

A cluster of overlapping triangles in shades of blue and teal in the top-left corner.

Introduction into AWS CDK

25/01/2024

A cluster of overlapping triangles in shades of blue and teal in the bottom-right corner.

About me

- Petar Repac

- Developer / DevOps / Cloud / APIs / ...
- Technical Team Lead - R&D

- AWS Certs


- AWS Solutions Architect - Professional
- AWS Security - Specialty

- Contact

- <https://www.linkedin.com/in/petarrepac/>
- petar.repac@gmail.com

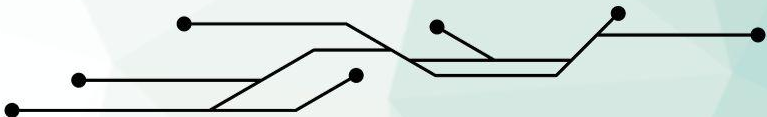
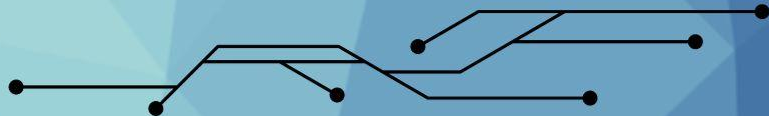


Agenda

- How did we got here?
 - What is the AWS CDK?
 - CDK Setup
 - Demo
 - Concepts
 - Demo 2
 - Resources
- 

How did we got here?

- Manual deployments
- Local scripts
- AWS CloudFormation
 - + local scripts
- AWS CDK
 - CI/CD pipeline



What is the AWS CDK?

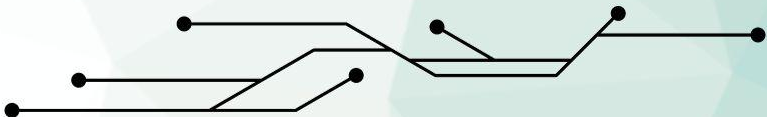
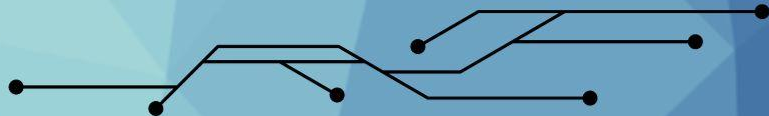
- framework for defining cloud infrastructure in code (IaC)
- via **AWS CloudFormation**
- open source (Apache-2.0)
- **expressive power of a programming language**
 - TypeScript, JavaScript, Python, Java, C#/.Net and Go
 - abstractions
 - abstractions over abstractions over abst....

History

- 07/2019 - AWS CDK reached GA (TypeScript and Python)
- 11/2019 - added Java and .NET
- 12/2021 - AWS CDK v2 reached GA
- 06/2023 - End support for CDK v1
- Github - 10.9k stars, 504 releases
- <https://github.com/aws/aws-cdk>

CDK Setup

- Install Node.js
 - `node -version`
- Install CDK Toolkit
 - `npm install -g aws-cdk`
 - `cdk --version`
- Bootstrapping
 - `cdk bootstrap aws://ACCOUNT-NUMBER/REGION`

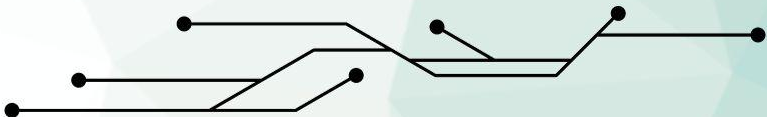
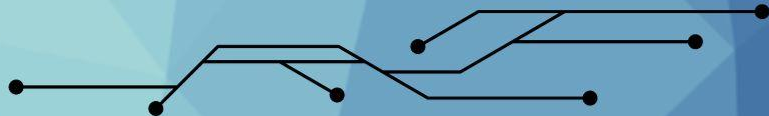


Demo HelloCdk

```
mkdir hello-cdk  
cd hello-cdk  
cdk init app --language csharp
```


Concepts

- **app**
 - application that uses the CDK to define AWS infrastructure
 - contains stack(s)
- **stack**
 - CloudFormation stacks
 - contains constructs
- **construct**
 - defines one or more concrete AWS resources



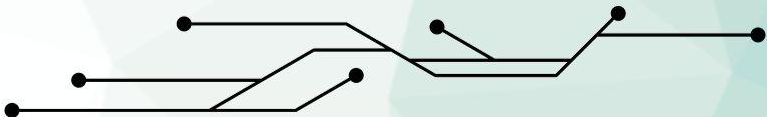
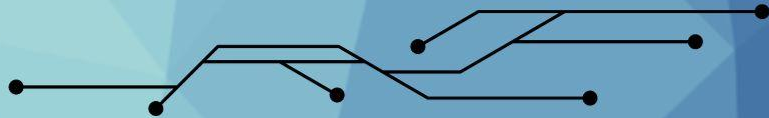
Concepts

- synthesis process

- CDK Toolkit converts one or more CDK stacks to AWS CloudFormation templates and related assets
- catches logical errors in defining your AWS resources

- deployment

- may find permission issues



App

Stack(s)

Construct



Amazon SQS
Queue



AWS Lambda
Function

Construct



Amazon S3
Bucket



Amazon
DynamoDB
Table

TypeScript
JavaScript
Python
Java
C#/.NET

Cloudformation
template

```
Resources:
  MyVpcSPBCKAP:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
      InstanceTenancy: default
      Tags:
        - Key: Name
          Value: MyEcsConstruct/MyVpc
```



AWS
CloudFormation

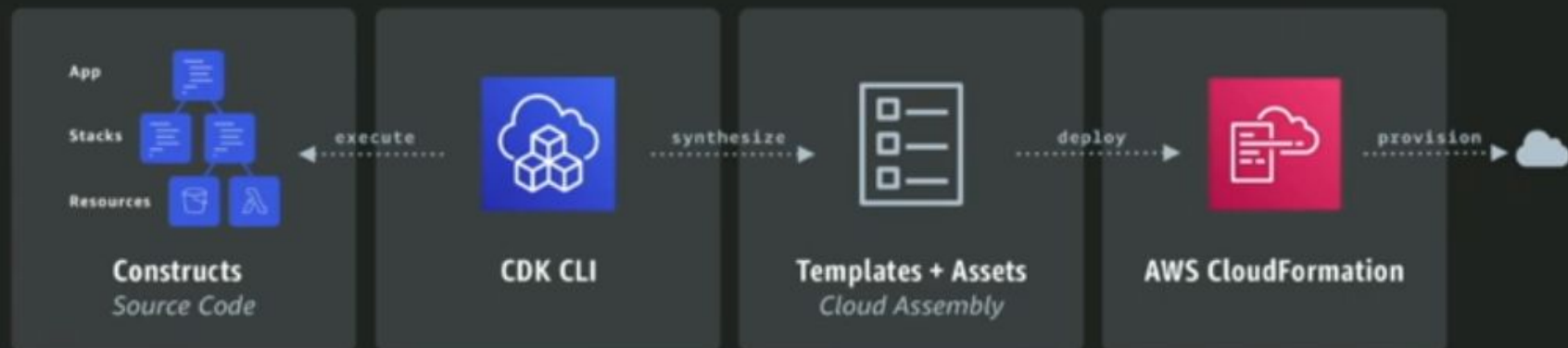
Resources



From constructs to the cloud

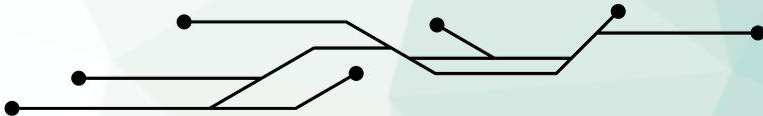
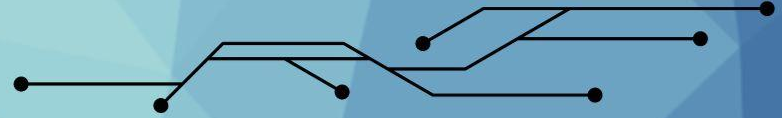


AWS CDK



Construct types

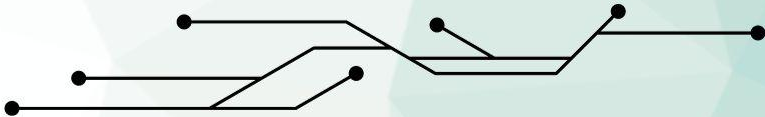
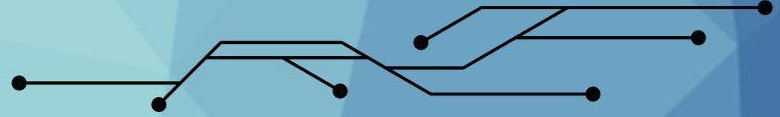
- **AWS CloudFormation-only or L1**
 - correspond directly to resource definition in AWS CloudFormation
 - always have names that begin with `Cfn`
 - E.g. `CfnBucket` is the L1 construct for an S3 bucket.



Construct types (cont.)

- **Curated or L2**

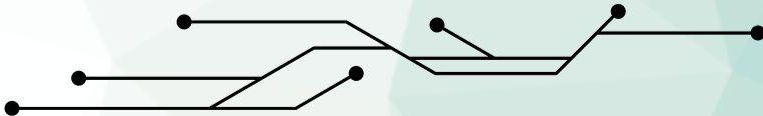
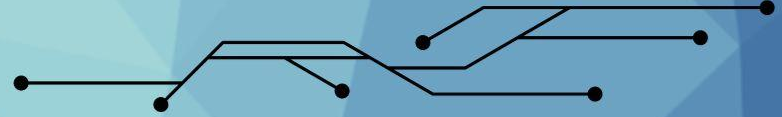
- abstractions to address specific use cases
- encapsulate L1 resources, provide sensible defaults and best practice security policies
- e.g. `Bucket` is the L2 construct for an S3 bucket



Construct types (cont.)

- **Patterns or L3**

- Patterns declare multiple resources to create entire AWS architectures for particular use cases
- E.g. `ApplicationLoadBalancedFargateService`



Benefits

- “**expressive power of a programming language**”
 - **compiler**, tooling, code completion, libraries system
- high-level constructs, sensible, secure defaults
 - AWS Construct Library
 - less code + code reuse
- programming idioms (“if”, “while”, composition, inheritance, ...)
 - create infra using building blocks

Benefits (cont.)

- application code, infra and config - all in one place
 - runtime code and infra stay in sync
- unit testing
- still uses the power of AWS CloudFormation
- share infrastructure design patterns
 - e.g. as libs
 - among teams, inside organization, with public

Below the surface

- AWS CDK is developed in one language (TypeScript)
 - Language bindings are generated for the other languages through the use of a tool called JSII
 - <https://aws.github.io/jsii/overview/runtime-architecture/>

Resources

- <https://www.cdkday.com/>
- <https://github.com/aws-samples/aws-cdk-examples>
- <https://constructs.dev/>
- <https://docs.aws.amazon.com/cdk/v2/guide/cli.html>