Serverless

Weniger ist mehr, der Weg zur moderneren Architektur?

Code-Days 2019 | Max Körbächer

REPLY

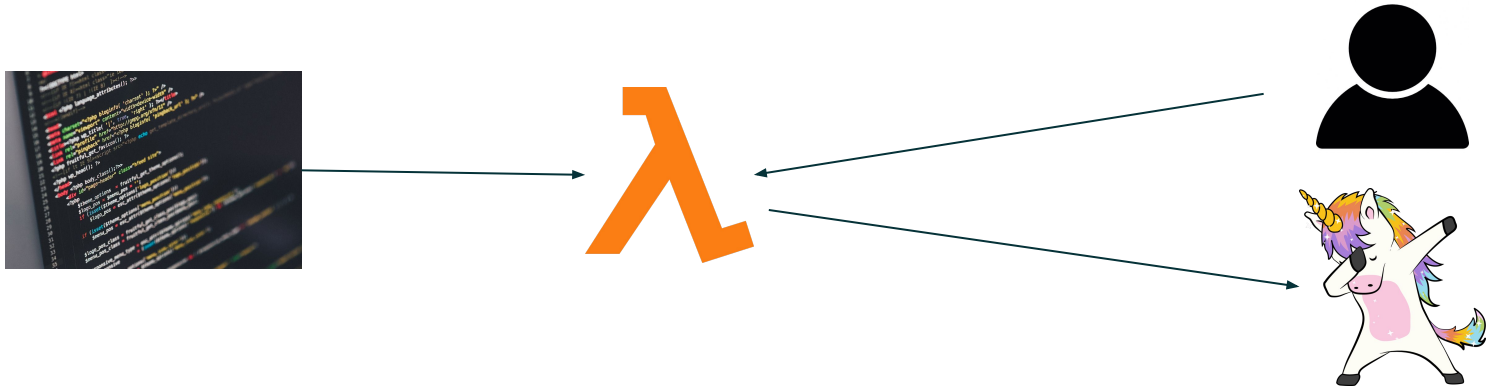
Hey!

Max Körbächer
Senior Cloud Solution Architect @ Storm Reply

- **Design and build cloud ready solutions**
  - microservice & event driven apps
  - serverless & kubernetes based
  - ♥ for  , GraphQL & NoSQL
- **Background as Enterprise Architect & Founder**
- **Visit me at: max.koerbaecher.io**

CODE DAYS

Was ist Serverless?



Was oft unter Serverless allgemein verstanden wird …

... ist in der Regel falsch.

Quelle: pixabay.com

Serverless architectures are application designs that incorporate third-party “Backend as a Service” (BaaS) services, and/or that include custom code run in managed, ephemeral containers on a “Functions as a Service” (FaaS) platform.

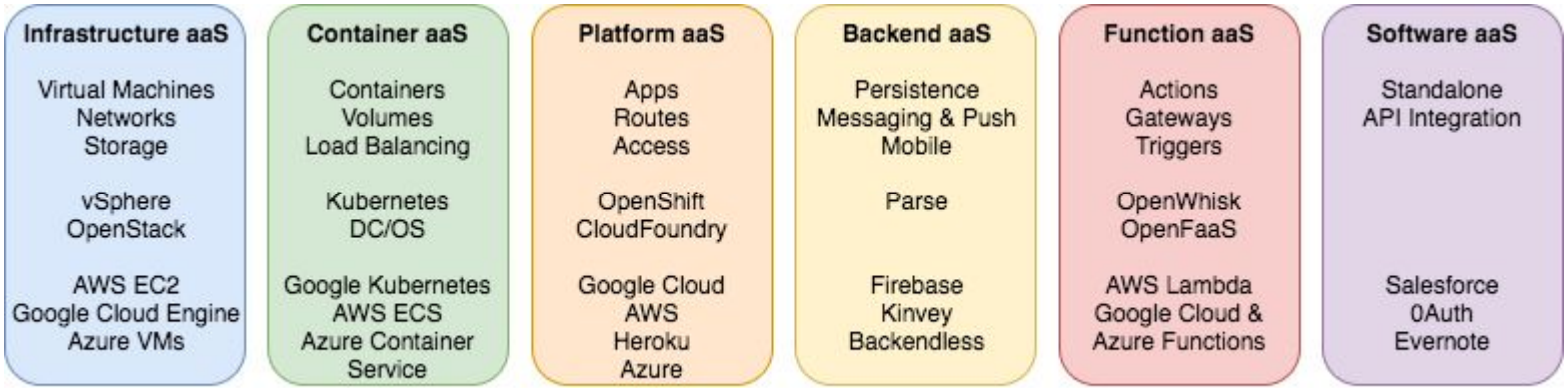
- Mike Roberts

FaaS != Serverless

FaaS im Allgemeinen kann als die Bereitstellung einer Plattform für die Ausführung von Code verstanden werden.

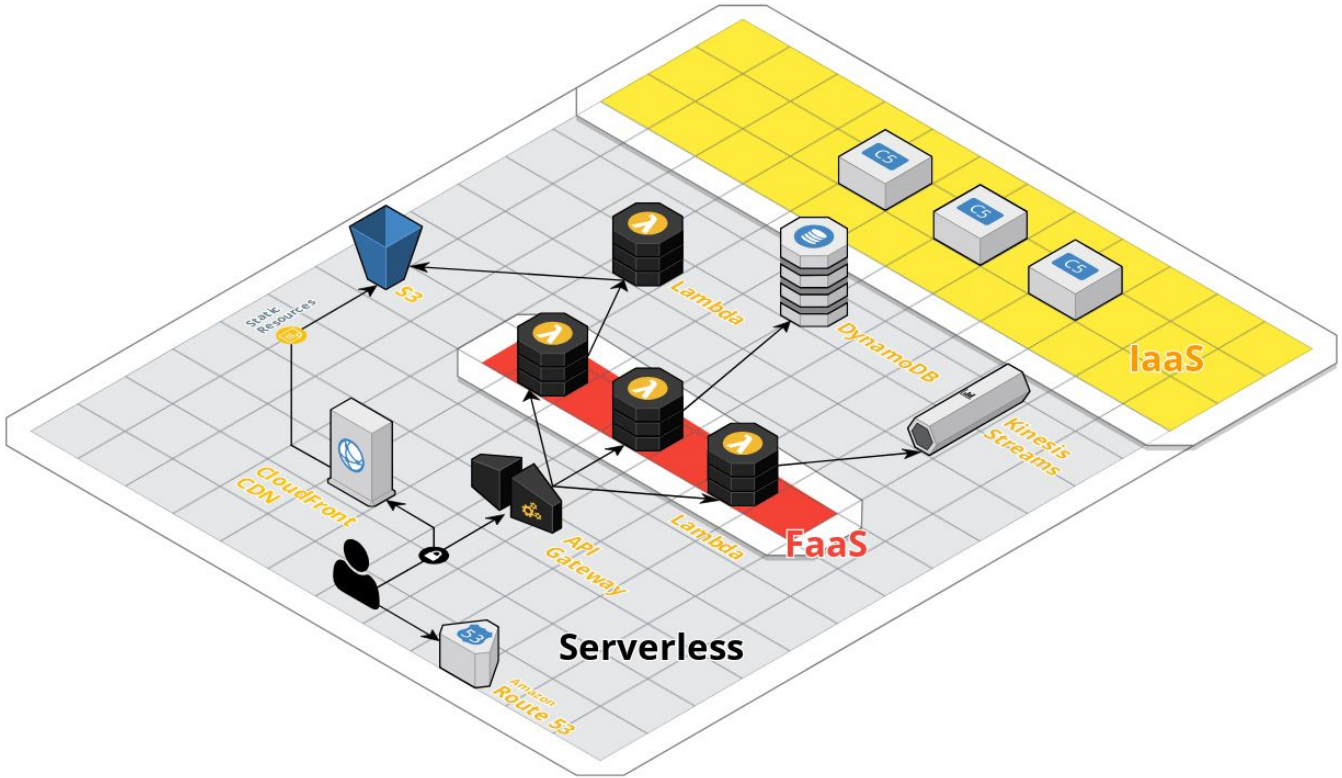
Serverless hingegen ist ein Architekturansatz, der mehrere Komponenten der XaaS Familie verwendet.

XaaS Familie



Serverless nutzt bzw. orchestriert XaaS

FaaS != Serverless II



CODE DAYS

Was verspricht Serverless zu sein?

Kosteneffizient, hochskalierbar & geringe Wartung

- Die Kosten einer serverlosen Anwendung basieren auf der Anzahl der Funktionsausführungen
- Kleinere Entwicklungskomponenten führen zu einer schnelleren Bereitstellung von Funktionen und erhöhen die Anpassungsfähigkeit
- Die Kosten für den Betrieb sinken, im Grunde keine Systemadministration notwendig
- Hochskalierbar, quasi grenzenlos
- Fördert Innovation, Microservices und SOA Prinzipien

“No server is easier to manage than no server”

Die Kehrseite von Serverless

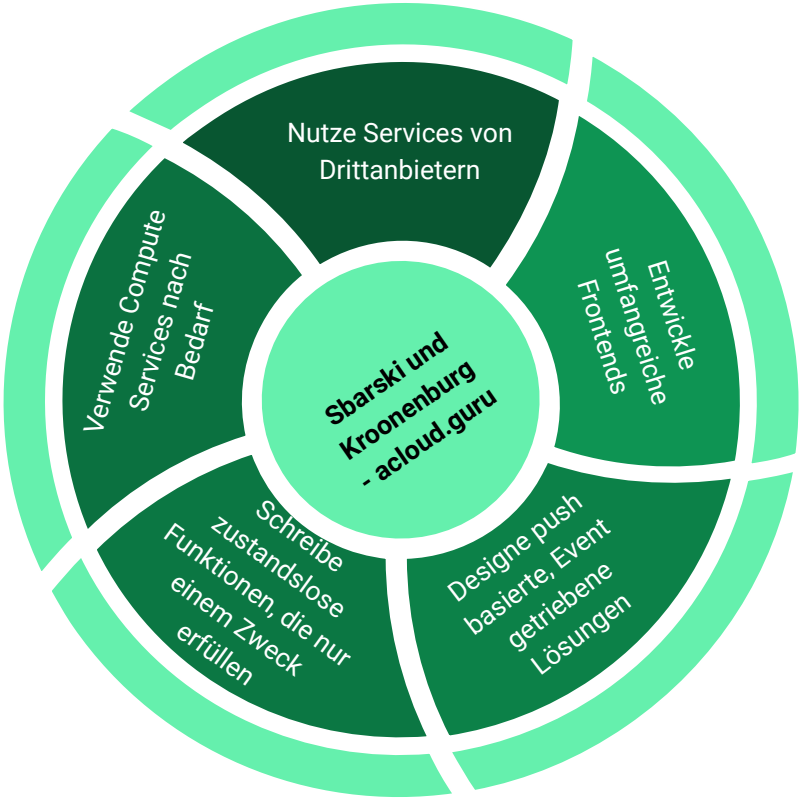
- Risiko bzgl. Sicherheit, Disaster Recovery, Vendor Lock-in
- Technologische Fragmentierung
  - Wirklich schlimm?
- Komplexität der Architektur steigt exponentiell an
- Zeitlich begrenzte Rechenzeit
- Testing, local state und multi tenancy ist eine Herausforderung

FaaS Universum

CODE DAYS

https://landscape.cncf.io/images/serverless.png

Serverless Prinzipien



Organisatorische Voraussetzungen für Serverless

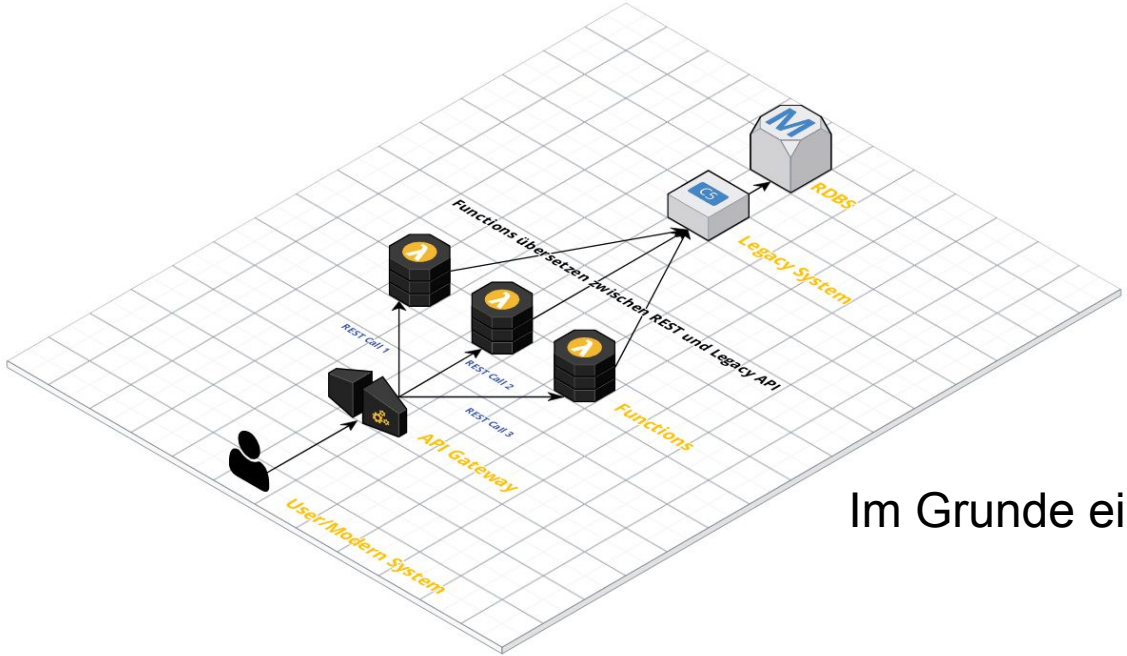
CODE DAYS


Mit Serverless zur
“moderneren”
Architektur?

Wo können wir Serverless einsetzen?

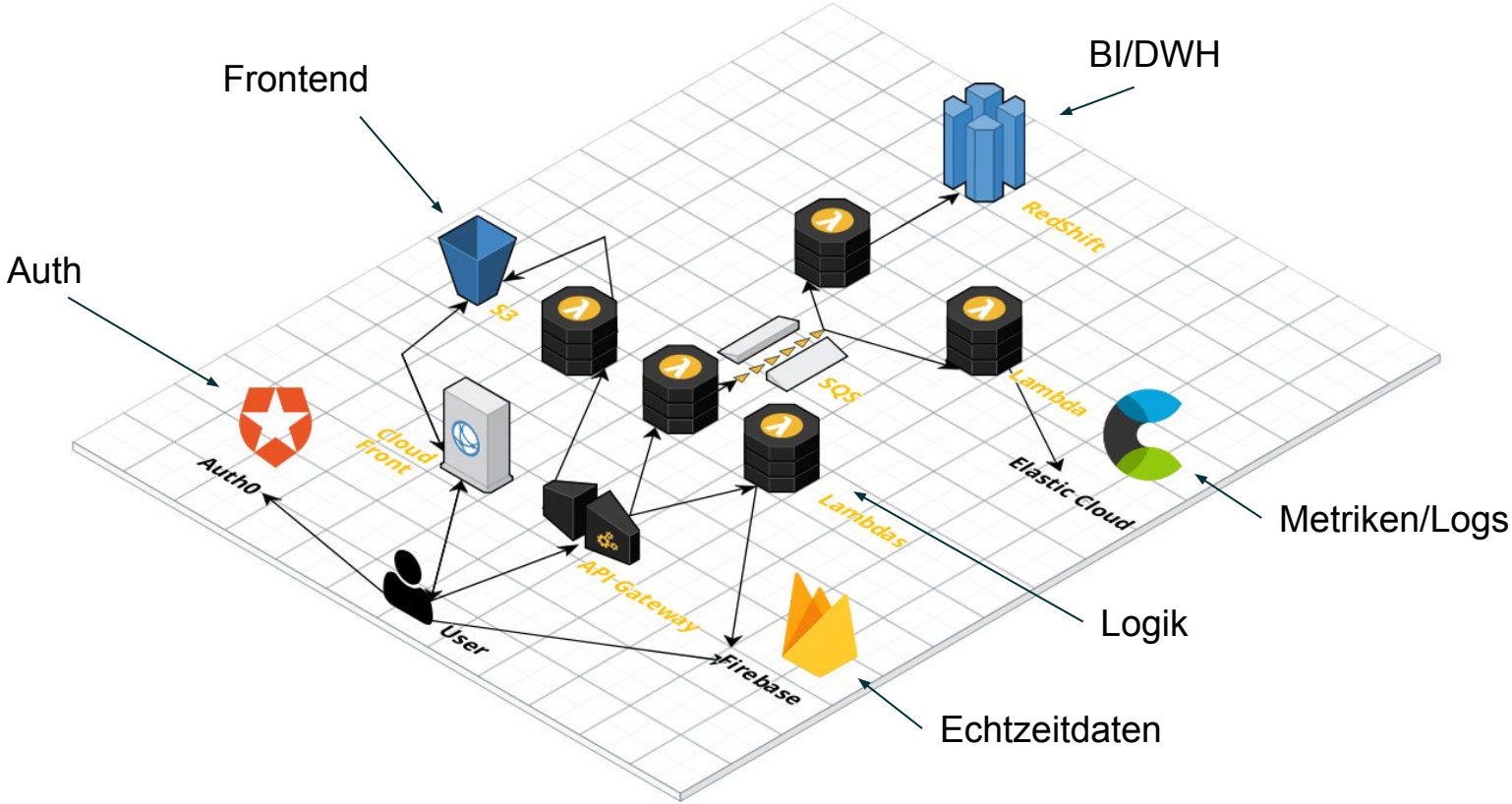
- Web Applikationen - statisch, komplexe Apps
- Backend - Mobile, IoT oder klassische Anwendungsbackend
- Chatbots & Sprachassistenten
- Datenverarbeitung - Batchprozesse & Echtzeitdatenverarbeitung
  - ETL Jobs sind am kommen
- System- & Infrastruktur Management

Beispiel:
API Proxy & Datenmanipulation



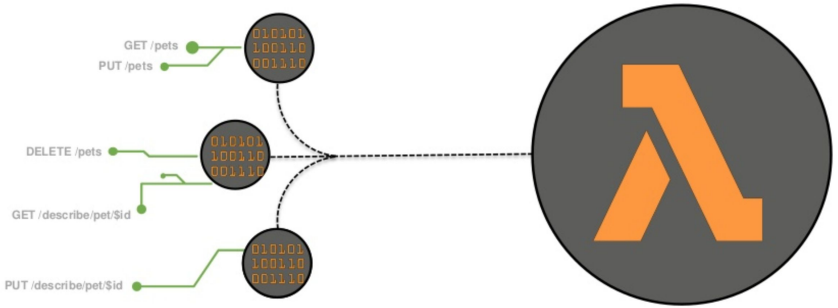
Im Grunde eine Fassade

Beispiel: Serverless Applikation



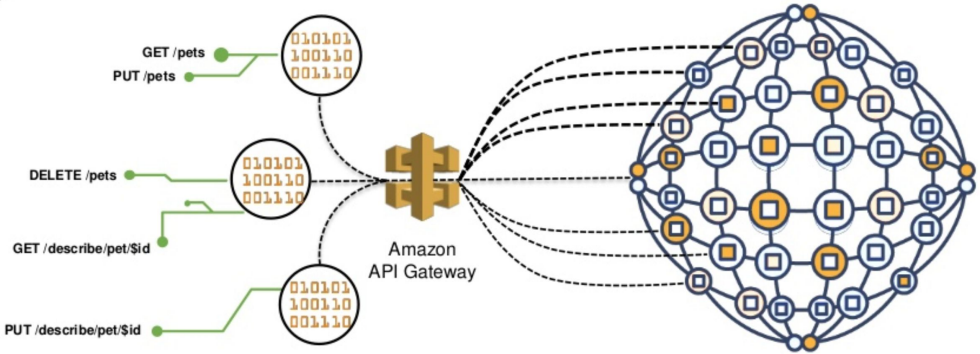
CODE DAYS

Design Anti-/Pattern



Anti-Pattern
Ein einzige große Funktion

Pattern
Mehrere orchestrierte Funktionen



Quelle: aws.com

CODE DAYS

Design Prinzipien

Speicher den Zustand (state) in einen persistenten Storage

Orchestriere die Funktionen mit Statemachines

Verwende Event-Getriebene Designs

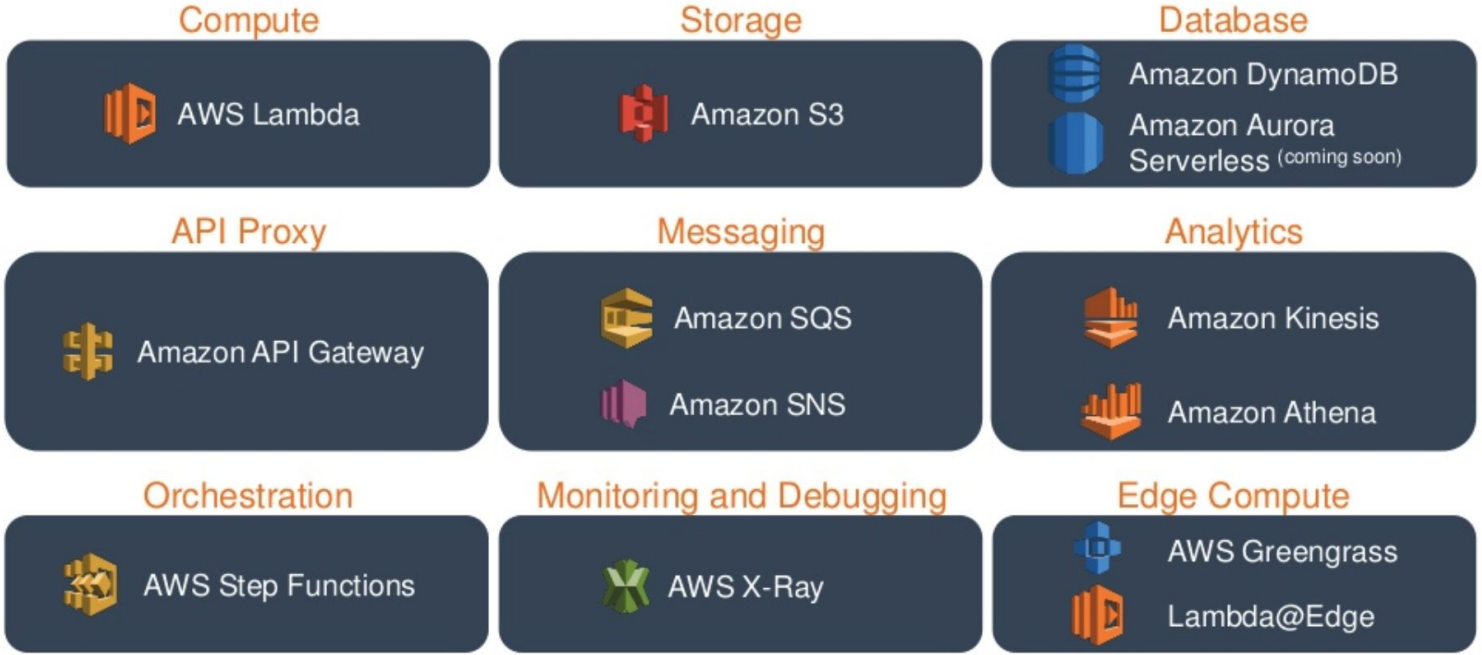
Designe für Fehler und Duplikate

Nutze das Saga Pattern

Verwende API Gateways

CODE DAYS

Cloud Umgebungen sind das perfekte Umfeld für Serverless



Security

ja, auch bei Serverless

- OWASP gilt auch hier bspw. Input Validation
- Eine Funktion = eine Nutzerrolle
- Speichern und Verschlüsseln sie Passwörter und Zertifikate mittels Key & Secret Management Tools
- Verwendung von WAFs und SIEMs

Erreichen wir eine modernere Architektur mit Serverless?

Take Aways

01	Weniger ist mehr!	• Reduziere das Package size! • Frameworks nur wenn es sein muss
02	Benutze ENVs	• Passe deine Funktionen durch ENVs an
03	Die richtige Sprache	• Interpretierte Sprachen sind schneller initialisiert • ABER nicht unbedingt schneller insgesamt
04	Lagere deine Abhängigkeiten aus	• bspw. .jar in /lib Ordner
05	Mehr Speicher, ist mehr CPU, ist mehr Bandbreite	• Ist die Funktion Speicher/CPU/Netzwerk lastig?

Grüße aus der Zukunft

Lambda@Edge

- Funktionen werden via CDN näher am Kunden ausgeführt
- Dynamische Anpassung von Webinhalten, Auth, Security etc.

AWS Greengrass - Lambdas auf IoT Devices

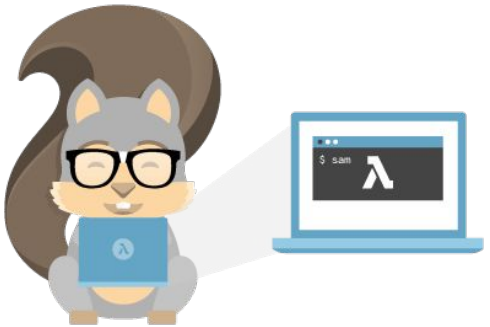
- Funktionen können wie gewohnt entwickelt werden
- Werden auf IoT Devices installiert
- Bspw. Implementierung trainierte Algorithmen (ML) “direkt an der Maschine”

Entwickeln und Bereitstellen

SAM

AWS Serverless Application Model

- CLI Tool für das lokale Entwicklung und Testen
- Aufrufen von AWS Services bspw. S3, DynamoDB
- Lokales API Gateway mit Hot Reload
- Boilerplate Templates




```
Transform: 'AWS::Serverless-2016-10-31'
Resources:
  ThumbnailFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      Runtime: nodejs6.10
      Handler: index.handler
      CodeUri: ./src
      Events:
        ThumbnailApi:
            Type: Api
            Properties:
                Path: /thumbnail
                Method: GET
```

CODE
DAYS

Weitere Tools

Serverless Framework

Weitverbreitetes Framework für Serverless Entwicklung
Open Source & Enterprise
Cloud Agnostisch

Zeit.co

Kostenlos mit Limitierung & Enterprise
Bietet Services wie Tests auf der Zeit Infrastruktur

APEX.run

Open Source
Breite Community

Danke!

Wenn du mehr über Storm Reply erfahren willst:

https://www.reply.com/storm-reply