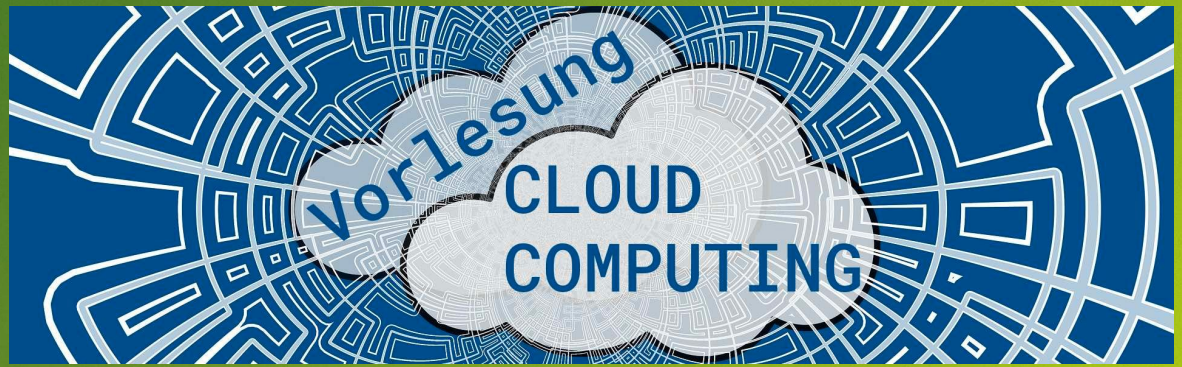
Hochschule **Rosenheim**
University of Applied Sciences

QA|WARE
SOFTWARE ENGINEERING

Serverless



Serverless

is the next logical evolution in Cloud Native Software Development

CNCF Cloud Native Landscape
2019-10-07T04:21:31Z 37ca95b

Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

Greyed logos are not open source

App Definition and Development

Database

Streaming & Messaging

Application Definition & Image Build

Continuous Integration & Delivery

Orchestration & Management

Scheduling & Orchestration

Coordination & Service Discovery

Remote Procedure Call

Service Proxy

API Gateway

Service Mesh

Runtime

Cloud Native Storage

Container Runtime

Cloud Native Network

Provisioning

Automation & Configuration

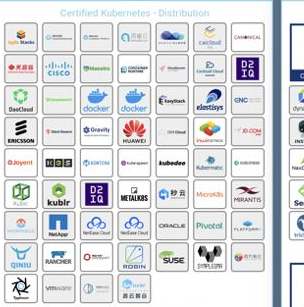
Container Registry

Security & Compliance

Key Management

Platform

Certified Kubernetes - Distribution



Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Observability and Analysis

Monitoring

Logging

Tracing

Chaos Engineering

Serverless

Members

Kubernetes Certified Service Provider

Kubernetes Training Partner

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path

l.cncf.io

CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE Landscape

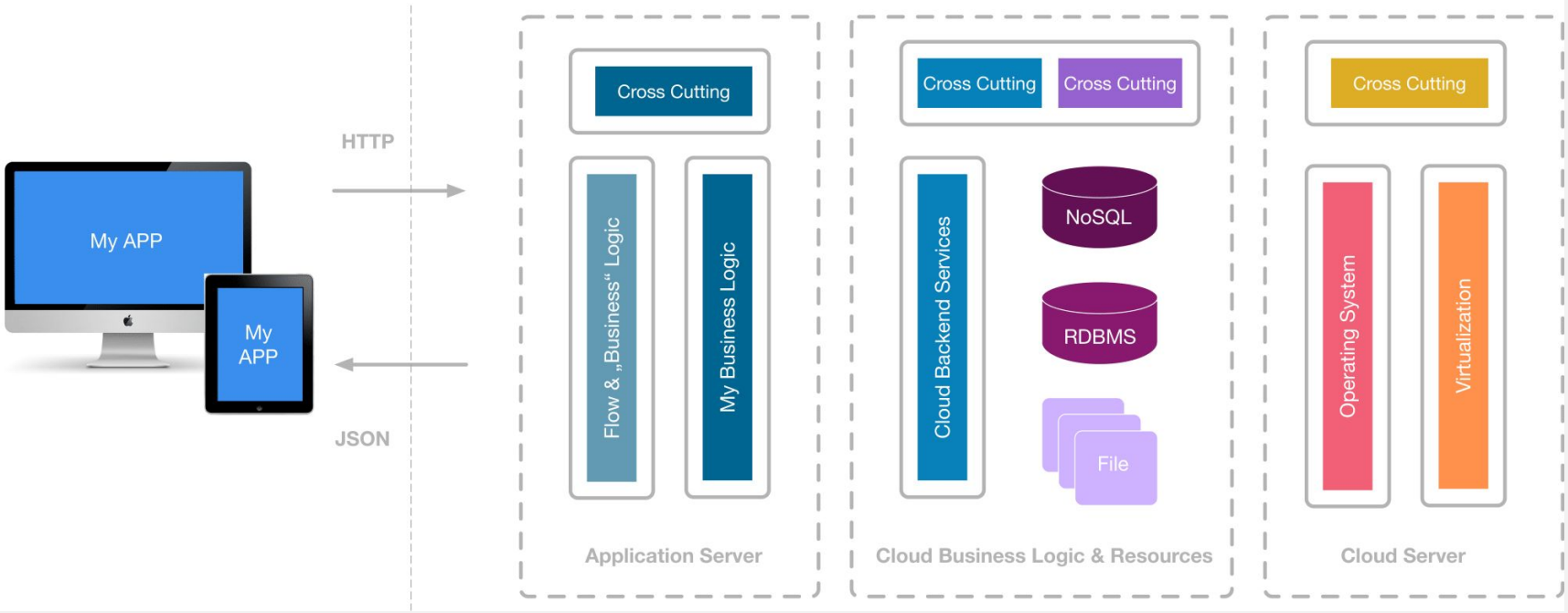
Redpoint Amplify

CLOUD NATIVE SOFTWARE DEVELOPMENT IS

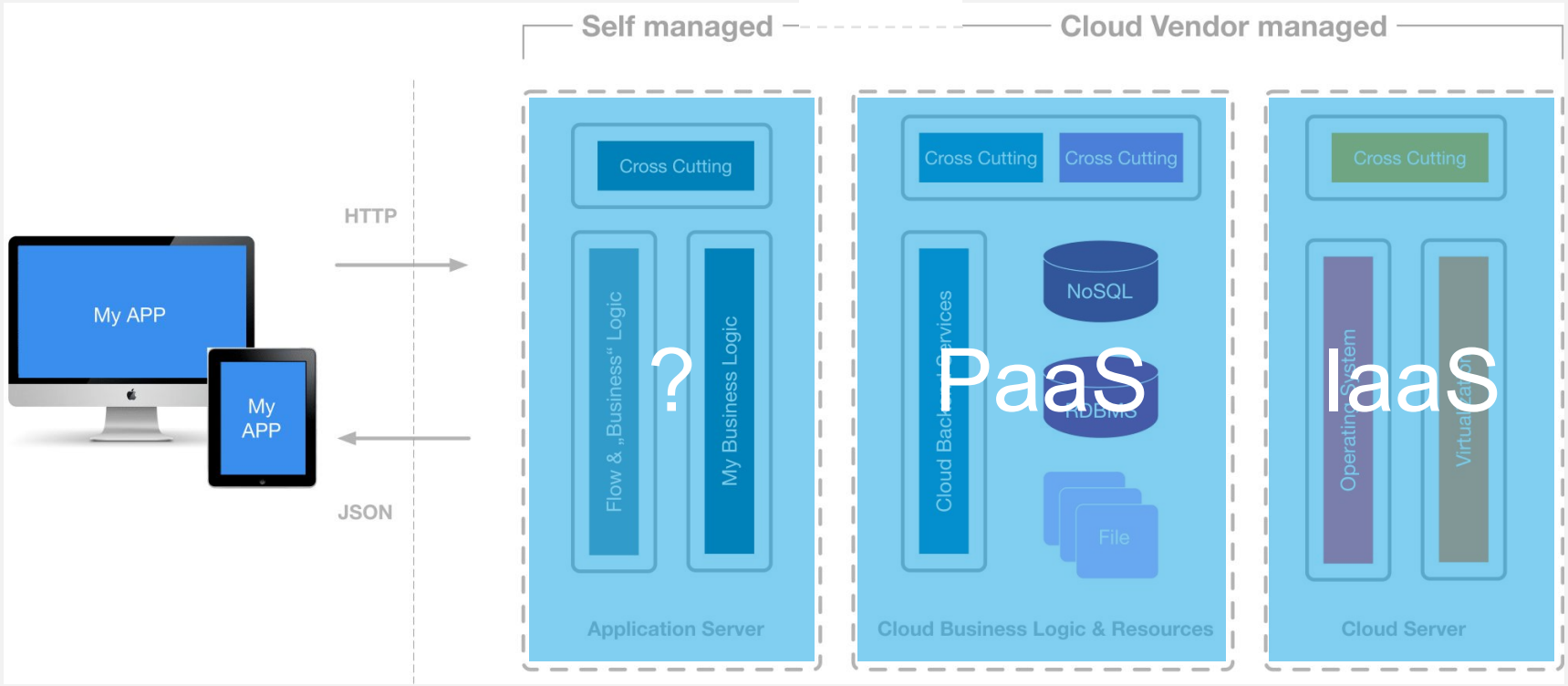
COMPLEX.

DOCKER, YAML, MICROSERVICES, KUBERNETES, ET.AL.

Traditionelle Cloud-basierte Anwendungsarchitektur



Traditionelle Cloud-basierte Anwendungsarchitektur



Kein Server ist einfacher zu verwalten, als kein Server!

Werner Vogels, CTO, Amazon

Bild: pavlinec – gettyimages.de

Serverless computing refers to a new model of cloud native computing,

https://landscape.cncf.io/

Serverless computing refers to a new model of cloud native computing, enabled by architectures that do not require server management to build and run applications.

https://landscape.cncf.io/

Serverless computing refers to a new model of cloud native computing, enabled by architectures that do not require server management to build and run applications. It leverages a finer-grained deployment model

https://landscape.cncf.io/

Serverless computing refers to a new model of cloud native computing, enabled by architectures that do not require server management to build and run applications. It leverages a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform

https://landscape.cncf.io/

Serverless computing refers to a new model of cloud native computing, enabled by architectures that do not require server management to build and run applications. It leverages a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.

https://landscape.cncf.io/

Serverless Computing - Definition

Serverless computing is a cloud computing execution model in which the cloud provider dynamically manages the allocation of machine resources. Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity.[1] It is a form of utility computing.

Serverless computing still requires servers, hence it's a misnomer.[1] The name "serverless computing" is used because the server management and capacity planning decisions are completely hidden from the developer or operator. Serverless code can be used in conjunction with code deployed in traditional styles, such as microservices. Alternatively, applications can be written to be purely serverless and use no provisioned services at all.[2]

wikipedia.org

Serverless computing allows you to build and run applications and services without thinking about servers. Serverless applications don't require you to provision, scale, and manage any servers. You can build them for nearly any type of application or backend service, and everything required to run and scale your application with high availability is handled for you.

amazon.com

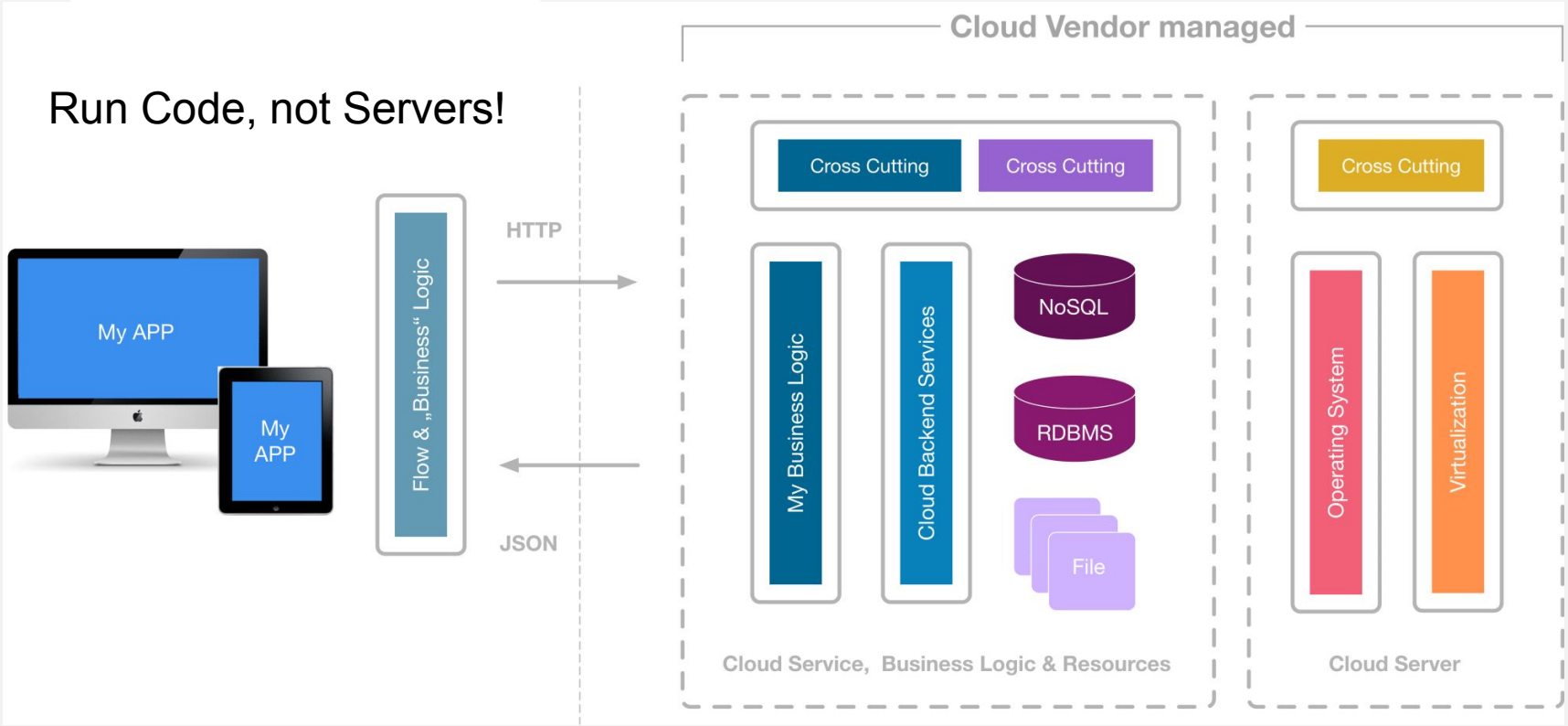
Serverless Computing – Überblick & Vergleich mit PaaS

- Serverless Computing wird häufing auch als Function as a Service (**Faas**) bezeichnet
- FaaS ist eine Spezialform von PaaS
- Deployment und Betrieb wird vom Cloud Betreiber durchgeführt. Hier ähnelt eine FaaS Plattform PaaS
- Ein Unterschied zu ‚klassischen' PaaS Platformen:
  - Der Betreiber garantiert nicht, dass eine einzelne Funktion ständig deployed ist. Häufig wird diese bei Bedarf erst geladen / deployed.
  - Es entfällt die feingranulare Administration einer PaaS.
  - Entwickler müssen sich nicht um die Laufzeitumgebung (bau des Containers, o.ä.) kümmern

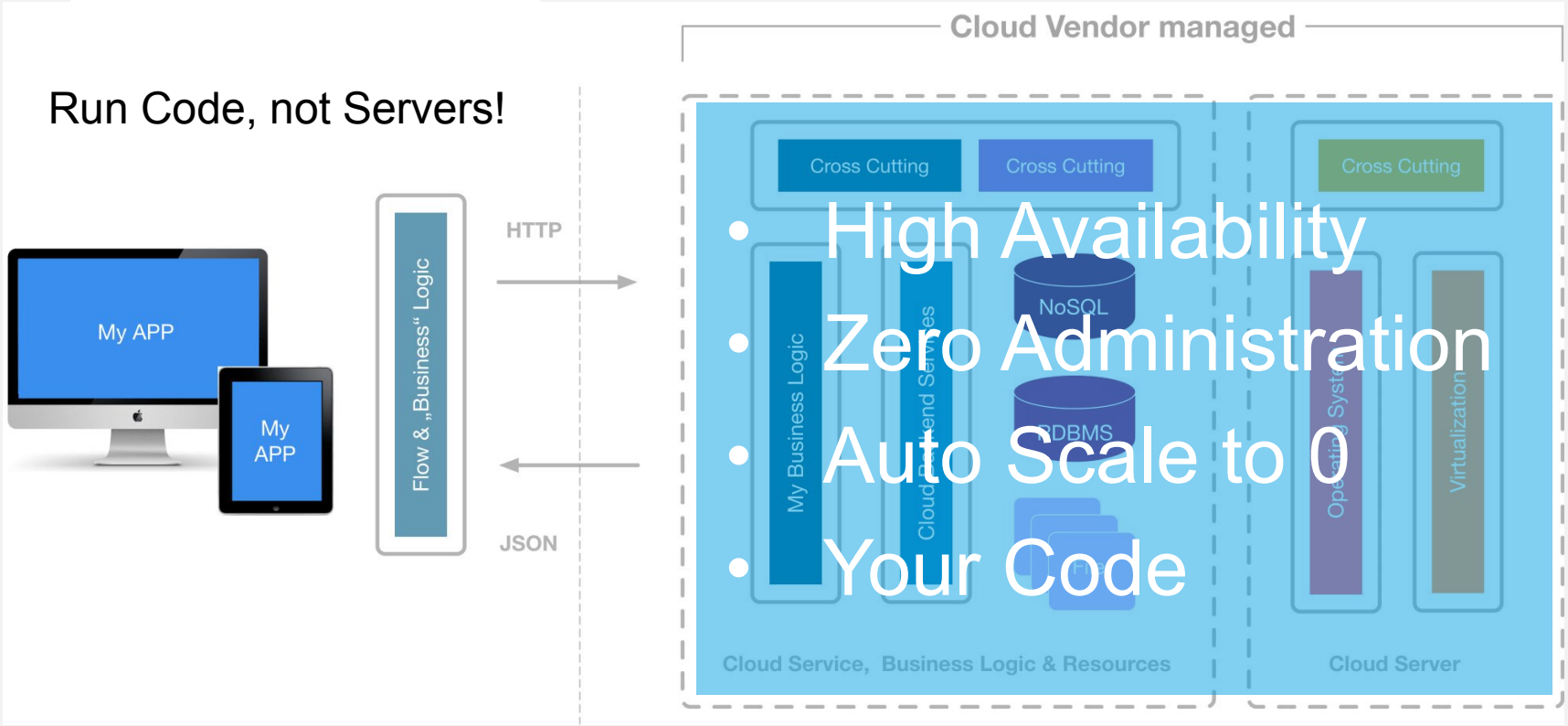
- Der primäre Architekturstil von FaaS ist Ereignisgetriebene Architektur (Event-driven Architecture / EDA)
- Die größten Anbieter sind Amazon mit AWS Lambda, Microsoft mit Azure Functions und Google mit App Engine :

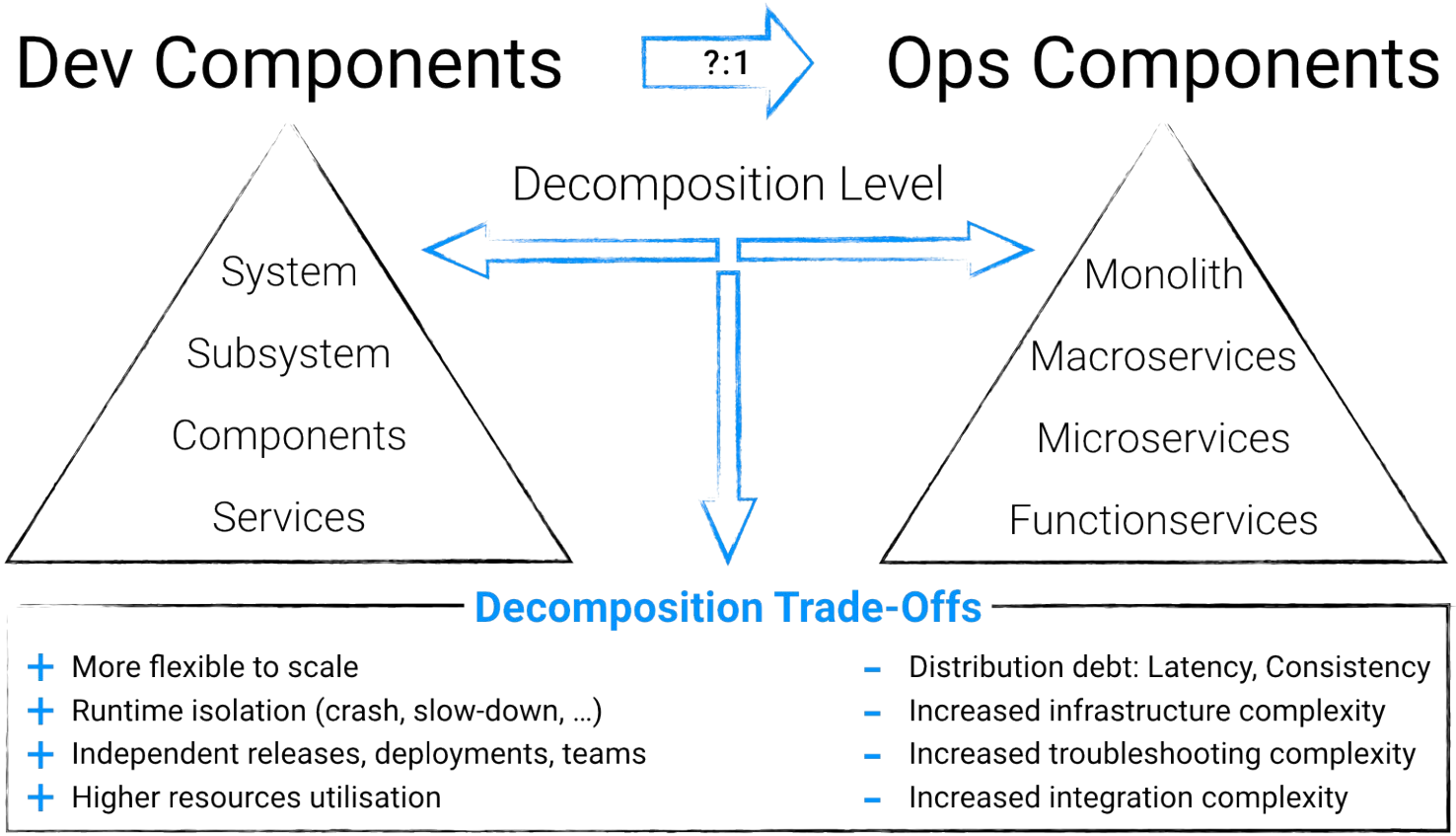


Serverless Anwendungsarchitektur



Serverless Anwendungsarchitektur





Decomposition Trade-Offs

- \+ More flexible to scale
- \+ Runtime isolation (crash, slow-down, …)
- \+ Independent releases, deployments, teams
- \+ Higher resources utilisation
- – Distribution debt: Latency, Consistency
- – Increased infrastructure complexity
- – Increased troubleshooting complexity
- – Increased integration complexity

Serverless Computing – Vor- und Nachteile

Vorteile:

- Kosten: Da einzelne Funktionen nur bei Benutzung deployed werden ist dies oft kosteneffektiver, als Server ständig zu betreiben
- Produktivität: Einzelne Funktionen können sehr schnell geschrieben, deployed und aktualisiert werden.
- Performance: Einzelne Funktionen können sehr feingranular skaliert werden.

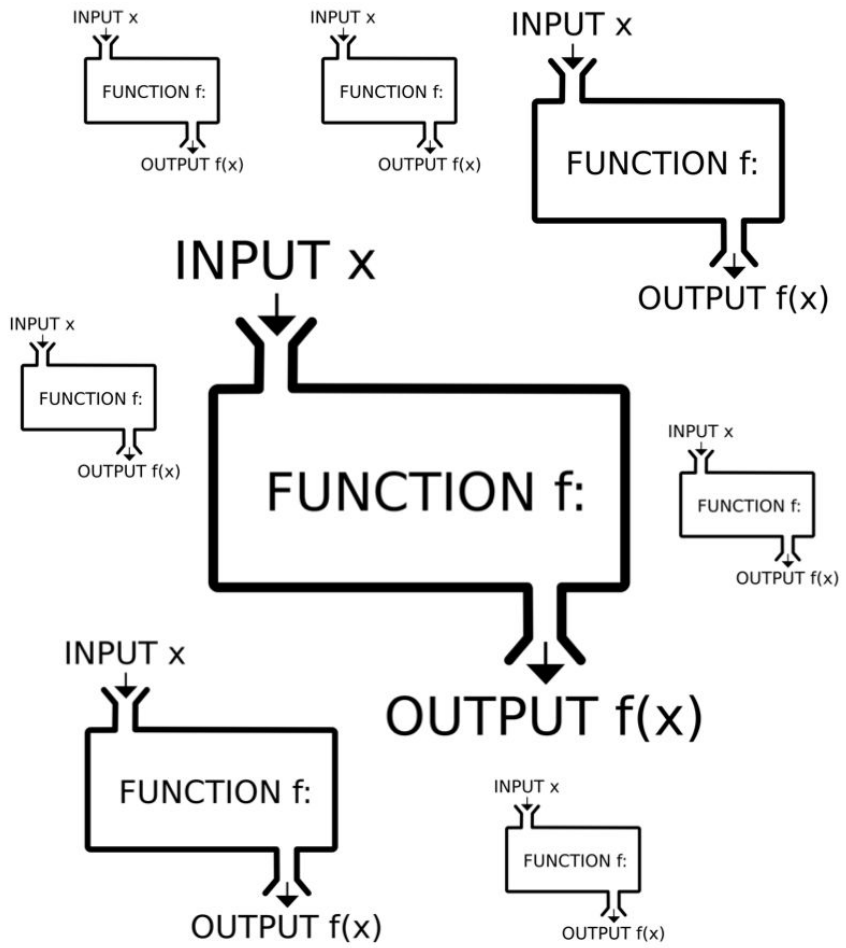
Nachteile:

- Performance: Da einzelne Funktionen evtl. erst bei Bedarf geladen werden, können starke Schwankungen bei der Ausführung auftreten.
- Debugging: Außer Fehlermeldungen und Log-Output, hat man weniger Möglichkeiten zur Diagnose. Dies erschwert das Debugging / Profiling der Anwendung.

Functions

as preferred Serverless Application

Programming Model

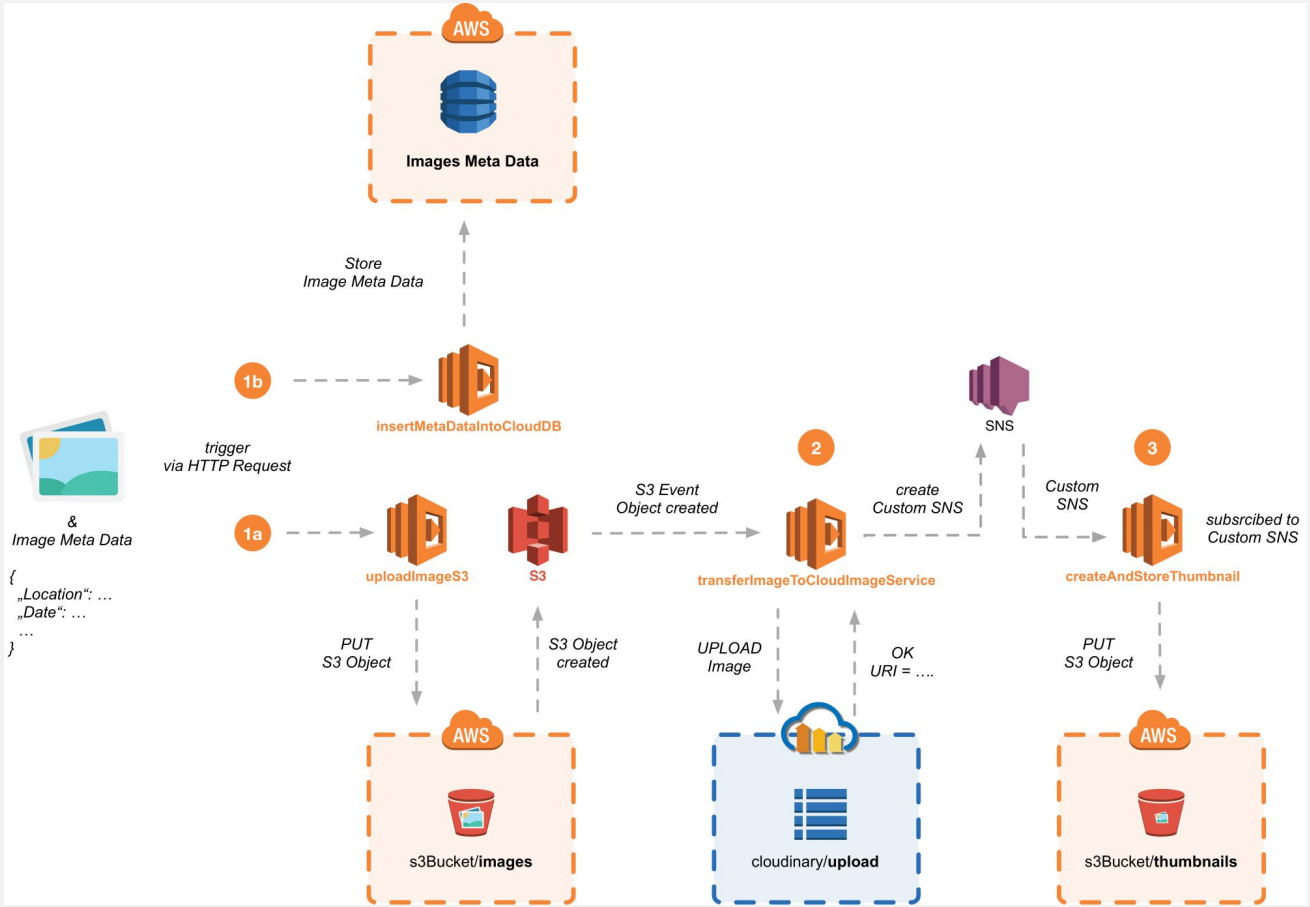


EVENT-DRIVEN ARCHITECTURE

enables loosly coupled reactive

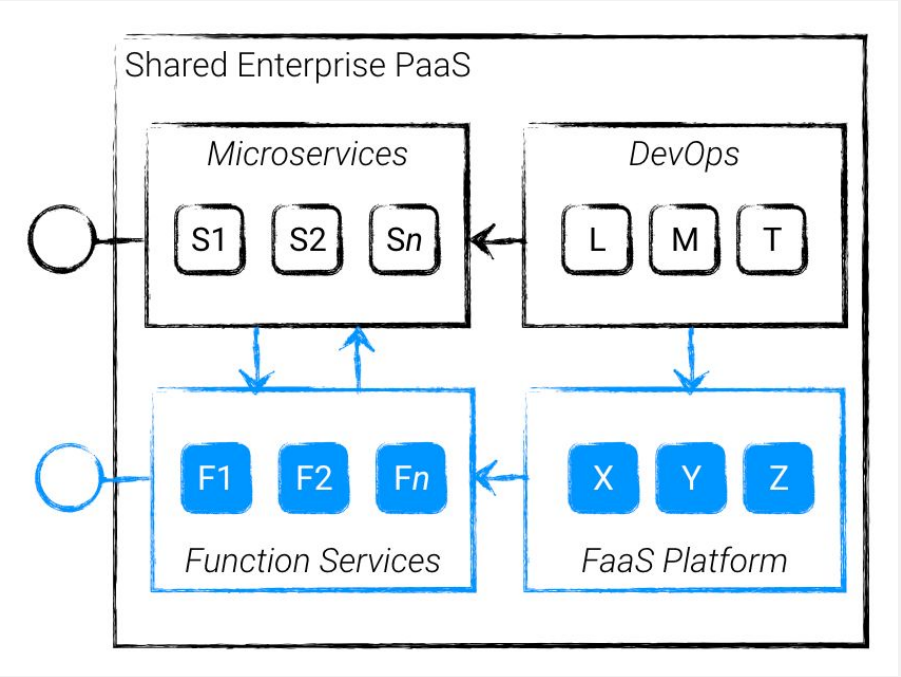
software components and services.

Create Thumbnails the AWS Lambda Way



Use Case 1
Hybrid Architectures

- Kombination von Microservice Architektur mit EDA
- Nutzung von Function Services für Event-getriebene Use Cases
- Reduzierter Ressourcen-Verbrauch per Scale-to-Zero
- Integration in bestehende Enterprise PaaS Umgebung



Use Case 2
Edge und Fog Computing

- Anbindung von Raum-Sensoren mittels Serverless Backend
- Couch Projekt: Nutzung von FaaS auf Low Power Devices
- Unterstützung von leichtgewichtigen Cluster Scheduler wie Docker Swarm

Self-hosted Serverless

- *Scale to zero*
- *React to events*

CNCF Serverless Landscape

Tools

Security

Framework

dapr — CNCF Incubating

Hosted Platform

Installable Platform

KEDA — CNCF Incubating

Knative — CNCF Incubating

s.cncf.io

Serverless computing refers to a new model of cloud native computing, enabled by architectures that do not require server management to build and run applications. This landscape illustrates a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment

CLOUD NATIVE Landscape

CLOUD NATIVE COMPUTING FOUNDATION

Redpoint

Cloud Native Landscape

OPENFAAS

fission

~~Kubeless~~

nuclio

Kyma

siehe auch https://bit.ly/2Mh1kxJ

Die Kandidaten

- OpenFaas
  https://www.openfaas.com
- Fission
  https://fission.io
- ~~Kubeless~~
  ~~https://kubeless.io~~
- Nuclio
  https://nuclio.io
- Knative
  https://knative.dev/
- Kyma
  https://kyma-project.io

	LANGUAGE	USE CASES	GENERATION	PLATFORMS	RUNTIMES	TRIGGERS
FISSION	GO	ENTERPRISE	2ND	K8S	GO, PYTHON, NODEJS, JAVA/JVM	CRON, HTTP, NATS, AZURE QUEUE STORAGE, KAFKA, KUBEWATCH
KUBELESS	GO	ENTERPRISE	2ND	K8S	NODEJS, JAVA, GO, JVM, PYTHON, PHP, RUBY, .NET CORE, BALLERINA, VERTX	CRON, HTTP, NATS, KINESIS, KAFKA
OPENFAAS	GO	ENTERPRISE, IOT	1ST	K8S, DOCKER	GO, C#, JAVA8, DOCKERFILE, NODEJS, PHP, PYTHON, RUBY	HTTP, CRON, KAFKA, AWS SNS, S3, CLOUDEVENTS, IFTTT, REDIS, MQTT, NATS
NUCLIO	GO	ENTERPRISE, IOT	2ND	DOCKER, K8S, AWS, GCP	.NET CORE, GO, JAVA, NODEJS, PYTHON, SHELL	CRON, EVENTHUB, HTTP, KAFKA, KINESIS, NATS, RABBITMQ, MQTT
OPENWHISK	SCALA	ENTERPRISE, HOSTED?	2ND	K8S, MESOS, DOCKER, OPENSHIFT	NODEJS, SWIFT, JAVA, GO, SCALA, PYTHON, PHP, RUBY, .NET CORE, BALLERINA	CLOUDANT, RSS, KAFKA, JIRA, BLUEMIX PUSH, SLACK, GITHUB
FN PROJECT	GO	ENTERPRISE, HOSTED?	1ST	DOCKER, K8S	JAVA, GO, NODEJS, PYTHON, RUBY	HTTP

IT DEPENDS ON YOUR USE CASE.

- FISSION IS A PRETTY COMPLETE PLATFORM.
- OPENFAAS IS VERY POPULAR WITH AN ACTIVE COMMUNITY. CURRENTLY THE ONLY ONE WITH SUPPORT FOR ARM DEVICES.
- NUCLIO IS FAST. LIGHTWEIGHT AND HAS SUPPORT FOR MANY TRIGGERS. PROMISING ROADMAP.
- KUBELESS ~~IS~~ *was* LIGHTWEIGHT AND SIMPLE.

Beispiel AWS Lambda

- Lambda ist ein AWS Service. Er wurde 2014 eingeführt.
- Zielgruppe: Vereinfachtes bauen von on-demand Applikationen.
- Ursprüngliches Ziel war die einfache Umsetzung von Use-Cases, wie z.B. Image-Upload in die AWS Cloud
- Unterstützte Sprachen:
  - Node.js
  - Python
  - Java
  - C#
  - Go
- AWS-Lambda Funktionen werden in Inkrementen von 100ms abgerechnet. Die EC2 dagegen wird in Stunden abgerechnet.