aws Cloud Development Kit

(AWS CDK v2)

Context.

- **GOAL:** Reliably and **consistently** provisioning and configuring infrastructure is foundational for DevOps and fast software delivery.
  - Multiple environments – Development, Test, Production
  - Multiple regions
- **PROBLEM:** Manual processes to create infrastructure can lack
  - consistency,
  - a single source of truth,
  - and reliable detection/remediation of provisioning errors**.**
- **SOLUTION:** Infrastructure as code (IaC)

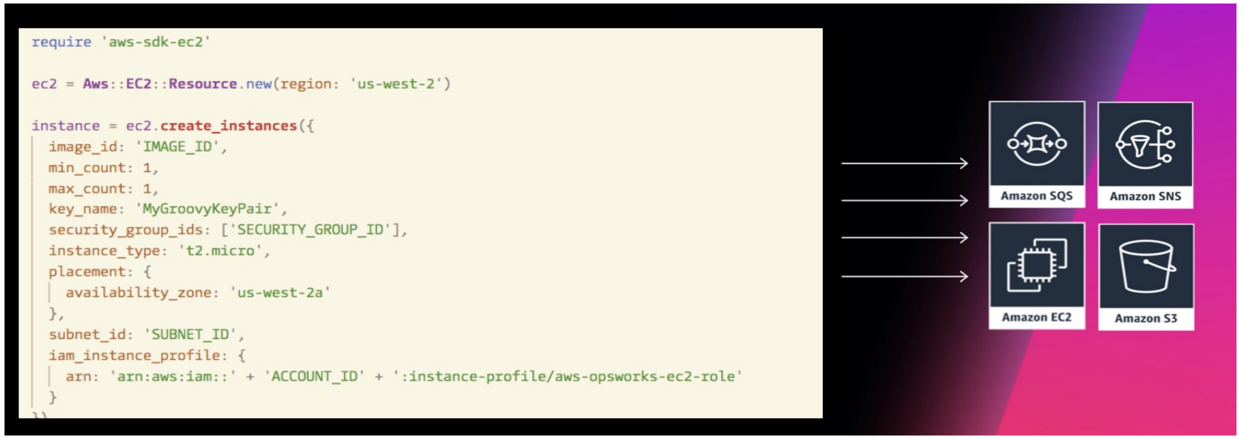
Infrastructure as code

- **Infrastructure as code allows organizations to automate and manage (cloud) resources consistently.**
  - **Resources – S3 bucket, EC2 instance, SQS, VPC, etc**

- **IaC allows us to:**
  1. **Use Version Controlled repositories as the single source of truth.**
  2. **Roll back changes to a previous version as needed.**
  3. **Shae and enforce best practices more consistently.**

The IaC journey.

- **Scripted.**

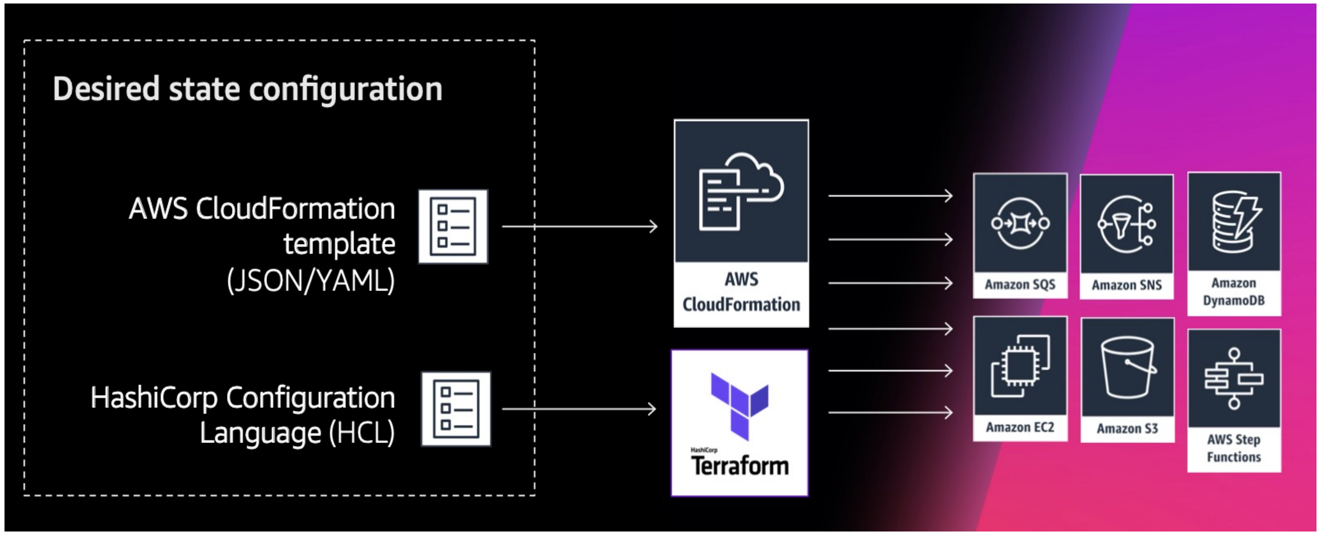


```ruby
require 'aws-sdk-ec2'

ec2 = Aws::EC2::Resource.new(region: 'us-west-2')

instance = ec2.create_instances({
  image_id: 'IMAGE_ID',
  min_count: 1,
  max_count: 1,
  key_name: 'MyGroovyKeyPair',
  security_group_ids: ['SECURITY_GROUP_ID'],
  instance_type: 't2.micro',
  placement: {
    availability_zone: 'us-west-2a'
  },
  subnet_id: 'SUBNET_ID',
  iam_instance_profile: {
    arn: 'arn:aws:iam::' + 'ACCOUNT_ID' + ':instance-profile/aws-opsworks-ec2-role'
  }
```

- **Problems:**
  1. **What happens if an API call fails?**
  2. **How do I make updates to the infrastructure?**
  3. **How do I know when a resource is ready?**
  4. **How do I roll back the infrastructure?**

The IaC journey.

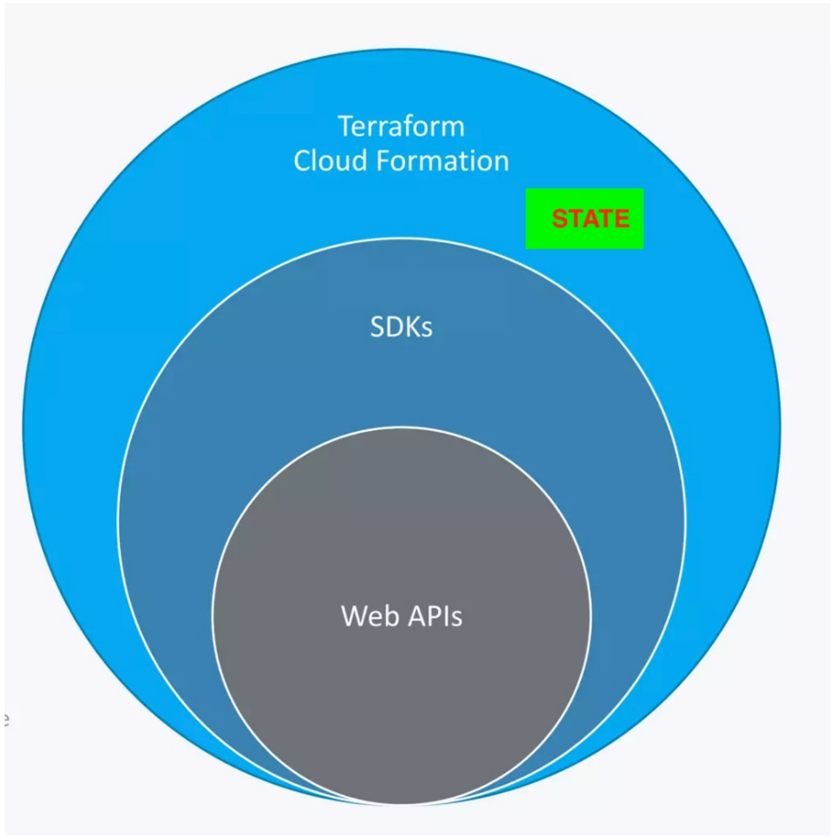
- **Resource Provisioning Engines.**



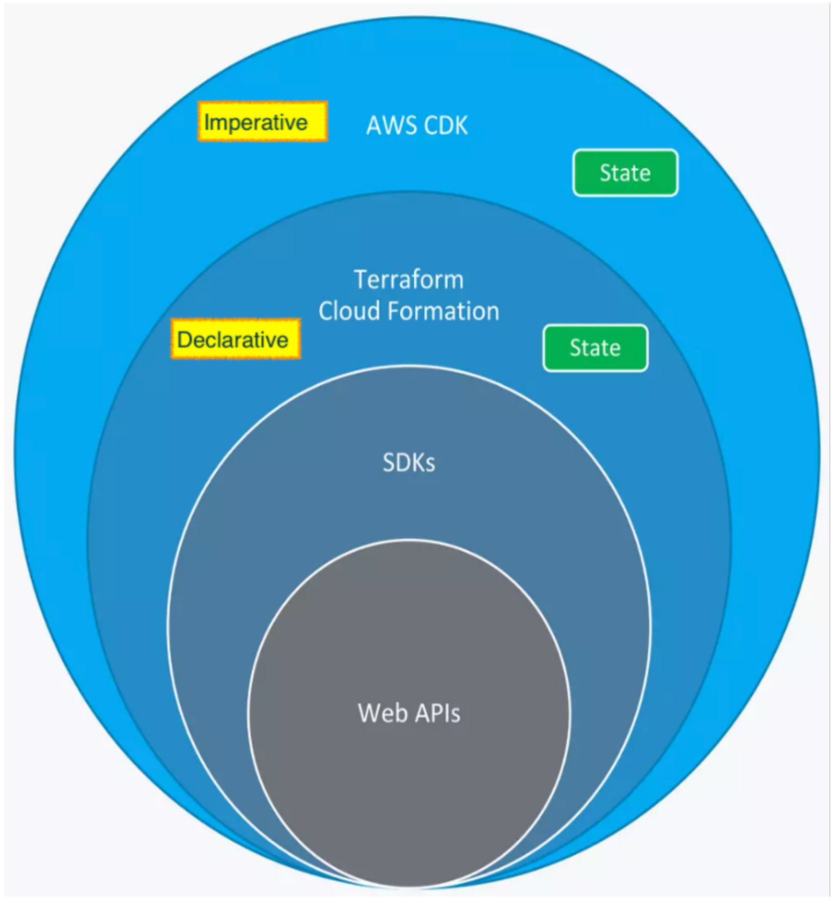
- **Advantages:**
  - **Easy to update the infrastructure.**
  - **Reproducible.**
- **Disadvantages**
  - **Configuration syntax.**
  - **No abstractions, therefore lots of details (no sensible defaults)**

The IaC journey.

- **Web APIs - AWS has exposed majority of their cloud services publicaly using <u>REST APIs</u>**
- **SDKs - Available in all the major programming languages.**
- **CloudFormation (2011) – next level abstraction of SDKs.**
  - **Provides a set of tools to define infrastructure <u>declaratively</u>.(YAML/JSON)**
  - **Manages updates to infrastructure <u>state</u>**
- **HCL TerraForm (2014) – Open source.**



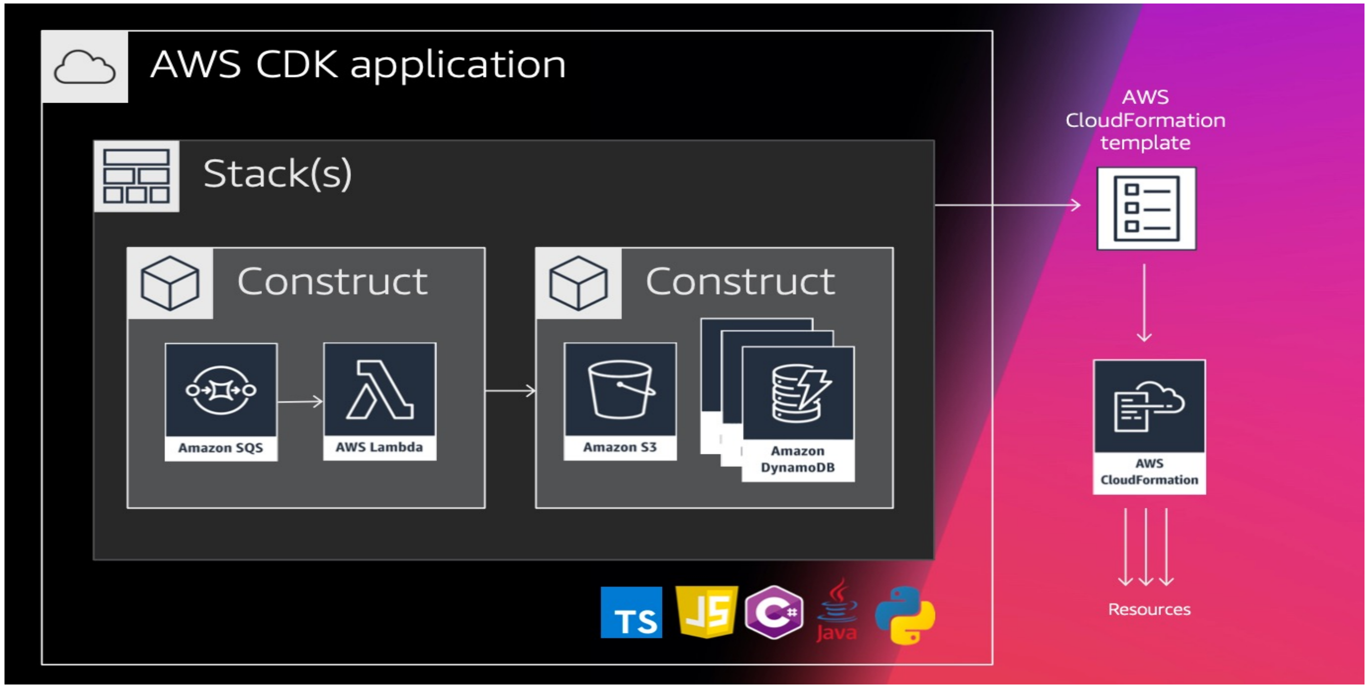
The CDK framework



- **August 2019 – proof of concept**
- **Goal - Describe infrastructure in an imperative language.**
  - **Supports Typescript, JS, Python, C#, Go, and growing.**
- **Class libraries of constructs with sensible defaults.**
  - **Abstractions-heavy.**
- **Improved Developer experience (DX).**
  - **IDE hinting/intelllisense.**
  - **LOC : CF >> CDK**
  - **Unit testing.**

CDK concepts

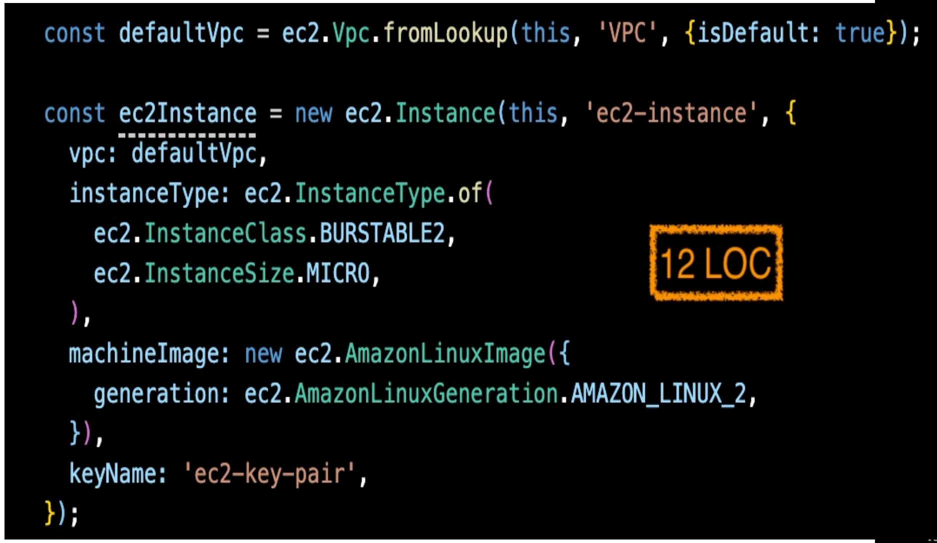
- **Application (App) >> Stack >> Construct >> Resources**



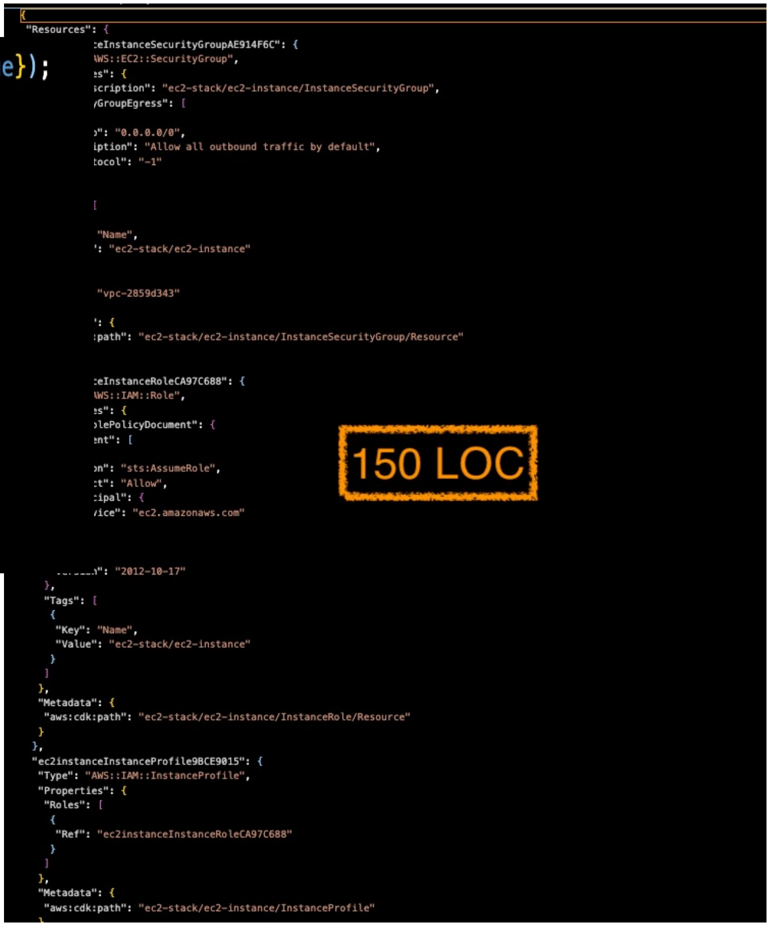
- **A stack is the unit of deployment, according to CloudFormation**

Developer Productivity (LOC)

- **Obj: Provision an EC2 instance with the default security policy, and located in the default VPC.**

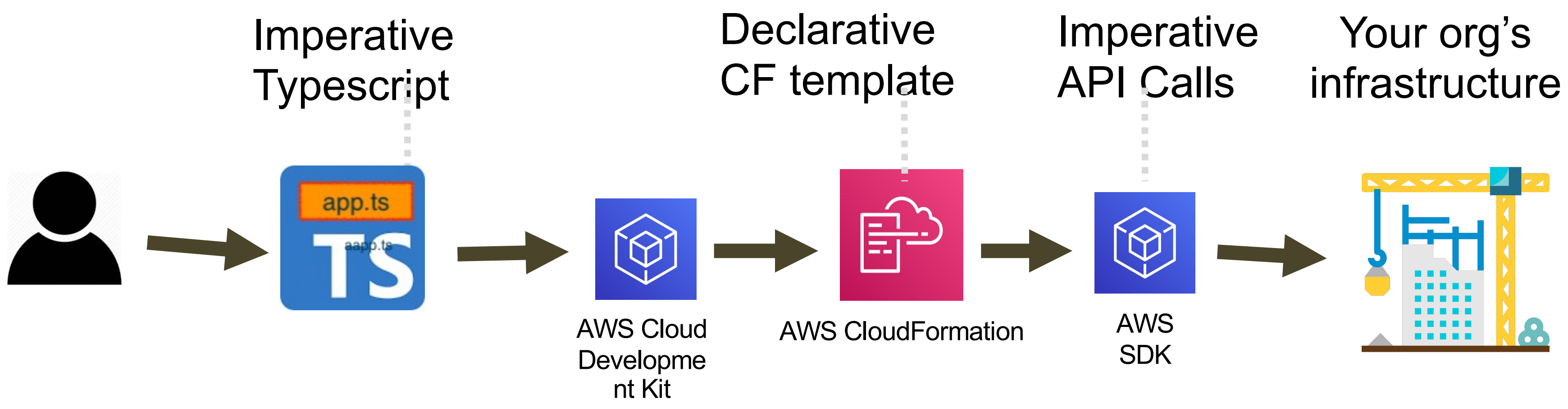


```
const defaultVpc = ec2.Vpc.fromLookup(this, 'VPC', {isDefault: true});

const ec2Instance = new ec2.Instance(this, 'ec2-instance', {
  vpc: defaultVpc,
  instanceType: ec2.InstanceType.of(
    ec2.InstanceClass.BURSTABLE2,
    ec2.InstanceSize.MICRO,
  ),
  machineImage: new ec2.AmazonLinuxImage({
    generation: ec2.AmazonLinuxGeneration.AMAZON_LINUX_2,
  }),
  keyName: 'ec2-key-pair',
});
```



```
{
 "Resources": {
           :eInstanceSecurityGroupAE914F6C": {
           WS::EC2::SecurityGroup",
           :s": {
           scription": "ec2-stack/ec2-instance/InstanceSecurityGroup",
           /GroupEgress": [

           ›": "0.0.0.0/0",
           iption": "Allow all outbound traffic by default",
           tocol": "-1"


           [


            "Name",
           ': "ec2-stack/ec2-instance"


            "vpc-2859d343"

           ': {
           :path": "ec2-stack/ec2-instance/InstanceSecurityGroup/Resource"


           :eInstanceRoleCA97C688": {
           WS::IAM::Role",
           :s": {
           »lePolicyDocument": {
           :nt": [

           on": "sts:AssumeRole",
           :t": "Allow",
           :ipal": {
           /ice": "ec2.amazonaws.com"



           n": "2012-10-17"
    },
    "Tags": [
     {
      "Key": "Name",
      "Value": "ec2-stack/ec2-instance"
     }
    ]
   },
   "Metadata": {
    "aws:cdk:path": "ec2-stack/ec2-instance/InstanceRole/Resource"
   }
  },
  "ec2instanceInstanceProfile9BCE9015": {
   "Type": "AWS::IAM::InstanceProfile",
   "Properties": {
    "Roles": [
     {
      "Ref": "ec2instanceInstanceRoleCA97C688"
     }
    ]
   },
   "Metadata": {
    "aws:cdk:path": "ec2-stack/ec2-instance/InstanceProfile"
```

CDK execution.



CDK workflow.

- **Workflow:**

  $ cdk init app --language typescript | python | go | java.

  # Scaffolding

  . . . . Write infrastructure code . . . . . .

  $ cdk synth # (Optional) Generate local copy of CF template

  $ cdk deploy # Deploy app stack(s)

  . . . . . Change infrastructure code . . . . . .

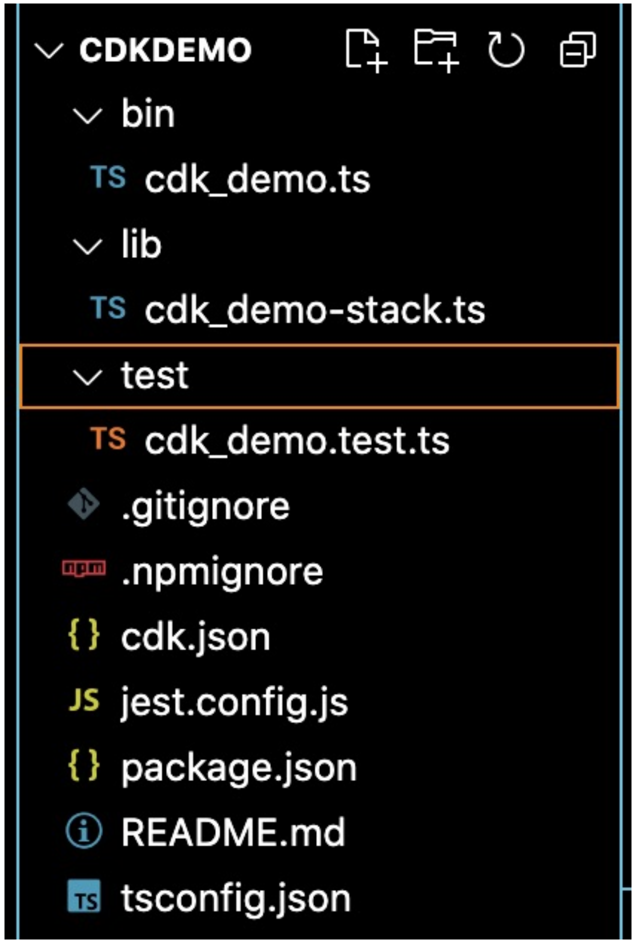
  $ cdk deploy. # Send updated template to CF to trigger state change

  . . . .

  $ cdk destroy # Request CF to destroy all stack resources

CDK app project structure

- *./bin/cdk_demo.ts*
    - **Entry point file used by the CDK framework.**
    - **Where you define your app's stack configuration..**
- *./lib* **folder**
    - **Contains the IaC that provisions the resources.**
    - **Required by** ./bin/cdk_demo.ts **during** synth **and** deploy **actions.**
- *./test/cdk_demo.test.ts*
    - **Template test code for app.**



```
CDKDEMO
  bin
    cdk_demo.ts
  lib
    cdk_demo-stack.ts
  test
    cdk_demo.test.ts
  .gitignore
  .npmignore
  cdk.json
  jest.config.js
  package.json
  README.md
  tsconfig.json
```

Construct Levels

- ***L1 – CloudFormation resources.***
  - ***1:1 relationship with CF template resources. No default configuration settings. No abstractions.***

- ***L2 – AWS constructs.***
  - ***1:M relationship with CF resources. Lots of default settings. High level abstraction.***

- ***L3 – Purpose-built constructs.***
  - ***Pattern-based. Optimized for particular use case. Community and AWS supplied.***

DEMO