:doctype: book

include::attributes.txt[]

// Attributes [.topic] :info\_titleabbrev: Projects :keywords: {aws} CDK, {aws} CloudFormation, IaC, Infrastructure as code, CDK projects, {aws} CDK concepts

[[projects,projects.title]] = {aws} CDK projects

== [abstract]

## An {aws} Cloud Development Kit ({aws} CDK) project represents the files and folders that contain your CDK code. Contents will vary based on your programming language.

// Content start

An {aws} Cloud Development Kit ({aws} CDK) project represents the files and folders that contain your CDK code. Contents will vary based on your programming language.

You can create your {aws} CDK project manually or with the {aws} CDK Command Line Interface ({aws} CDK CLI) cdk init command. In this topic, we will refer to the project structure and naming conventions of files and folders created by the {aws} CDK CLI. You can customize and organize your CDK projects to fit your needs.

= [NOTE]

Project structure created by the {aws} CDK CLI may vary across versions over time.

====

[#projects-universal] == Universal files and folders

[[projects-universal-git]] .git:: If you have git installed, the {aws} CDK CLI automatically initializes a [.noloc]Git repository for your project. The .git directory contains information about the repository.

[[projects-universal-gitignore]] .gitignore:: Text file used by [.noloc]Git to specify files and folders to ignore.

[[projects-universal-readme]] README.md:: Text file that provides you with basic guidance and important information for managing your {aws} CDK project. Modify this file as necessary to document important information regarding your CDK project.

[[projects-universal-cdk]] cdk.json:: Configuration file for the {aws} CDK. This file provides instruction to the {aws} CDK CLI regarding how to run your app.

[#projects-specific] == Language-specific files and folders

The following files and folders are unique to each supported programming language.

==== [role=“tablist”] TypeScript:: The following is an example project created in the my-cdk-ts-project directory using the cdk init --language typescript command: + [source,none,subs=“verbatim,attributes”] — my-cdk-ts-project ├── .git ├── .gitignore ├── .npmignore ├── README.md ├── bin │ └── my-cdk-ts-project.ts ├── cdk.json ├── jest.config.js ├── lib │ └── my-cdk-ts-project-stack.ts ├── node\_modules ├── package-lock.json ├── package.json ├── test │ └── my-cdk-ts-project.test.ts └── tsconfig.json —

.npmignore::: File that specifies which files and folders to ignore when publishing a package to the npm registry. This file is similar to .gitignore, but is specific to npm packages.

bin/my-cdk-ts-project.ts::: The *application file* defines your CDK app. CDK projects can contain one or more application files. Application files are stored in the bin folder. + The following is an example of a basic application file that defines a CDK app: + [source,javascript,subs=“verbatim,attributes”] — #!/usr/bin/env node import ‘source-map-support/register’; import \* as cdk from ‘aws-cdk-lib’; import { MyCdkTsProjectStack } from ‘../lib/my-cdk-ts-project-stack’;

const app = new cdk.App(); new MyCdkTsProjectStack(app, ‘MyCdkTsProjectStack’); —

jest.config.js::: Configuration file for [.noloc]Jest. [.noloc]*Jest* is a popular JavaScript testing framework.

lib/my-cdk-ts-project-stack.ts::: The *stack file* defines your CDK stack. Within your stack, you define {aws} resources and properties using constructs. + The following is an example of a basic stack file that defines a CDK stack: + [source,javascript,subs=“verbatim,attributes”] — import \* as cdk from ‘aws-cdk-lib’; import { Construct } from ‘constructs’;

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

// code that defines your resources and properties go here } } —

node\_modules::: Common folder in [.noloc]Node.js projects that contain dependencies for your project.

package-lock.json::: Metadata file that works with the package.json file to manage versions of dependencies.

package.json::: Metadata file that is commonly used in [.noloc]Node.js projects. This file contains information about your CDK project such as the project name, script definitions, dependencies, and other import project-level information.

test/my-cdk-ts-project.test.ts::: A test folder is created to organize tests for your CDK project. A sample test file is also created. + You can write tests in TypeScript and use [.noloc]Jest to compile your TypeScript code before running tests.

tsconfig.json::: Configuration file used in TypeScript projects that specifies compiler options and project settings.

JavaScript:: The following is an example project created in the my-cdk-js-project directory using the cdk init --language javascript command: + [source,none,subs=“verbatim,attributes”] — my-cdk-js-project ├── .git ├── .gitignore ├── .npmignore ├── README.md ├── bin │ └── my-cdk-js-project.js ├── cdk.json ├── jest.config.js ├── lib │ └── my-cdk-js-project-stack.js ├── node\_modules ├── package-lock.json ├── package.json └── test └── my-cdk-js-project.test.js —

.npmignore::: File that specifies which files and folders to ignore when publishing a package to the npm registry. This file is similar to .gitignore, but is specific to npm packages.

bin/my-cdk-js-project.js::: The *application file* defines your CDK app. CDK projects can contain one or more application files. Application files are stored in the bin folder. + The following is an example of a basic application file that defines a CDK app: + [source,javascript,subs=“verbatim,attributes”] — #!/usr/bin/env node

const cdk = require(‘aws-cdk-lib’); const { MyCdkJsProjectStack } = require(‘../lib/my-cdk-js-project-stack’);

const app = new cdk.App(); new MyCdkJsProjectStack(app, ‘MyCdkJsProjectStack’); —

jest.config.js::: Configuration file for [.noloc]Jest. [.noloc]*Jest* is a popular JavaScript testing framework.

lib/my-cdk-js-project-stack.js::: The *stack file* defines your CDK stack. Within your stack, you define {aws} resources and properties using constructs. + The following is an example of a basic stack file that defines a CDK stack: + [source,javascript,subs=“verbatim,attributes”] — const { Stack, Duration } = require(‘aws-cdk-lib’);

class MyCdkJsProjectStack extends Stack { constructor(scope, id, props) { super(scope, id, props);

// code that defines your resources and properties go here } }

== module.exports = { MyCdkJsProjectStack }

node\_modules::: Common folder in [.noloc]Node.js projects that contain dependencies for your project.

package-lock.json::: Metadata file that works with the package.json file to manage versions of dependencies.

package.json::: Metadata file that is commonly used in [.noloc]Node.js projects. This file contains information about your CDK project such as the project name, script definitions, dependencies, and other import project-level information.

test/my-cdk-js-project.test.js::: A test folder is created to organize tests for your CDK project. A sample test file is also created. + You can write tests in JavaScript and use [.noloc]Jest to compile your JavaScript code before running tests.

Python:: The following is an example project created in the my-cdk-py-project directory using the cdk init --language python command: + [source,none,subs=“verbatim,attributes”] — my-cdk-py-project ├── .git ├── .gitignore ├── .venv ├── README.md ├── app.py ├── cdk.json ├── my\_cdk\_py\_project │ ├── *init*.py │ └── my\_cdk\_py\_project\_stack.py ├── requirements-dev.txt ├── requirements.txt ├── source.bat └── tests ├── *init*.py └── unit —

.venv::: The CDK CLI automatically creates a virtual environment for your project. The .venv directory refers to this virtual environment.

app.py::: The *application file* defines your CDK app. CDK projects can contain one or more application files. + The following is an example of a basic application file that defines a CDK app: + [source,python,subs=“verbatim,attributes”] — #!/usr/bin/env python3 import os

import aws\_cdk as cdk

from my\_cdk\_py\_project.my\_cdk\_py\_project\_stack import MyCdkPyProjectStack

app = cdk.App() MyCdkPyProjectStack(app, “MyCdkPyProjectStack”)

== app.synth()

my\_cdk\_py\_project::: Directory that contains your *stack files*. The CDK CLI creates the following here: + –

* +*init*.py+ – An empty Python package definition file.
* {blank}
* == my\_cdk\_py\_project – File that defines your CDK stack. You then define {aws} resources and properties within the stack using constructs.
* The following is an example of a stack file:

## [source,python,subs=“verbatim,attributes”]

* from aws\_cdk import Stack

from constructs import Construct

class MyCdkPyProjectStack(Stack): def *init*(self, scope: Construct, construct\_id: str, **kwargs) -> None: super().*init*(scope, construct\_id,** kwargs)

== # code that defines your resources and properties go here

requirements-dev.txt::: File similar to requirements.txt, but used to manage dependencies specifically for development purposes rather than production.

requirements.txt::: Common file used in Python projects to specify and manage project dependencies.

source.bat::: Batch file for [.noloc]Windows that is used to set up the Python virtual environment.

tests::: Directory that contains tests for your CDK project. + The following is an example of a unit test: + [source,python,subs=“verbatim,attributes”] — import aws\_cdk as core import aws\_cdk.assertions as assertions

from my\_cdk\_py\_project.my\_cdk\_py\_project\_stack import MyCdkPyProjectStack

def test\_sqs\_queue\_created(): app = core.App() stack = MyCdkPyProjectStack(app, “my-cdk-py-project”) template = assertions.Template.from\_stack(stack)

template.has\_resource\_properties(“{aws}::SQS::Queue”, { “VisibilityTimeout”: 300 }) —

Java:: The following is an example project created in the my-cdk-java-project directory using the cdk init --language java command: + [source,none,subs=“verbatim,attributes”] — my-cdk-java-project ├── .git ├── .gitignore ├── README.md ├── cdk.json ├── pom.xml └── src ├── main └── test —

pom.xml::: File that contains configuration information and metadata about your CDK project. This file is a part of [.noloc]Maven.

src/main::: Directory containing your *application* and *stack* files. + The following is an example application file: + [source,java,subs=“verbatim,attributes”] — package com.myorg;

import software.amazon.awscdk.App; import software.amazon.awscdk.Environment; import software.amazon.awscdk.StackProps;

import java.util.Arrays;

public class MyCdkJavaProjectApp { public static void main(final String[] args) { App app = new App();

new MyCdkJavaProjectStack(app, “MyCdkJavaProjectStack”, StackProps.builder() .build());

app.synth(); } } — + The following is an example stack file: + [source,java,subs=“verbatim,attributes”] — package com.myorg;

import software.constructs.Construct; import software.amazon.awscdk.Stack; import software.amazon.awscdk.StackProps;

public class MyCdkJavaProjectStack extends Stack { public MyCdkJavaProjectStack(final Construct scope, final String id) { this(scope, id, null); }

public MyCdkJavaProjectStack(final Construct scope, final String id, final StackProps props) { super(scope, id, props);

// code that defines your resources and properties go here } } —

src/test::: Directory containing your test files. The following is an example: + [source,java,subs=“verbatim,attributes”] — package com.myorg;

import software.amazon.awscdk.App; import software.amazon.awscdk.assertions.Template; import java.io.IOException;

import java.util.HashMap;

import org.junit.jupiter.api.Test;

public class MyCdkJavaProjectTest {

@Test public void testStack() throws IOException { App app = new App(); MyCdkJavaProjectStack stack = new MyCdkJavaProjectStack(app, “test”);

Template template = Template.fromStack(stack);

template.hasResourceProperties(“{aws}::SQS::Queue”, new HashMap<String, Number>() {{ put(“VisibilityTimeout”, 300); }}); } } —

C#:: The following is an example project created in the my-cdk-csharp-project directory using the cdk init --language csharp command: + [source,none,subs=“verbatim,attributes”] — my-cdk-csharp-project ├── .git ├── .gitignore ├── README.md ├── cdk.json └── src ├── MyCdkCsharpProject └── MyCdkCsharpProject.sln —

src/MyCdkCsharpProject::: Directory containing your *application* and *stack* files. + The following is an example application file: + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using System; using System.Collections.Generic; using System.Linq;

namespace MyCdkCsharpProject { sealed class Program { public static void Main(string[] args) { var app = new App(); new MyCdkCsharpProjectStack(app, “MyCdkCsharpProjectStack”, new StackProps{}); app.Synth(); } } } — + The following is an example stack file: + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using Constructs;

namespace MyCdkCsharpProject { public class MyCdkCsharpProjectStack : Stack { internal MyCdkCsharpProjectStack(Construct scope, string id, IStackProps props = null) : base(scope, id, props) { // code that defines your resources and properties go here } } } — + This directory also contains the following: + –

* GlobalSuppressions.cs – File used to suppress specific compiler warnings or errors across your project.
* {blank}
* == .csproj – XML-based file used to define project settings, dependencies, and build configurations.

src/MyCdkCsharpProject.sln::: [.noloc]Microsoft Visual Studio Solution File used to organize and manage related projects.

[.noloc]Go:: The following is an example project created in the my-cdk-go-project directory using the cdk init --language go command: + [source,none,subs=“verbatim,attributes”] — my-cdk-go-project ├── .git ├── .gitignore ├── README.md ├── cdk.json ├── go.mod ├── my-cdk-go-project.go └── my-cdk-go-project\_test.go —

go.mod::: File that contains module information and is used to manage dependencies and versioning for your [.noloc]Go project.

my-cdk-go-project.go::: File that defines your CDK application and stacks. + The following is an example: + [source,go,subs=“verbatim,attributes”] — package main import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/constructs-go/constructs/v10” “github.com/aws/jsii-runtime-go” )

type MyCdkGoProjectStackProps struct { awscdk.StackProps }

func NewMyCdkGoProjectStack(scope constructs.Construct, id string, props \*MyCdkGoProjectStackProps) awscdk.Stack { var sprops awscdk.StackProps if props != nil { sprops = props.StackProps } stack := awscdk.NewStack(scope, &id, &sprops) // The code that defines your resources and properties go here

return stack }

func main() { defer jsii.Close() app := awscdk.NewApp(nil) NewMyCdkGoProjectStack(app, “MyCdkGoProjectStack”, &MyCdkGoProjectStackProps{ awscdk.StackProps{ Env: env(), }, }) app.Synth(nil) }

func env() \*awscdk.Environment {

return nil } —

my-cdk-go-project\_test.go::: File that defines a sample test. + The following is an example: + [source,go,subs=“verbatim,attributes”] — package main

import ( “testing”

“github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/aws-cdk-go/awscdk/v2/assertions” “github.com/aws/jsii-runtime-go” )

func TestMyCdkGoProjectStack(t \*testing.T) {

// GIVEN app := awscdk.NewApp(nil)

// WHEN stack := NewMyCdkGoProjectStack(app, “MyStack”, nil)

// THEN template := assertions.Template\_FromStack(stack, nil) template.HasResourceProperties(jsii.String(“{aws}::SQS::Queue”), map[string]interface{}{ “VisibilityTimeout”: 300, }) } — ====