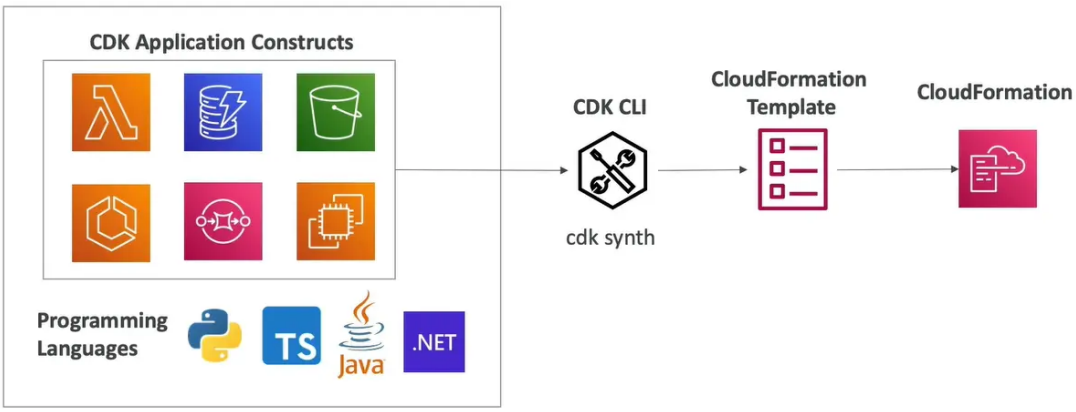
Cloud Development Kit (CDK)

AWS Cloud Development Kit (CDK)

- define your cloud infrastructure using a familiar language:
  - javascript/typescript,python,Java,and .NET
- contains high level components called constructs
- the code is ”compiled” into a CloudFormation template (JSON / YAML)
- you can therefore deploy infrastructure and application runtime code together
  - great for Lambda functions
  - great for Docker containers in ECS / EKS

```
export class MyEcsConstructStack extends core.Stack {
  constructor(scope: core.App, id: string, props?: core.StackProps)
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, "MyVpc", {
      maxAzs: 3 // Default is all AZs in region
    });

    const cluster = new ecs.Cluster(this, "MyCluster", {
      vpc: vpc
    });

    // Create a load-balanced Fargate service and make it public
    new ecs_patterns.ApplicationLoadBalancedFargateService(this, "My
      cluster: cluster, // Required
      cpu: 512, // Default is 256
      desiredCount: 6, // Default is 1
      taskImageOptions: { image: ecs.ContainerImage.fromRegistry("am
      memoryLimitMiB: 2048, // Default is 512
      publicLoadBalancer: true // Default is false
    });
  }
}
```

CDK in a diagram



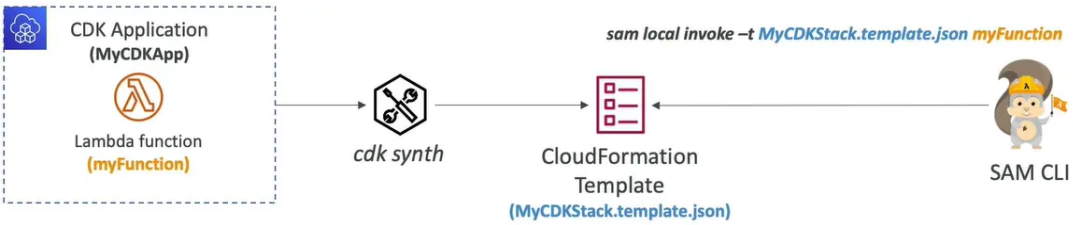
CDK vs SAM

- SAM:
  - serverless focused

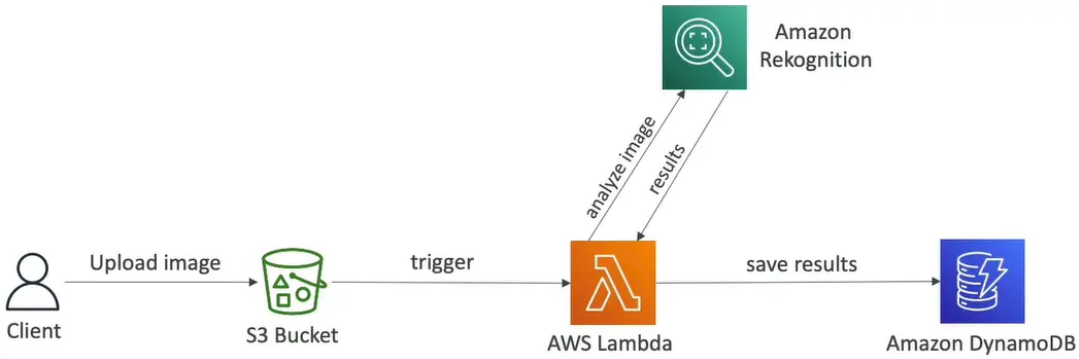
    - wirte your template declaratively [声明式] in JSON or YAML
    - great for quickly getting started with Lambda
    - leverage CloudFormation
- CDK
    - all AWS services
    - write infra in a programming language JavaScript / TypeScript,python,Java and .NET
    - leverage Cloudformation

CDK + SAM

- you can use SAM CLI to locally test your CDK apps
- you must first run cdk synth



CDK Hands–on



CDK Constructs

- CDK Construct is component that encapsulates [封装] everything CDK needs to create the final CloudFormation stack
- can represent [表示] a single AWS resource (eg, S3 bucket) or multiple related resources (eg, worker queue with compute)
- AWS Construct Library
    - a collection of Contructs included in AWS SDK which contains Contructus for every AWS resource
    - contains 3 different levels of Constructs available (L1,L2,L3)
- Construct Hub – contains additional constructs from AWS, 3rd parties, and open–source CDK community

CDK Constructs – Layer 1 Constructs (L1)

```
const bucket = new s3.CfnBucket(this, "MyBucket", {
    bucketName: "MyBucket"
});
```

- can be called *CFN Resources* which represents all resources directly available in CloudFormation
- constructs are periodically generated from CloudFormation Resource Specification
- construct names start with Cfn (eg, CfnBucket)
- you must explicitly [明确的] configure all resource properties

CDK Constructs – Layer 2 Constructs (L2)

```
const s3 = require('aws-cdk-lib/aws-s3');

const bucket = new s3.Bucket(this, 'MyBucket', {
  versioned: true,
  encryption: s3.BucketEncryption.KMS
});

// Returns the HTTPS URL of an S3 Object
const objectUrl = bucket.urlForObject('MyBucket/MyObject');
```

- represents AWS resources but with a higher level (intent-based API)
- similar functionality as L1 but with convenient defaults and boilerplate [样板]
  - you don't need to know all the details about the resourece properties
- provide methods that make it simper to work with the source (eg, bucket.addLifeCycleRule())

CDK Constructs – Layer 3 Constructs (L3)

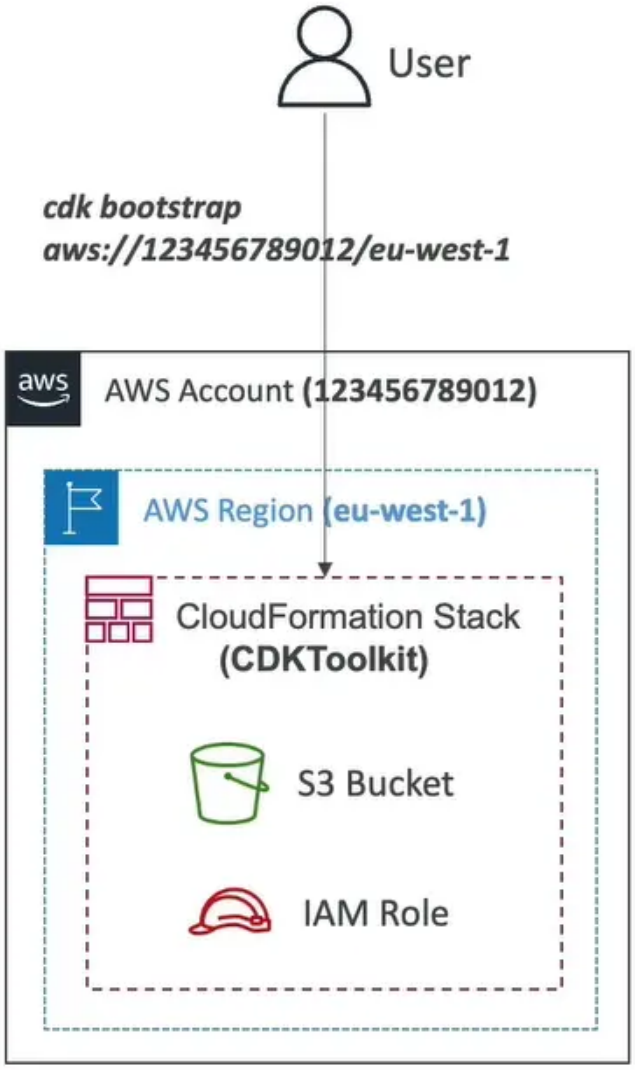
```
const api = new apigateway.LambdaRestApi(this, 'myapi', {
  handler: backend,
  proxy: false
});

const items = api.root.addResource('items');
items.addMethod('GET');  // GET /items
items.addMethod('POST'); // POST /items

const item = items.addResource('{item}');
item.addMethod('GET');   // GET /items/{item}

item.addMethod('DELETE', new apigateway.HttpIntegration('http://amazon.com'));
```

- can be called *Patterns*， which represents multiple related resources
- helps you complete common tasks in AWS
- examples
  - *aws-apigateway.LambdaRestApi* represents an API Gateway backed by a Lambda function
  - *aws-ecs-patterns.ApplicationLoadBalancerFargateService* which represents an architecture that includes a Fargate cluster with Application Load Balancer

CDK – Important Commands to know

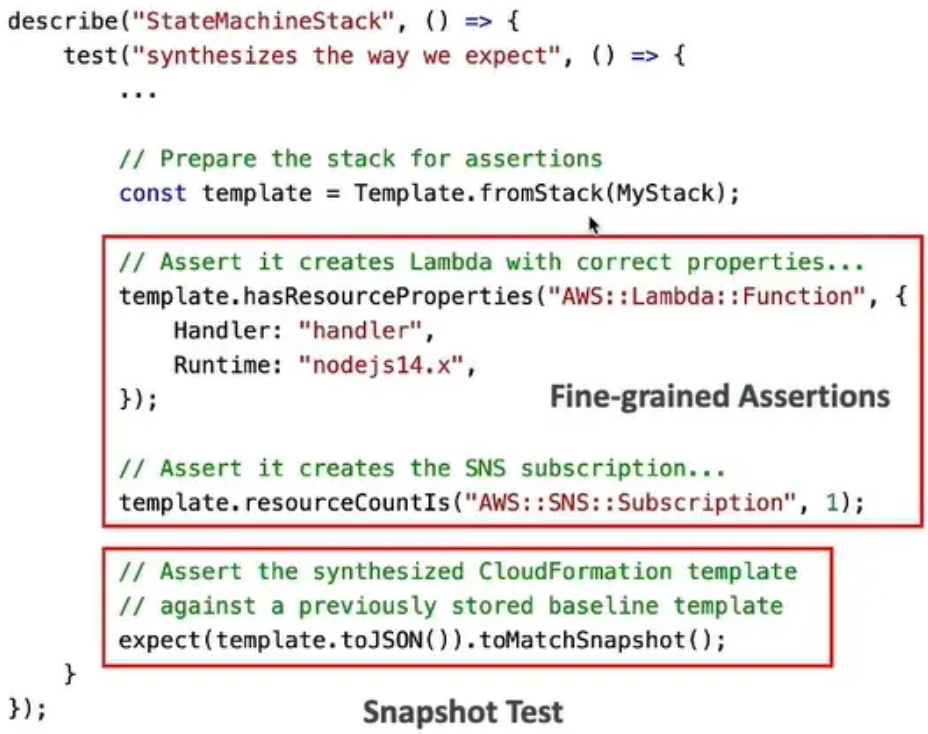
Command	Description
npm install -g aws-cdk-lib	Install the CDK CLI and libraries
cdk init app	Create a new CDK project from a specified template
cdk synth	Synthesizes and prints the CloudFormation template
cdk bootstrap	Deploys the CDK Toolkit staging Stack
cdk deploy	Deploy the Stack(s)
cdk diff	View differences of local CDK and deployed Stack
cdk destroy	Destroy the Stack(s)

CDK – Boostrapping [增强]



- the process of provisioning resources for CDK before you can deploy CDK apps into an AWS environment]
- AWS Environment = account & region
- CloudFormation stack called CDKToolkit is created and containes
  - S3 Bucket – to store files
  - IAM Roles – to grant [授予] permissions to perform deployments
- you must run the following command for each new environment
  - cdk boostrap aws://<aws_account>/<aws_region>
- otherwise, you will get an error "Policy contains a statement with one or more invalid principal"

CDK Testing



- to test CDK apps , use CDK Assertions [断言] Module combined with popular test framworks suck as Jest (JavaScript) or Pytest (Python)
- verify we have specific resources, rules conditions , parameters
- two types of tests
  - Fine-grained [细粒度] Assertions (common) – test specific aspects of the CloudFormation template (eg, check if a resource has this property with this value)
  - Snapshot Tests – test the synthesized CloudFormation template against a previously stored baseline template
- to import a template
  - Template.fromStack(MyStack): stack built in CDK
  - Tempate.fromString(mystring): stack build outside CDK