[#cdk-pipeline-define] == Define a pipeline

Your CDK Pipelines application will include at least two stacks: one that represents the pipeline itself, and one or more stacks that represent the application deployed through it. Stacks can also be grouped into *stages*, which you can use to deploy copies of infrastructure stacks to different environments. For now, we’ll consider the pipeline, and later delve into the application it will deploy.

The construct +link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.pipelines.CodePipeline.html[CodePipeline]+ is the construct that represents a CDK Pipeline that uses {aws} CodePipeline as its deployment engine. When you instantiate CodePipeline in a stack, you define the source location for the pipeline (such as a GitHub repository). You also define the commands to build the app.

For example, the following defines a pipeline whose source is stored in a GitHub repository. It also includes a build step for a TypeScript CDK application. Fill in the information about your GitHub repo where indicated.

= [NOTE]

By default, the pipeline authenticates to GitHub using a personal access token stored in Secrets Manager under the name github-token.

====

You’ll also need to update the instantiation of the pipeline stack to specify the {aws} account and Region.

==== [role=“tablist”] TypeScript:: In lib/my-pipeline-stack.ts (may vary if your project folder isn’t named my-pipeline): + [source,javascript,subs=“verbatim,attributes”] — import \* as cdk from ‘aws-cdk-lib’; import { Construct } from ‘constructs’; import { CodePipeline, CodePipelineSource, ShellStep } from ‘aws-cdk-lib/pipelines’;

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

const pipeline = new CodePipeline(this, ‘Pipeline’, { pipelineName: ‘MyPipeline’, synth: new ShellStep(‘Synth’, { input: CodePipelineSource.gitHub(‘OWNER/REPO’, ‘main’), commands: [‘npm ci’, ‘npm run build’, ‘npx cdk synth’] }) }); } } —- + In bin/my-pipeline.ts (may vary if your project folder isn’t named my-pipeline): + [source,javascript,subs=“verbatim,attributes”] —- #!/usr/bin/env node import \* as cdk from ‘aws-cdk-lib’; import { MyPipelineStack } from ‘../lib/my-pipeline-stack’;

const app = new cdk.App(); new MyPipelineStack(app, ‘MyPipelineStack’, { env: { account: ‘111111111111’, region: ‘eu-west-1’, } });

== app.synth();

JavaScript:: In lib/my-pipeline-stack.js (may vary if your project folder isn’t named my-pipeline): + [source,javascript,subs=“verbatim,attributes”] — const cdk = require(‘aws-cdk-lib’); const { CodePipeline, CodePipelineSource, ShellStep } = require(‘aws-cdk-lib/pipelines’);

class MyPipelineStack extends cdk.Stack { constructor(scope, id, props) { super(scope, id, props);

const pipeline = new CodePipeline(this, ‘Pipeline’, { pipelineName: ‘MyPipeline’, synth: new ShellStep(‘Synth’, { input: CodePipelineSource.gitHub(‘OWNER/REPO’, ‘main’), commands: [‘npm ci’, ‘npm run build’, ‘npx cdk synth’] }) }); } }

== module.exports = { MyPipelineStack }

* In bin/my-pipeline.js (may vary if your project folder isn’t named my-pipeline):

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

* #!/usr/bin/env node

const cdk = require(‘aws-cdk-lib’); const { MyPipelineStack } = require(‘../lib/my-pipeline-stack’);

const app = new cdk.App(); new MyPipelineStack(app, ‘MyPipelineStack’, { env: { account: ‘111111111111’, region: ‘eu-west-1’, } });

== app.synth();

Python:: In my-pipeline/my-pipeline-stack.py (may vary if your project folder isn’t named my-pipeline): + + [source,python,subs=“verbatim,attributes”] — import aws\_cdk as cdk from constructs import Construct from aws\_cdk.pipelines import CodePipeline, CodePipelineSource, ShellStep

class MyPipelineStack(cdk.Stack):

…. def **init**(self, scope: Construct, construct\_id: str, \*\*kwargs) -> None: super().\_\_init\_\_(scope, construct\_id, \*\*kwargs)

pipeline = CodePipeline(self, "Pipeline",  
 pipeline\_name="MyPipeline",  
 synth=ShellStep("Synth",  
 input=CodePipelineSource.git\_hub("OWNER/REPO", "main"),  
 commands=["npm install -g aws-cdk",  
 "python -m pip install -r requirements.txt",  
 "cdk synth"]  
 )  
 ) ---- + In `app.py`: + [source,python,subs="verbatim,attributes"] ---- #!/usr/bin/env python3 import aws\_cdk as cdk from my\_pipeline.my\_pipeline\_stack import MyPipelineStack

….

app = cdk.App() MyPipelineStack(app, “MyPipelineStack”, env=cdk.Environment(account=“111111111111”, region=“eu-west-1”) )

== app.synth()

Java:: In src/main/java/com/myorg/MyPipelineStack.java (may vary if your project folder isn’t named my-pipeline): + [source,java,subs=“verbatim,attributes”] — package com.myorg;

import java.util.Arrays; import software.constructs.Construct; import software.amazon.awscdk.Stack; import software.amazon.awscdk.StackProps; import software.amazon.awscdk.pipelines.CodePipeline; import software.amazon.awscdk.pipelines.CodePipelineSource; import software.amazon.awscdk.pipelines.ShellStep;

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

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

CodePipeline pipeline = CodePipeline.Builder.create(this, "pipeline")  
 .pipelineName("MyPipeline")  
 .synth(ShellStep.Builder.create("Synth")  
 .input(CodePipelineSource.gitHub("OWNER/REPO", "main"))  
 .commands(Arrays.asList("npm install -g aws-cdk", "cdk synth"))  
 .build())  
 .build();

} } —- + In src/main/java/com/myorg/MyPipelineApp.java (may vary if your project folder isn’t named my-pipeline): + [source,java,subs=“verbatim,attributes”] —- package com.myorg; ….

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

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

…. new MyPipelineStack(app, “PipelineStack”, StackProps.builder() .env(Environment.builder() .account(“111111111111”) .region(“eu-west-1”) .build()) .build());

app.synth();

} } —- ….

C#:: In src/MyPipeline/MyPipelineStack.cs (may vary if your project folder isn’t named my-pipeline): + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using Amazon.CDK.Pipelines;

namespace MyPipeline { public class MyPipelineStack : Stack { internal MyPipelineStack(Construct scope, string id, IStackProps props = null) : base(scope, id, props) { var pipeline = new CodePipeline(this, “pipeline”, new CodePipelineProps { PipelineName = “MyPipeline”, Synth = new ShellStep(“Synth”, new ShellStepProps { Input = CodePipelineSource.GitHub(“OWNER/REPO”, “main”), Commands = new string[] { “npm install -g aws-cdk”, “cdk synth” } }) }); } } } — + In src/MyPipeline/Program.cs (may vary if your project folder isn’t named my-pipeline): + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK;

namespace MyPipeline { sealed class Program { public static void Main(string[] args) { var app = new App(); new MyPipelineStack(app, “MyPipelineStack”, new StackProps { Env = new Amazon.CDK.Environment { Account = “111111111111”, Region = “eu-west-1” } });

app.Synth();  
 }

} } —-

Go:: + [source,go,subs=“verbatim,attributes”] — package main

import ( “github.com/aws/aws-cdk-go/awscdk/v2” codebuild “github.com/aws/aws-cdk-go/awscdk/v2/awscodebuild” ssm “github.com/aws/aws-cdk-go/awscdk/v2/awsssm” pipeline “github.com/aws/aws-cdk-go/awscdk/v2/pipelines” “github.com/aws/constructs-go/constructs/v10” “github.com/aws/jsii-runtime-go” “os” )

// my CDK Stack with resources func NewCdkStack(scope constructs.Construct, id *string, props* awscdk.StackProps) awscdk.Stack { stack := awscdk.NewStack(scope, id, props)

…. // create an example ssm parameter \_ = ssm.NewStringParameter(stack, jsii.String(“ssm-test-param”), &ssm.StringParameterProps{ ParameterName: jsii.String(“/testparam”), Description: jsii.String(“ssm parameter for demo”), StringValue: jsii.String(“my test param”), })

return stack } ….

// my CDK Application func NewCdkApplication(scope constructs.Construct, id *string, props* awscdk.StageProps) awscdk.Stage { stage := awscdk.NewStage(scope, id, props)

…. \_ = NewCdkStack(stage, jsii.String(“cdk-stack”), &awscdk.StackProps{Env: props.Env})

return stage } ….

// my CDK Pipeline func NewCdkPipeline(scope constructs.Construct, id *string, props* awscdk.StackProps) awscdk.Stack { stack := awscdk.NewStack(scope, id, props)

…. // GitHub repo with owner and repository name githubRepo := pipeline.CodePipelineSource\_GitHub(jsii.String(“owner/repo”), jsii.String(“main”), &pipeline.GitHubSourceOptions{ Authentication: awscdk.SecretValue\_SecretsManager(jsii.String(“my-github-token”), nil), })

// self mutating pipeline myPipeline := pipeline.NewCodePipeline(stack, jsii.String(“cdkPipeline”), &pipeline.CodePipelineProps{ PipelineName: jsii.String(“CdkPipeline”), // self mutation true - pipeline changes itself before application deployment SelfMutation: jsii.Bool(true), CodeBuildDefaults: &pipeline.CodeBuildOptions{ BuildEnvironment: &codebuild.BuildEnvironment{ // image version 6.0 recommended for newer go version BuildImage: codebuild.LinuxBuildImage\_FromCodeBuildImageId(jsii.String(“aws/codebuild/standard:6.0”)), }, }, Synth: pipeline.NewCodeBuildStep(jsii.String(“Synth”), &pipeline.CodeBuildStepProps{ Input: githubRepo, Commands: &[]\*string{ jsii.String(“npm install -g aws-cdk”), jsii.String(“cdk synth”), }, }), })

// deployment of actual CDK application myPipeline.AddStage(NewCdkApplication(stack, jsii.String(“MyApplication”), &awscdk.StageProps{ Env: targetAccountEnv(), }), &pipeline.AddStageOpts{ Post: &[]pipeline.Step{ pipeline.NewCodeBuildStep(jsii.String(“Manual Steps”), &pipeline.CodeBuildStepProps{ Commands: &[]\*string{ jsii.String(“echo "My CDK App deployed, manual steps go here … "”), }, }), }, })

return stack } ….

// main app func main() { defer jsii.Close()

…. app := awscdk.NewApp(nil)

// call CDK Pipeline NewCdkPipeline(app, jsii.String(“CdkPipelineStack”), &awscdk.StackProps{ Env: pipelineEnv(), })

app.Synth(nil) } ….

// env determines the {aws} environment (account+region) in which our stack is to // be deployed. For more information see: https://docs.aws.amazon.com/cdk/latest/guide/environments.html func pipelineEnv() \*awscdk.Environment { return &awscdk.Environment{ Account: jsii.String(os.Getenv(“CDK\_DEFAULT\_ACCOUNT”)), Region: jsii.String(os.Getenv(“CDK\_DEFAULT\_REGION”)), } }

func targetAccountEnv() \*awscdk.Environment { return &awscdk.Environment{ Account: jsii.String(os.Getenv(“CDK\_DEFAULT\_ACCOUNT”)), Region: jsii.String(os.Getenv(“CDK\_DEFAULT\_REGION”)), } } — ====

You must deploy a pipeline manually once. After that, the pipeline keeps itself up to date from the source code repository. So be sure that the code in the repo is the code you want deployed. Check in your changes and push to GitHub, then deploy:

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

git add –all git commit -m “initial commit” git push cdk deploy —

= [TIP]

Now that you’ve done the initial deployment, your local {aws} account no longer needs administrative access. This is because all changes to your app will be deployed via the pipeline. All you need to be able to do is push to GitHub.

====