Cloud infrastructure with AWS CDK

June 12, 2025

SREday Cologne

Bohdan Pohorilets, @bpohoriletz

Agenda

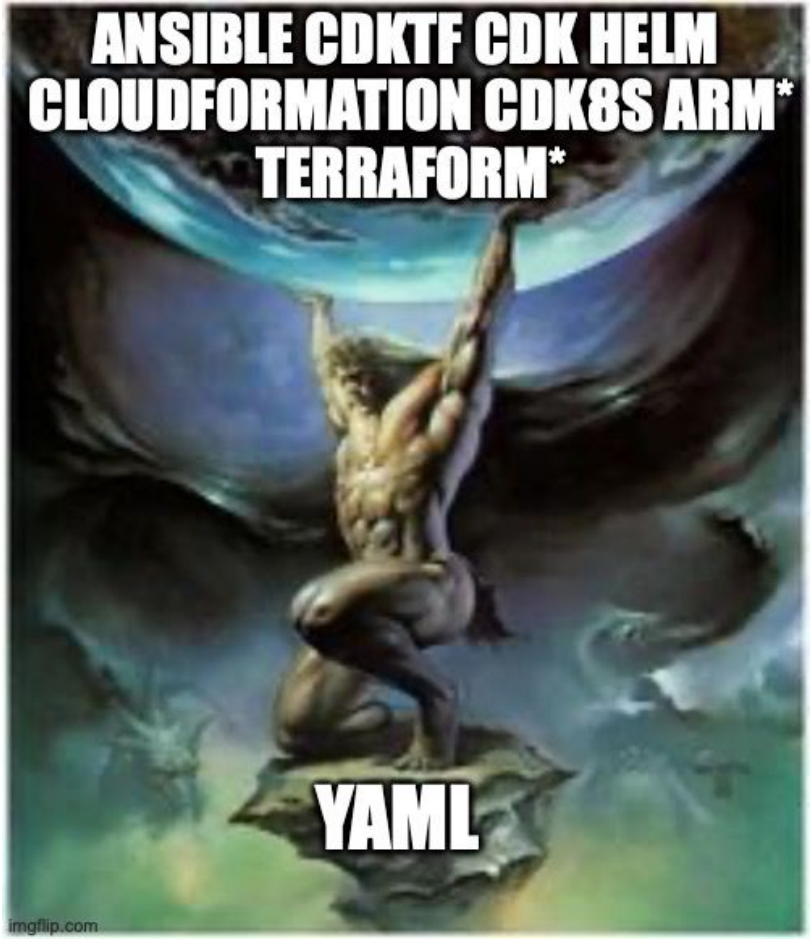
1. Introduction to CDK
2. Live deploy
3. Constructs
4. Use cases
5. Homework!

AWS Cloud Development Kit (AWS CDK)

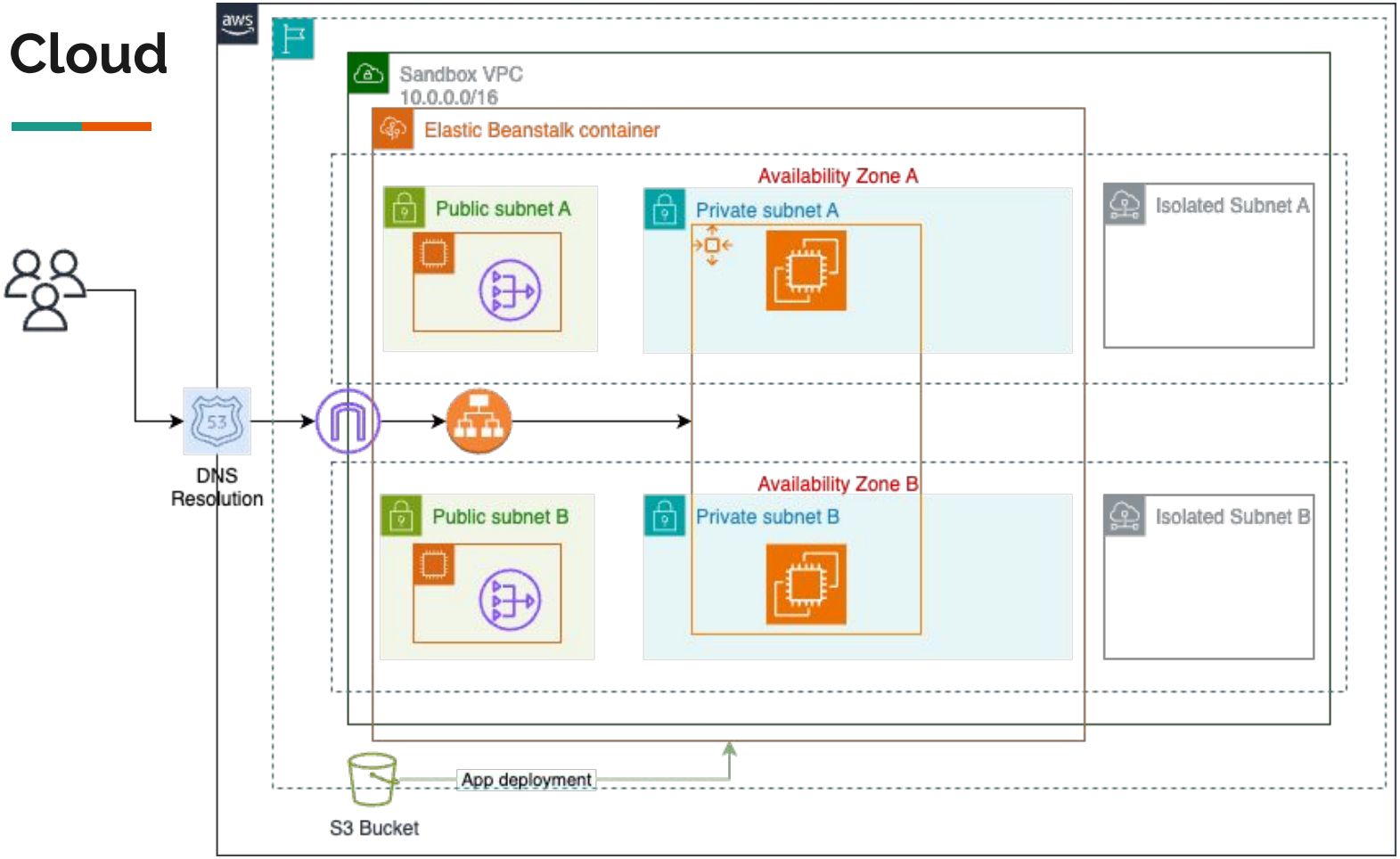
- The AWS Cloud Development Kit (AWS CDK) is an open-source software development framework developed by Amazon Web Services (AWS) for defining and provisioning cloud infrastructure resources using familiar programming languages.
- First release 2018
- AWS CDK supports TypeScript, JavaScript, Python, Java, Go, and C#

Feels Like

Is Like


ANSIBLE CDKTF CDK HELM
CLOUDFORMATION CDK8S ARM*
TERRAFORM*
YAML
imgflip.com

Cloud


aws
Sandbox VPC
10.0.0.0/16
Elastic Beanstalk container
Availability Zone A
Public subnet A
Private subnet A
Isolated Subnet A
DNS
Resolution
Availability Zone B
Public subnet B
Private subnet B
Isolated Subnet B
App deployment
S3 Bucket

Preparaton Step

- make check
- make install
- make bootstrap

Workflow

Drop resources - otherwise AWS will charge you

- npm run cdk diff Sandbox
- npm run cdk deploy Sandbox
- npm run cdk diff App
- npm run deploy-version App
- npm run destroy App
- npm run destroy Sandbox

bin/cloud

```
.. (up a dir)
</Documents/prototype/
▸ app/
▸ bin/
▾ cloud/
  ▸ app_versions/
  ▾ bin/
      cloud.ts
  ▸ cdk.out/
  ▸ cert/
  ▸ lib/
  ▸ node_modules/
  ▸ test/
    cdk.context.json
    cdk.json
    credentials.example
    eslint.config.mjs
    jest.config.js
    package-lock.json
```

```
1  #!/usr/bin/env node
2  import { App } from "aws-cdk-lib";
3  import { NonProdStack } from "../lib/non-prod-stack";
4  import { RailsStack } from "../lib/rails-stack";
5
6  async function Main() {
7    const app = new App();
8    const cdkEnv = {
9      account: process.env.CDK_DEFAULT_ACCOUNT,
10     region: process.env.CDK_DEFAULT_REGION
11   };
12
13   const sandboxStack = new NonProdStack(app, "Sandbox", {env: cdkEnv});
14   sandboxStack.synth("sandbox");
15
16   const railsStack = new RailsStack(app, "App", {env: cdkEnv});
17   railsStack.synth("sandbox", "prototype");
18 }
19
20 Main();
```

Sandbox Stack

```
1  import { Stack, StackProps } from "aws-cdk-lib";
2  import { Construct } from "constructs";
3  // import * as ssm from "aws-cdk-lib/aws-ssm";
4  // import * as s3 from "aws-cdk-lib/aws-s3";
5
6  // import { createNonprodDatabase } from "./aws-cdk-kit/rds/postgres"
7  import { createVpc } from "./aws-cdk-kit/ec2/vpc";
8
9  export class NonProdStack extends Stack {
10   constructor(scope: Construct, id: string, props?: StackProps) {
11     super(scope, id, props);
12   }
13
14   public async synth(stackName: string) {
15     const resourceNamePrefix = [stackName];
16     // Step 1: Create VPC
17     const _nonProdVpc = createVpc(resourceNamePrefix, this);
18     // Step H: Import ElasticBeanstalk bucket
```

Rails Stack

```
15 export class RailsStack extends Stack {
16   constructor(scope: Construct, id: string, props?: StackProps) {
17     super(scope, id, props);
18   }
19
20   public async synth(stackName: string, projectName: string) {
21     const environmentName = "demo";
22     const resourceNamePrefix = [stackName, projectName, environmentName];
23     // const resourceTags: string[] = [this.stackId, this.region, this.account];
24     // Step 0 (tricky): Create regional bucket for ElasticBeanstalk
25     const regionalEbBucket = Bucket.fromBucketName(this, "RegionalEbBucket", `elasticbeanstalk-${this.region}-${this.account}`);
26     // Step 1: Fetch VPC
27     // const vpcID = ssm.StringParameter.fromStringParameterName(this, "vpcID", `/${stackName}/VpcID`).stringValue;
28     const preProductionVpc = ec2.Vpc.fromLookup(this, "RailsVpc", {vpcName: con.ec2VpcName([stackName])})
29     // Step H: Fetch RDS data
30     // Step 2: Create ElasticBeanstalk application
31     const appRole = createAppRole(resourceNamePrefix, this);
32     const preProductionApp = createApplication(resourceNamePrefix, appRole, this);
33     // Step 3: Create ElasticBeanstalk environment
34     const instanceProfile = createEc2InstanceProfile(resourceNamePrefix, [regionalEbBucket.bucketArn], this);
35     const [demoEnv, _demoSg] = await createEnvironment(preProductionApp, resourceNamePrefix, instanceProfile, preProductionVpc, th
36     // Step 4 (optional): Create S3 bucket for ElasticBeanstalk environment
 5.8.2cloud/lib/rails-stack.ts  Line:36/45[80%]Col:10Buf:#26[101][0x65]
```

Constructs L1, L2, L3

```
9  export async function createEnvironment(
10   application: CfnApplication,
11   resourceNamePrefix: string[],
12   instanceProfile: iam.IInstanceProfile,
13   vpc: ec2.IVpc,
14   stack: Stack,
15   solutionStackName: string) : Promise<[CfnEnvironment, ec2.SecurityGroup]> {
16     const environmentName = con.ebEnvironmentName(resourceNamePrefix);
17     const securityGroups = createSecurityGroups(resourceNamePrefix, vpc, stack);
18
19     const env = new CfnEnvironment(stack, environmentName, {
20       applicationName: application.applicationName!,
21       environmentName: environmentName,
22       solutionStackName: solutionStackName,
23       optionSettings: await envOptionSettings(resourceNamePrefix, instanceProfile, securityGroups, vpc),
24     });
25     const cfnInstanceProfile = instanceProfile.node.defaultChild as iam.CfnInstanceProfile;
26     env.addDependency(cfnInstanceProfile);
27     env.addDependency(application);
28
TSC 5.8.2cloud/lib/aws-cdk-kit/eb/env.ts  Line:24/71[33%]Col:7Buf:#31[59][0x3B]
```

Use Cases

- AWS - Use
- Not AWS - Not Use

Alternatives

- CDK for Terraform
- Pulumi

Strengths

- Distributable projects
- Conventions
- Dynamic loading
- Compilation errors
- ?Tagging
- ?Tests

Homework

1. Provision RDS
2. Change VPC to use NatGateway
3. Provision S3 bucket for environment
4. Make tagging part of S3 bucket provisioning
5. Implement GitHub deployment
6. Change ElasticBenastalk EC2 configuration
7. Change VPC CIDR configuration
8. Destroy Sandbox stack before App

Learning Resources

- AWS CDK Immersion Day Workshop
- The AWS CDK layer guide
- Construct Hub
- aws-samples/aws-cdk-examples (GitHub)

Thank you for your time and attention!