AWS Cloud Development Kit (CDK) V2

Context

- **GOAL**: Reliable and consistent provisioning and configuring of cloud infrastructure is foundational for DevOps and rapid software delivery.
  - Multiple environments – Development, Test, Production.
  - Multiple regions.

- **PROBLEM**: Manual processes to create infrastructure can lack:
  - consistency,
  - a single source of truth,
  - reliable detection/remediation of provisioning errors.

- **SOLUTION**: Automation → Infrastructure as Code (IaC)

Infrastructure As Code (IaC)

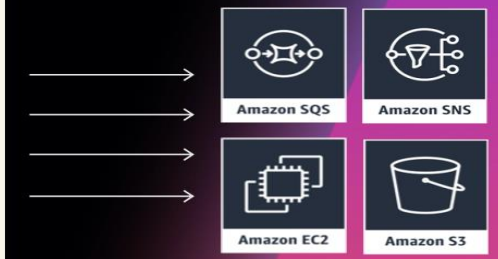
- Infrastructure as code allows organizations to automate and manage (cloud) infrastructure resources consistently.
  - Resources – S3 bucket, EC2 instance, SQS queue, VPC, etc.

- Advantages of IaC:
  - Provides a single source of truth.
  - Use Version Control tools to manage change.
  - Roll back changes.
  - Share and enforce best practices.

The IaC journey

- 1st generation: The Scripted approach.

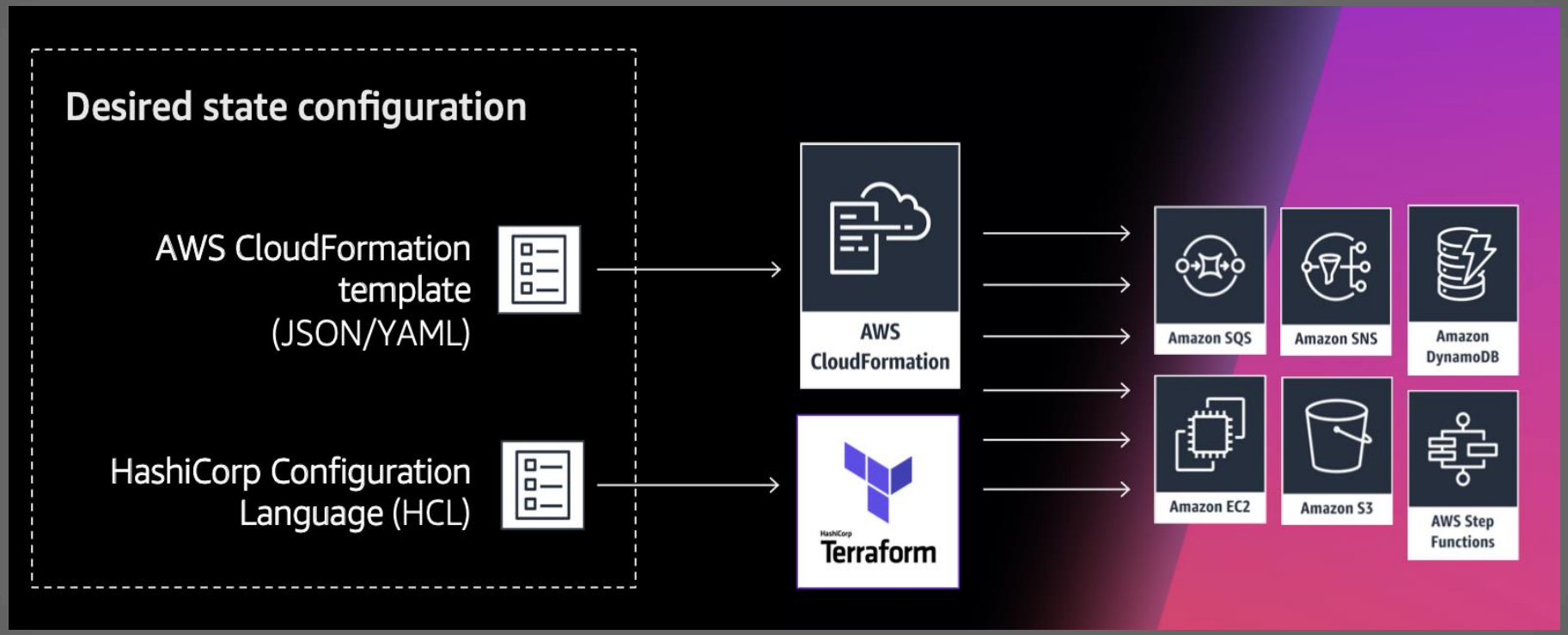
```ruby
require 'aws-sdk-ec2'

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:
  - What happens if an API call fails?
  - How do I make updates to the infrastructure?
  - How do I know when a resource is ready?
  - How do I roll back the infrastructure?

The IaC journey

- 2nd generation: Resource Provisioning Engines.

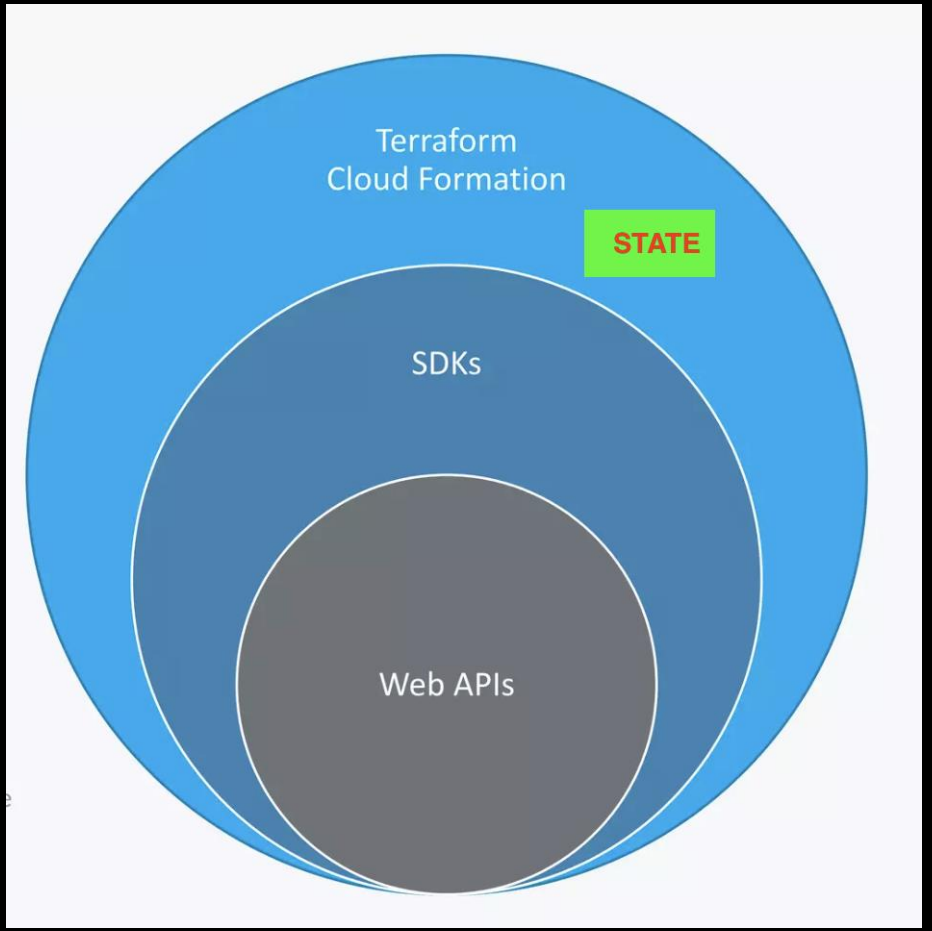


The IaC journey

- Resource Provisioning Engines.
- Advantages:
  - Easy to update the infrastructure.
  - Reproducible.
- Disadvantages
  - Configuration syntax not developer-friendly.
  - No abstractions or sensible default values.
  - Verbose.

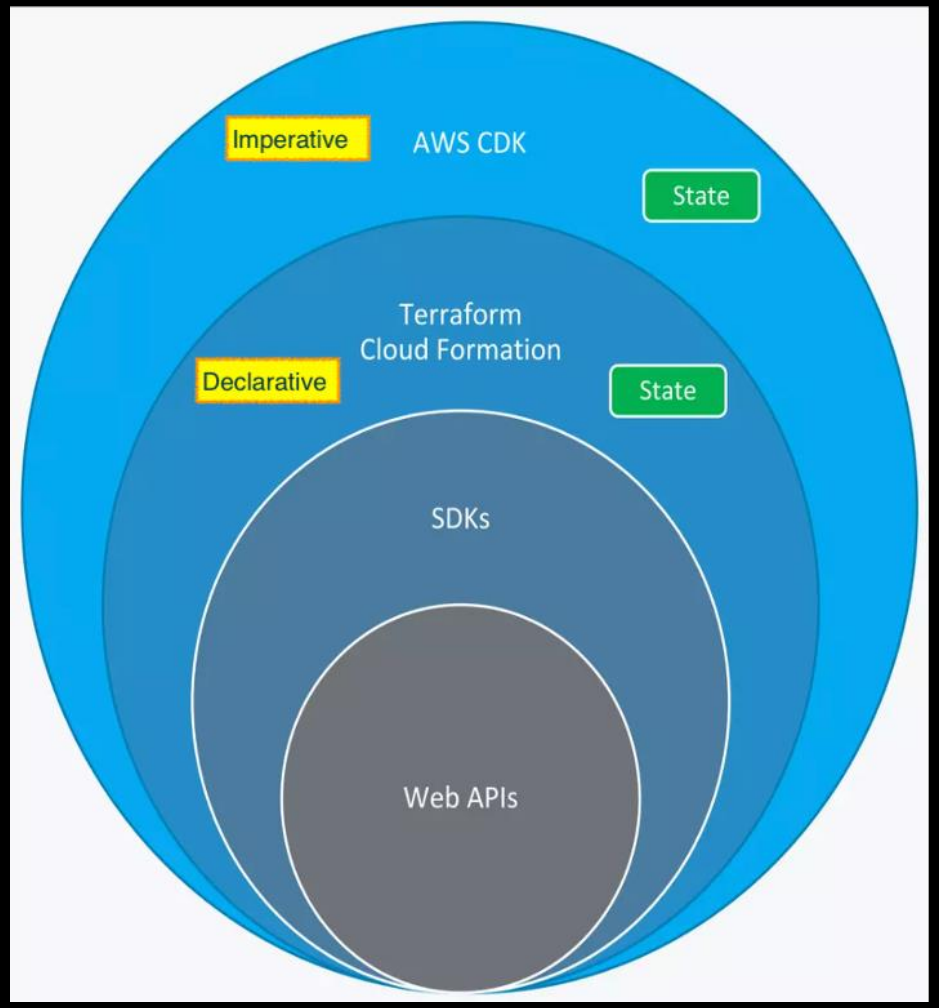
The IaC journey

1. Web APIs - AWS exposed majority of their services publicly using REST APIs.
2. SDKs – AWS provided SDKs in all the major programming languages.
3. CloudFormation (2011) – next level abstraction of SDKs.
   - Provides a set of tools to define infrastructure declaratively.(YAML/JSON)
   - Manages updates to infrastructure state.
4. HCL TerraForm (2014) – Open source.



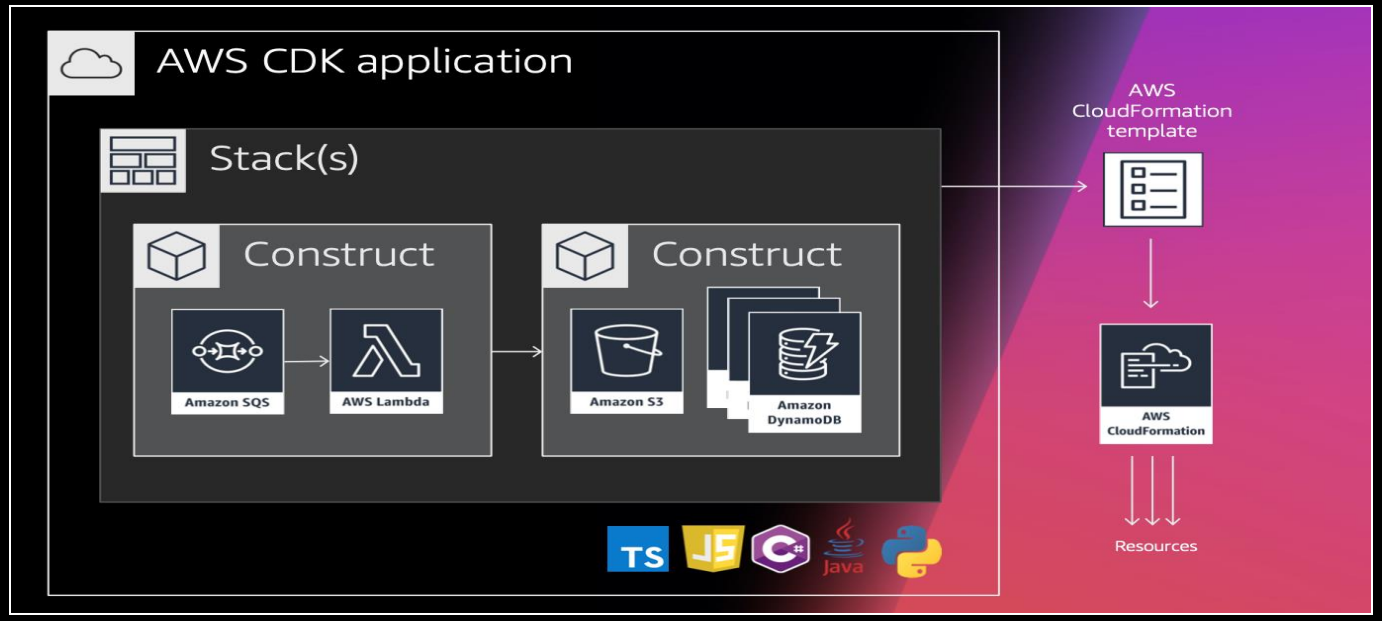
CDK framework – 3rd generation IaC

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



CDK coding concepts

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



- A stack is the unit of deployment, similar to CloudFormation.

Developer Productivity

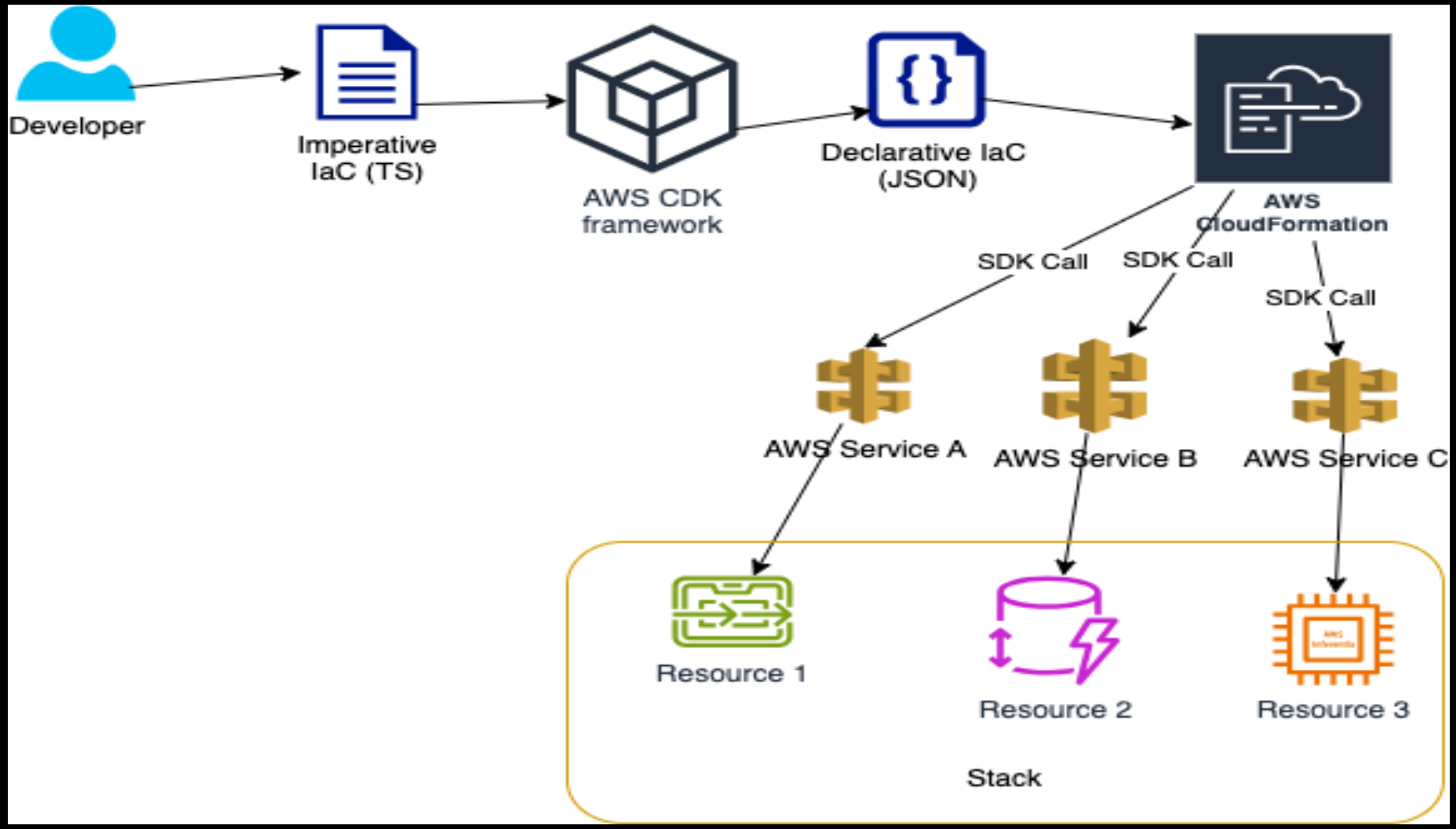
- Ex: Provision an EC2 instance with the default security policy and attached to the default VPC.

```
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',
});
```

12 LOC

150 LOC

CDK execution flow.



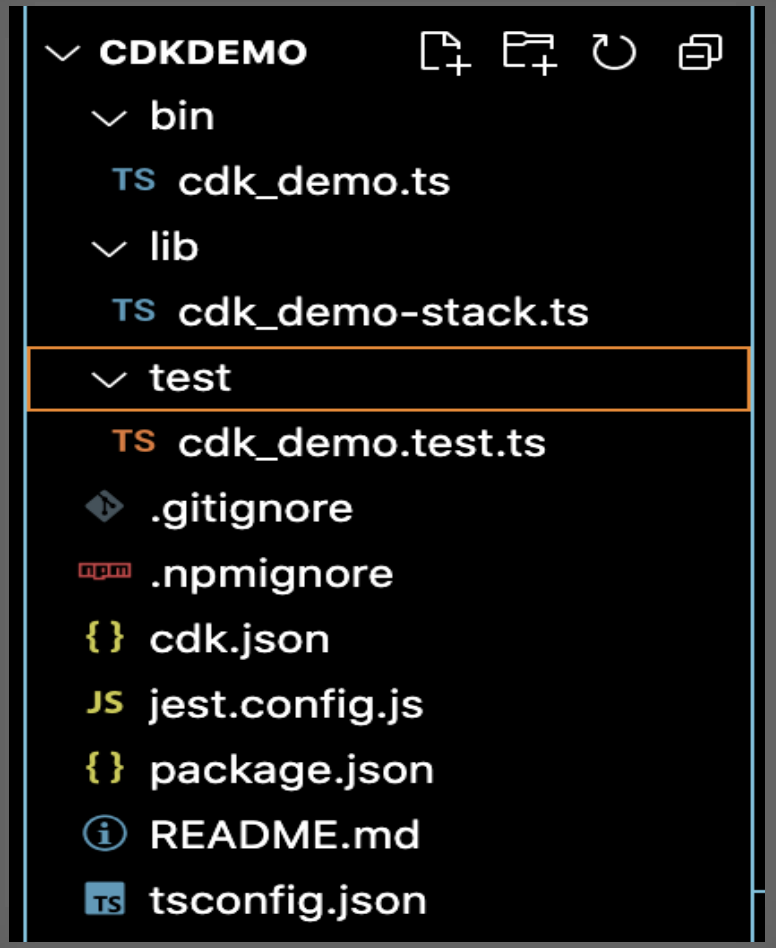
CDK workflow

- Workflow:

$ cdk init app --language typescript (python, go) # Scaffolding
. . . . Write infrastructure code . . . . . .
$ cdk synth   # (Optional) Generate local copy of CF template
$ cdk deploy   # Deploy app stack(s)
. . . . . Change infrastructure code . . . . . .
$ cdk deploy.   # Updates CF template and trigger stack update
. . . . . .
$ 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 composition.
- *./lib* folder
  - Contains the IaC that describes the infrastructure resources.
  - Used by bin file during deploy action.
- *./test/cdk_demo.test.ts*
  - Template test code for app.



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