**Create Two Stack in single project**

The whole procedure will be the same as we created an app using cdk. As prerequisites all set up completely and then:

1. Create a new directory for CDK project:> mkdir cdk-project
2. Change the directory to newly created directory: >cd cdk-project
3. Initialization of CDK project: > cdk init --language typescript
4. Adding some line of code on bin folder project.ts :

*new* Stack1(app, 'Stack1');

*new* Stack2(app, 'Stack2');

5. Now need to add two files (that will import by project.ts file) to create that stack object in Lib folder namely:

stack1.ts : have some bucket code(Do not you same bucket name)

stack2.ts:have also some bucket code

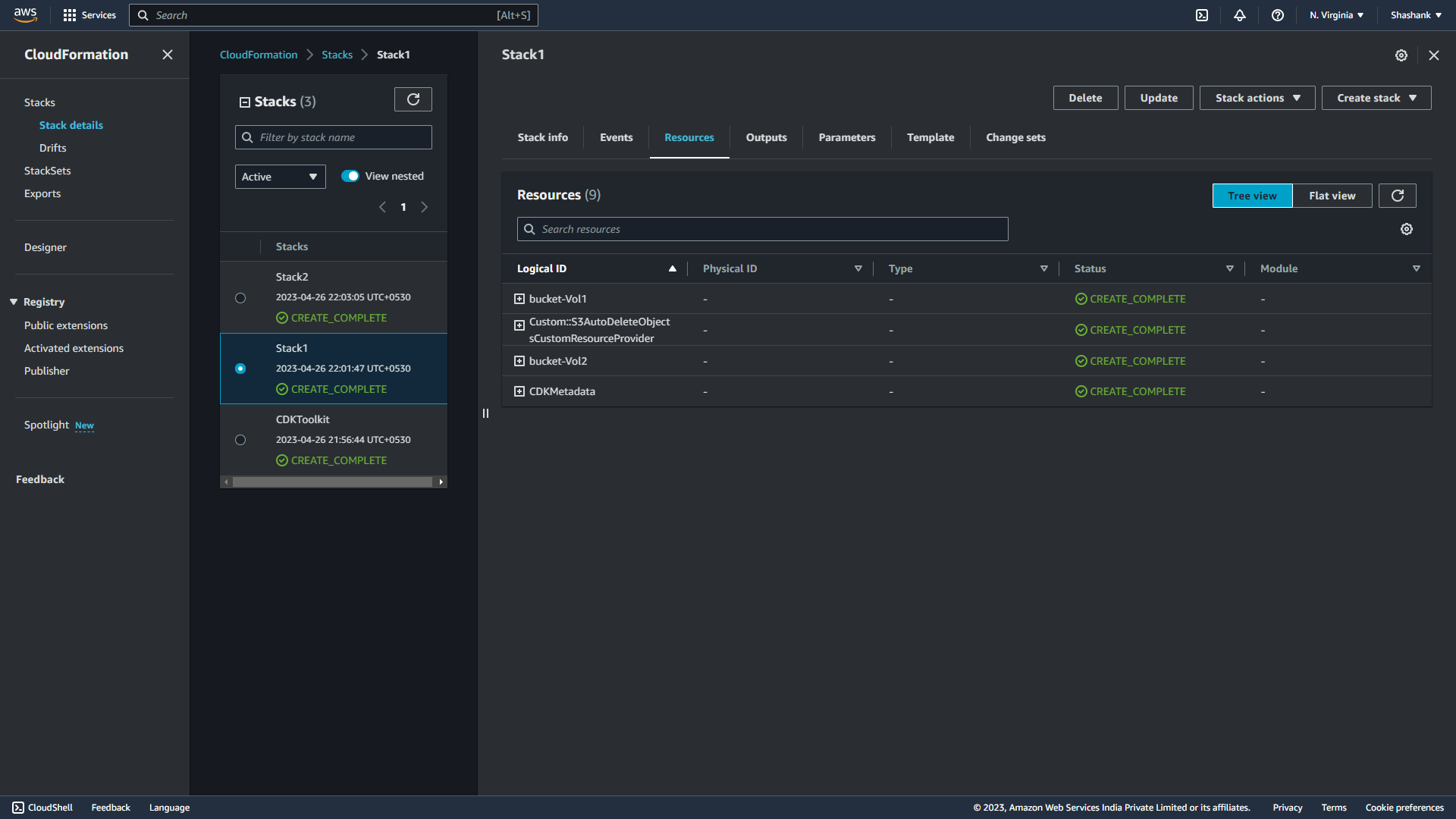
6. Now there is need to create executable of these files in my case it's in JavaScript :> npm run build

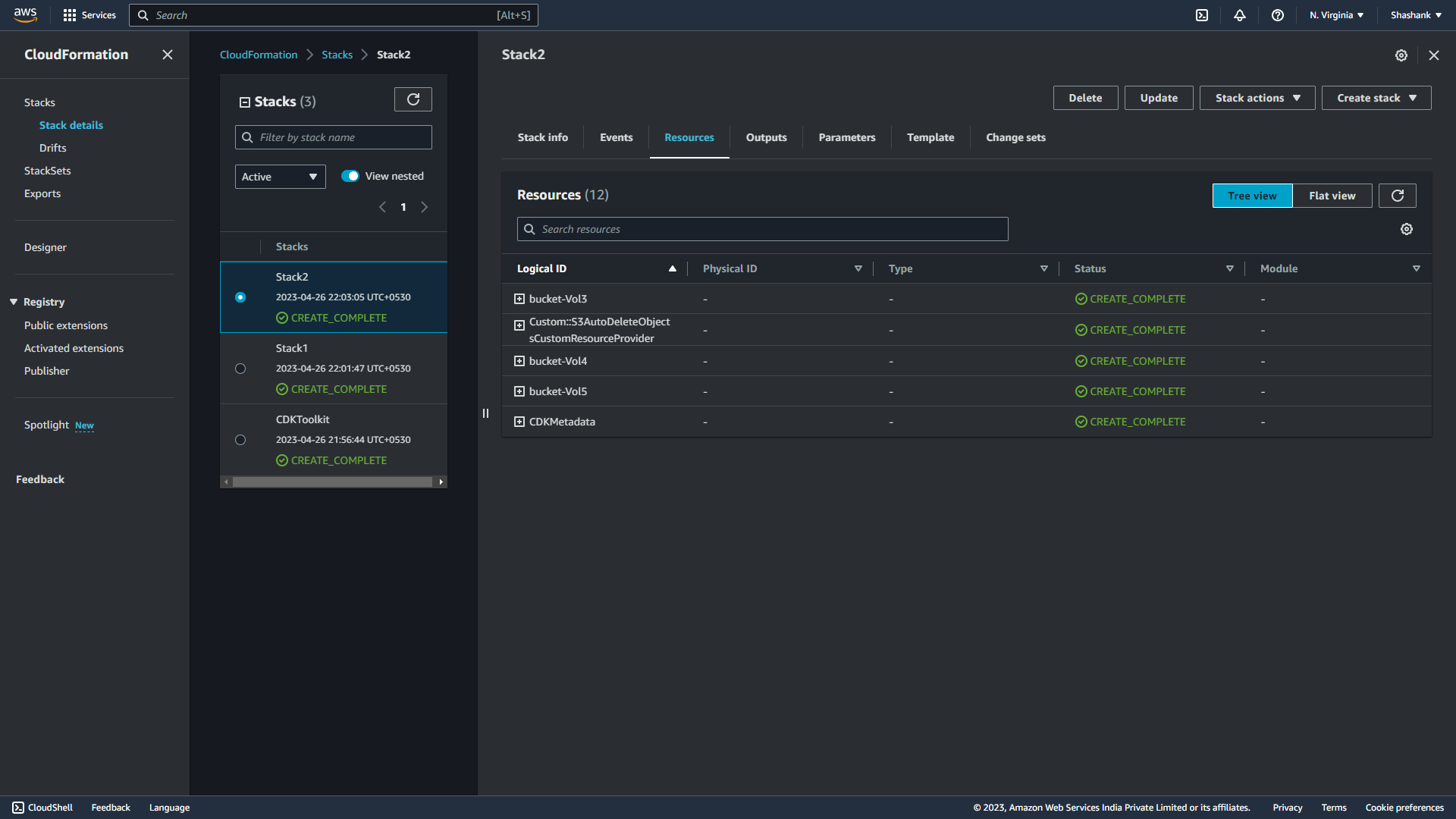
7. Next we will **synthesise** to create cloud formation template : cdk synth

8. Then we need to bootstrap:cdk bootstrap

9. finally we have to deploy it :>cdk deploy –all (because here we have created two stack)

Following result :





**Create Three app in single project**

The whole procedure will be the same as we created a stack using cdk. As prerequisites all set up completely and then:

1. Create a new directory for CDK project:> mkdir cdk-project
2. Change the directory to newly created directory: >cd cdk-project
3. Initialization of CDK project: > cdk init --language typescript
4. Adding some line of code on bin folder app\_project.ts :

*import* { OneStack } *from* '../lib/one-stack';//here I import stack resources code

*import* { TwoStack } *from* '../lib/two-stack';//here I import stack resources code

*import* { ThreeStack } *from* '../lib/three-stack'; //here I import stack resources code

// creating application3

*const* app3 *=* *new* cdk.App();

*new* ThreeStack(app3, 'ThreeStack');

app3.synth()

// creating application2

*const* app2 *=* *new* cdk.App();

*new* TwoStack(app2, 'TwoStack');

app2.synth()

// creaing application1

*const* app1 *=* *new* cdk.App();

*new* OneStack(app1, 'OneStack');

app1.synth()

5. Now need to add two files (that will import by project.ts file) to create that stack object in Lib folder as every app has their own stack namely:

One-stack.ts : have some bucket code(Do not you same bucket name)

Two-stack.ts:have also some bucket code

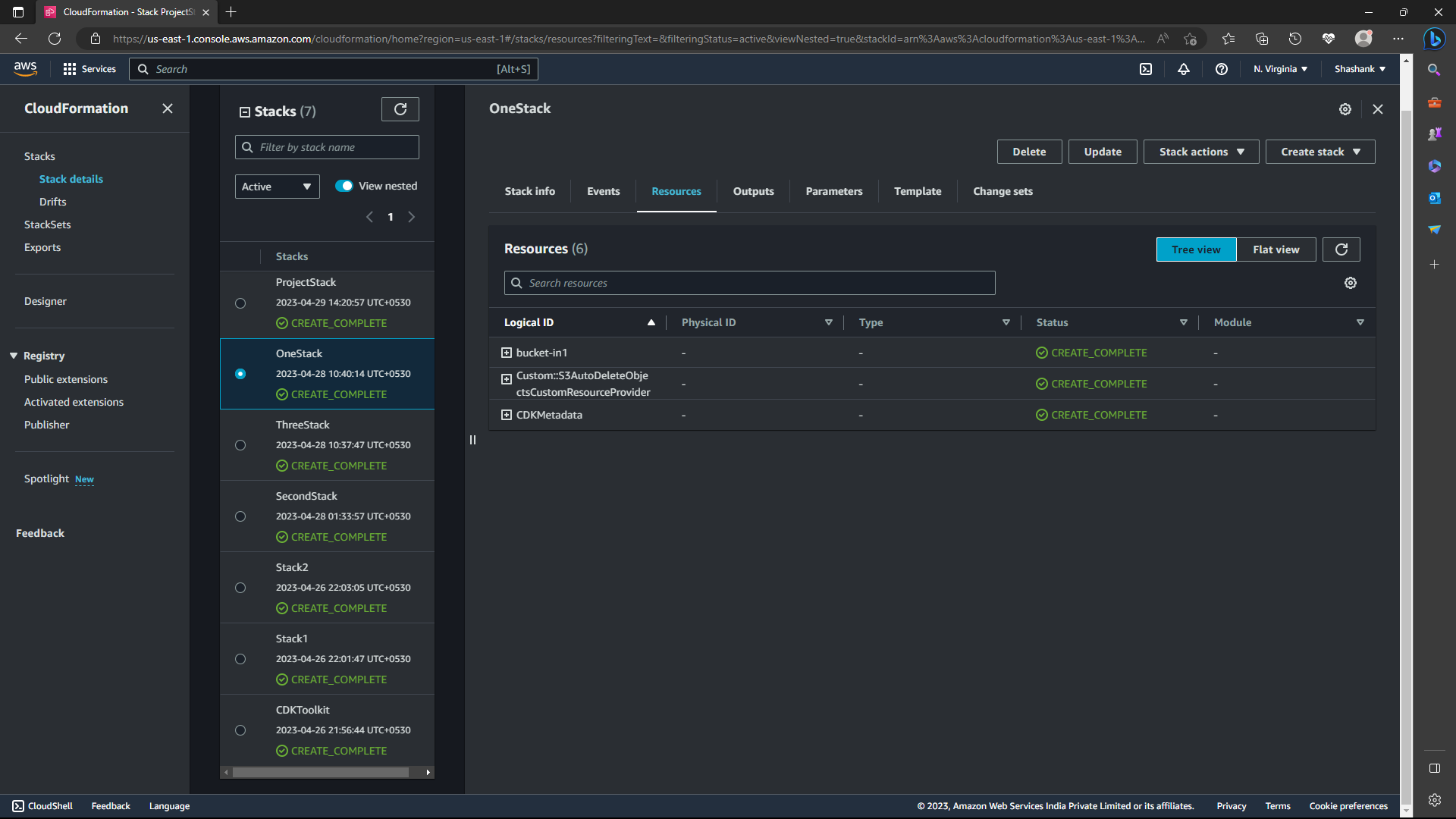
6. Now there is need to create executable of these files in my case it's in JavaScript :> npm run build

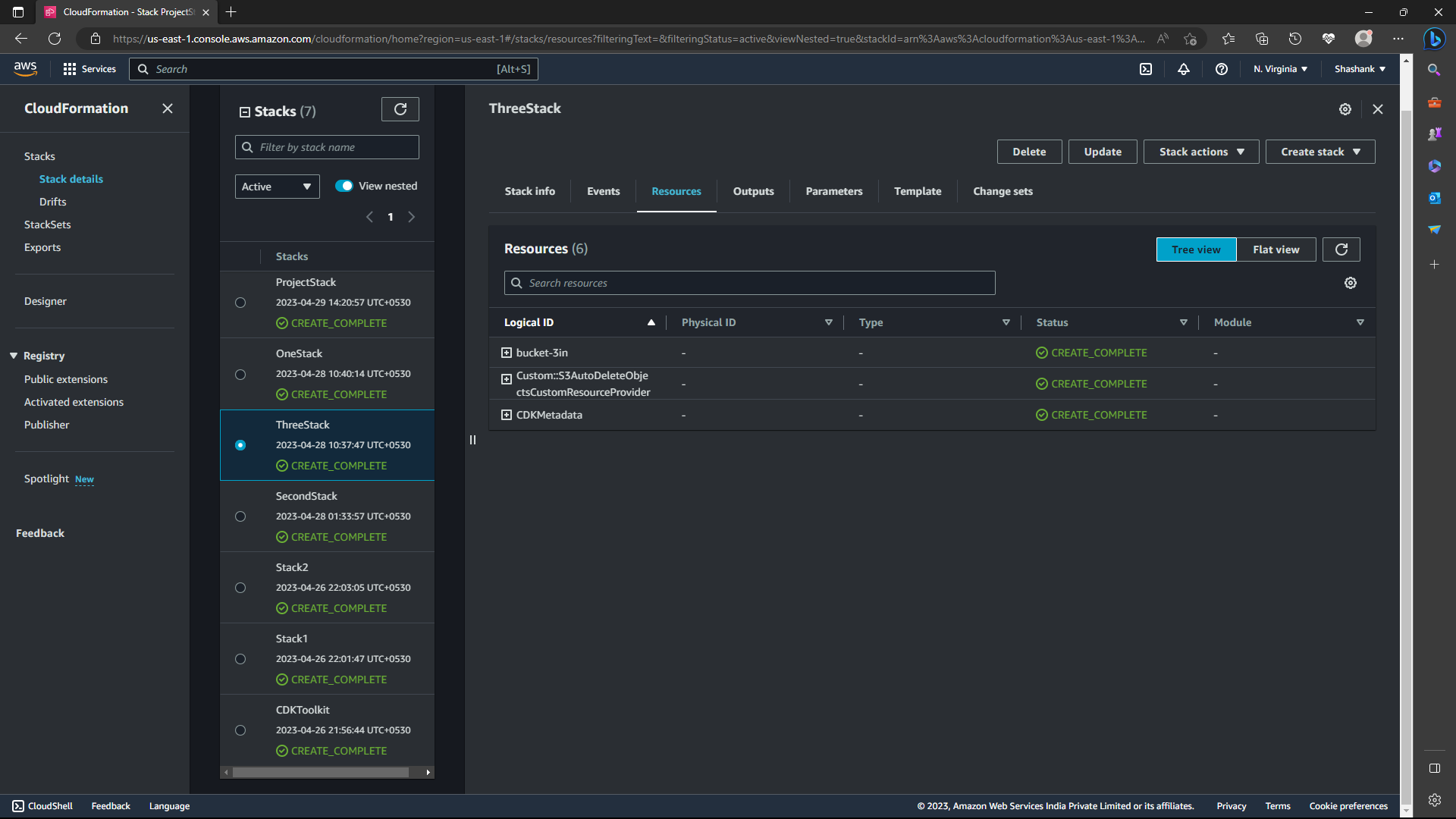
7. Next we will **synthesise** to create cloud formation template : cdk synth

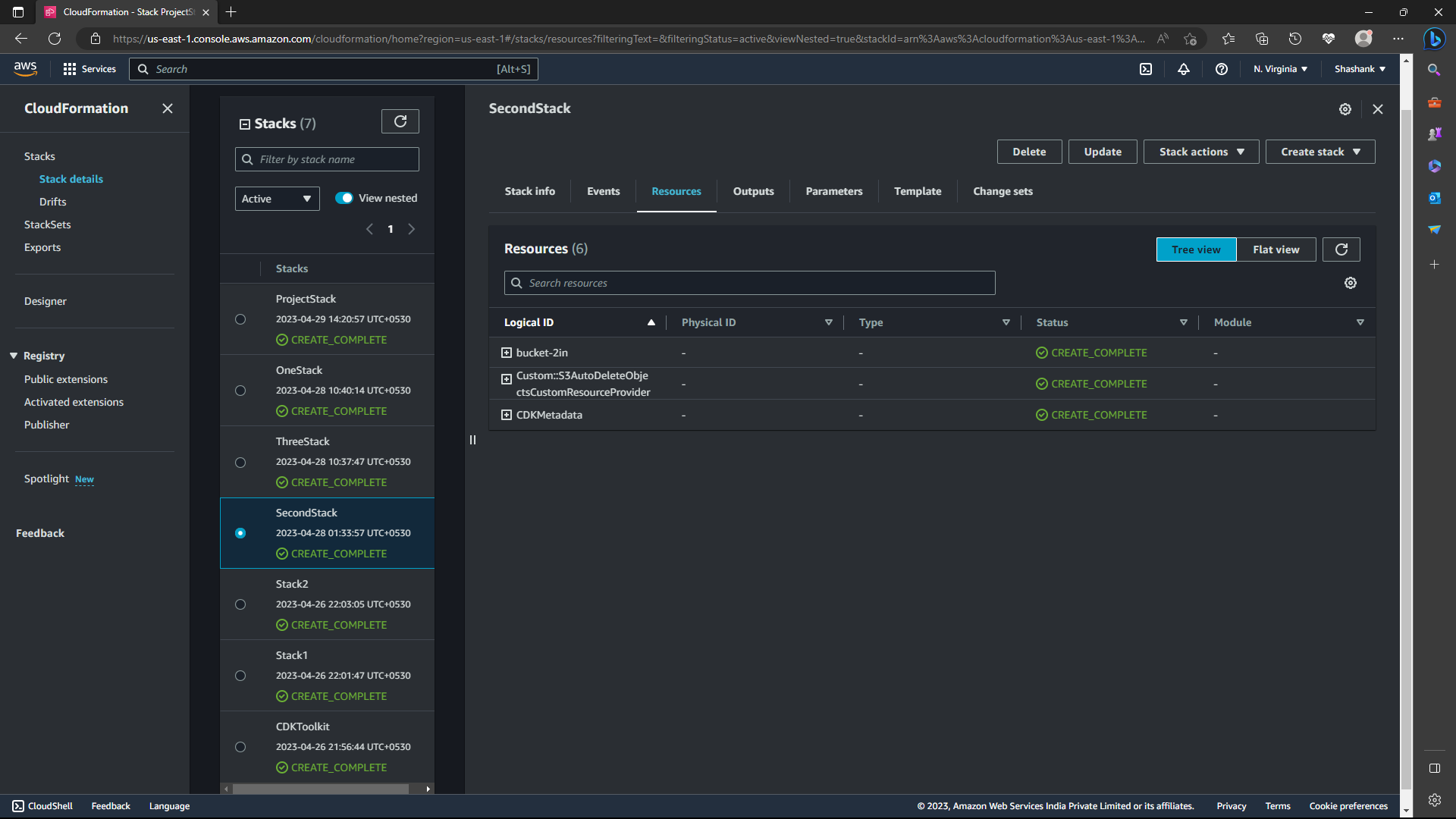
8. Then we need to bootstrap:cdk bootstrap

9. finally we have to deploy it :>cdk deploy –all (because here we have created two stack)

Following result :







**Create Resources (S3, Dynamodb, Lambda,VPC) in a project**

The whole procedure will be the same as we created a stack using cdk. As prerequisites all set up completely and then:

1. Create a new directory for CDK project:> mkdir cdk-project
2. Change the directory to newly created directory: >cd cdk-project
3. Initialization of CDK project: > cdk init --language typescript
4. Adding some line of code resources on lib folder project-stack.ts :

*import* { Construct } *from* 'constructs';// importing resources required template from node module folder

*import* \* *as* cdk *from* 'aws-cdk-lib/core'; // importing resources required template from node module folder

*import* \* *as* lambda *from* 'aws-cdk-lib/aws-lambda'; // importing resources required template from node module folder

*import* \* *as* s3 *from* 'aws-cdk-lib/aws-s3'; // importing resources required template from node module folder

*import* \* *as* dynamodb *from* 'aws-cdk-lib/aws-dynamodb'; // importing resources required template from node module folder

*import* \* *as* ec2 *from* 'aws-cdk-lib/aws-ec2'; // importing resources required template from node module folder

*import* \* *as* rds *from* 'aws-cdk-lib/aws-rds'; // importing resources required template from node module folder

*export* *class* ProjectStack *extends* cdk.Stack {

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

*super*(scope, id, props);

*// Create a VPC with two public subnets and two private subnets*

*const* vpc *=* *new* ec2.Vpc(*this*, 'VPC-Vol1', {

maxAzs: 2,

subnetConfiguration: [

{

name: 'public',

subnetType: ec2.SubnetType.PUBLIC,

},

{

name: 'private',

subnetType: ec2.SubnetType.PRIVATE\_ISOLATED,

},

],

});

*// Create a Lambda function*

*const* lambdaFunction *=* *new* lambda.Function(*this*, 'LambdaSetOne', {

runtime: lambda.Runtime.NODEJS\_14\_X,

code: lambda.Code.fromAsset('lambda'),//set path lambda program

handler: 'hello.handler',

vpc: vpc,

vpcSubnets: {

subnetGroupName: 'private',

},

environment: {

TABLE\_NAME: 'my-dynamodb-tab-One',

},

});

*// Create an S3 bucket*

*new* s3.Bucket(*this*, 'bucket-Set1', {

versioned: true, *//version control of an object in bucket*

bucketName: 'bucket-1.10', *//provide name to my bucket*

blockPublicAccess: s3.BlockPublicAccess.BLOCK\_ALL, *//build to block public access*

});

*// Create a DynamoDB table*

*const* dynamoDbTable *=* *new* dynamodb.Table(*this*, 'DynamoDbTabOne', {

partitionKey: { name: 'id', type: dynamodb.AttributeType.STRING },

billingMode: dynamodb.BillingMode.PAY\_PER\_REQUEST,

encryption: dynamodb.TableEncryption.AWS\_MANAGED,

});

*// Create an RDS database instance*

*const* dbInstance *=* *new* rds.DatabaseInstance(*this*, 'DbInstaOne', {

engine: rds.DatabaseInstanceEngine.mysql({ version: rds.MysqlEngineVersion.VER\_8\_0\_23 }),

instanceType: ec2.InstanceType.of(ec2.InstanceClass.BURSTABLE2, ec2.InstanceSize.SMALL),

vpc: vpc,

vpcSubnets: {

subnetGroupName: 'private',

},

allocatedStorage: 20,

storageEncrypted: true,

1. In case of lambda we need to crate a directory name lambda in root of project directory (in my case) and the we need to create a file inside hello.js:

exports.handler *=* *async* *function*(*event*)

{

console.log("request:", JSON.stringify(event, undefined, 2));

*return*{

statusCode: 200,

headers: {"Content-Type": "text/plain"},

body: 'Hello, CDK! You`ve hit ${event.path}\n'

};

};

This is a simple Lambda function which returns the text “Hello, CDK! You’ve hit [url path]”. The function’s output also includes the HTTP status code and HTTP headers. These are used by API Gateway to formulate the HTTP response to the user.

6. Now need to add two files (that will import by project.ts file) to create that stack object in Lib folder as every app has their own stack namely:

One-stack.ts : have some bucket code(Do not you same bucket name)

Two-stack.ts:have also some bucket code

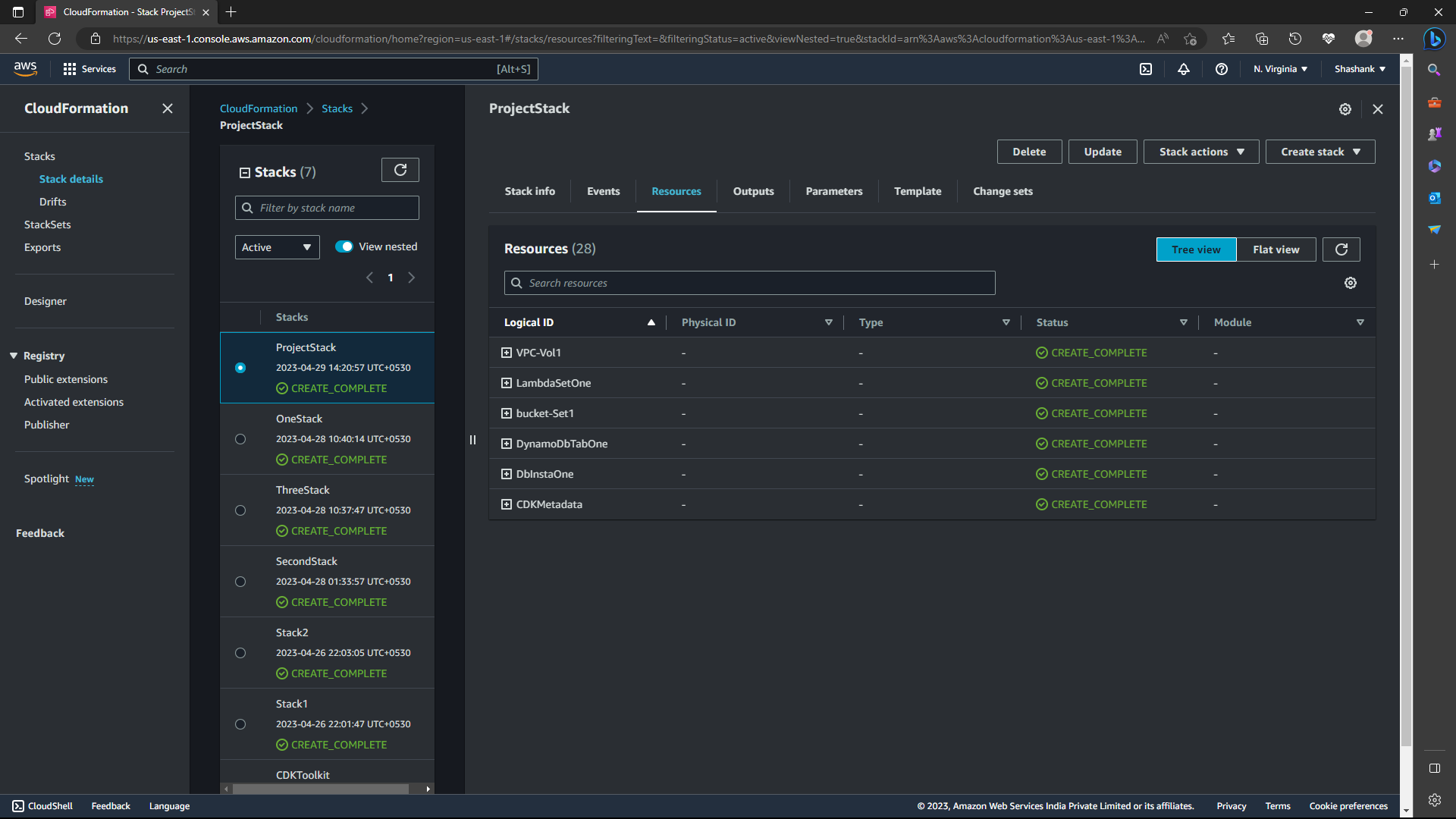
7. Now there is need to create executable of these files in my case it's in JavaScript :> npm run build

8. Next we will **synthesise** to create cloud formation template : cdk synth

9. Then we need to bootstrap:cdk bootstrap

10. finally we have to deploy it :>cdk deploy –all (because here we have created two stack)

Following result :



});

}

}