Goodbye YAML

Hello AWS Cloud Development Kit

Ken Winner
@KenWin0x539

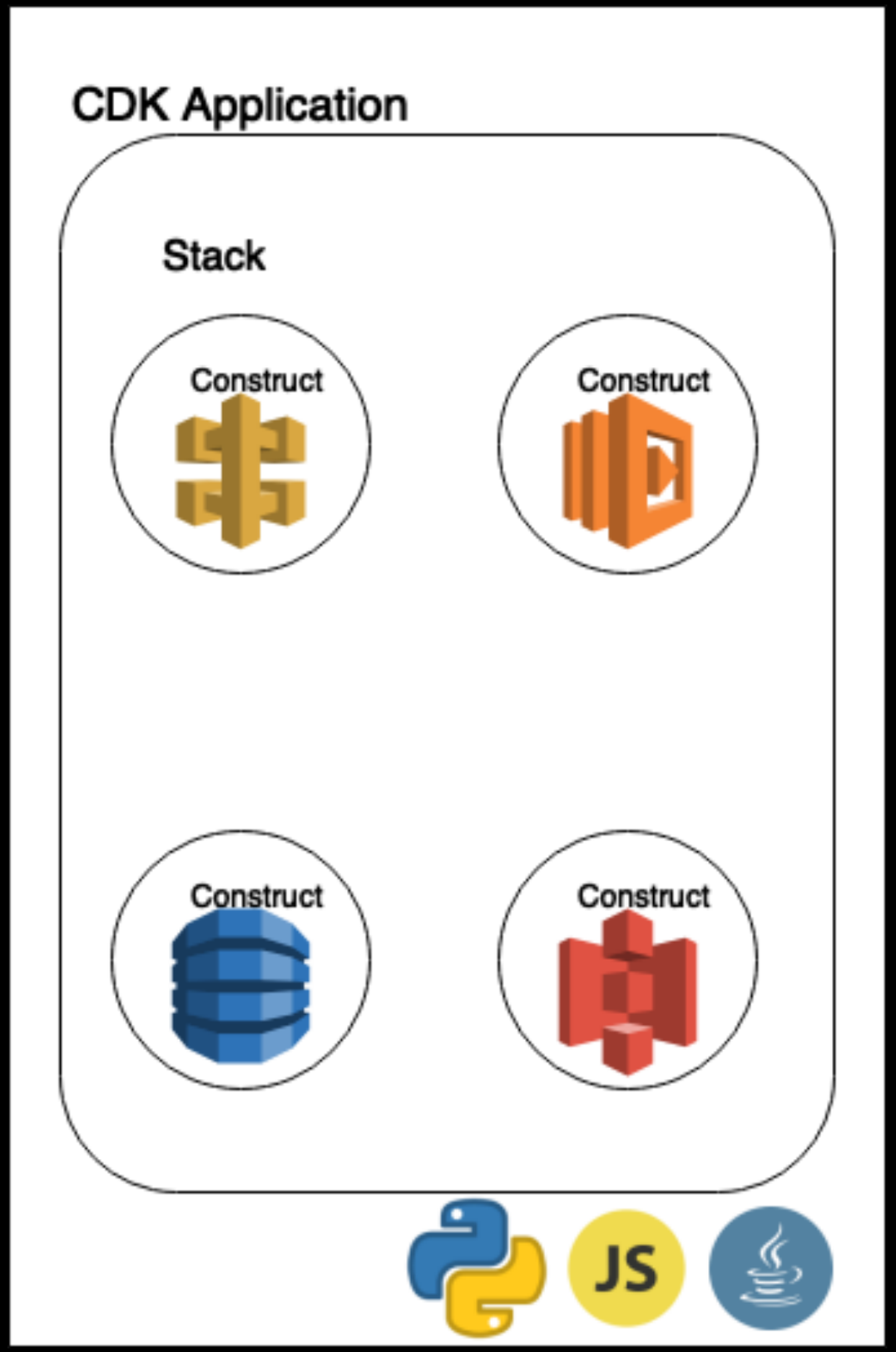
https://github.com/kcwinner/icc-demo/

Agenda

- What is the CDK?
- Features
- Demo
- Circle Back
- Questions

What is the AWS CDK?

- Create and provision AWS infrastructure
- Use a template file to create and delete a collection of resources
- Define your resources in a familiar programming language



Supported Languages

- JavaScript
- TypeScript
- Python
- C#/.NET
- Java

Context

- Used to retrieve information about your AWS account
- Used to pass in command line parameters

```
STAGE=${STAGE:-"dev"}
REGION=${REGION:-“us-east-1"}
ACCOUNT=`aws sts get-caller-identity --query 'Account' --output text`

TABLE_NAME="icc-global-${STAGE}"

cdk synth \
    -c ACCOUNT=${ACCOUNT} \
    -c STAGE=${STAGE} \
    -c REGION=${REGION} \
    -c TABLE_NAME=${TABLE_NAME}
```

SSM

```
const STAGE = this.node.tryGetContext('STAGE');
this.isProd = STAGE === 'prod';

// Retrieve a specific version of the secret (SecureString) parameter.
// 'version' is always required.
const secretValue = StringParameter.fromSecureStringParameterAttributes(this, 'secret', {
    parameterName: 'my-secret-ssm-value',
    version: 1
});

let cluster = new CfnDBCluster(this, "DatabaseCluster", {
    engine: "aurora",
    engineMode: "serverless",
    engineVersion: "5.6",
    databaseName: `mydb`,
    dbClusterParameterGroupName: 'default.aurora5.6',
    masterUsername: `myuser`,
    // Output: "{{resolve:ssm-secure:my-secret-ssm-value:1}}"
    masterUserPassword: secretValue.stringValue,
    dbSubnetGroupName: 'subnet-group-name',
    vpcSecurityGroupIds: ['vpcSecurityGroupIds'],
    storageEncrypted: true,
    backupRetentionPeriod: this.isProd ? 30 : 7, // Max of 35 days
    scalingConfiguration: {
        autoPause: STAGE === 'prod' ? false : true,
        secondsUntilAutoPause: 600, // 10 minutes
        minCapacity: this.isProd ? 2 : 1,
        maxCapacity: this.isProd ? 4 : 2
    },
```

Loops

```
const groups = [
    { name: 'Admins', precedence: 0, description: 'Group for Admins' },
    { name: 'Users', precedence: 10, description: 'Group for Users' },
    { name: 'Guests', precedence: 100, description: 'Group for Guests' }
]
```

```
function createGroups(stage) {
    groups.forEach(group => {
        new CfnUserPoolGroup(this, `demo-${group.name.toLowerCase()}-${stage}`, {
            userPoolId: this.userPool.userPoolId,
            groupName: group.name,
            precedence: group.precedence,
            description: group.description
        })
    })
}
```

Conditions

```
export class ICCDemo extends cdk.Stack {
    public readonly isProd: boolean;
    public readonly region: string;

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

        const STAGE = this.node.tryGetContext('STAGE');
        this.isProd = STAGE === 'prod';
        this.region = props?.env?.region || 'us-east-1';

        let ec2Instance = new CfnInstance(this, 'demo', {
            imageId: regionMap[this.region]
        });

        if (this.isProd) {
            new CfnVolumeAttachment(this, 'demo', {
                instanceId: ec2Instance.ref,
                device: '/dev/sdh',
                volumeId: 'ref'
            })
        }
    }
}
```

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Mappings:
  RegionMap:
    us-east-1:
      AMI: "ami-0ff8a91507f77f867"
      TestAz: "us-east-1a"
Parameters:
  Stage:
    Description: Environment stage
    Default: dev
    Type: String
    AllowedValues:
      - dev
      - test
      - prod
    ConstraintDescription: Must specify dev, test, or prod
Conditions:
  IsProd: !Equals [ !Ref EnvType, prod ]
Resources:
  EC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", AMI]
  MountPoint:
    Type: "AWS::EC2::VolumeAttachment"
    Condition: IsProd
    Properties:
      InstanceId:
        !Ref EC2Instance
      VolumeId:
        !Ref NewVolume
      Device: /dev/sdh
```

Constructs

- Basic building blocks of the CDK

```
let regionalApi = new LambdaRestApi(this, `icc-api-${STAGE}`, {
  handler: lambdaFunction,
  domainName: {
    certificate: dnsCert,
    domainName: DOMAIN_NAME,
    endpointType: EndpointType.REGIONAL
  }
})

let record = new ICCRegionalRecord(this, `icc-api-regional-record-${STAGE}`, {
  api: regionalApi,
  region: REGION || '',
  domainName: 'kennethwinner.com'
})
```

Cloudformation Constructs vs Native Constructs

```
let lambdaFunction = new Function(this, 'icc-api-function', {
  runtime: Runtime.NODEJS_12_X,
  code: Code.asset('lambda'),
  handler: 'api.handler',
  environment: {
    TABLE_NAME: globalTableStack.tableName
  }
});

let cfnFunction = new CfnFunction(this, 'cfn-example', {
  runtime: 'nodejs12.x',
  code: {
    s3Bucket: 'some-bucket',
    s3Key: 'code-asset-key'
  },
  handler: 'api.handler',
  role: 'my-awesome-role-arn'
})
```

“Persuasion is clearly a sort of demonstration, since we are most fully persuaded when we consider a thing to have been demonstrated.”

–Aristotle

Demo

- cdk setup
- active-active api with global tables

Things I Do Not Love

- Opinionated
- Often have to fall back on “Cfn” resources instead of native CDK constructs
- Unclear errors

Things I Love

- “Code” defined
- Re-usable constructs
- Readable
- Code completion
- Import existing resources through context
- Execute code that is NOT Cloudformation
    - Example - store secrets in SSM

Questions?

@KenWin0x539

https://github.com/kcwinner/icc-demo/