AWS Cloud Development Kit

AGENDA

- Warum Infrastruktur mit Code?
- Was gibt es bereits für AWS?
- Prinzip, Lifecycle, Projektaufbau
- Setup, Demos
- Testing, Continuous Integration
- Learnings
- Weitere Werkzeuge
- Construct Library
- Ausblick

INFRASTRUCTURE

Warum Infrastruktur ~~as~~ with Code?

- Cloud App + Infrastruktur in bevorzugter Sprache
- Hohe Abstraktion vermeidet wachsende Deskriptoren mit steigender Komplexität
- Keine markup-artiges Yaml, Json or low-level provider-spezifische Template Sprache
- Geringere Wahrscheinlichkeit von Kopien statt Wiederverwendung (komplexer Infrastruktur)
- AWS Cross Region Bedarf für eine Anwendung mit Logik und Multi-Stacks einfach zu decken

WEITERE VORTEILE (AWS CDK)

- Logik (if, for-loops etc.) zur Infrastrukturdefinition
- Objektorientierte Techniken zur Modellierung
- Teilen und Wiederverwenden von Komponenten/Bibliotheken
- Organisation von Projekten in logisch-fachliche Module
- Testing von Infrastruktur-Code mit Standardmittel
- Bestehende Code Review Workflows nutzbar
- Code Completion in der IDE

(MÖGLICHE) NACHTEILE

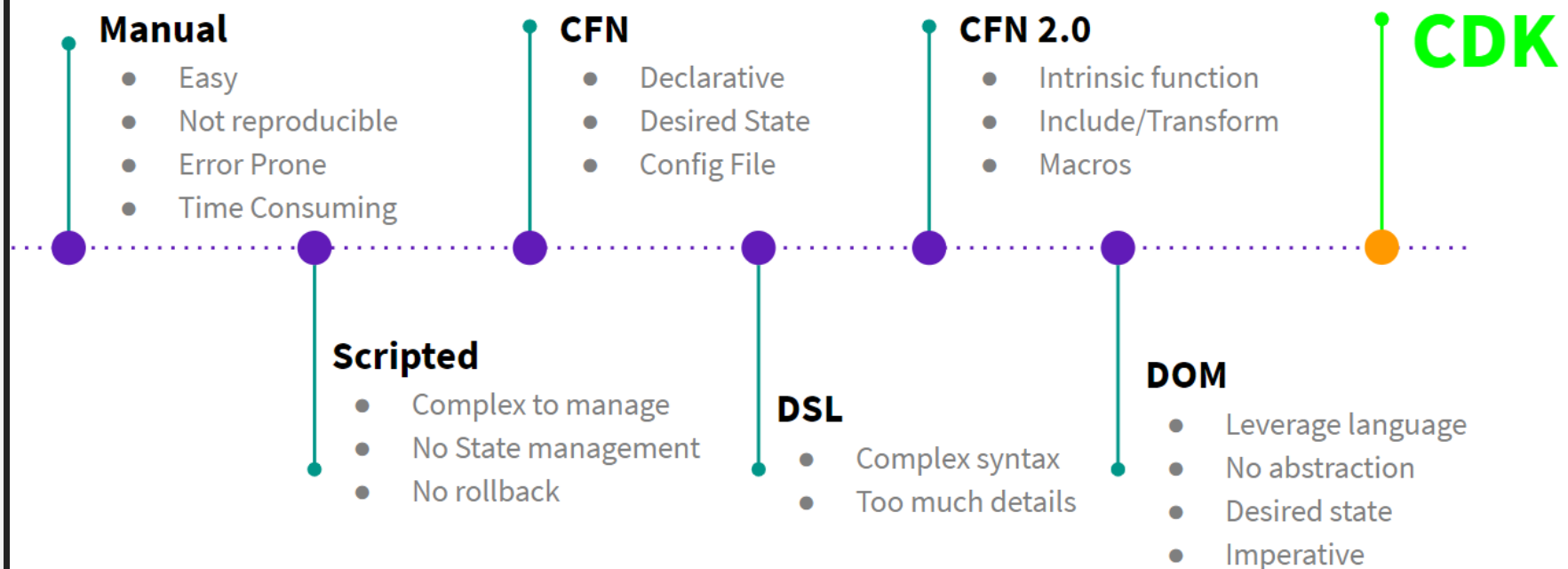
- Neues SDK/API, ggf. mit unvollständiger Abdeckung des Cloud Ressourcen.
- Modularisierungskonzepte eventuell nicht übertragbar.
- Development Kit nicht auf andere Provider übertragbar.
- Akzeptanz bei versierten Terraformern und Serverless/SAM Nutzern (DevOps)

WERKZEUGE

Was gibt es bereits für AWS?

- Mit gewohnten Programmiersprachen:
  - Pulumi (multi cloud, polyglott)
  - Troposhere, Sceptre (Python)
  - ...?
- Deklarativ, ohne Programmiersprachen:
  - Cloudformation, Serverless, SAM
  - Cfn Modules, Stacker, Stack Deployment Tool
  - Terraform
  - ...?

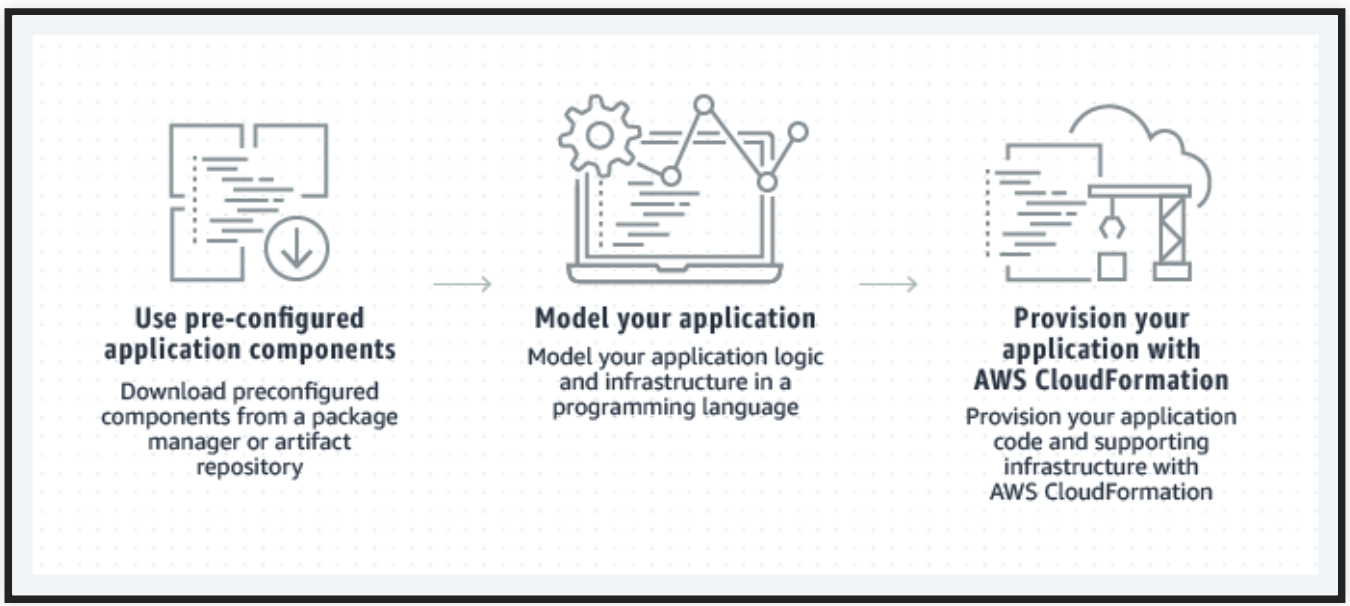
Evolution of infra deployment on AWS



Was ist AWS CDK?

- AWS Cloud Development Kit ist ein Open-Source-Framework für die Softwareentwicklung
- Damit wird Cloud-Infrastruktur als Code mit modernen Programmiersprachen definiert und über AWS Cloudformation bereitgestellt
- Unterstütze Sprachen: TypeScript, JavaScript, Python, Java, and .NET (C#, F#)
- Entwickler erstellen in ihrer Sprache wiederverwendbare Komponenten (Constructs) und führen sie in Stacks und Apps zusammen

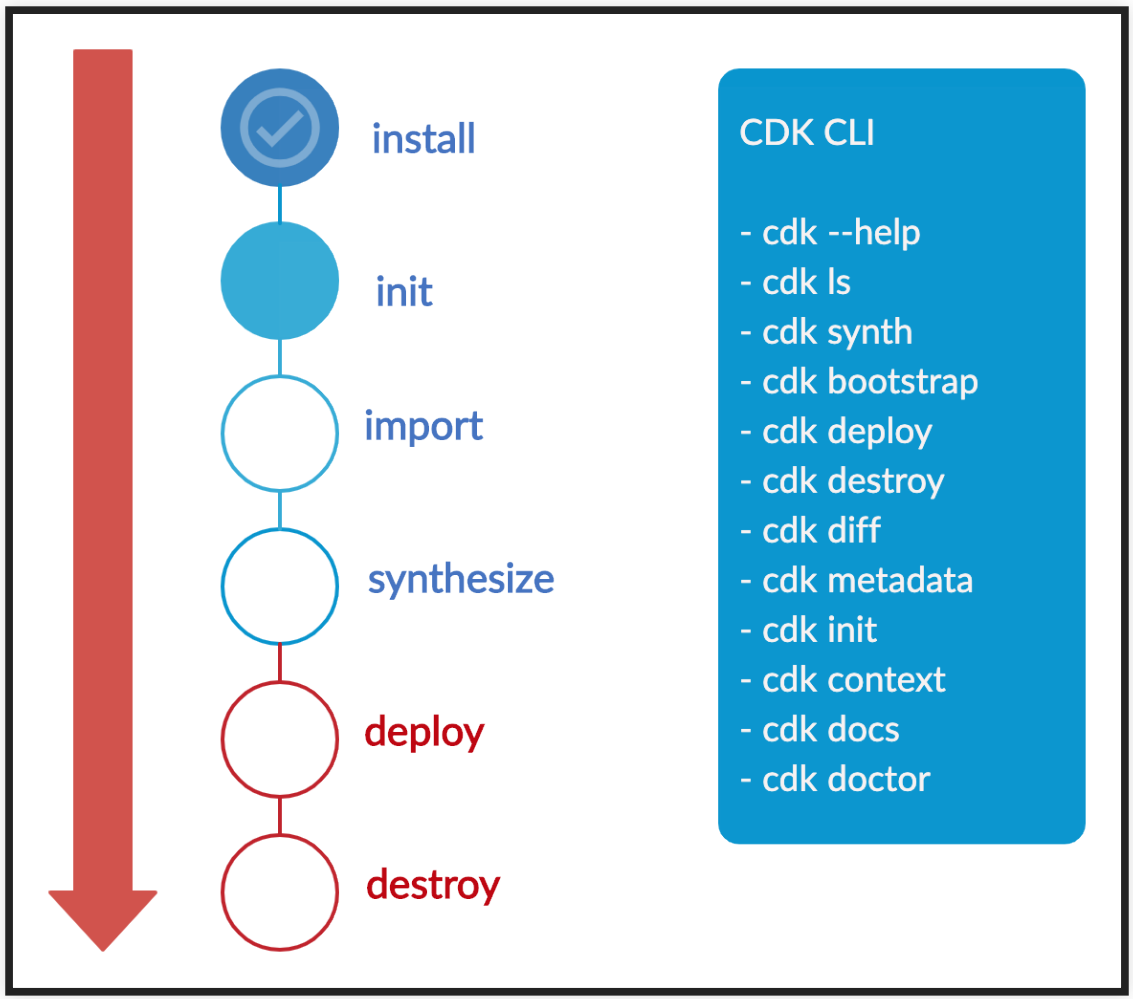
FUNKTIONSPRINZIP



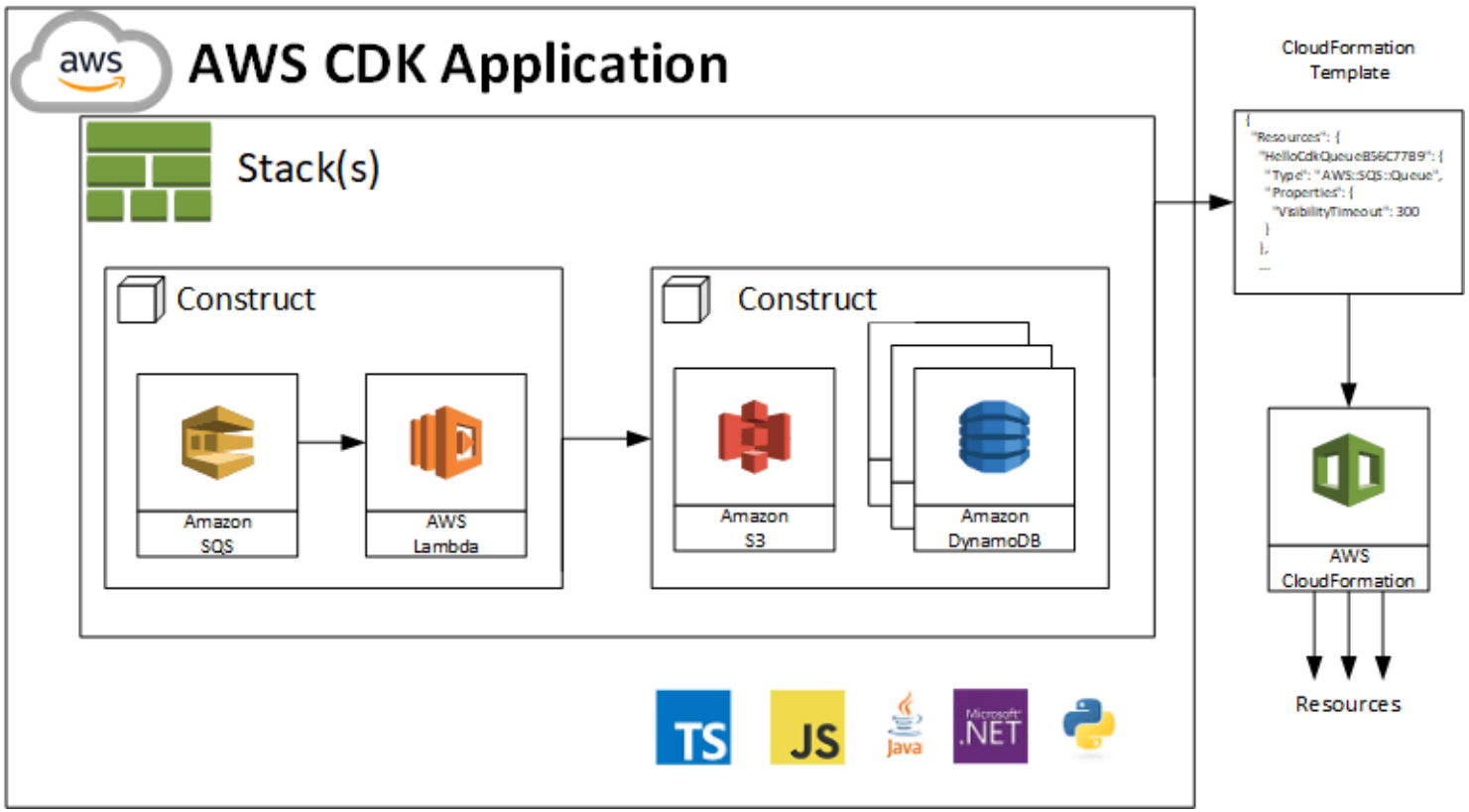
Erwartete Qualität & AWS Abdeckung

Aktueller Zustand (stabil) & Abdeckung (teilweise)

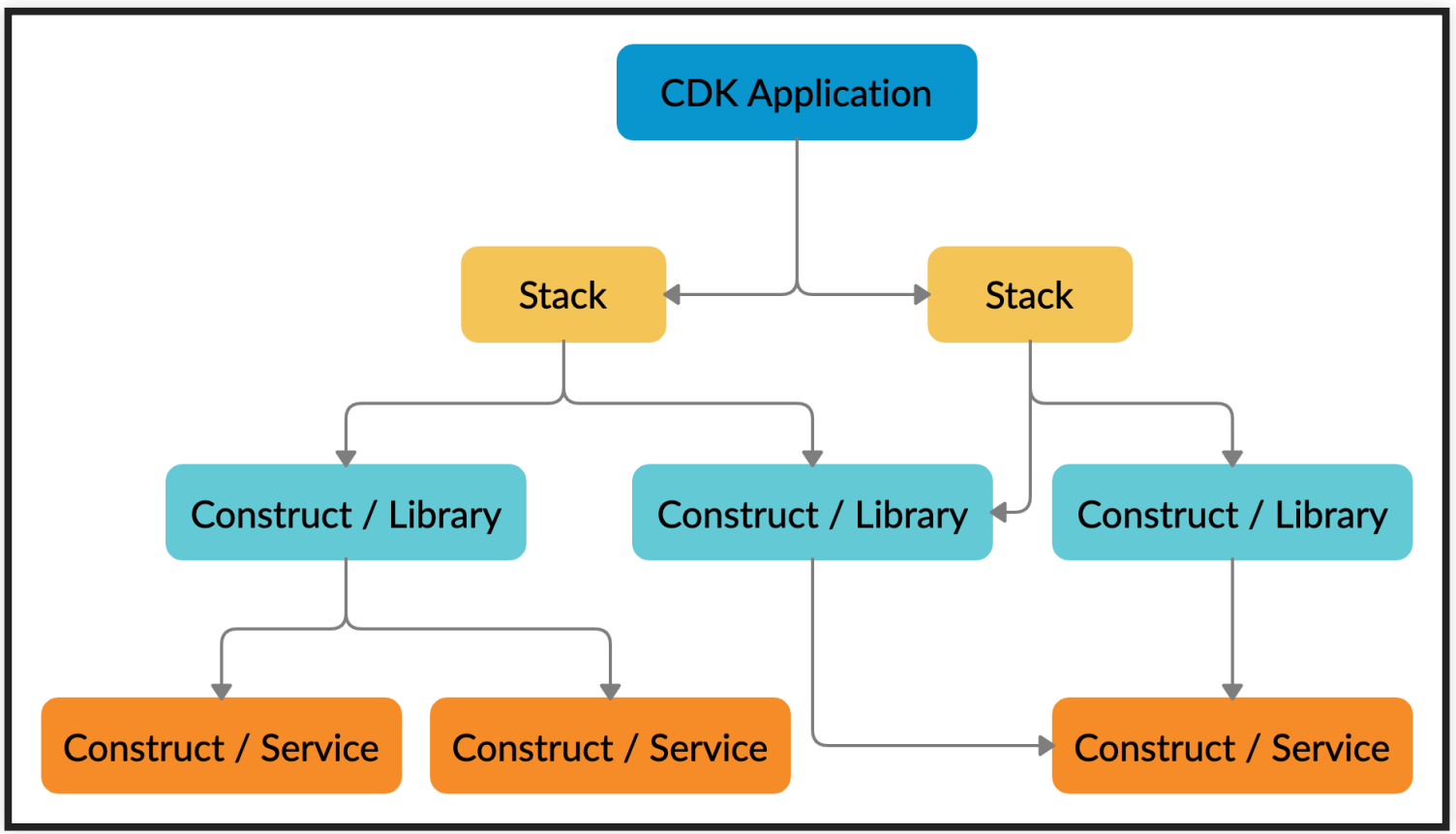
LEBENSZYKLUS



PROJEKTÜBERBLICK



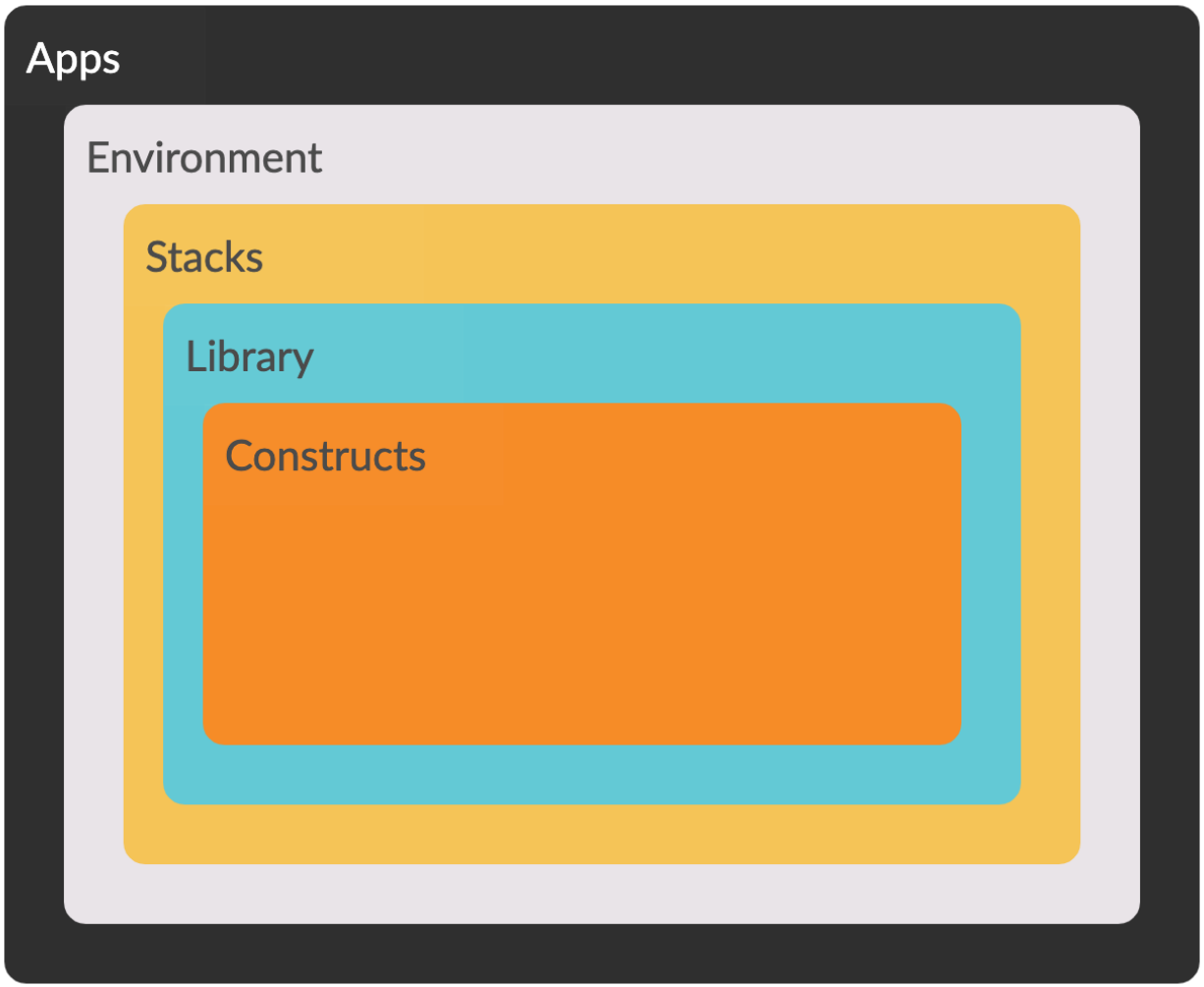
PROJEKTHIERARCHIE



PROJEKTHIERARCHIE

- App - Projektursprung, stellt Stacks bereit
  - Environment - Multi-Account, Multi-Region
- Stack - Bereitstellbare Einheit einer CDK App
- Construct - AWS Resource Repräsentation
  - High Level (Bibliothek wie VPC oder CustomResource)
  - Low level (CfnResources) wie Subnets, Gateways etc.

PROJEKTHIERARCHIE



SETUP

- AWS CLI und ein oder mehrere AWS Profile
- Node.js > 10.3.0 (für alle Sprachen, JS Bindings)
- AWS CDK
  - Ref: Getting started with CDK

```
npm install -g aws-cdk
cdk --version
```

DEMOS

- Polyglot Blueprints
- CDK synth, deploy, diff, destroy
- Sample Stack: API Cors Lambda Crud DynamoDb
- Amplify App from Github
- Vue.js SPA Deployment to S3
- Sprachsynthesis App with Amazon Polly
- Brand new: CDK Watchful Construct Library

POLYGLOTTE BLUEPRINTS

- Aktuelle Sprachbibliotheken und Lifecycle/Dependency Manager

```
mkdir demo && cd demo
cdk init sample-app language=typescript
npm install
npm run build && npm run test
```

```
mkdir demojava && cd demojava
cdk init lib language=java
mvn package
```

CDK SYNTHESE UND DEPLOYMENT

```
cdk ls
cdk synth
cdk deploy
cdk diff
cdk destroy
```

```
cdk synth '*' --profile any-profile-name
```

```
cdk synth StackOne StackTwo
```

SAMPLE: CDK MULTISTACK DIFF

```
Stack Cloudfront-Stack
Resources
[~] AWS::Route53::RecordSet CnameRecord-www.wohngemeinschaft.shared-preview.immocloud.io-wohngemeinschaft.shared-preview.immocloud.io CnameRecordwwwwohng
aftsharedpreviewimmocloudiowohngemeinschaftsharedpreviewimmocloudio84E5DE86 replace
└─ [~] HostedZoneId (requires replacement)
     ├─ [-] /hostedzone/Z2BXJ6TP3W1FB0
     └─ [+] Z2BXJ6TP3W1FB0
[~] AWS::Route53::RecordSet CnameRecord-ratgeber.wohngemeinschaft.shared-preview.immocloud.io-wohngemeinschaft.shared-preview.immocloud.io CnameRecordrat
ngemeinschaftsharedpreviewimmocloudiowohngemeinschaftsharedpreviewimmocloudioE080A036 replace
└─ [~] HostedZoneId (requires replacement)
     ├─ [-] /hostedzone/Z2BXJ6TP3W1FB0
     └─ [+] Z2BXJ6TP3W1FB0
[~] AWS::S3::Bucket Cloudfront-Target-Bucket-wohngemeinschaft-de CloudfrontTargetBucketwohngemeinschaftde342112DD
└─ [~] WebsiteConfiguration
     ├─ [-] Removed: .ErrorDocument
     ├─ [-] Removed: .IndexDocument
     └─ [+] Added: .RedirectAllRequestsTo
[~] AWS::CloudFront::Distribution SiteDistribution/CFDistribution SiteDistributionCFDistribution209CF7F5
└─ [~] DistributionConfig
     └─ [~] .DefaultCacheBehavior:
          └─ [-] Removed: .LambdaFunctionAssociations
[~] AWS::Lambda::Function Custom::CDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C81C01536
└─ [~] Metadata
     └─ [~] .aws:asset:path:
          ├─ [-] asset.2405af9f7d79e8aaf1cbe504cbb8cc3f89c292c82c5bc487b26c0c0c1e9f3f15.zip
          └─ [+] asset.99ffb81915bf0df7ed403582b7b1302f4d9cf22ba770ee5a158781217fc7aa24.zip
[~] AWS::Route53::RecordSet AliasRecord-wohngemeinschaft.shared-preview.immocloud.io AliasRecordwohngemeinschaftsharedpreviewimmocloudio67911F24 replace
└─ [~] HostedZoneId (requires replacement)
     ├─ [-] /hostedzone/Z2BXJ6TP3W1FB0
     └─ [+] Z2BXJ6TP3W1FB0

Stack DynamoDb-Stack
There were no differences
Stack LambdaEdge-Stack
```

DEMO: SAMPLE STACK ON GITHUB

API-Cors-Lambda-Crud-DynamoDb Sample

DEMO: AMPLIFY CONSOLE APP

Amplify Console App: Deploy a static site from Github

AWS Amplify: Hosting for full stack serverless web apps with continuous deployment

DEMO: SPA DEPLOYMENT TO AWS S3

Eine Vue.js App mit API-Zugriff

DEMO: SPRACHSYNTHESIS APP WITH AMAZON POLLY

Build a Text to Speech App with Amazon Polly

DEMO: CDK WATCHFUL

Monitor CDK Apps

Github -> CDK Watchful

TESTING

Sample: Jest mit Snapshots

```
describe('New DynamoDb Resource can be setup', () => {
  test('Synthesized Stack matches snapshot', () => {
    const stack = prepareTestStack(new cdk.App());
    expectCDK(stack).notTo(matchTemplate({
      "Resources": {}
    }, MatchStyle.EXACT));
    expect(SynthUtils.toCloudFormation(stack)).toMatchSnapshot
  });
  test('BillingMode is Pay_per_request', () => {
    const stack = prepareTestStack(new cdk.App());
    expectCDK(stack).to(haveResource("AWS::DynamoDB::Table", {
      BillingMode: "PAY_PER_REQUEST"
    }));
  });
}o
```

TESTING

Sample: Snapshot Test Result

```
- Snapshot
+ Received

@@ -163,18 +163,10 @@
            ],
          },
        },
        "Type": "AWS::SSM::Parameter",
      },
-     "LambdaVersionFA49E61E": Object {
-       "Properties": Object {
-         "FunctionName": Object {
-           "Ref": "LambdaD247545B",
-         },
-       },
-       "Type": "AWS::Lambda::Version",
-     },
      "LambdaVersionsha256a6bfe6f8fa4ddf97dcff8826f636b543836576066259ba2d54a939177ee31b64BABFA42F": Object {
        "Properties": Object {
          "FunctionName": Object {
            "Ref": "LambdaD247545B",
          },

  24 |       "Resources": {}
  25 |     }, MatchStyle.EXACT));
> 26 |     expect(SynthUtils.toCloudFormation(stack)).toMatchSnapshot();
     |                                                ^
```

CI/CD PIPELINE - JENKINS CDK DEPLOYMENT

```
stage('AWS CDK synthesis & deployment'){
  agent { docker {
      image 'robertd/alpine-aws-cdk'
      args '-it -v $WORKSPACE:/app -w /app'
  } }
  steps {
    withCredentials([file(credentialsId: 'PROFILES',
                          variable: 'CREDENTIALS_FILE')]) {
      script {
        sh "cdk --app 'npx ts-node bin/edge-multi-stack.ts'
                synth '*'"
        sh "cdk --app 'npx ts-node bin/edge-multi-stack.ts'
                deploy Route53-Stack LambdaEdge-Stack Cloudfro
      }
    }
```

CI/CD PIPELINE - CDK BOOTSTRAPPING

```
script {
  bootStrapUsEast1 = sh(returnStdout: true, script:
     "cdk bootstrap aws://${accountIdDev}/us-east-1")
  bootStrapEuCentral1 = sh(returnStdout: true, script:
     "cdk bootstrap aws://${accountIdDev}/eu-central-1")
  }
```

LEARNINGS

- Thoughtworks hat recht (vermeide händische Cloudformation Vorlagen)
- Steile Lernkurve, schnelle Einarbeitung
- Bequemes Arbeiten z.B. mit TypeScript
- Keine Hürden für sauberes Testen
- Staging, Cross Region und Cross Account Deployments inkl. CI/CD pipeline möglich.
- Nutze Docker Container mit AWS SDK, CDK und Typescript Toolstack
- und mehr...

EVEN MORE LEARNINGS

- Bessere Mikroarchitektur durch Unit-Tests und Multistack Deployment
- MultiStack mit Route53, ACM, Cloudfront, Lambda@Edge + DynamoDB war erfolgreich
- CDK für CloudFront mit Lambda@Edge ggf. problematisch
    - Lambda Assoziationen und Löschungen
    - Timing
    - Cloudfront Konstrukt teilweise zu unflexibel
    - Nicht alles Cross-stack modularisierbar

WEITERE WERKZEUGE

- Disassembler cdk-dasm (experimentell)
- AWS Jsii (stabil, CDK Kern-Tool)
- Docker Container zur Arbeit mit CDK

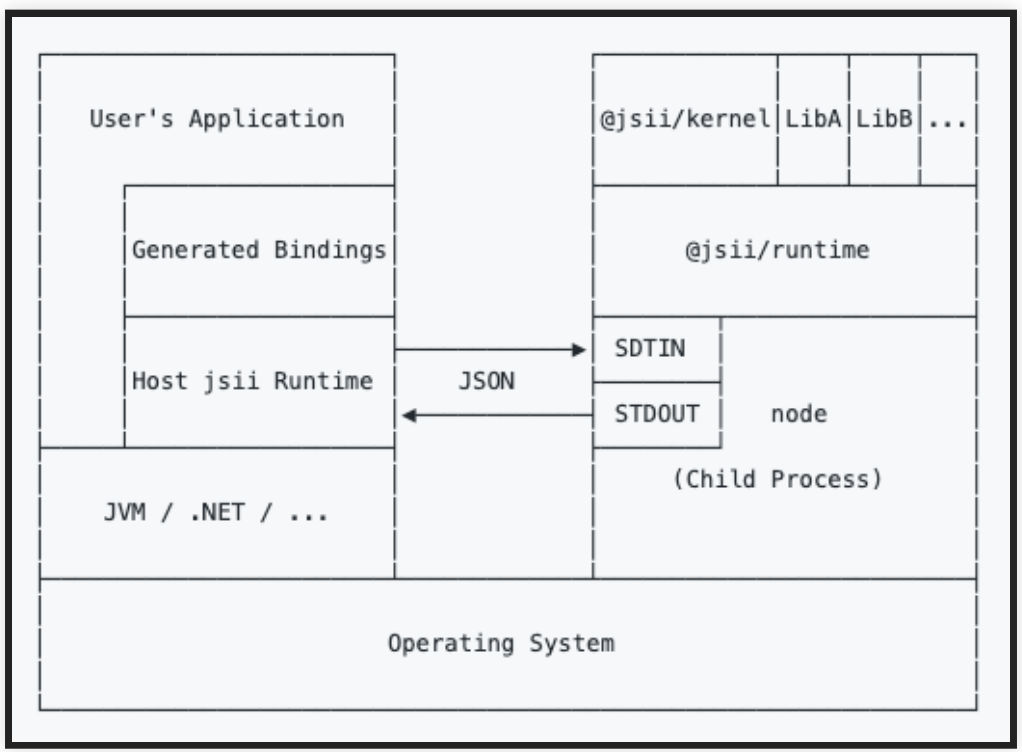
DISASSEMBLER CDK-DASM

- Cloudformation Disassembler
- Erzeugt Typescript code aus Cfn Templates
- Experimentell, nicht empfohlen für Produktivsysteme
- Nützlich zum Starten mit CDK
- NPM Package cdk-asm

```
cdk-dasm < any-stack-template > any-stack.ts
```

AWS JSII

- Produktion polyglotter CDK Bibliotheken aus einer Codebase (TypeScript)
- Github AWS Jsii



```
npm init -y
npm i --save-dev jsii jsii-pacmak
# now configure package.json for polyglot output
npm run build
npm run package
```

DOCKER CONTAINER

- Es gibt diverse Docker Container für die Arbeit mit AWS CDK
  - z.B. Docker Image - Alpine-AWS-CDK
- Zusätzliche Werkzeuge für eine Umgebung mit Node.js und TypeScript
  - z.B. Docker Image - Node-NPX-TypeScript

AWS CONSTRUCT LIBRARY

- Konstrukte bereitgestellt in AWS Construct Library, abstrahiert Cloud-Infrastrukturlogik
- Für alle Sprachen ist eine JS Runtime erforderlich
- Lokale Definition oder via Paket Manager
- AWS CDK Construct Library
  - Stabil: AWS Lambda
  - Experimentell: AWS Kinesis
- Alternative Konstrukte
  - z.B. Destroyable Bucket

WEITERE RESSOURCEN

- AWS CDK
- CDK Workshop
- g aws cdk blog
- ...

AUSBLICK

- Aktuelles Release
- CDK ist stabil und nutzbar
- Einige Konstrukte sind noch experimentell
- Es gibt ein Angebot an 3rd Party Konstrukten (z.B. DestroyableBucket)
- Ökosystem, Stabilität & Abdeckung wachsen
- Multi-Cloud Tools wie Pulumi sind eine Alternative
- aber...
- ...zur Produktion würde ich noch Terraform bevorzugen

Vielen Dank für Eure Aufmerksamkeit

- E-mail: klaus@pittig.de
- Twitter, Github: @jforge