|  |
| --- |
| **CI/CD Deployment for Springboot Application.**  **By**  **ANANT KHAIRNAR** |

**Date of Submission: 22 -NOV -2023**

**Source code Git Locatio**

**https://github.com/anantkhairnar/CI-CD-Deployment-for-Springboot-Application.git**

**Source Code:**

**CI/CD Deployment for Springboot Application.**

import { Repository } from 'aws-cdk-lib/aws-codecommit'

import { BuildSpec } from 'aws-cdk-lib/aws-codebuild'

import { PolicyStatement } from 'aws-cdk-lib/aws-iam'

import { CodeBuildStep, CodePipeline, CodePipelineSource, ShellStep } from 'aws-cdk-lib/pipelines'

import { CfnOutput, Stack, StackProps } from 'aws-cdk-lib'

import { Construct } from 'constructs'

import { Deployment } from './stages'

export class CodePipelineStack extends Stack {

constructor (scope: Construct, id: string, props?: StackProps) {

super(scope, id, props)

const repo = new Repository(this, 'Repository', {

repositoryName: 'SampleRepository',

description: 'This is sample repository for the project.'

})

const validatePolicy = new PolicyStatement({

actions: [

'cloudformation:DescribeStacks',

'events:DescribeEventBus'

],

resources: ['\*']

})

const pipeline = new CodePipeline(this, 'Pipeline', {

synth: new ShellStep('Synth', {

input: CodePipelineSource.codeCommit(repo, 'main'),

installCommands: [

'make warming'

],

commands: [

'make build'

]

})

})

// Add dev deployment

const devStage = new Deployment(this, 'Dev')

pipeline.addStage(devStage, {

// Execute all sequence of actions before deployment

pre: [

new CodeBuildStep('Linting', {

installCommands: [

'make warming'

],

commands: [

'make linting'

]

}),

new CodeBuildStep('UnitTest', {

installCommands: [

'make warming'

],

commands: [

'make unittest'

],

partialBuildSpec: BuildSpec.fromObject({

reports: {

coverage: {

files: [

'./coverage/clover.xml'

],

'file-format': 'CLOVERXML'

},

unittest: {

files: [

'./test-report.xml'

],

'file-format': 'JUNITXML'

}

}

}),

rolePolicyStatements: [

new PolicyStatement({

actions: [

'codebuild:CreateReportGroup',

'codebuild:CreateReport',

'codebuild:UpdateReport',

'codebuild:BatchPutTestCases',

'codebuild:BatchPutCodeCoverages'

],

resources: ['\*']

})

]

}),

new CodeBuildStep('Security', {

installCommands: [

'make warming',

'gem install cfn-nag'

],

commands: [

'make build',

'make security'

],

partialBuildSpec: BuildSpec.fromObject({

phases: {

install: {

'runtime-versions': {

ruby: '2.6'

}

}

}

})

})

],

// Execute validation check for post-deployment

post: [

new CodeBuildStep('Validate', {

env: {

STAGE: devStage.stageName

},

installCommands: [

'make warming'

],

commands: [

'make validate'

],

rolePolicyStatements: [validatePolicy]

})

]

})

// Add test deployment

const testStage = new Deployment(this, 'Test')

pipeline.addStage(testStage, {

// Execute validation check for post-deployment

post: [

new CodeBuildStep('Validate', {

env: {

STAGE: testStage.stageName

},

installCommands: [

'make warming'

],

commands: [

'make validate'

],

rolePolicyStatements: [validatePolicy]

})

]

})

// Add prod deployment

const prodStage = new Deployment(this, 'Prod')

pipeline.addStage(prodStage, {

// Execute validation check for post-deployment

post: [

new CodeBuildStep('Validate', {

env: {

STAGE: prodStage.stageName

},

installCommands: [

'make warming'

],

commands: [

'make validate'

],

rolePolicyStatements: [validatePolicy]

})

]

})

// Output

new CfnOutput(this, 'RepositoryName', {

value: repo.repositoryName

})

}

}

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |