:doctype: book

include::attributes.txt[]

// Attributes [.topic] :info\_titleabbrev: CDK stages :info\_abstract: An {aws} Cloud Development Kit ({aws} CDK) stage represents a group of one or more CDK stacks that are configured to + deploy together. Use stages to deploy the same grouping of stacks to multiple environments, such as development, testing, + and production. :keywords: {aws} CDK, CDK stages, Deployment, CDK stacks

[#stages] = Introduction to {aws} CDK stages

== [abstract]

## An {aws} Cloud Development Kit ({aws} CDK) *stage* represents a group of one or more CDK stacks that are configured to deploy together. Use stages to deploy the same grouping of stacks to multiple environments, such as development, testing, and production.

// Content start

An {aws} Cloud Development Kit ({aws} CDK) *stage* represents a group of one or more CDK stacks that are configured to deploy together. Use stages to deploy the same grouping of stacks to multiple environments, such as development, testing, and production.

To configure a CDK stage, import and use the https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.Stage.html[Stage] construct.

The following is a basic example that defines a CDK stage named MyAppStage. We add two CDK stacks, named AppStack and DatabaseStack to our stage. For this example, AppStack contains application resources and DatabaseStack contains database resources. We then create two instances of MyAppStage, for development and production environments:

==== [role=“tablist”] TypeScript:: In cdk-demo-app/lib/app-stack.ts: + [source,javascript,subs=“verbatim,attributes”] — import \* as cdk from ‘aws-cdk-lib’; import { Construct } from ‘constructs’;

// Define the app stack export class AppStack extends cdk.Stack { constructor(scope: Construct, id: string, props?: cdk.StackProps) { super(scope, id, props); // The code that defines your application goes here } } — +

In cdk-demo-app/lib/database-stack.ts: + [source,javascript,subs=“verbatim,attributes”] — import \* as cdk from ‘aws-cdk-lib’; import { Construct } from ‘constructs’;

// Define the database stack export class DatabaseStack extends cdk.Stack { constructor(scope: Construct, id: string, props?: cdk.StackProps) { super(scope, id, props); // The code that defines your database goes here } } — +

In cdk-demo-app/lib/my-stage.ts: + [source,javascript,subs=“verbatim,attributes”] — import \* as cdk from ‘aws-cdk-lib’; import { Construct } from ‘constructs’; import { Stage } from ‘aws-cdk-lib’; import { AppStack } from ‘./app-stack’; import { DatabaseStack } from ‘./database-stack’;

// Define the stage export class MyAppStage extends Stage { constructor(scope: Construct, id: string, props?: cdk.StageProps) { super(scope, id, props);

// Add both stacks to the stage new AppStack(this, ‘AppStack’); new DatabaseStack(this, ‘DatabaseStack’); } } —- +

In cdk-demo-app/bin/cdk-demo-app.ts: + [source,javascript,subs=“verbatim,attributes”] — #!/usr/bin/env node import ‘source-map-support/register’; import \* as cdk from ‘aws-cdk-lib’; import { MyAppStage } from ‘../lib/my-stage’;

// Create a CDK app const app = new cdk.App();

// Create the development stage new MyAppStage(app, ‘Dev’, { env: { account: ‘123456789012’, region: ‘us-east-1’ } });

// Create the production stage new MyAppStage(app, ‘Prod’, { env: { account: ‘098765432109’, region: ‘us-east-1’ } }); —

JavaScript:: In cdk-demo-app/lib/app-stack.js: + [source,javascript,subs=“verbatim,attributes”] — const { Stack } = require(‘aws-cdk-lib’);

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

// The code that defines your application goes here } }

== module.exports = { AppStack }

In cdk-demo-app/lib/database-stack.js: + [source,javascript,subs=“verbatim,attributes”] — const { Stack } = require(‘aws-cdk-lib’);

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

// The code that defines your database goes here } }

== module.exports = { DatabaseStack }

In cdk-demo-app/lib/my-stage.js: + [source,javascript,subs=“verbatim,attributes”] — const { Stage } = require(‘aws-cdk-lib’); const { AppStack } = require(‘./app-stack’); const { DatabaseStack } = require(‘./database-stack’);

// Define the stage class MyAppStage extends Stage { constructor(scope, id, props) { super(scope, id, props);

// Add both stacks to the stage new AppStack(this, ‘AppStack’); new DatabaseStack(this, ‘DatabaseStack’); } }

== module.exports = { MyAppStage };

In cdk-demo-app/bin/cdk-demo-app.js: + [source,javascript,subs=“verbatim,attributes”] — #!/usr/bin/env node

const cdk = require(‘aws-cdk-lib’); const { MyAppStage } = require(‘../lib/my-stage’);

// Create the CDK app const app = new cdk.App();

// Create the development stage new MyAppStage(app, ‘Dev’, { env: { account: ‘123456789012’, region: ‘us-east-1’, }, });

// Create the production stage new MyAppStage(app, ‘Prod’, { env: { account: ‘098765432109’, region: ‘us-east-1’, }, }); —

Python:: In cdk-demo-app/cdk\_demo\_app/app\_stack.py: + [source,python,subs=“verbatim,attributes”] — from aws\_cdk import Stack from constructs import Construct

= Define the app stack

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

# The code that defines your application goes here ---- +

In cdk-demo-app/cdk\_demo\_app/database\_stack.py: + [source,python,subs=“verbatim,attributes”] — from aws\_cdk import Stack from constructs import Construct

= Define the database stack

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

# The code that defines your database goes here ---- +

In cdk-demo-app/cdk\_demo\_app/my\_stage.py: + [source,python,subs=“verbatim,attributes”] — from aws\_cdk import Stage from constructs import Construct from .app\_stack import AppStack from .database\_stack import DatabaseStack

= Define the stage

class MyAppStage(Stage): def *init*(self, scope: Construct, id: str, **kwargs) -> None: super().*init*(scope, id,** kwargs)

# Add both stacks to the stage  
 AppStack(self, "AppStack")  
 DatabaseStack(self, "DatabaseStack") ---- +

In cdk-demo-app/app.py: + [source,python,subs=“verbatim,attributes”] — #!/usr/bin/env python3 import os

import aws\_cdk as cdk

from cdk\_demo\_app.my\_stage import MyAppStage

= Create a CDK app

app = cdk.App()

= Create the development stage

MyAppStage(app, ‘Dev’, env=cdk.Environment(account=‘123456789012’, region=‘us-east-1’), )

= Create the production stage

MyAppStage(app, ‘Prod’, env=cdk.Environment(account=‘098765432109’, region=‘us-east-1’), )

== app.synth()

Java:: In cdk-demo-app/src/main/java/com/myorg/AppStack.java: + [source,java,subs=“verbatim,attributes”] — package com.myorg;

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

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

…. public AppStack(final Construct scope, final String id, final StackProps props) { super(scope, id, props);

// The code that defines your application goes here

} } —- + ….

In cdk-demo-app/src/main/java/com/myorg/DatabaseStack.java: + [source,java,subs=“verbatim,attributes”] — package com.myorg;

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

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

…. public DatabaseStack(final Construct scope, final String id, final StackProps props) { super(scope, id, props);

// The code that defines your database goes here

} } —- + ….

In cdk-demo-app/src/main/java/com/myorg/MyAppStage.java: + [source,java,subs=“verbatim,attributes”] — package com.myorg;

import software.amazon.awscdk.Stage; import software.amazon.awscdk.StageProps; import software.constructs.Construct;

// Define the stage public class MyAppStage extends Stage { public MyAppStage(final Construct scope, final String id, final software.amazon.awscdk.Environment env) { super(scope, id, StageProps.builder().env(env).build());

// Add both stacks to the stage  
 new AppStack(this, "AppStack");  
 new DatabaseStack(this, "DatabaseStack");

} } —- +

In cdk-demo-app/src/main/java/com/myorg/CdkDemoAppApp.java: + [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 CdkDemoAppApp { public static void main(final String[] args) {

…. // Create a CDK app App app = new App();

// Create the development stage  
new MyAppStage(app, "Dev", Environment.builder()  
 .account("123456789012")  
 .region("us-east-1")  
 .build());  
  
// Create the production stage  
new MyAppStage(app, "Prod", Environment.builder()  
.account("098765432109")  
.region("us-east-1")  
.build());  
  
app.synth();

} } —- ….

C#:: In cdk-demo-app/src/CdkDemoApp/AppStack.cs: + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using Constructs;

namespace CdkDemoApp { public class AppStack : Stack { internal AppStack(Construct scope, string id, IStackProps props = null) : base(scope, id, props) { // The code that defines your application goes here } } } — +

In cdk-demo-app/src/CdkDemoApp/DatabaseStack.cs: + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using Constructs;

namespace CdkDemoApp { public class DatabaseStack : Stack { internal DatabaseStack(Construct scope, string id, IStackProps props = null) : base(scope, id, props) { // The code that defines your database goes here } } } — +

In cdk-demo-app/src/CdkDemoApp/MyAppStage.cs: + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using Constructs;

namespace CdkDemoApp { // Define the stage public class MyAppStage : Stage { internal MyAppStage(Construct scope, string id, Environment env) : base(scope, id, new StageProps { Env = env }) { // Add both stacks to the stage new AppStack(this, “AppStack”); new DatabaseStack(this, “DatabaseStack”); } } } — +

In cdk-demo-app/src/CdkDemoApp/program.cs: + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using System; using System.Collections.Generic; using System.Linq;

namespace CdkDemoApp { sealed class Program { public static void Main(string[] args) { // Create a CDK app var app = new App();

…. // Create the development stage new MyAppStage(app, “Dev”, new Amazon.CDK.Environment { Account = “123456789012”, Region = “us-east-1” });

// Create the production stage  
 new MyAppStage(app, "Prod", new Amazon.CDK.Environment  
 {  
 Account = "098765432109",  
 Region = "us-east-1"  
 });  
  
 app.Synth();  
}

} } —- ….

Go:: In cdk-demo-app/cdk-demo-app.go: + [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” )

// Define the app stack type AppStackProps struct { awscdk.StackProps }

func NewAppStack(scope constructs.Construct, id string, props \*AppStackProps) awscdk.Stack { stack := awscdk.NewStack(scope, &id, &props.StackProps)

…. // The code that defines your application goes here

return stack } ….

// Define the database stack type DatabaseStackProps struct { awscdk.StackProps }

func NewDatabaseStack(scope constructs.Construct, id string, props \*DatabaseStackProps) awscdk.Stack { stack := awscdk.NewStack(scope, &id, &props.StackProps)

…. // The code that defines your database goes here

return stack } ….

// Define the stage type MyAppStageProps struct { awscdk.StageProps }

func NewMyAppStage(scope constructs.Construct, id string, props \*MyAppStageProps) awscdk.Stage { stage := awscdk.NewStage(scope, &id, &props.StageProps)

…. // Add both stacks to the stage NewAppStack(stage, “AppStack”, &AppStackProps{ StackProps: awscdk.StackProps{ Env: props.Env, }, })

NewDatabaseStack(stage, “DatabaseStack”, &DatabaseStackProps{ StackProps: awscdk.StackProps{ Env: props.Env, }, })

return stage } ….

func main() { defer jsii.Close()

…. // Create a CDK app app := awscdk.NewApp(nil)

// Create the development stage NewMyAppStage(app, “Dev”, &MyAppStageProps{ StageProps: awscdk.StageProps{ Env: &awscdk.Environment{ Account: jsii.String(“123456789012”), Region: jsii.String(“us-east-1”), }, }, })

// Create the production stage NewMyAppStage(app, “Prod”, &MyAppStageProps{ StageProps: awscdk.StageProps{ Env: &awscdk.Environment{ Account: jsii.String(“098765432109”), Region: jsii.String(“us-east-1”), }, }, })

app.Synth(nil) } ….

func env() \*awscdk.Environment { return nil } — ====

When we run cdk synth, two cloud assemblies are created in cdk.out. These two cloud assemblies contain the synthesized {aws} CloudFormation template and assets for each stage. The following is snippet of our project directory:

==== [role=“tablist”] TypeScript:: + [source,none,subs=“verbatim,attributes”] — cdk-demo-app ├── bin │ └── cdk-demo-app.ts ├── cdk.out │ ├── assembly-Dev │ │ ├── DevAppStack++++++.assets.json │ │ ├── DevAppStack++++++.template.json │ │ ├── DevDatabaseStack++++++.assets.json │ │ ├── DevDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json │ ├── assembly-Prod │ │ ├── ProdAppStack++++++.assets.json │ │ ├── ProdAppStack++++++.template.json │ │ ├── ProdDatabaseStack++++++.assets.json │ │ ├── ProdDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json └── lib ├── app-stack.ts ├── database-stack.ts └── my-stage.ts —-++++++++++++++++++++++++++++++++++++++++++++++++

JavaScript:: + [source,none,subs=“verbatim,attributes”] — cdk-demo-app ├── bin │ └── cdk-demo-app.js ├── cdk.out │ ├── assembly-Dev │ │ ├── DevAppStack++++++.assets.json │ │ ├── DevAppStack++++++.template.json │ │ ├── DevDatabaseStack++++++.assets.json │ │ ├── DevDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json │ ├── assembly-Prod │ │ ├── ProdAppStack++++++.assets.json │ │ ├── ProdAppStack++++++.template.json │ │ ├── ProdDatabaseStack++++++.assets.json │ │ ├── ProdDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json └── lib ├── app-stack.js ├── database-stack.js └── my-stage.js —-++++++++++++++++++++++++++++++++++++++++++++++++

Python:: + [source,none,subs=“verbatim,attributes”] — cdk-demo-app ├── app.py ├── cdk.out │ ├── assembly-Dev │ │ ├── DevAppStack++++++.assets.json │ │ ├── DevAppStack++++++.template.json │ │ ├── DevDatabaseStack++++++.assets.json │ │ ├── DevDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json │ ├── assembly-Prod │ │ ├── ProdAppStack++++++.assets.json │ │ ├── ProdAppStack++++++.template.json │ │ ├── ProdDatabaseStack++++++.assets.json │ │ ├── ProdDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json │ ├── cdk.out │ ├── manifest.json │ └── tree.json └── cdk\_demo\_app ├── **init**.py ├── app\_stack.py ├── database\_stack.py └── my\_stage.py —-++++++++++++++++++++++++++++++++++++++++++++++++

Java:: + [source,none,subs=“verbatim,attributes”] — cdk-demo-app ├── cdk.out │ ├── assembly-Dev │ │ ├── DevAppStack++++++.assets.json │ │ ├── DevAppStack++++++.template.json │ │ ├── DevDatabaseStack++++++.assets.json │ │ ├── DevDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json │ ├── assembly-Prod │ │ ├── ProdAppStack++++++.assets.json │ │ ├── ProdAppStack++++++.template.json │ │ ├── ProdDatabaseStack++++++.assets.json │ │ ├── ProdDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json │ ├── cdk.out │ ├── manifest.json │ └── tree.json └── src └── main └── java └── com └── myorg ├── AppStack.java ├── CdkDemoAppApp.java ├── DatabaseStack.java └── MyAppStage.java —-++++++++++++++++++++++++++++++++++++++++++++++++

C#:: + [source,none,subs=“verbatim,attributes”] — cdk-demo-app ├── cdk.out │ ├── assembly-Dev │ │ ├── DevAppStack++++++.assets.json │ │ ├── DevAppStack++++++.template.json │ │ ├── DevDatabaseStack++++++.assets.json │ │ ├── DevDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json │ ├── assembly-Prod │ │ ├── ProdAppStack++++++.assets.json │ │ ├── ProdAppStack++++++.template.json │ │ ├── ProdDatabaseStack++++++.assets.json │ │ ├── ProdDatabaseStack++++++.template.json │ │ ├── cdk.out │ │ └── manifest.json │ ├── cdk.out │ ├── manifest.json │ └── tree.json └── src └── CdkDemoApp ├── AppStack.cs ├── DatabaseStack.cs ├── MyAppStage.cs └── Program.cs —-++++++++++++++++++++++++++++++++++++++++++++++++

Go:: + [source,none,subs=“verbatim,attributes”] — cdk-demo-app ├── cdk-demo-app.go └── cdk.out ├── assembly-Dev │ ├── DevAppStack++++++.assets.json │ ├── DevAppStack++++++.template.json │ ├── DevDatabaseStack++++++.assets.json │ ├── DevDatabaseStack++++++.template.json │ ├── cdk.out │ └── manifest.json ├── assembly-Prod │ ├── ProdAppStack++++++.assets.json │ ├── ProdAppStack++++++.template.json │ ├── ProdDatabaseStack++++++.assets.json │ ├── ProdDatabaseStack++++++.template.json │ ├── cdk.out │ └── manifest.json ├── cdk.out ├── manifest.json └── tree.json —-++++++++++++++++++++++++++++++++++++++++++++++++

When we list our stacks with cdk list, we see a total of four stacks:

== [source,none,subs=“verbatim,attributes”]

$ cdk list Dev/AppStack (Dev-AppStack) Dev/DatabaseStack (Dev-DatabaseStack) Prod/AppStack (Prod-AppStack) Prod/DatabaseStack (Prod-DatabaseStack) —

To deploy a specific stage, we run cdk deploy and provide the stacks to deploy. The following is an example that uses the /\* wildcard to deploy both stacks in our Dev stage:

== [source,none,subs=“verbatim,attributes”]

$ cdk deploy <“Dev/\*“>

✨ Synthesis time: 3.18s

Dev/AppStack (Dev-AppStack) Dev/AppStack (Dev-AppStack): deploying… [1/2]

✅ Dev/AppStack (Dev-AppStack)

✨ Deployment time: 1.11s

Stack ARN: …

✨ Total time: 4.29s

Dev/DatabaseStack (Dev-DatabaseStack) Dev/DatabaseStack (Dev-DatabaseStack): deploying… [2/2]

✅ Dev/DatabaseStack (Dev-DatabaseStack)

✨ Deployment time: 1.09s

Stack ARN: …

== ✨ Total time: 4.27s

====