application. Data is processed and enriched in the system as follows:

1. Users publish messages to an Amazon Simple Notification Service (Amazon SNS) topic. Messages are encrypted at rest using an AWS Key Management Service (AWS KMS) customer-managed key.
2. Amazon Simple Queue Service (Amazon SQS) queue is subscribed to the Amazon SNS topic, where published messages are delivered.
3. AWS Lambda consumes messages from the Amazon SQS queue, adding additional data to the message. Messages that cannot be processed successfully are sent to a dead-letter queue.
4. Successfully enriched messages are stored in an Amazon DynamoDB table by the Lambda function.

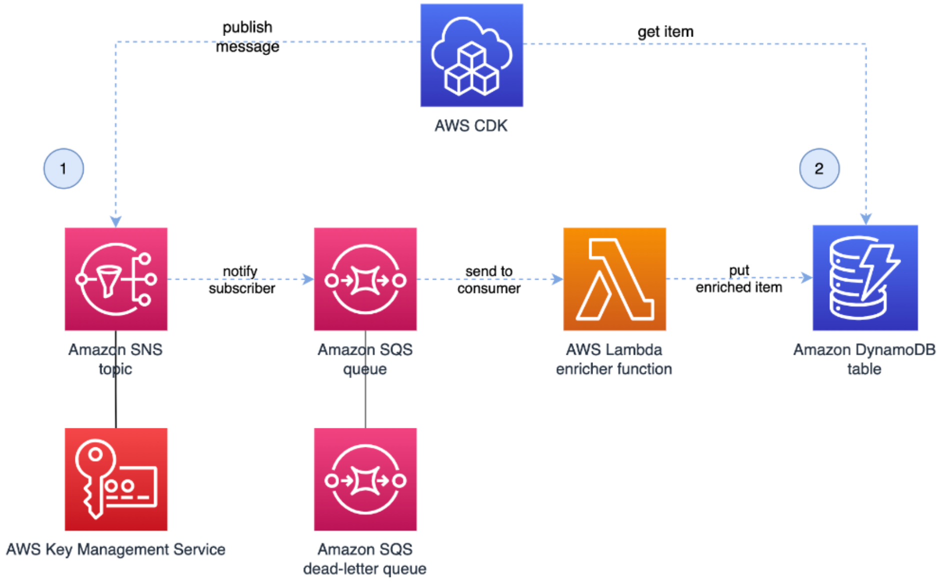


Figure 2: Integration test with one assertion

For this sample application, we will use AWS CDK's integration testing framework to validate the processing for a single message as shown in Figure 2. To run the test, we configure the test framework to do the following steps:

1. Publish a message to the Amazon SNS topic. Wait for the application to process the message and save to DynamoDB.
2. Periodically check the Amazon DynamoDB table and verify that the saved message was enriched.

Prerequisites

The following are the required to deploy this solution:

- An AWS account
- Node.js v16 or later and npm version 9 or later are installed
- Install AWS CDK version 2.73.0 or later
- Clone the GitHub repository and install the dependencies
- Run CDK Bootstrap on your AWS Account, in N. Virginia (us-east-1) region

The structure of the sample AWS CDK application repository is as follows:

- **/bin** folder contains the top-level definition of the AWS CDK app.
- **/lib** folder contains the stack definition of the application under test which defines the application described in the section above.
- **/lib/functions** contains the Lambda function runtime code.
- **/integ-tests** contains the integration test stack where we define and configure our test cases.

The repository is a typical AWS CDK application except that it has one additional directory for the test case definitions. For the remainder of this blog post, we focus on the integration test definition in /integ-tests/integ.sns-sqs-ddb.ts and walk you through its creation and the execution of the integration test.

Writing integration tests

An integration test should validate expected behavior of your AWS CDK application. You can define an integration test for your application as follows:

1. Create a stack under test from the *CdkIntegTestsDemoStack* definition and map it to the application.

TypeScript

```
// CDK App for Integration Tests
const app = new cdk.App();

// Stack under test
const stackUnderTest = new CdkIntegTestsDemoStack(app, ‘IntegrationTestStack’, {
  setDestroyPolicyToAllResources: true,
  description:
    “This stack includes the application’s resources for integration testing.”,
});
```

2. Define the integration test construct with a list of test cases. This construct offers the ability to customize the behavior of the integration runner tool. For example, you can force the integ-runner to destroy the resources after the test run to force the cleanup.

TypeScript

```
// Initialize Integ Test construct
const integ = new IntegTest(app, ‘DataFlowTest’, {
  testCases: [stackUnderTest], // Define a list of cases for this test
  cdkCommandOptions: {
    // Customize the integ-runner parameters
    destroy: {
      args: {
        force: true,
      },
    },
  },
  regions: [stackUnderTest.region],
});
```

3. Add an assertion to validate the test results. In this example, we validate the single message flow from the Amazon SNS topic to the Amazon DynamoDB table. The assertion publishes the message object to the Amazon SNS topic using the AwsApiCall method. In the background this method utilizes a Lambda-backed CloudFormation custom resource to execute the Amazon SNS Publish API call with the AWS SDK for JavaScript.

TypeScript

```
/**
 * Assertion:
 * The application should handle single message and write the enriched item to the DynamoDB table.
 */
const id = 'test-id-1';
const message = 'This message should be validated';
/**
 * Publish a message to the SNS topic.
 * Note - SNS topic ARN is a member variable of the
 * application stack for testing purposes.
 */
const assertion = integ.assertions
  .awsApiCall('SNS', 'publish', {
    TopicArn: stackUnderTest.topicArn,
    Message: JSON.stringify({
      id: id,
      message: message,
    }),
  })
```

4. Use the *next* helper method to chain API calls. In our example, a second Amazon DynamoDB GetItem API call gets the item whose primary key equals the message id. The result from the second API call is expected to match the message object including the additional attribute added as a result of the data enrichment.

TypeScript

```
/**
 * Validate that the DynamoDB table contains the enriched message.
 */
  .next(
    integ.assertions
      .awsApiCall('DynamoDB', 'getItem', {
        TableName: stackUnderTest.tableName,
        Key: { id: { S: id } },
      })
      /**
       * Expect the enriched message to be returned.
       */
      .expect(
        ExpectedResult.objectLike({
          Item: { id: { S: id, },
            message: { S: message, },
            additionalAttr: { S: 'enriched', },
          },
        }),
      )
```

5. Since it may take a while for the message to be passed through the application, we run the assertion asynchronously by calling the *waitForAssertions* method. This means that the Amazon DynamoDB GetItem API call is called in intervals until the expected result is met or the total timeout is reached.

TypeScript

```
/**
 * Timeout and interval check for assertion to be true.
 * Note - Data may take some time to arrive in DynamoDB.
 * Iteratively executes API call at specified interval.
 */
      .waitForAssertions({
        totalTimeout: Duration.seconds(25),
        interval: Duration.seconds(3),
      }),
  );
```

6. The AwsApiCall method automatically adds the correct IAM permissions for both API calls to the AWS Lambda function. Given that the example application's Amazon SNS topic is encrypted using an AWS KMS key, additional permissions are required to publish the message.

TypeScript

```
// Add the required permissions to the api call
assertion.provider.addToRolePolicy({
  Effect: 'Allow',
  Action: [
    'kms:Encrypt',
    'kms:ReEncrypt*',
    'kms:GenerateDataKey*',
    'kms:Decrypt',
  ],
  Resource: [stackUnderTest.kmsKeyArn],
});
```

The full code for this blog is available on this GitHub project.

Running integration tests

In this section, we show how to run integration test for the introduced sample application using the integ-runner to execute the test case and report on the assertion results.

Install and build the project.

Bash
```bash
npm install

npm run build
```

Run the following command to initiate the test case execution with a list of options.

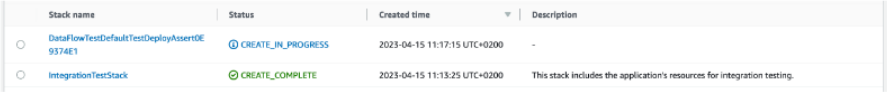
Bash
```bash
npm run integ-test
```

The *directory* option specifies in which location the integ-runner needs to recursively search for test definition files. The *parallel-regions* option allows to define a list of regions to run tests in. We set this to us-east-1 and ensure that the AWS CDK bootstrapping has previously been performed in this region. The *update-on-failed* option allows to rerun the integration tests if the snapshot fails. A full list of available options can be found in the integ-runner Github repository.

Hint: if you want to retain your test stacks during development for debugging, you can specify the *no-clean* option to retain the test stack after the test run.

The integ-runner initially checks the integration test snapshots to determine if any changes have occurred since the last execution. Since there are no previous snapshots for the initial run, the snapshot verification fails. As a result, the integ-runner begins executing the integration tests using the ephemeral test stack and displays the result.

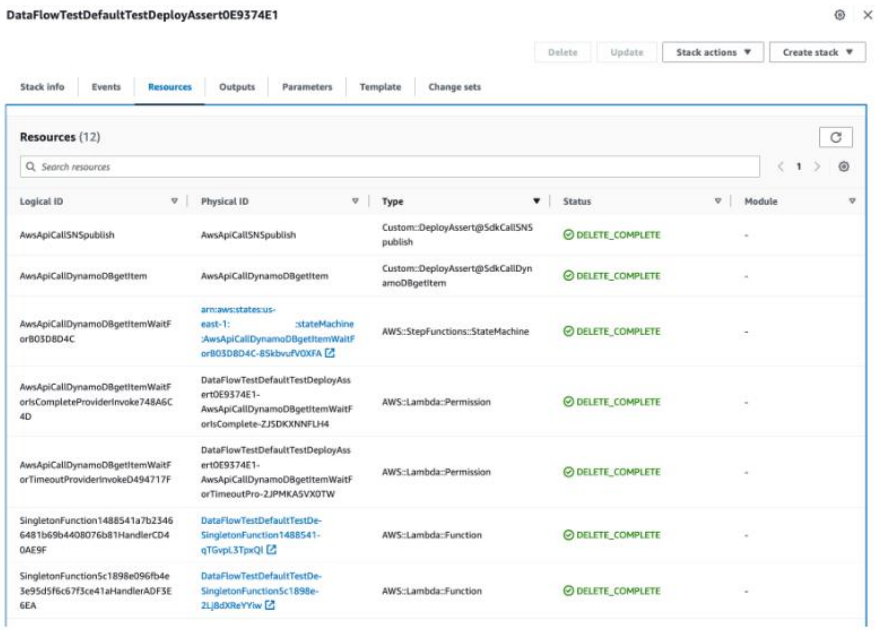
Bash
```bash
Verifying integration test snapshots...

  NEW        integ.sns-sqs-ddb 2.863s

Snapshot Results:

Tests:    1 failed, 1 total

Running integration tests for failed tests...

Running in parallel across regions: us-east-1
Running test <your-path>/cdk-integ-tests-demo/integ-tests/integ.sns-sqs-ddb.js in us-east-1
  SUCCESS    integ.sns-sqs-ddb-DemoTest/DefaultTest 587.295s
       AssertionResultsAwsApiCallDynamoDBgetItem - success

Test Results:

Tests:    1 passed, 1 total
```



	Stack name	Status	Created time	Description
○	DataFlowTestDefaultTestDeployAssert0E9374E1	ⓘ CREATE_IN_PROGRESS	2023-04-15 11:17:15 UTC+0200	-
○	IntegrationTestStack	✓ CREATE_COMPLETE	2023-04-15 11:13:25 UTC+0200	This stack includes the application's resources for integration testing.

Figure 3: AWS CloudFormation deploying the IntegrationTestStack and DataFlowDefaultTestDeployAssert stacks

The integ-runner generates two AWS CloudFormation stacks, as shown in Figure 3. The *IntegrationTestStack* stack includes the resources from our sample application, which serves as an isolated application representing the stack under test. The *DataFlowDefaultTestDeployAssert* stack contains the resources required for executing the integration tests as shown in Figure 4.



DataFlowTestDefaultTestDeployAssert0E9374E1

Resources (12)

Logical ID	Physical ID	Type	Status	Module
AwsApiCallSNSpublish	AwsApiCallSNSpublish	Custom::DeployAssert@SdkCallSNSpublish	✓ DELETE_COMPLETE	-
AwsApiCallDynamoDBgetItem	AwsApiCallDynamoDBgetItem	Custom::DeployAssert@SdkCallDynamoDBgetItem	✓ DELETE_COMPLETE	-
AwsApiCallDynamoDBgetItemWaitForB03D8D4C	arn:aws:states:us-east-1: :stateMachine:AwsApiCallDynamoDBgetItemWaitForB03D8D4C-8SkbvufV0XFA	AWS::StepFunctions::StateMachine	✓ DELETE_COMPLETE	-
AwsApiCallDynamoDBgetItemWaitForIsCompleteProviderInvoke748A6C4D	DataFlowTestDefaultTestDeployAssert0E9374E1-AwsApiCallDynamoDBgetItemWaitForIsComplete-ZJSDKXNNFLH4	AWS::Lambda::Permission	✓ DELETE_COMPLETE	-
AwsApiCallDynamoDBgetItemWaitForTimeoutProviderInvokeD494717F	DataFlowTestDefaultTestDeployAssert0E9374E1-AwsApiCallDynamoDBgetItemWaitForTimeoutPro-2JPMKA5VX0TW	AWS::Lambda::Permission	✓ DELETE_COMPLETE	-
SingletonFunction1488541a7b23466481b69b4408076b81HandlerCD40AE9F	DataFlowTestDefaultTestDe-SingletonFunction1488541-qTGvpL3TpxQI	AWS::Lambda::Function	✓ DELETE_COMPLETE	-
SingletonFunction5c1898e096fb4e3e95d5f6c67f3ce41aHandlerADF3E6EA	DataFlowTestDefaultTestDe-SingletonFunction5c1898e-2LjBdXReYYiw	AWS::Lambda::Function	✓ DELETE_COMPLETE	-

Figure 4: AWS CloudFormation resources for the DataFlowDefaultTestDeployAssert stack

Cleaning up

Based on the specified RemovalPolicy, the resources are automatically destroyed as the stack is removed. Some resources such as Amazon DynamoDB tables have the default *RemovalPolicy* set to *Retain* in AWS CDK. To set the removal policy to *Destroy* for the integration test resources, we leverage Aspects.

TypeScript
```typescript
/**
 * Aspect for setting all removal policies to DESTROY
 */
class ApplyDestroyPolicyAspect implements cdk.IAspect {
  public visit(node: IConstruct): void {
    if (node instanceof CfnResource) {
      node.applyRemovalPolicy(cdk.RemovalPolicy.DESTROY);
    }
  }
}
```

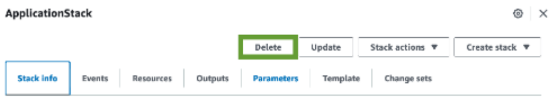


Figure 5: Deleting AWS CloudFormation stacks from the AWS Console

If you set the *no-clean* argument as part of the integ-runner CLI options, you need to manually destroy the stacks. This can be done from the AWS Console, via AWS CloudFormation as shown in Figure 5 or by using the following command.

Bash
```bash
cdk destroy --all
```

To clean up the code repository build files, you can run the following script.

Bash
```bash
npm run clean
```

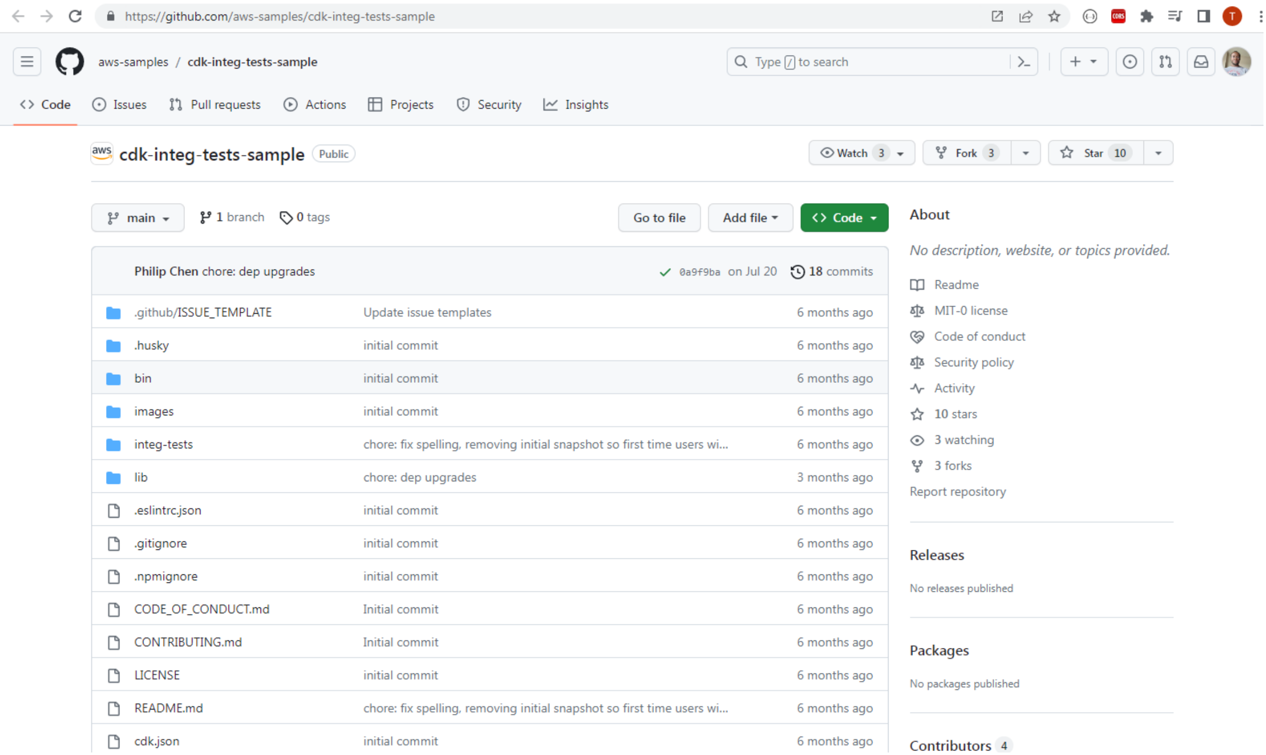
Conclusion

The AWS CDK integ-tests construct is a valuable tool for defining and conducting automated integration tests for your AWS CDK applications. In this blog post, we have introduced a practical code example showcasing how AWS CDK integration tests can be used to validate the expected application behavior when deployed to the cloud. You can leverage the techniques in this guide to write your own AWS CDK integration tests and improve the quality and reliability of your application releases.

For information on how to get started with these constructs, please refer to the following documentation.

Call to Action

Integ-runner and integ-tests constructs are experimental and subject to change. The release notes for both stable and experimental modules are available in the AWS CDK Github release notes. As always, we welcome bug reports, feature requests, and pull requests on the aws-cdk GitHub repository to further shape these alpha constructs based on your feedback.



No description, website, or topics provided.

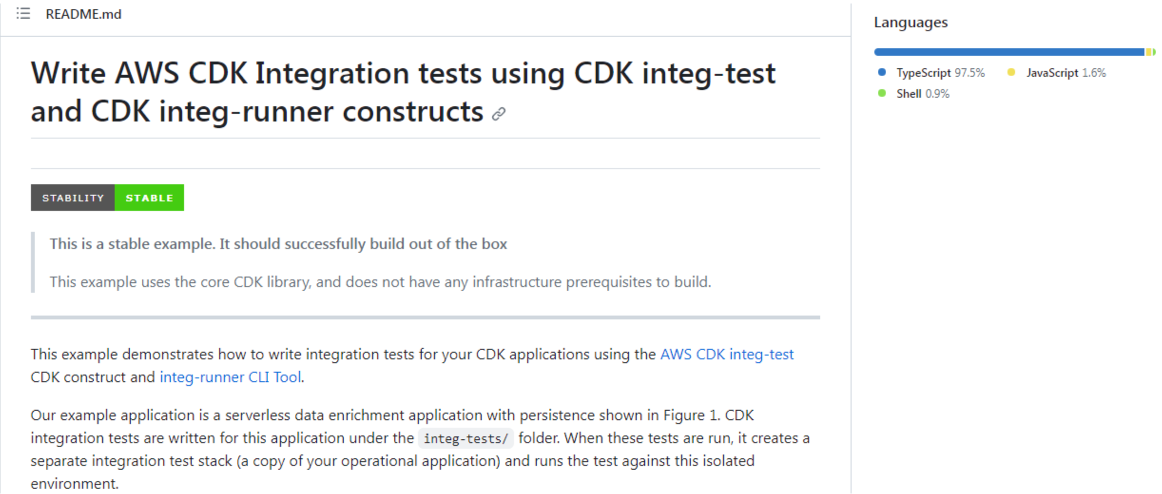
No releases published

No packages published

Philip Chen chore: dep upgrades ✓ 0a9f9ba on Jul 20 18 commits

.github/ISSUE_TEMPLATE	Update issue templates	6 months ago
.husky	initial commit	6 months ago
bin	initial commit	6 months ago
images	initial commit	6 months ago
integ-tests	chore: fix spelling, removing initial snapshot so first time users wi...	6 months ago
lib	chore: dep upgrades	3 months ago
.eslintrc.json	initial commit	6 months ago
.gitignore	initial commit	6 months ago
.npmignore	initial commit	6 months ago
CODE_OF_CONDUCT.md	Initial commit	6 months ago
CONTRIBUTING.md	Initial commit	6 months ago
LICENSE	initial commit	6 months ago
README.md	chore: fix spelling, removing initial snapshot so first time users wi...	6 months ago
cdk.json	initial commit	6 months ago

https://github.com/aws-samples/cdk-integ-tests-sample



README.md

Write AWS CDK Integration tests using CDK integ-test and CDK integ-runner constructs

STABILITY STABLE

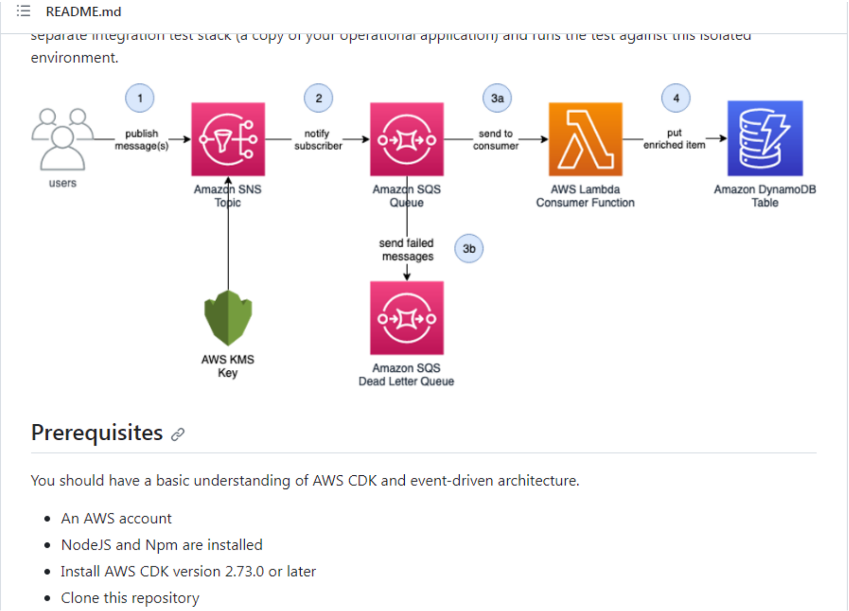
> This is a stable example. It should successfully build out of the box
>
> This example uses the core CDK library, and does not have any infrastructure prerequisites to build.

This example demonstrates how to write integration tests for your CDK applications using the AWS CDK integ-test CDK construct and integ-runner CLI Tool.

Our example application is a serverless data enrichment application with persistence shown in Figure 1. CDK integration tests are written for this application under the `integ-tests/` folder. When these tests are run, it creates a separate integration test stack (a copy of your operational application) and runs the test against this isolated environment.

Languages

- TypeScript 97.5%
- JavaScript 1.6%
- Shell 0.9%



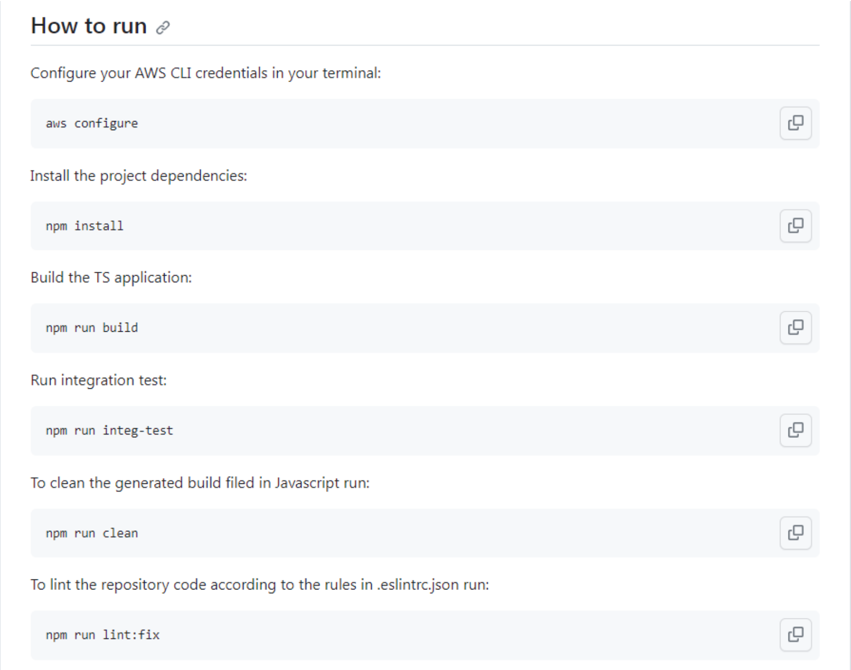
README.md

separate integration test stack (a copy of your operational application) and runs the test against this isolated environment.

Prerequisites

You should have a basic understanding of AWS CDK and event-driven architecture.

- An AWS account
- NodeJS and Npm are installed
- Install AWS CDK version 2.73.0 or later
- Clone this repository



How to run

Configure your AWS CLI credentials in your terminal:

```
aws configure
```

Install the project dependencies:

```
npm install
```

Build the TS application:

```
npm run build
```

Run integration test:

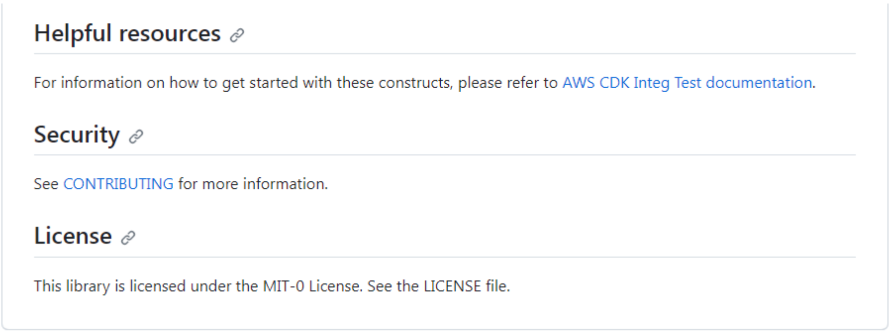
```
npm run integ-test
```

To clean the generated build filed in Javascript run:

```
npm run clean
```

To lint the repository code according to the rules in .eslintrc.json run:

```
npm run lint:fix
```



Helpful resources

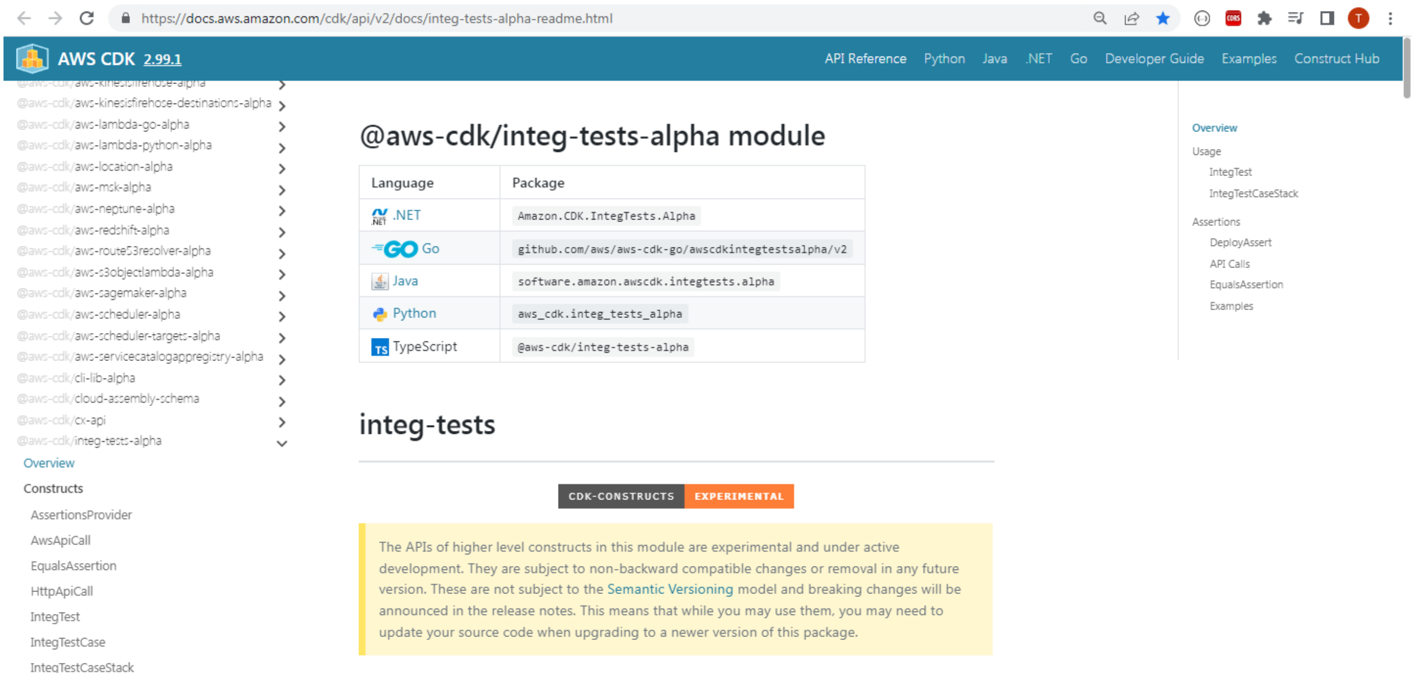
For information on how to get started with these constructs, please refer to AWS CDK Integ Test documentation.

Security

See CONTRIBUTING for more information.

License

This library is licensed under the MIT-0 License. See the LICENSE file.



https://docs.aws.amazon.com/cdk/api/v2/docs/integ-tests-alpha-readme.html
AWS CDK 2.99.1
API Reference Python Java .NET Go Developer Guide Examples Construct Hub

@aws-cdk/aws-kinesisfirehose-alpha
@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack

@aws-cdk/integ-tests-alpha module

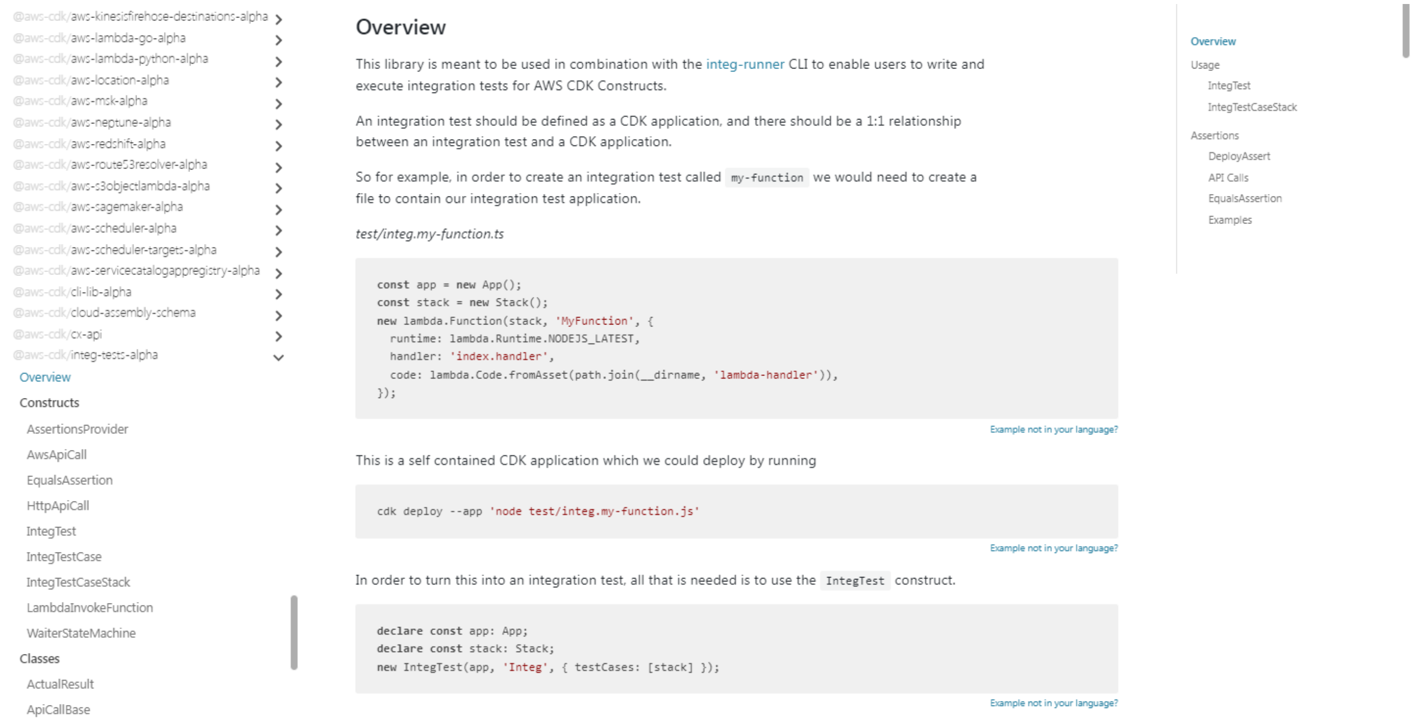
Language	Package
.NET	`Amazon.CDK.IntegTests.Alpha`
Go	`github.com/aws/aws-cdk-go/awscdkintegtestsalpha/v2`
Java	`software.amazon.awscdk.integtests.alpha`
Python	`aws_cdk.integ_tests_alpha`
TypeScript	`@aws-cdk/integ-tests-alpha`

integ-tests

CDK-CONSTRUCTS EXPERIMENTAL

> The APIs of higher level constructs in this module are experimental and under active development. They are subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model and breaking changes will be announced in the release notes. 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.

https://docs.aws.amazon.com/cdk/api/v2/docs/integ-tests-alpha-readme.html



@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Classes
ActualResult
ApiCallBase

Overview

This library is meant to be used in combination with the integ-runner CLI to enable users to write and execute integration tests for AWS CDK Constructs.

An integration test should be defined as a CDK application, and there should be a 1:1 relationship between an integration test and a CDK application.

So for example, in order to create an integration test called `my-function` we would need to create a file to contain our integration test application.

test/integ.my-function.ts

```
const app = new App();
const stack = new Stack();
new lambda.Function(stack, 'MyFunction', {
  runtime: lambda.Runtime.NODEJS_LATEST,
  handler: 'index.handler',
  code: lambda.Code.fromAsset(path.join(__dirname, 'lambda-handler')),
});
```

Example not in your language?

This is a self contained CDK application which we could deploy by running

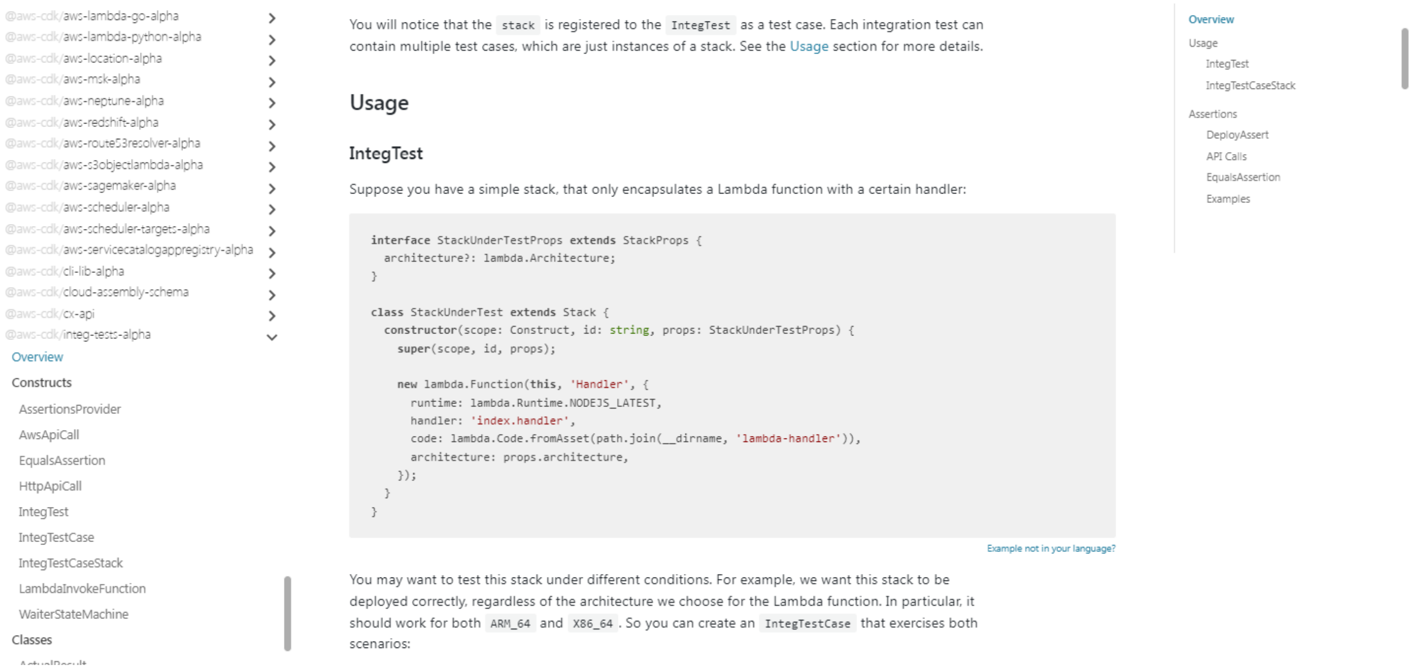
```
cdk deploy --app 'node test/integ.my-function.js'
```

Example not in your language?

In order to turn this into an integration test, all that is needed is to use the `IntegTest` construct.

```
declare const app: App;
declare const stack: Stack;
new IntegTest(app, 'Integ', { testCases: [stack] });
```

Example not in your language?


@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Classes
Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples

You will notice that the `stack` is registered to the `IntegTest` as a test case. Each integration test can contain multiple test cases, which are just instances of a stack. See the Usage section for more details.

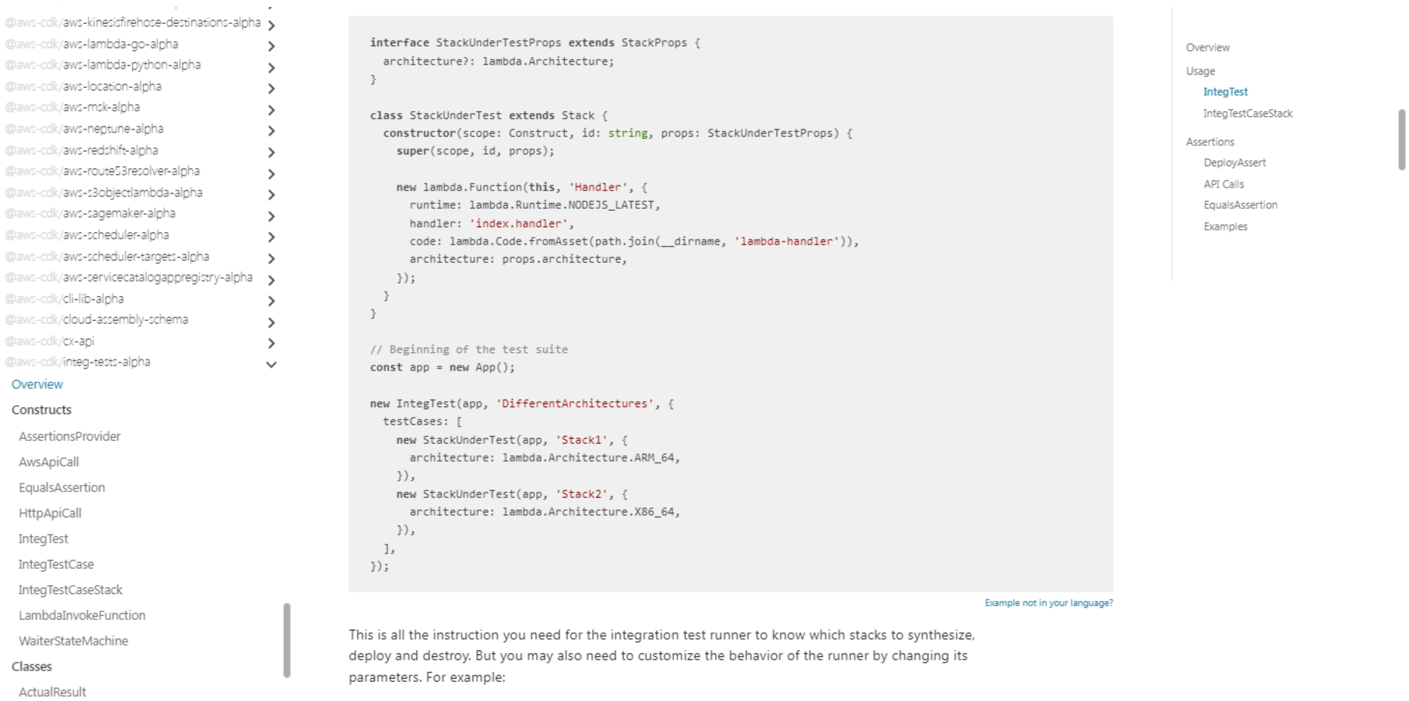
Usage

IntegTest

Suppose you have a simple stack, that only encapsulates a Lambda function with a certain handler:

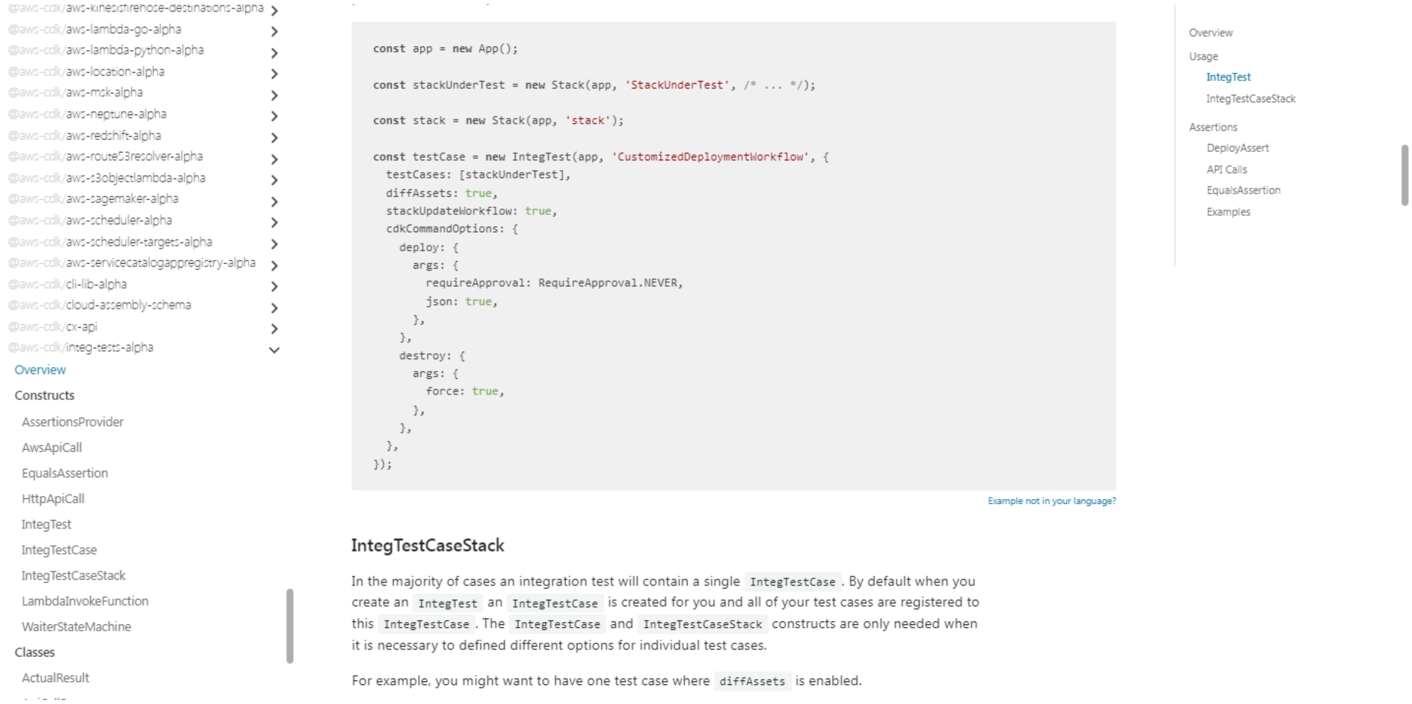
```
interface StackUnderTestProps extends StackProps {
  architecture?: lambda.Architecture;
}

class StackUnderTest extends Stack {
  constructor(scope: Construct, id: string, props: StackUnderTestProps) {
    super(scope, id, props);

    new lambda.Function(this, 'Handler', {
      runtime: lambda.Runtime.NODEJS_LATEST,
      handler: 'index.handler',
      code: lambda.Code.fromAsset(path.join(__dirname, 'lambda-handler')),
      architecture: props.architecture,
    });
  }
}
```

You may want to test this stack under different conditions. For example, we want this stack to be deployed correctly, regardless of the architecture we choose for the Lambda function. In particular, it should work for both `ARM_64` and `X86_64`. So you can create an `IntegTestCase` that exercises both scenarios:


@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Classes
ActualResult
Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples

```
interface StackUnderTestProps extends StackProps {
  architecture?: lambda.Architecture;
}

class StackUnderTest extends Stack {
  constructor(scope: Construct, id: string, props: StackUnderTestProps) {
    super(scope, id, props);

    new lambda.Function(this, 'Handler', {
      runtime: lambda.Runtime.NODEJS_LATEST,
      handler: 'index.handler',
      code: lambda.Code.fromAsset(path.join(__dirname, 'lambda-handler')),
      architecture: props.architecture,
    });
  }
}

// Beginning of the test suite
const app = new App();

new IntegTest(app, 'DifferentArchitectures', {
  testCases: [
    new StackUnderTest(app, 'Stack1', {
      architecture: lambda.Architecture.ARM_64,
    }),
    new StackUnderTest(app, 'Stack2', {
      architecture: lambda.Architecture.X86_64,
    }),
  ],
});
```

This is all the instruction you need for the integration test runner to know which stacks to synthesize, deploy and destroy. But you may also need to customize the behavior of the runner by changing its parameters. For example:


@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Classes
ActualResult
Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples

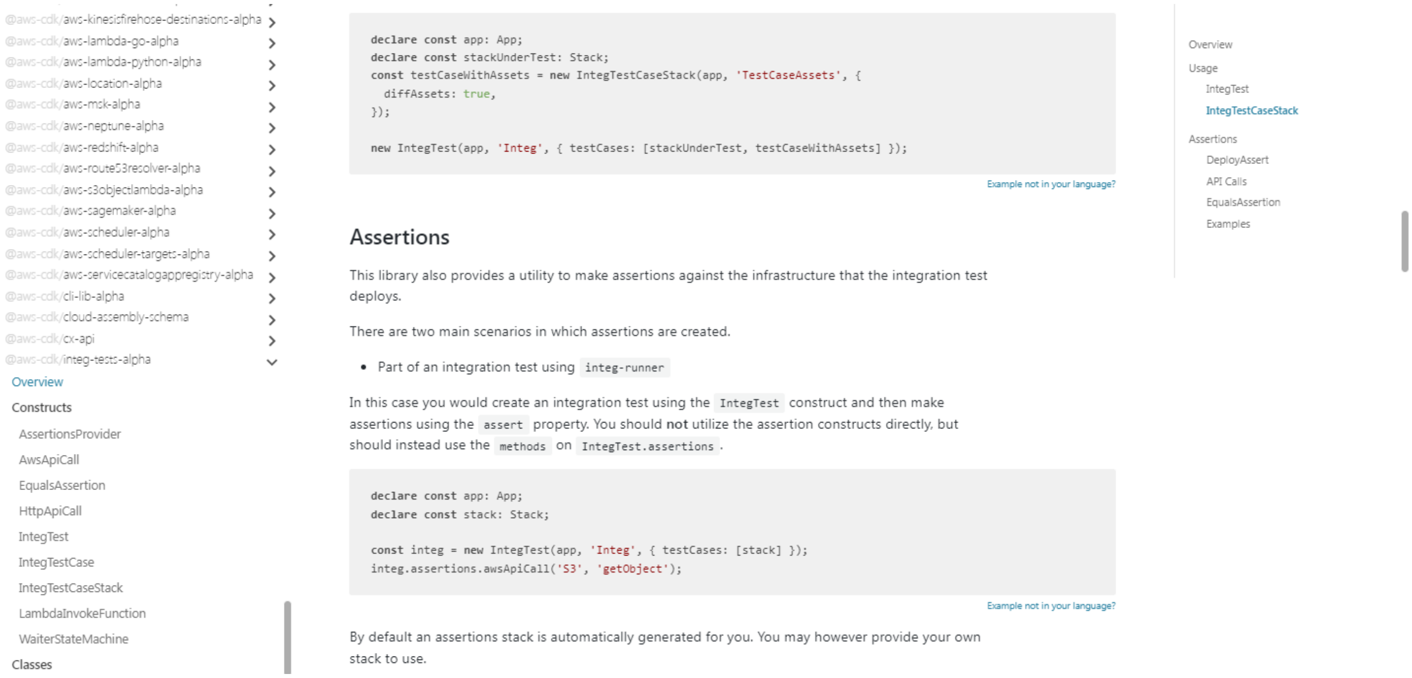
```
const app = new App();

const stackUnderTest = new Stack(app, 'StackUnderTest', /* ... */);

const stack = new Stack(app, 'stack');

const testCase = new IntegTest(app, 'CustomizedDeploymentWorkflow', {
  testCases: [stackUnderTest],
  diffAssets: true,
  stackUpdateWorkflow: true,
  cdkCommandOptions: {
    deploy: {
      args: {
        requireApproval: RequireApproval.NEVER,
        json: true,
      },
    },
    destroy: {
      args: {
        force: true,
      },
    },
  },
});
```

Example not in your language?

IntegTestCaseStack

In the majority of cases an integration test will contain a single `IntegTestCase`. By default when you create an `IntegTest` an `IntegTestCase` is created for you and all of your test cases are registered to this `IntegTestCase`. The `IntegTestCase` and `IntegTestCaseStack` constructs are only needed when it is necessary to defined different options for individual test cases.

For example, you might want to have one test case where `diffAssets` is enabled.


@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Classes
Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples

```
declare const app: App;
declare const stackUnderTest: Stack;
const testCaseWithAssets = new IntegTestCaseStack(app, 'TestCaseAssets', {
  diffAssets: true,
});

new IntegTest(app, 'Integ', { testCases: [stackUnderTest, testCaseWithAssets] });
```

Example not in your language?

Assertions

This library also provides a utility to make assertions against the infrastructure that the integration test deploys.

There are two main scenarios in which assertions are created.

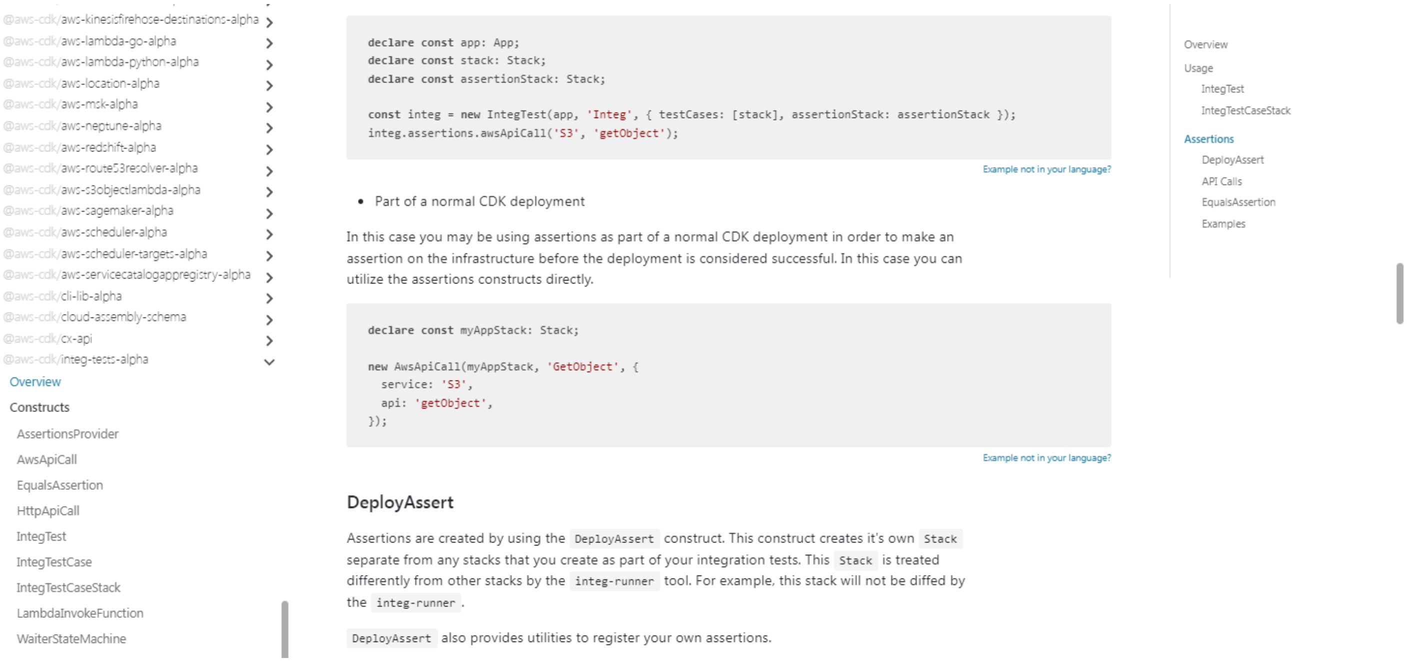
- Part of an integration test using `integ-runner`

In this case you would create an integration test using the `IntegTest` construct and then make assertions using the `assert` property. You should **not** utilize the assertion constructs directly, but should instead use the `methods` on `IntegTest.assertions`.

```
declare const app: App;
declare const stack: Stack;

const integ = new IntegTest(app, 'Integ', { testCases: [stack] });
integ.assertions.awsApiCall('S3', 'getObject');
```

Example not in your language?

By default an assertions stack is automatically generated for you. You may however provide your own stack to use.


@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples

```
declare const app: App;
declare const stack: Stack;
declare const assertionStack: Stack;

const integ = new IntegTest(app, 'Integ', { testCases: [stack], assertionStack: assertionStack });
integ.assertions.awsApiCall('S3', 'getObject');
```

Example not in your language?

- Part of a normal CDK deployment

In this case you may be using assertions as part of a normal CDK deployment in order to make an assertion on the infrastructure before the deployment is considered successful. In this case you can utilize the assertions constructs directly.

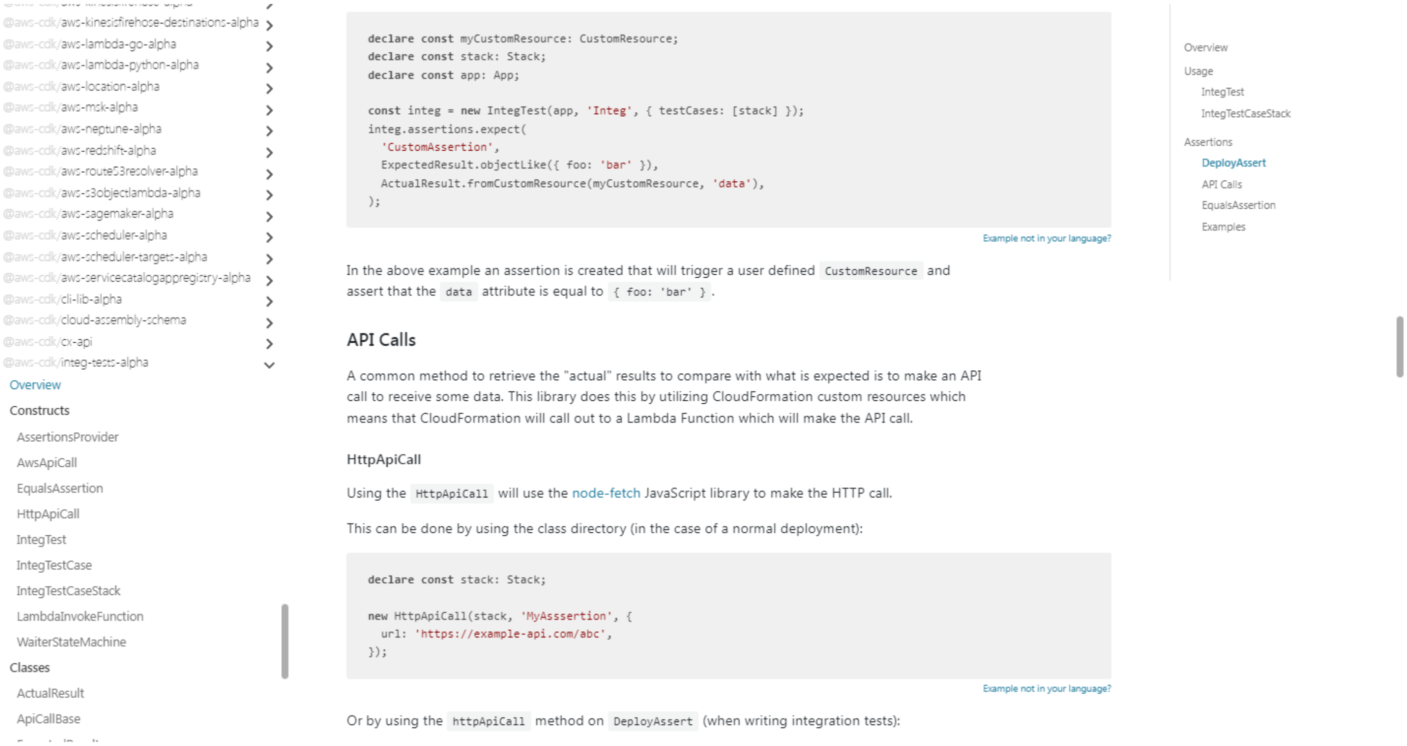
```
declare const myAppStack: Stack;

new AwsApiCall(myAppStack, 'GetObject', {
  service: 'S3',
  api: 'getObject',
});
```

Example not in your language?

DeployAssert

Assertions are created by using the `DeployAssert` construct. This construct creates it's own `Stack` separate from any stacks that you create as part of your integration tests. This `Stack` is treated differently from other stacks by the `integ-runner` tool. For example, this stack will not be diffed by the `integ-runner`.

`DeployAssert` also provides utilities to register your own assertions.


@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Classes
ActualResult
ApiCallBase
Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples

```
declare const myCustomResource: CustomResource;
declare const stack: Stack;
declare const app: App;

const integ = new IntegTest(app, 'Integ', { testCases: [stack] });
integ.assertions.expect(
  'CustomAssertion',
  ExpectedResult.objectLike({ foo: 'bar' }),
  ActualResult.fromCustomResource(myCustomResource, 'data'),
);
```

Example not in your language?

In the above example an assertion is created that will trigger a user defined `CustomResource` and assert that the `data` attribute is equal to `{ foo: 'bar' }`.

API Calls

A common method to retrieve the "actual" results to compare with what is expected is to make an API call to receive some data. This library does this by utilizing CloudFormation custom resources which means that CloudFormation will call out to a Lambda Function which will make the API call.

HttpApiCall

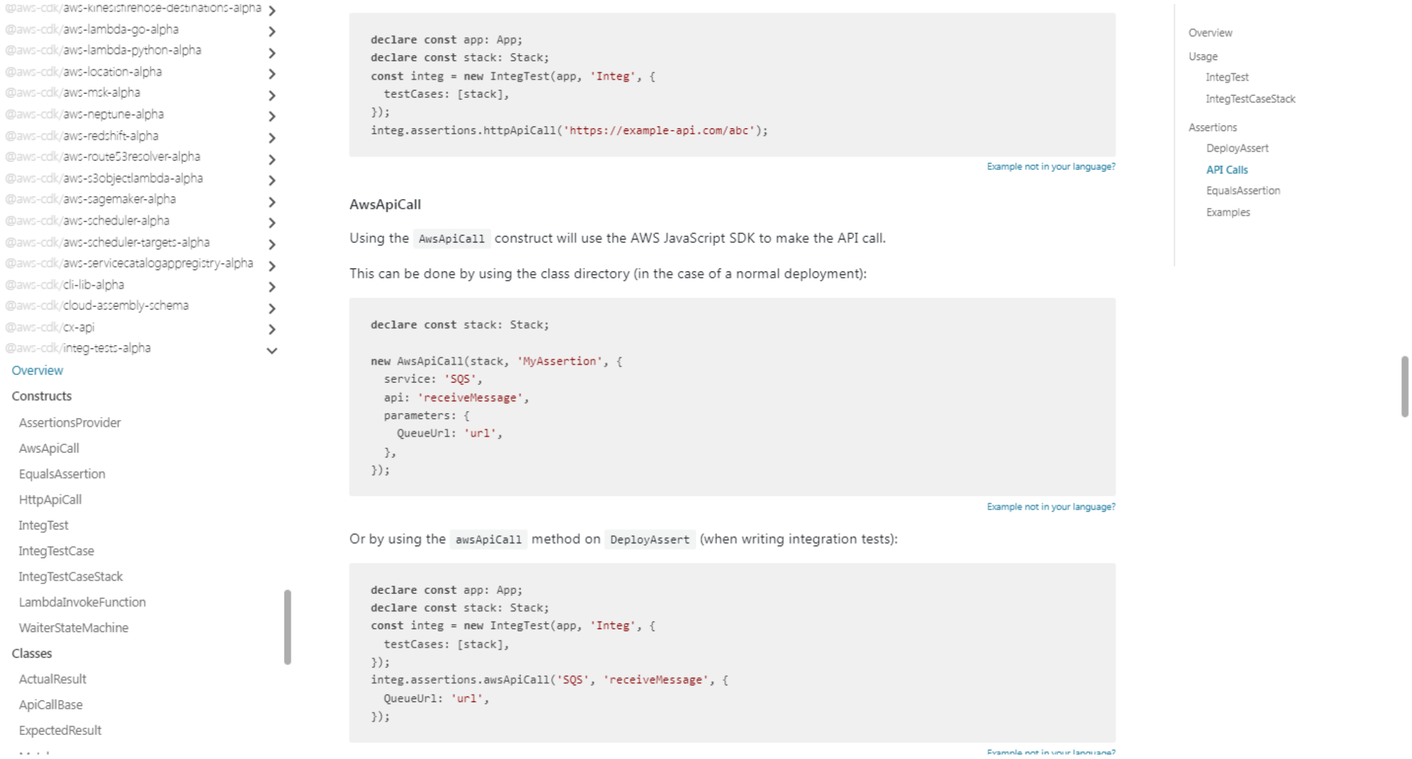
Using the `HttpApiCall` will use the node-fetch JavaScript library to make the HTTP call.

This can be done by using the class directory (in the case of a normal deployment):

```
declare const stack: Stack;

new HttpApiCall(stack, 'MyAsssertion', {
  url: 'https://example-api.com/abc',
});
```

Example not in your language?

Or by using the `httpApiCall` method on `DeployAssert` (when writing integration tests):



@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Classes
ActualResult
ApiCallBase
ExpectedResult

```
declare const app: App;
declare const stack: Stack;
const integ = new IntegTest(app, 'Integ', {
  testCases: [stack],
});
integ.assertions.httpApiCall('https://example-api.com/abc');
```

Example not in your language?

AwsApiCall

Using the `AwsApiCall` construct will use the AWS JavaScript SDK to make the API call.

This can be done by using the class directory (in the case of a normal deployment):

```
declare const stack: Stack;

new AwsApiCall(stack, 'MyAssertion', {
  service: 'SQS',
  api: 'receiveMessage',
  parameters: {
    QueueUrl: 'url',
  },
});
```

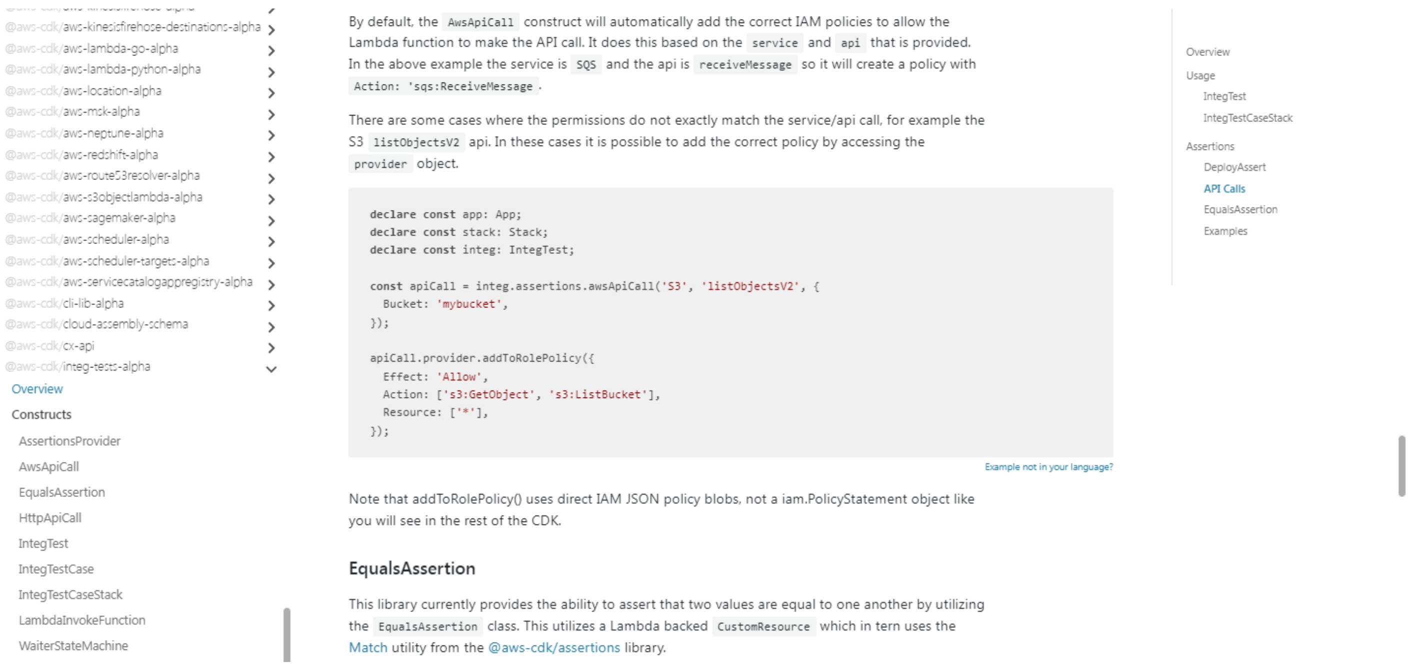
Example not in your language?

Or by using the `awsApiCall` method on `DeployAssert` (when writing integration tests):

```
declare const app: App;
declare const stack: Stack;
const integ = new IntegTest(app, 'Integ', {
  testCases: [stack],
});
integ.assertions.awsApiCall('SQS', 'receiveMessage', {
  QueueUrl: 'url',
});
```

Example not in your language?

Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples



@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine

By default, the `AwsApiCall` construct will automatically add the correct IAM policies to allow the Lambda function to make the API call. It does this based on the `service` and `api` that is provided. In the above example the service is `SQS` and the api is `receiveMessage` so it will create a policy with `Action: 'sqs:ReceiveMessage'`.

There are some cases where the permissions do not exactly match the service/api call, for example the S3 `listObjectsV2` api. In these cases it is possible to add the correct policy by accessing the `provider` object.

```
declare const app: App;
declare const stack: Stack;
declare const integ: IntegTest;

const apiCall = integ.assertions.awsApiCall('S3', 'listObjectsV2', {
  Bucket: 'mybucket',
});

apiCall.provider.addToRolePolicy({
  Effect: 'Allow',
  Action: ['s3:GetObject', 's3:ListBucket'],
  Resource: ['*'],
});
```

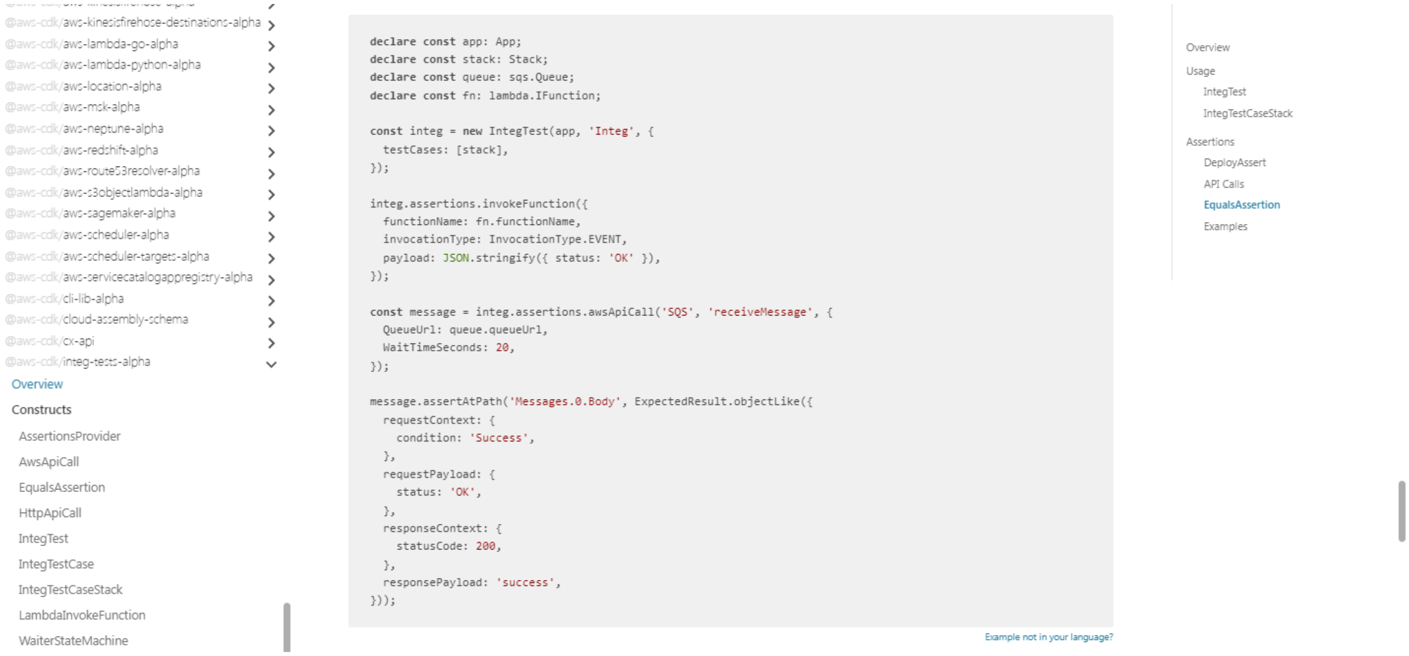
Example not in your language?

Note that addToRolePolicy() uses direct IAM JSON policy blobs, not a iam.PolicyStatement object like you will see in the rest of the CDK.

EqualsAssertion

This library currently provides the ability to assert that two values are equal to one another by utilizing the `EqualsAssertion` class. This utilizes a Lambda backed `CustomResource` which in tern uses the Match utility from the @aws-cdk/assertions library.

Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples

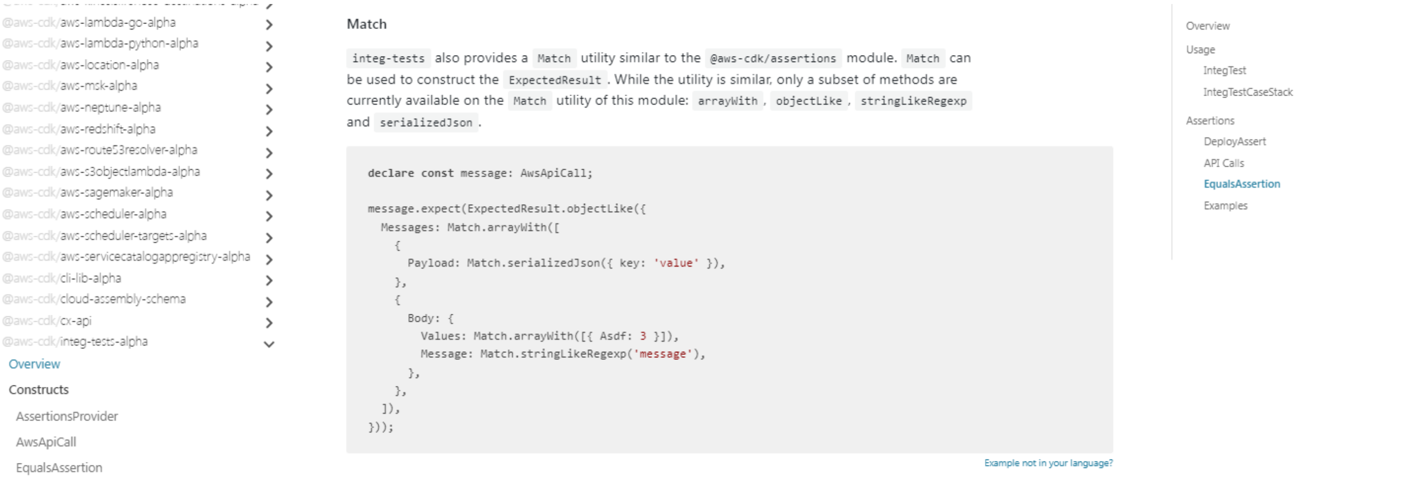


@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine

```
declare const app: App;
declare const stack: Stack;
declare const queue: sqs.Queue;
declare const fn: lambda.IFunction;

const integ = new IntegTest(app, 'Integ', {
  testCases: [stack],
});

integ.assertions.invokeFunction({
  functionName: fn.functionName,
  invocationType: InvocationType.EVENT,
  payload: JSON.stringify({ status: 'OK' }),
});

const message = integ.assertions.awsApiCall('SQS', 'receiveMessage', {
  QueueUrl: queue.queueUrl,
  WaitTimeSeconds: 20,
});

message.assertAtPath('Messages.0.Body', ExpectedResult.objectLike({
  requestContext: {
    condition: 'Success',
  },
  requestPayload: {
    status: 'OK',
  },
  responseContext: {
    statusCode: 200,
  },
  responsePayload: 'success',
}));
```

Example not in your language?

Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples



@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion

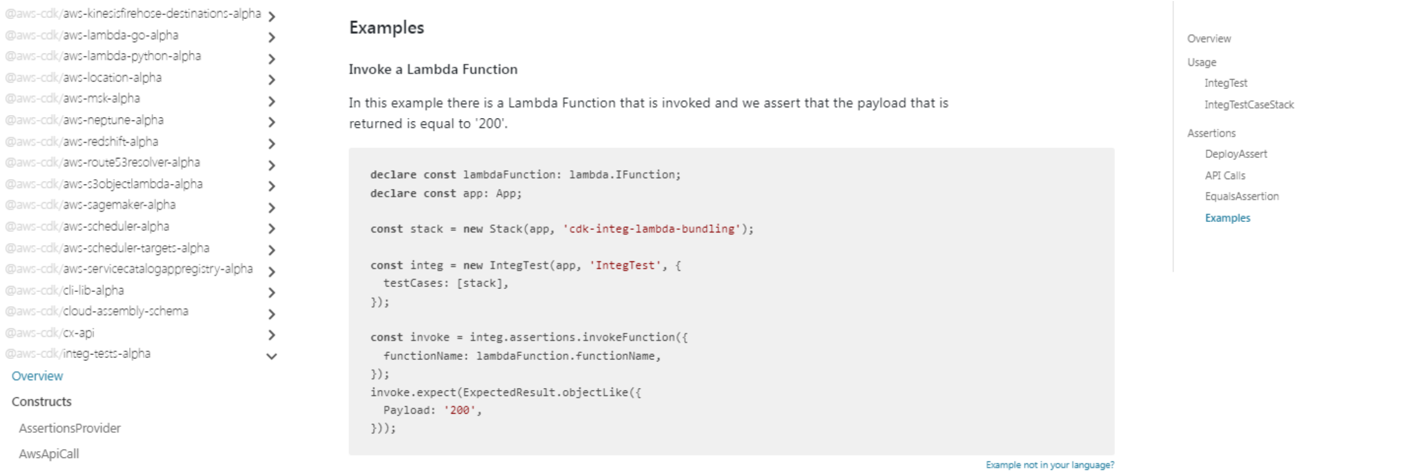
Match

`integ-tests` also provides a `Match` utility similar to the `@aws-cdk/assertions` module. `Match` can be used to construct the `ExpectedResult`. While the utility is similar, only a subset of methods are currently available on the `Match` utility of this module: `arrayWith`, `objectLike`, `stringLikeRegexp` and `serializedJson`.

```
declare const message: AwsApiCall;

message.expect(ExpectedResult.objectLike({
  Messages: Match.arrayWith([
    {
      Payload: Match.serializedJson({ key: 'value' }),
    },
    {
      Body: {
        Values: Match.arrayWith([{ Asdf: 3 }]),
        Message: Match.stringLikeRegexp('message'),
      },
    },
  ]),
}));
```

Example not in your language?

Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples



@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall

Examples

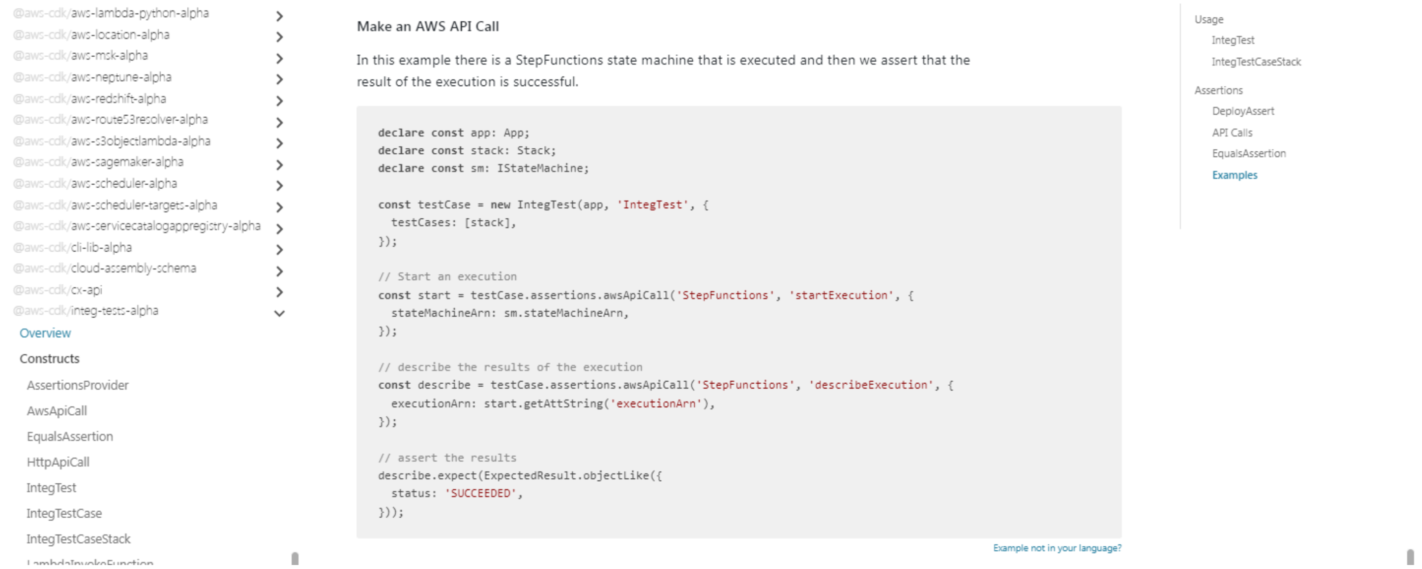
Invoke a Lambda Function

In this example there is a Lambda Function that is invoked and we assert that the payload that is returned is equal to '200'.

```
declare const lambdaFunction: lambda.IFunction;
declare const app: App;

const stack = new Stack(app, 'cdk-integ-lambda-bundling');

const integ = new IntegTest(app, 'IntegTest', {
  testCases: [stack],
});

const invoke = integ.assertions.invokeFunction({
  functionName: lambdaFunction.functionName,
});
invoke.expect(ExpectedResult.objectLike({
  Payload: '200',
}));
```

Example not in your language?

Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples



@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction

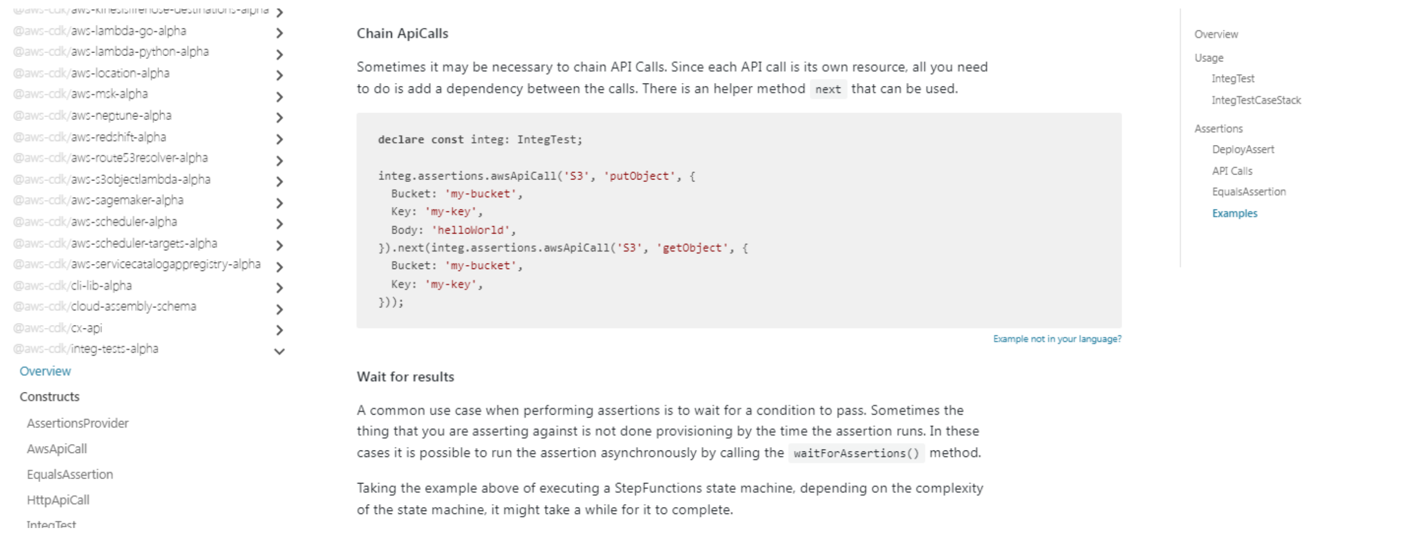
Make an AWS API Call

In this example there is a StepFunctions state machine that is executed and then we assert that the result of the execution is successful.

```
declare const app: App;
declare const stack: Stack;
declare const sm: IStateMachine;

const testCase = new IntegTest(app, 'IntegTest', {
  testCases: [stack],
});

// Start an execution
const start = testCase.assertions.awsApiCall('StepFunctions', 'startExecution', {
  stateMachineArn: sm.stateMachineArn,
});

// describe the results of the execution
const describe = testCase.assertions.awsApiCall('StepFunctions', 'describeExecution', {
  executionArn: start.getAttString('executionArn'),
});

// assert the results
describe.expect(ExpectedResult.objectLike({
  status: 'SUCCEEDED',
}));
```

Example not in your language?

Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples



@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest

Chain ApiCalls

Sometimes it may be necessary to chain API Calls. Since each API call is its own resource, all you need to do is add a dependency between the calls. There is an helper method `next` that can be used.

```
declare const integ: IntegTest;

integ.assertions.awsApiCall('S3', 'putObject', {
  Bucket: 'my-bucket',
  Key: 'my-key',
  Body: 'helloWorld',
}).next(integ.assertions.awsApiCall('S3', 'getObject', {
  Bucket: 'my-bucket',
  Key: 'my-key',
}));
```

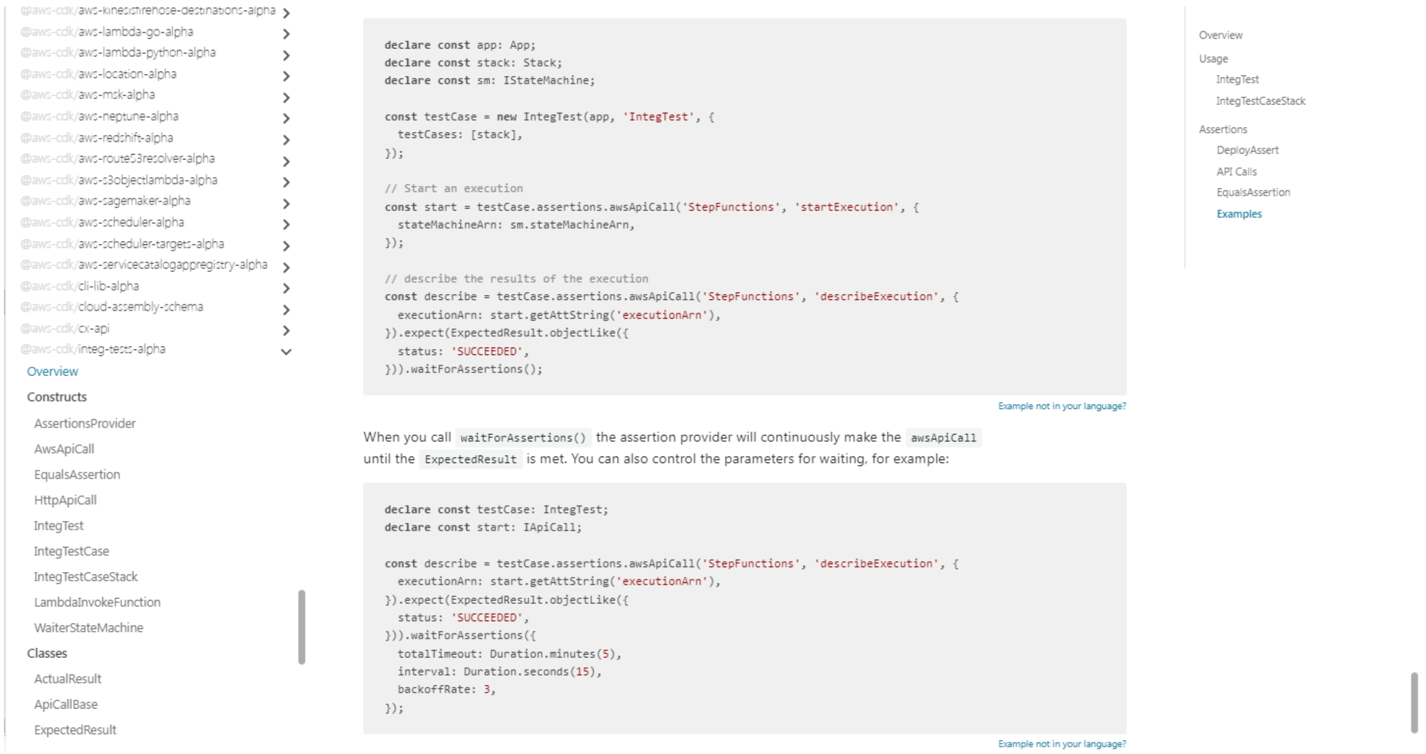
Example not in your language?

Wait for results

A common use case when performing assertions is to wait for a condition to pass. Sometimes the thing that you are asserting against is not done provisioning by the time the assertion runs. In these cases it is possible to run the assertion asynchronously by calling the `waitForAssertions()` method.

Taking the example above of executing a StepFunctions state machine, depending on the complexity of the state machine, it might take a while for it to complete.

Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples



@aws-cdk/aws-kinesisfirehose-destinations-alpha
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Classes
ActualResult
ApiCallBase
ExpectedResult

```
declare const app: App;
declare const stack: Stack;
declare const sm: IStateMachine;

const testCase = new IntegTest(app, 'IntegTest', {
  testCases: [stack],
});

// Start an execution
const start = testCase.assertions.awsApiCall('StepFunctions', 'startExecution', {
  stateMachineArn: sm.stateMachineArn,
});

// describe the results of the execution
const describe = testCase.assertions.awsApiCall('StepFunctions', 'describeExecution', {
  executionArn: start.getAttString('executionArn'),
}).expect(ExpectedResult.objectLike({
  status: 'SUCCEEDED',
})).waitForAssertions();
```

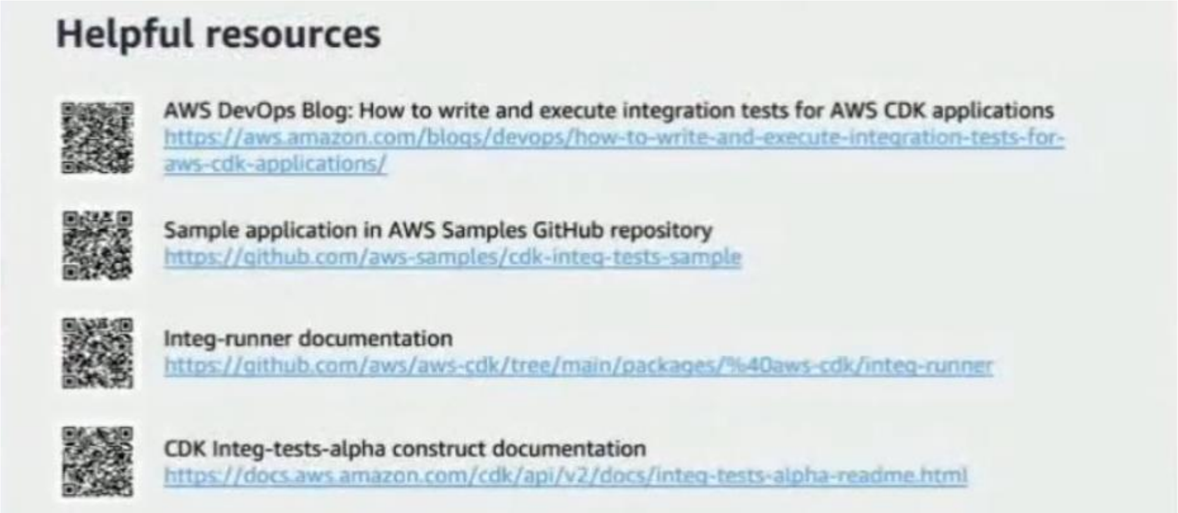
Example not in your language?

When you call `waitForAssertions()` the assertion provider will continuously make the `awsApiCall` until the `ExpectedResult` is met. You can also control the parameters for waiting, for example:

```
declare const testCase: IntegTest;
declare const start: IApiCall;

const describe = testCase.assertions.awsApiCall('StepFunctions', 'describeExecution', {
  executionArn: start.getAttString('executionArn'),
}).expect(ExpectedResult.objectLike({
  status: 'SUCCEEDED',
})).waitForAssertions({
  totalTimeout: Duration.minutes(5),
  interval: Duration.seconds(15),
  backoffRate: 3,
});
```

Example not in your language?

Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples



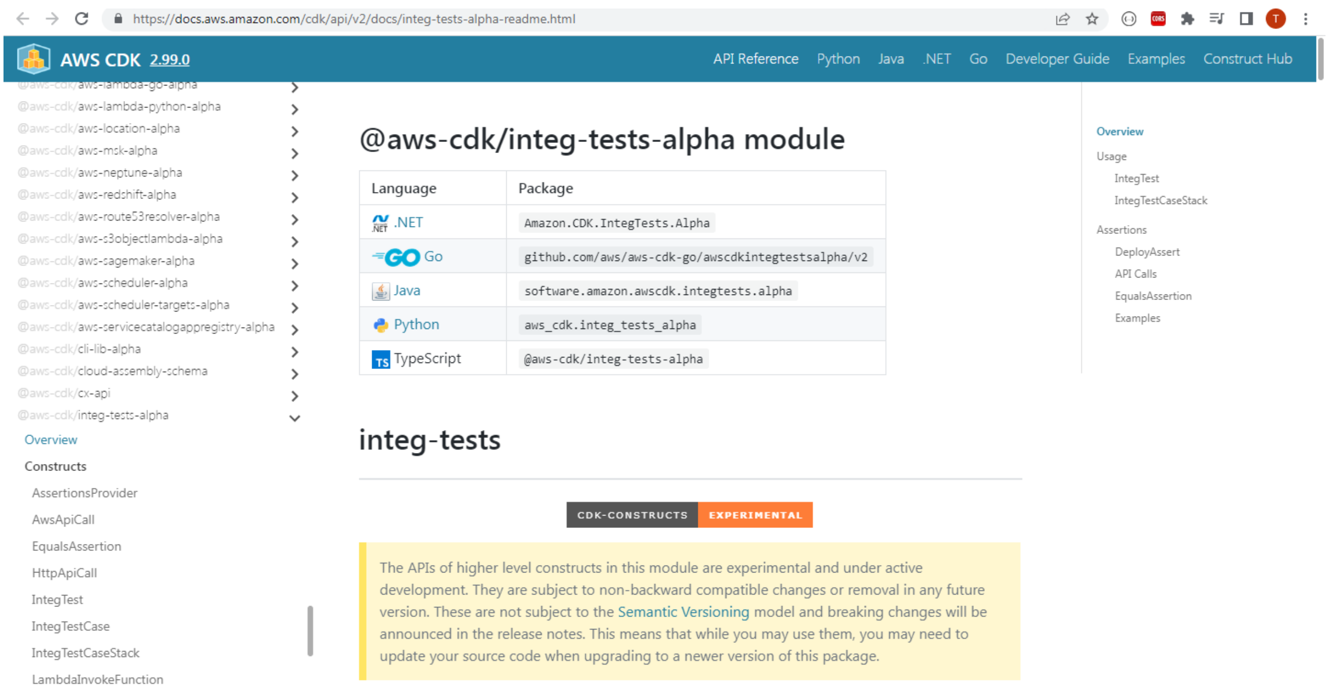
Helpful resources

AWS DevOps Blog: How to write and execute integration tests for AWS CDK applications
https://aws.amazon.com/blogs/devops/how-to-write-and-execute-integration-tests-for-aws-cdk-applications/

Sample application in AWS Samples GitHub repository
https://github.com/aws-samples/cdk-integ-tests-sample

Integ-runner documentation
https://github.com/aws/aws-cdk/tree/main/packages/%40aws-cdk/integ-runner

CDK Integ-tests-alpha construct documentation
https://docs.aws.amazon.com/cdk/api/v2/docs/integ-tests-alpha-readme.html



https://docs.aws.amazon.com/cdk/api/v2/docs/integ-tests-alpha-readme.html
AWS CDK 2.99.0
API Reference
Python
Java
.NET
Go
Developer Guide
Examples
Construct Hub
@aws-cdk/aws-lambda-go-alpha
@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples

@aws-cdk/integ-tests-alpha module

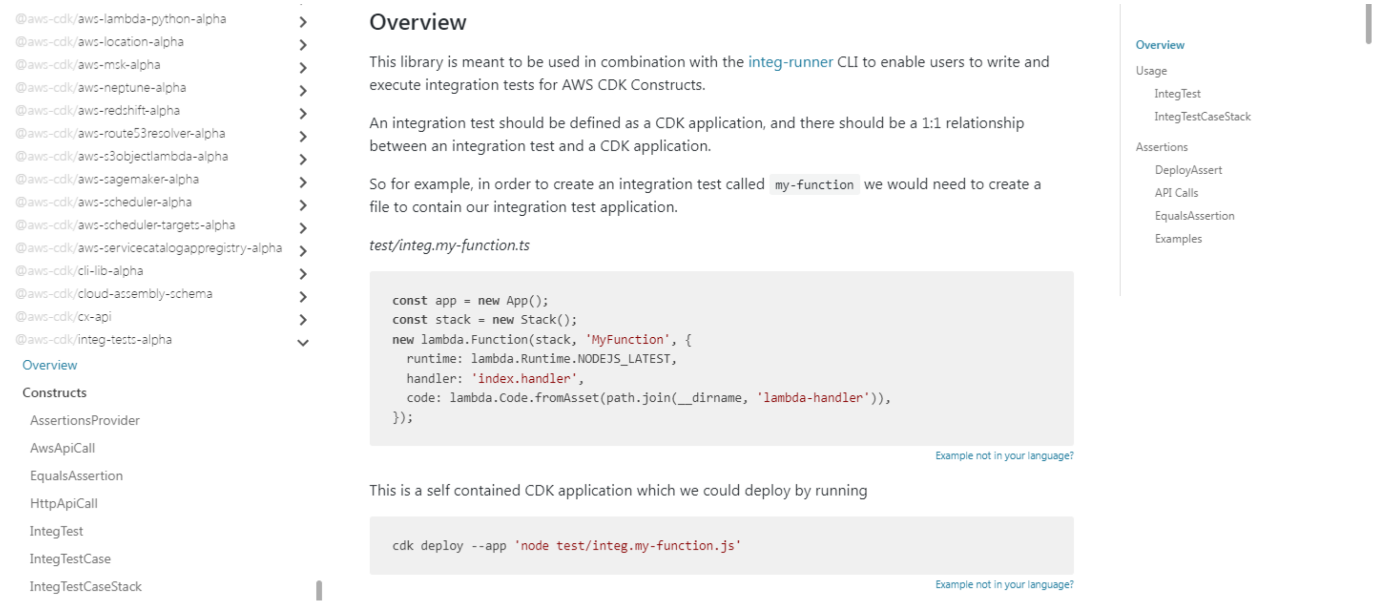
Language	Package
.NET	`Amazon.CDK.IntegTests.Alpha`
Go	`github.com/aws/aws-cdk-go/awscdkintegtestsalpha/v2`
Java	`software.amazon.awscdk.integtests.alpha`
Python	`aws_cdk.integ_tests_alpha`
TypeScript	`@aws-cdk/integ-tests-alpha`

integ-tests

CDK-CONSTRUCTS EXPERIMENTAL

The APIs of higher level constructs in this module are experimental and under active development. They are subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model and breaking changes will be announced in the release notes. 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.

https://docs.aws.amazon.com/cdk/api/v2/docs/integ-tests-alpha-readme.html



@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
Overview
Usage
IntegTest
IntegTestCaseStack
Assertions
DeployAssert
API Calls
EqualsAssertion
Examples

Overview

This library is meant to be used in combination with the integ-runner CLI to enable users to write and execute integration tests for AWS CDK Constructs.

An integration test should be defined as a CDK application, and there should be a 1:1 relationship between an integration test and a CDK application.

So for example, in order to create an integration test called `my-function` we would need to create a file to contain our integration test application.

test/integ.my-function.ts

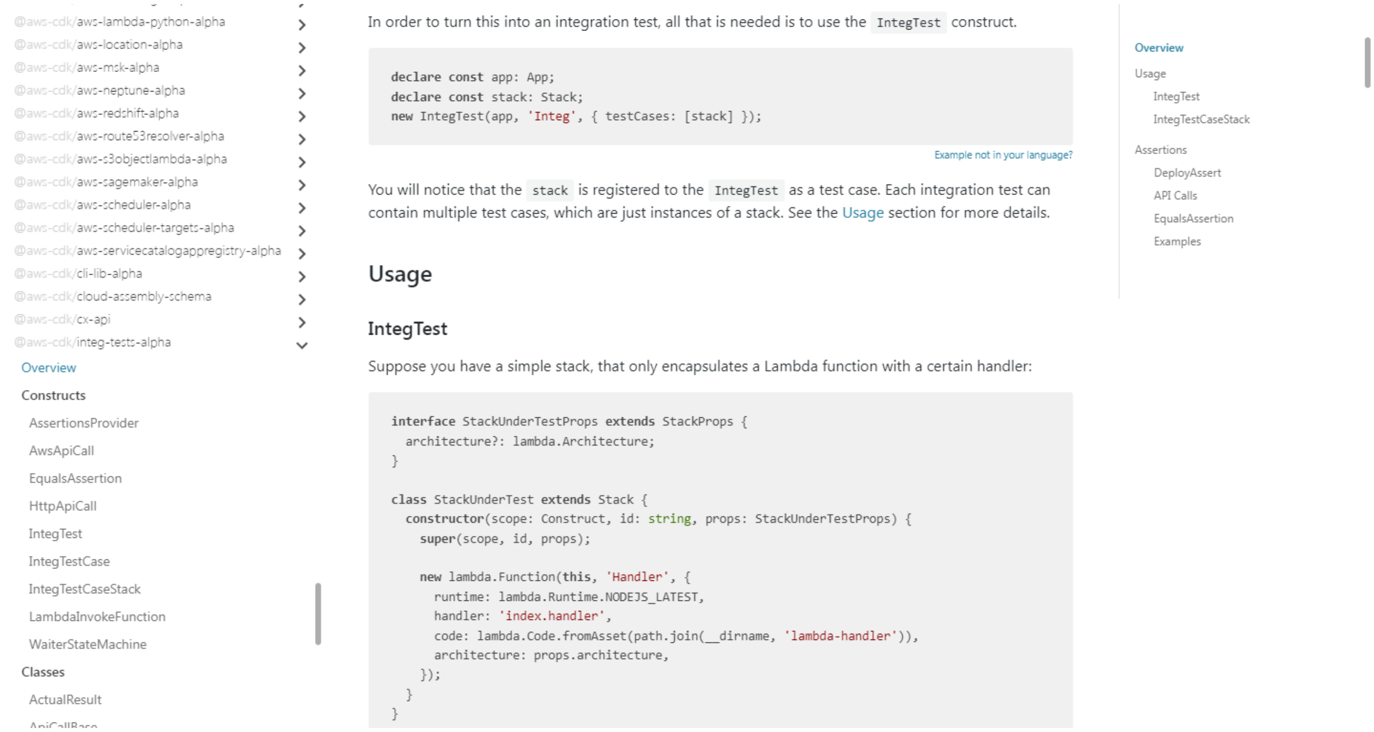
```
const app = new App();
const stack = new Stack();
new lambda.Function(stack, 'MyFunction', {
  runtime: lambda.Runtime.NODEJS_LATEST,
  handler: 'index.handler',
  code: lambda.Code.fromAsset(path.join(__dirname, 'lambda-handler')),
});
```

Example not in your language?

This is a self contained CDK application which we could deploy by running

```
cdk deploy --app 'node test/integ.my-function.js'
```

Example not in your language?



@aws-cdk/aws-lambda-python-alpha
@aws-cdk/aws-location-alpha
@aws-cdk/aws-msk-alpha
@aws-cdk/aws-neptune-alpha
@aws-cdk/aws-redshift-alpha
@aws-cdk/aws-route53resolver-alpha
@aws-cdk/aws-s3objectlambda-alpha
@aws-cdk/aws-sagemaker-alpha
@aws-cdk/aws-scheduler-alpha
@aws-cdk/aws-scheduler-targets-alpha
@aws-cdk/aws-servicecatalogappregistry-alpha
@aws-cdk/cli-lib-alpha
@aws-cdk/cloud-assembly-schema
@aws-cdk/cx-api
@aws-cdk/integ-tests-alpha
Overview
Constructs
AssertionsProvider
AwsApiCall
EqualsAssertion
HttpApiCall
IntegTest
IntegTestCase
IntegTestCaseStack
LambdaInvokeFunction
WaiterStateMachine
Classes
ActualResult
ApiCallBase

In order to turn this into an integration test, all that is needed is to use the `IntegTest` construct.

```
declare const app: App;
declare const stack: Stack;
new IntegTest(app, 'Integ', { testCases: [stack] });
```

Example not in your language?

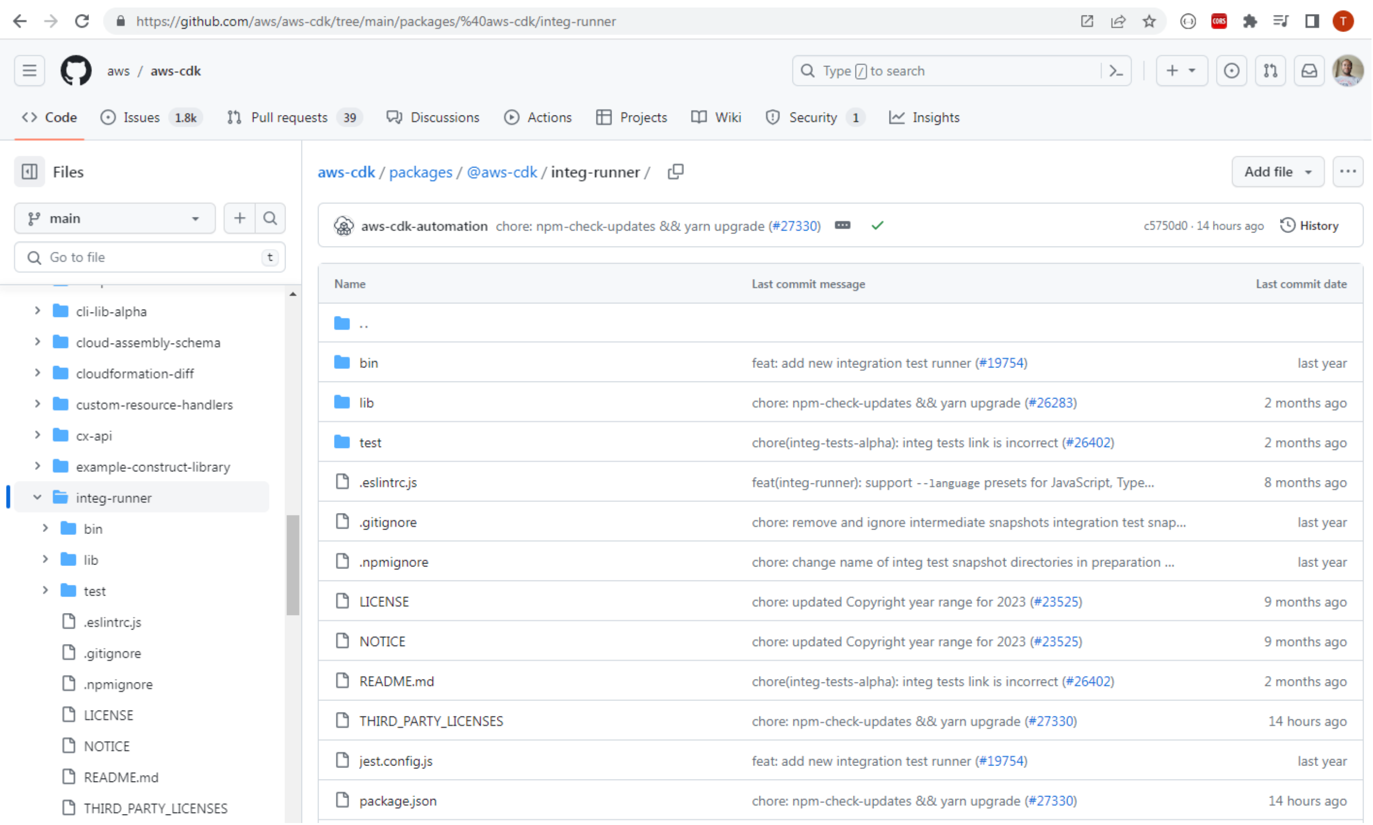
You will notice that the `stack` is registered to the `IntegTest` as a test case. Each integration test can contain multiple test cases, which are just instances of a stack. See the Usage section for more details.

Usage

IntegTest

Suppose you have a simple stack, that only encapsulates a Lambda function with a certain handler:

```
interface StackUnderTestProps extends StackProps {
  architecture?: lambda.Architecture;
}

class StackUnderTest extends Stack {
  constructor(scope: Construct, id: string, props: StackUnderTestProps) {
    super(scope, id, props);

    new lambda.Function(this, 'Handler', {
      runtime: lambda.Runtime.NODEJS_LATEST,
      handler: 'index.handler',
      code: lambda.Code.fromAsset(path.join(__dirname, 'lambda-handler')),
      architecture: props.architecture,
    });
  }
}
```

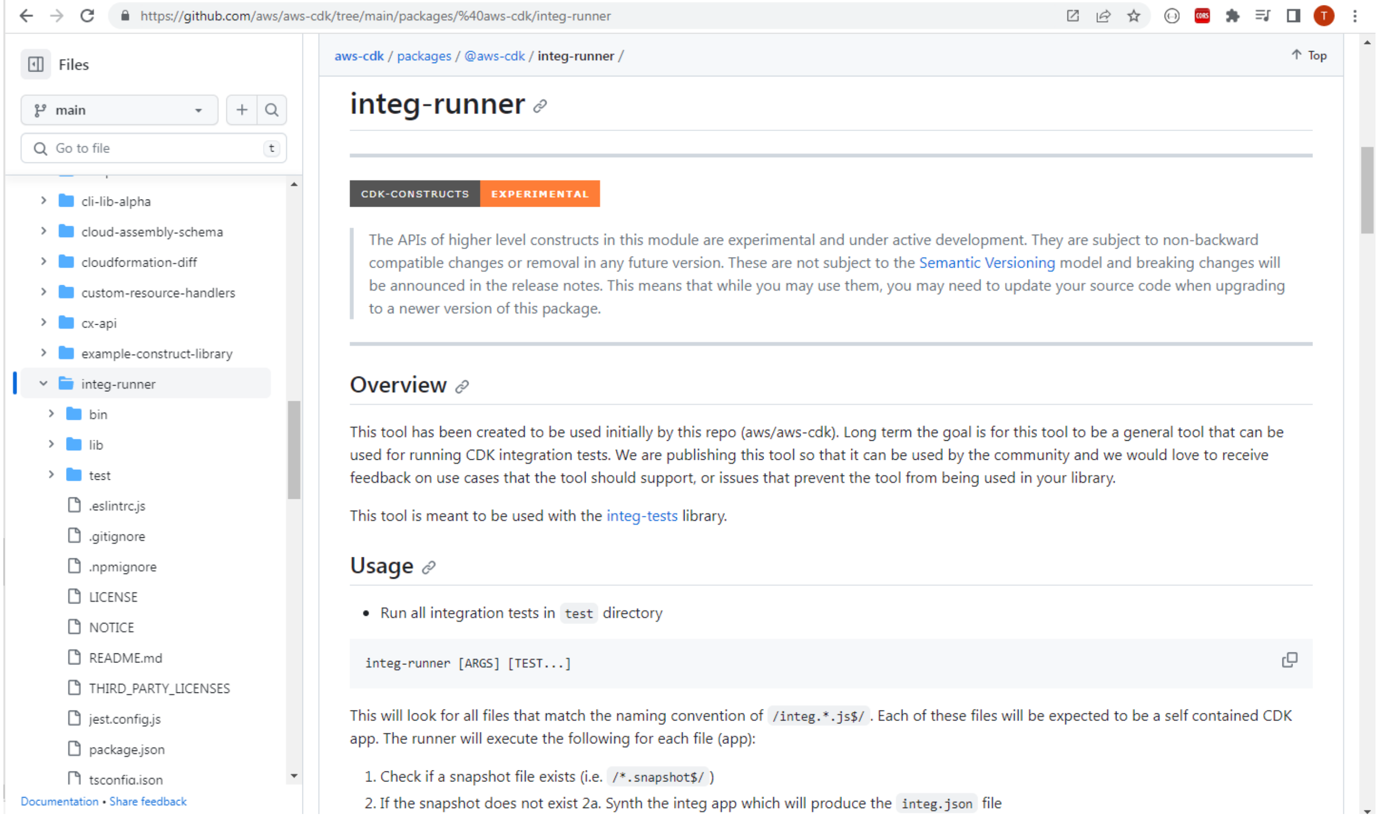


aws-cdk / packages / @aws-cdk / integ-runner /

aws-cdk-automation chore: npm-check-updates && yarn upgrade (#27330) ✓ c5750d0 · 14 hours ago

Name	Last commit message	Last commit date
..		
bin	feat: add new integration test runner (#19754)	last year
lib	chore: npm-check-updates && yarn upgrade (#26283)	2 months ago
test	chore(integ-tests-alpha): integ tests link is incorrect (#26402)	2 months ago
.eslintrc.js	feat(integ-runner): support `--language` presets for JavaScript, Type...	8 months ago
.gitignore	chore: remove and ignore intermediate snapshots integration test snap...	last year
.npmignore	chore: change name of integ test snapshot directories in preparation ...	last year
LICENSE	chore: updated Copyright year range for 2023 (#23525)	9 months ago
NOTICE	chore: updated Copyright year range for 2023 (#23525)	9 months ago
README.md	chore(integ-tests-alpha): integ tests link is incorrect (#26402)	2 months ago
THIRD_PARTY_LICENSES	chore: npm-check-updates && yarn upgrade (#27330)	14 hours ago
jest.config.js	feat: add new integration test runner (#19754)	last year
package.json	chore: npm-check-updates && yarn upgrade (#27330)	14 hours ago

https://github.com/aws/aws-cdk/tree/main/packages/%40aws-cdk/integ-runner



integ-runner

> The APIs of higher level constructs in this module are experimental and under active development. They are subject to non-backward compatible changes or removal in any future version. These are not subject to the Semantic Versioning model and breaking changes will be announced in the release notes. 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 tool has been created to be used initially by this repo (aws/aws-cdk). Long term the goal is for this tool to be a general tool that can be used for running CDK integration tests. We are publishing this tool so that it can be used by the community and we would love to receive feedback on use cases that the tool should support, or issues that prevent the tool from being used in your library.

This tool is meant to be used with the integ-tests library.

Usage

- Run all integration tests in `test` directory

```
integ-runner [ARGS] [TEST...]
```

This will look for all files that match the naming convention of `/integ.*.js$/`. Each of these files will be expected to be a self contained CDK app. The runner will execute the following for each file (app):

1. Check if a snapshot file exists (i.e. `/*.snapshot$/`)
2. If the snapshot does not exist 2a. Synth the integ app which will produce the `integ.json` file