qaware.de

Hochschule Rosenheim
University of Applied Sciences

QA|WARE
SOFTWARE ENGINEERING

Serverless Computing

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

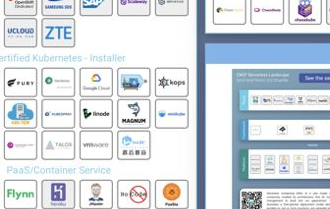
Key Management

Platform

Certified Kubernetes - Distribution

Certified Kubernetes - Hosted

Certified Kubernetes - Installer



PaaS/Container Service

Observability and Analysis

Monitoring

Logging

Tracing

Chaos Engineering

Serverless

Members

Special

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

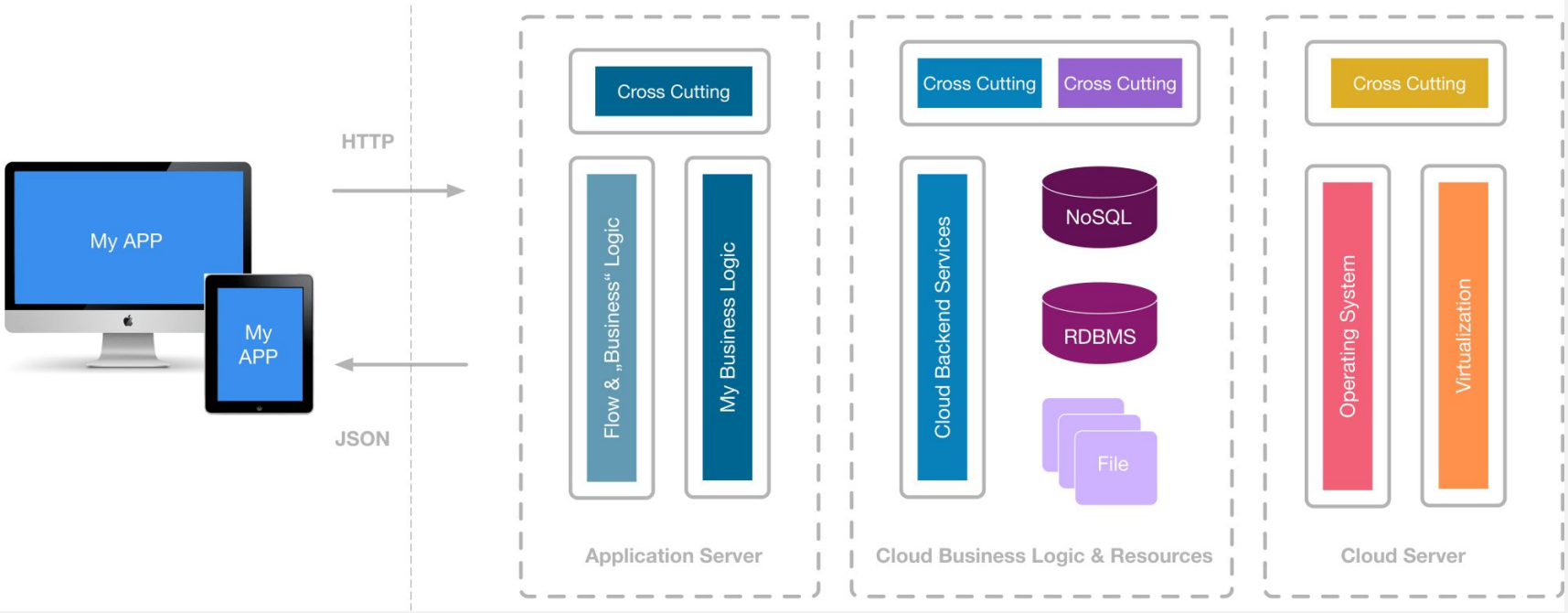
Redpoint Amplify

CLOUD NATIVE SOFTWARE DEVELOPMENT IS

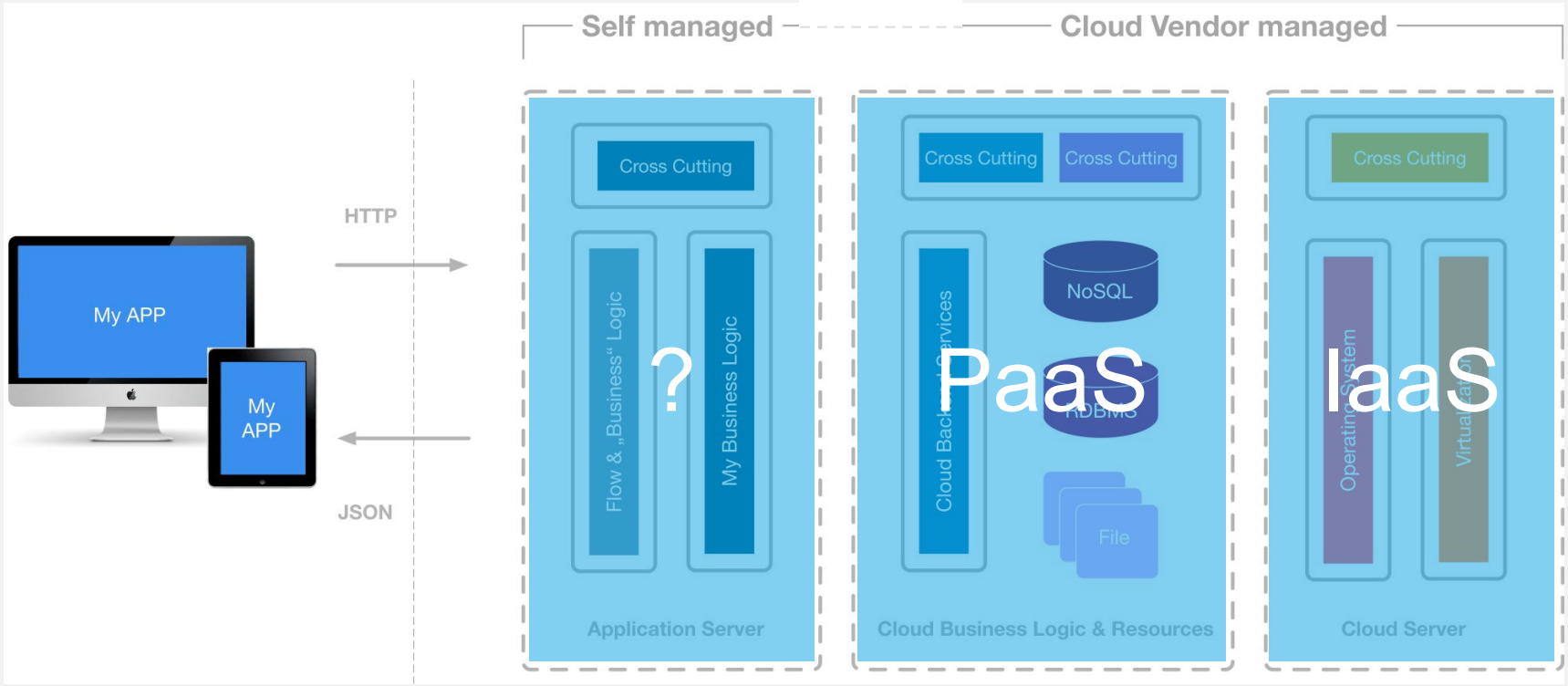
COMPLEX.

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

Traditional cloud-based architecture



Traditional cloud-based architecture



No server is easier to manage than no 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

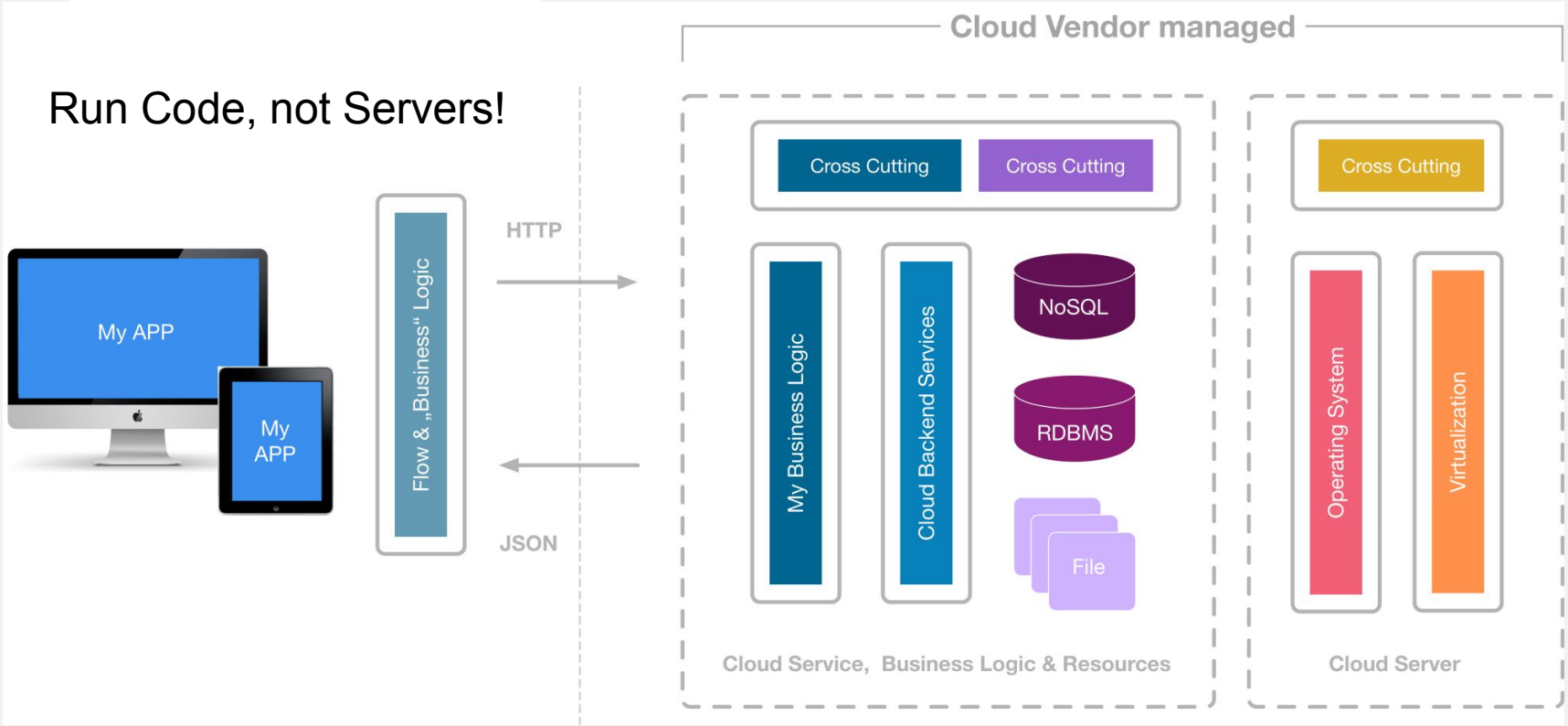
Serverless computing allows you to build and run applications and services without thinking about servers. Serverless applications don't require you to provision, scale, and manage any servers. You can build them for nearly any type of application or backend service, and everything required to run and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

amazon.com

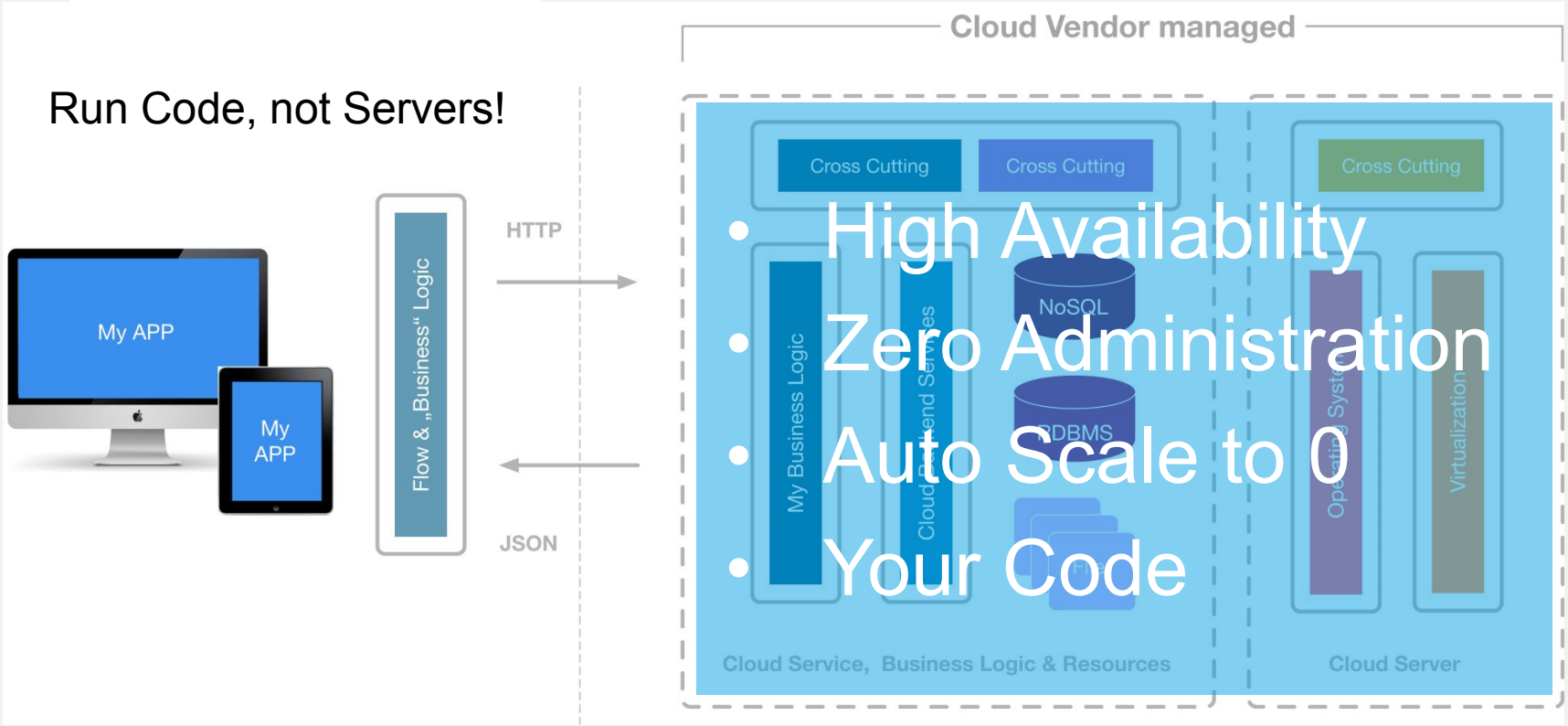
Serverless Computing – Overview & Comparison to PaaS

- Serverless computing is often referred to as Function as a Service (FaaS).
    - FaaS is a specialized form of PaaS.
    - Deployment and operations are handled by the cloud provider, making a FaaS platform similar to PaaS.
- A key difference from 'traditional' PaaS platforms:
    - The provider does not guarantee that a single function is constantly deployed. Often, the function is only loaded/deployed when needed.
- Fine-grained administration of a PaaS is unnecessary.
- Developers in general do not need to worry about the runtime environment (e.g., building containers). However, not true for all languages and runtimes.
- The primary architectural style of FaaS is event-driven architecture (EDA).
  The largest providers are Amazon with AWS Lambda, Microsoft with Azure Functions, and Google with App Engine.

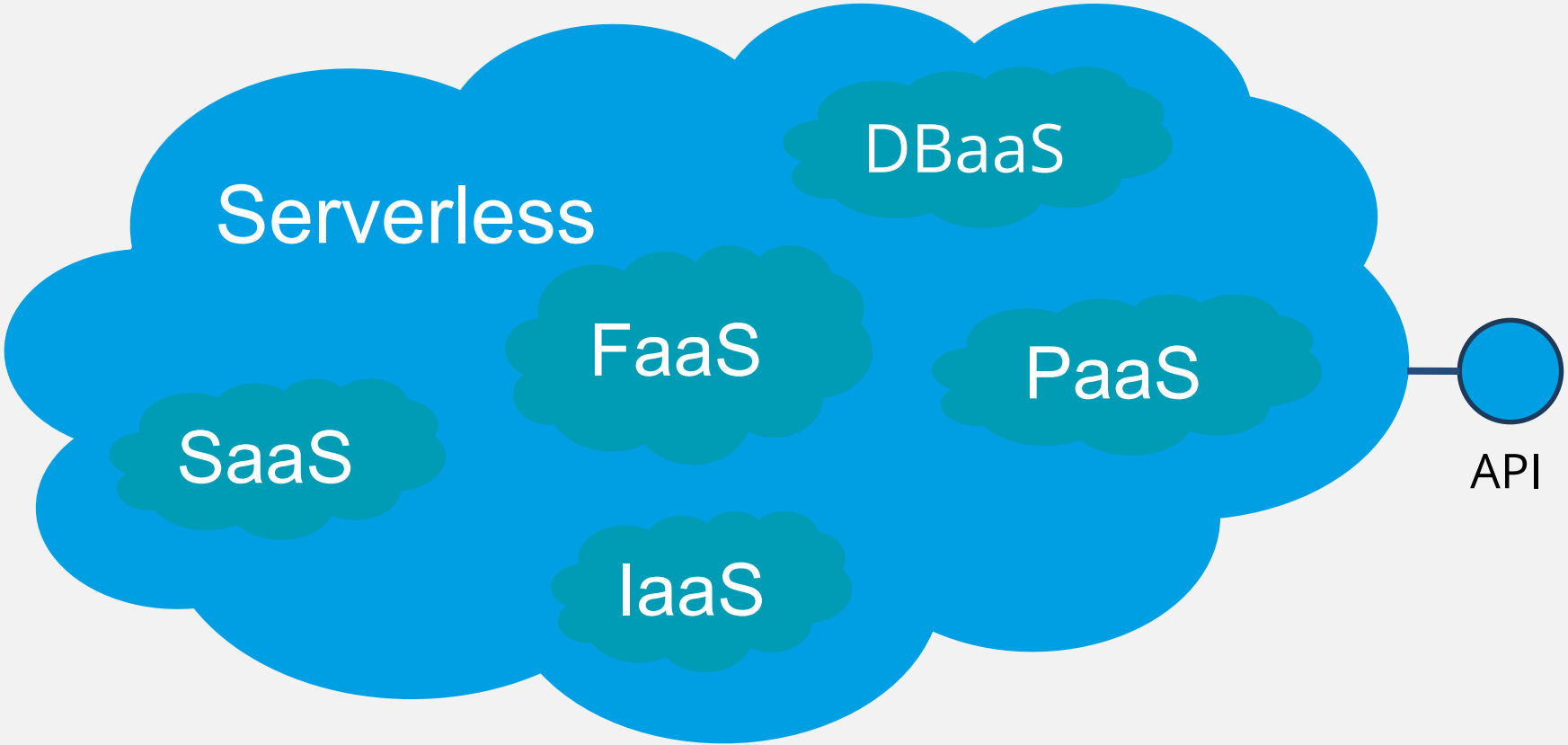
Serverless application architecture

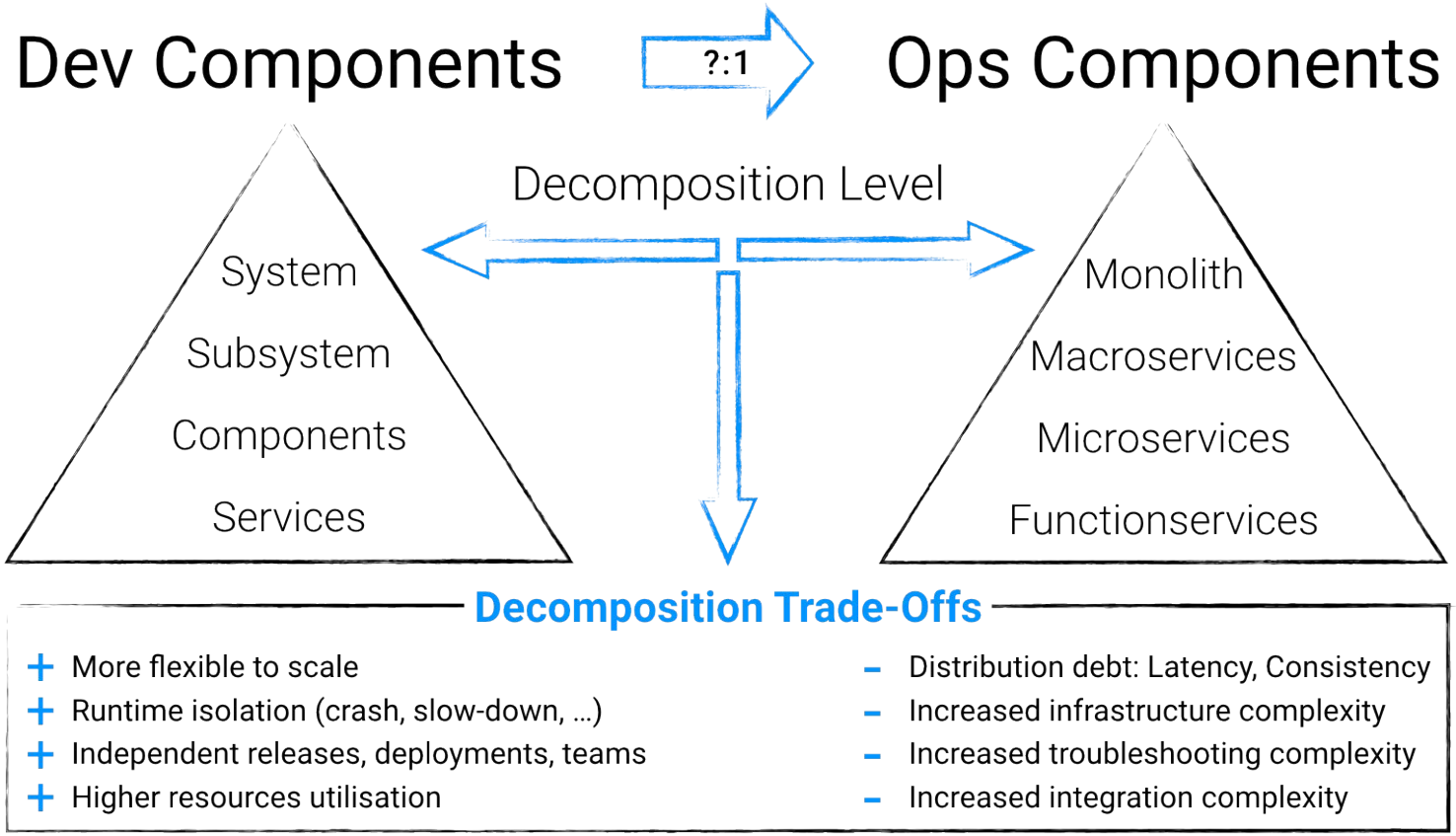


Serverless application architecture



Out-of the Box Self-scaling Fully Managed Backend





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 – Advantages & Disadvantages

Advantages:

- **Cost:** Since individual functions are only deployed when used, this is often more cost-effective than running servers continuously.
- **Productivity:** Individual functions can be written, deployed, and updated very quickly.
- **Performance:** Individual functions can be scaled very granularly.

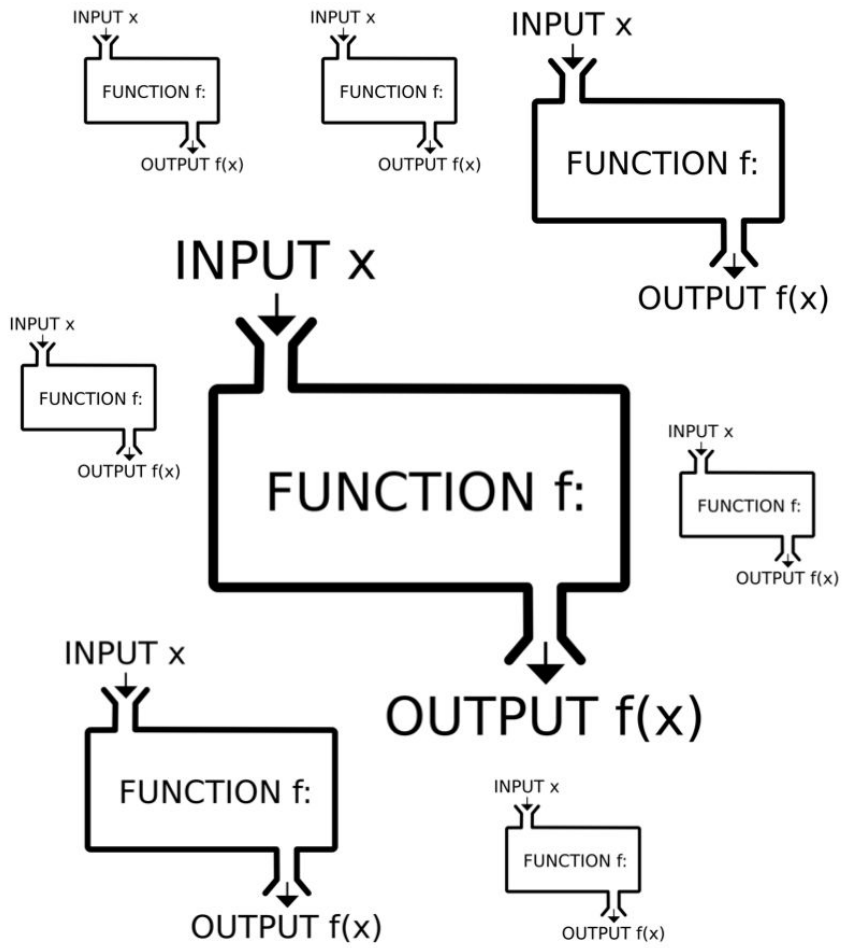
Disadvantages:

- **Performance:** Since functions may only be loaded on demand, significant fluctuations in execution time can occur.
- **Debugging:** Beyond error messages and log output, there are fewer diagnostic options available, making debugging and profiling the application more difficult.

Functions

as preferred Serverless Application

Programming Model

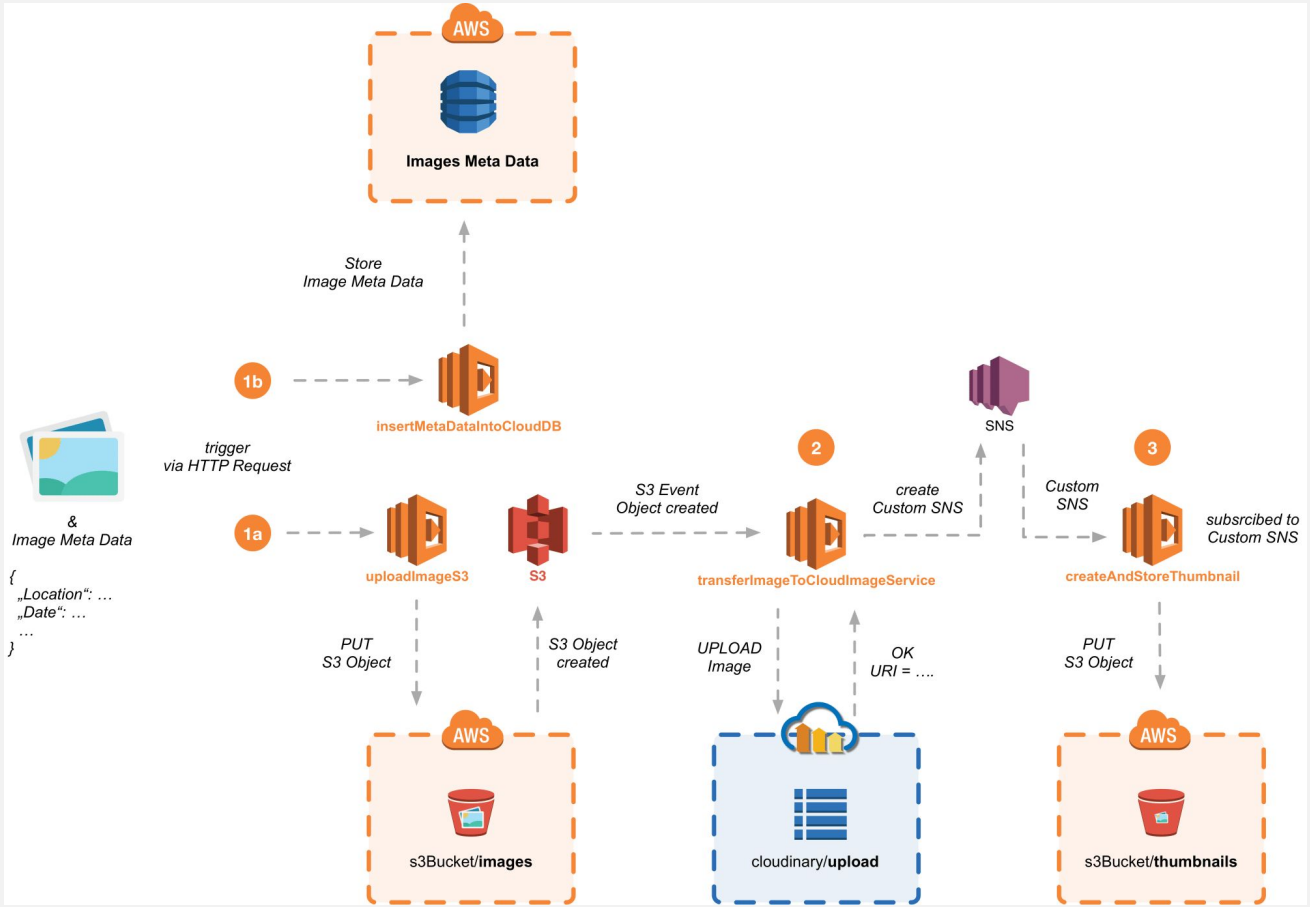


EVENT-DRIVEN ARCHITECTURE

enables loosely coupled reactive

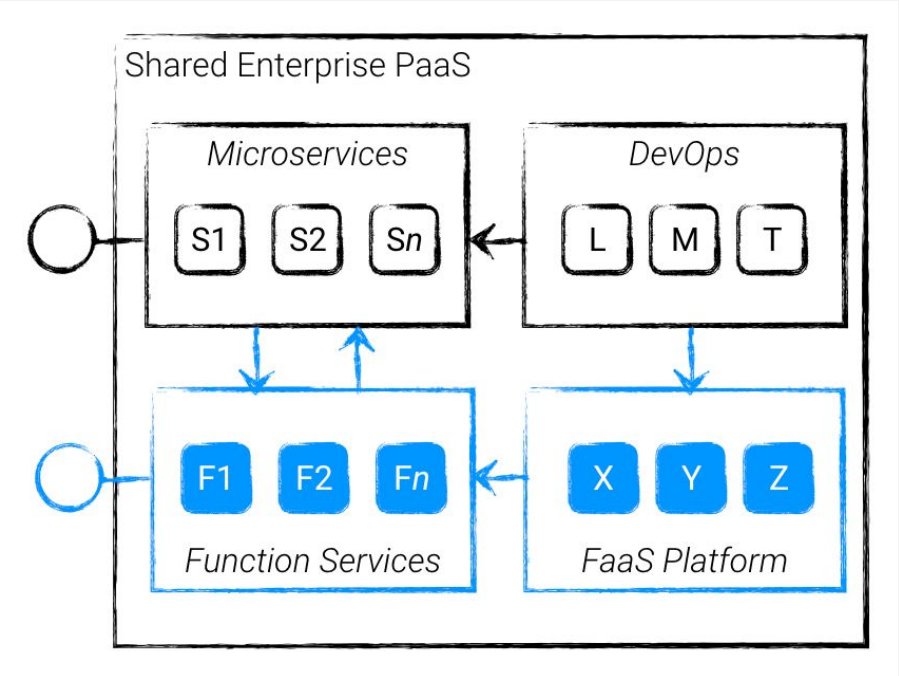
software components and services.

Create Thumbnails the AWS Lambda Way



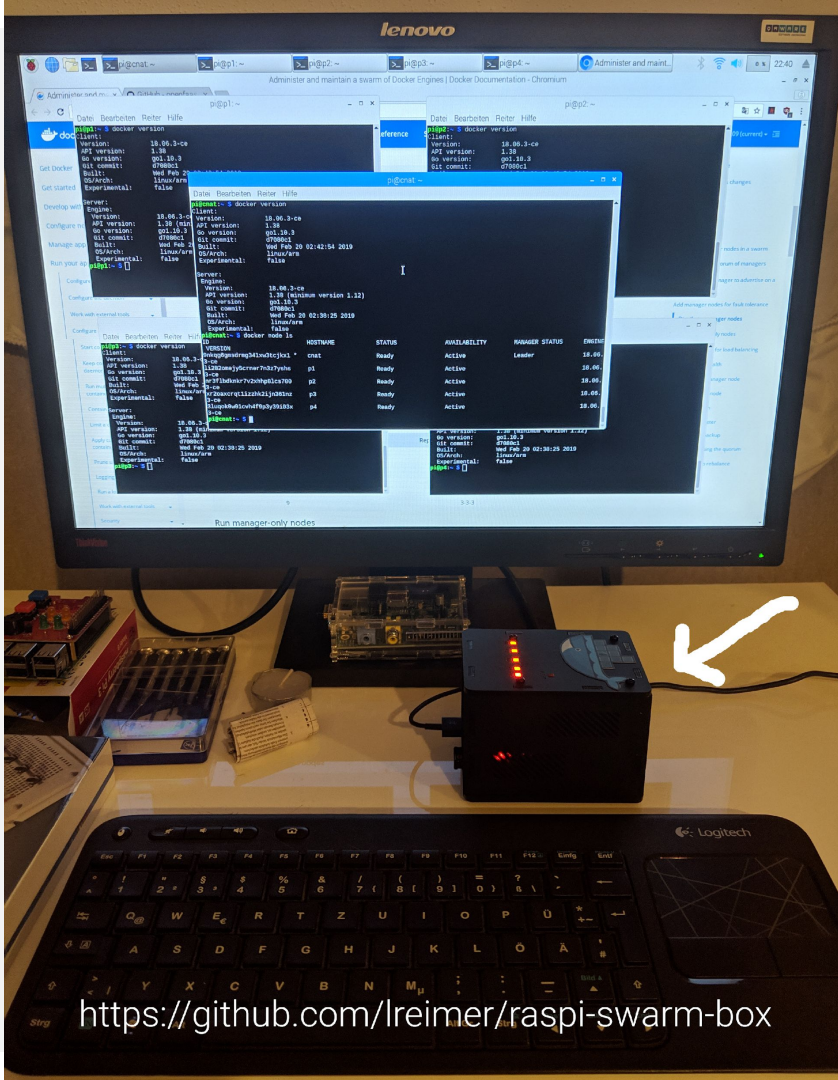
Use Cases:
Hybrid Architectures

- combination of microservice architectures with EDA
- usage of function services for event-driven use cases
- reduce resource usage by leveraging scale-to-zero
- integration into existing enterprise PaaS environments



Use Cases:

- Edge and fog computing
- Full Serverless application architecture
- etc.



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

The candidates

- 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.

Example AWS Lambda

Lambda is an AWS service introduced in 2014.

Target audience: Simplified development of on-demand applications.
The original goal was to enable easy implementation of use cases, such as image uploads to the AWS cloud.

Supported languages:

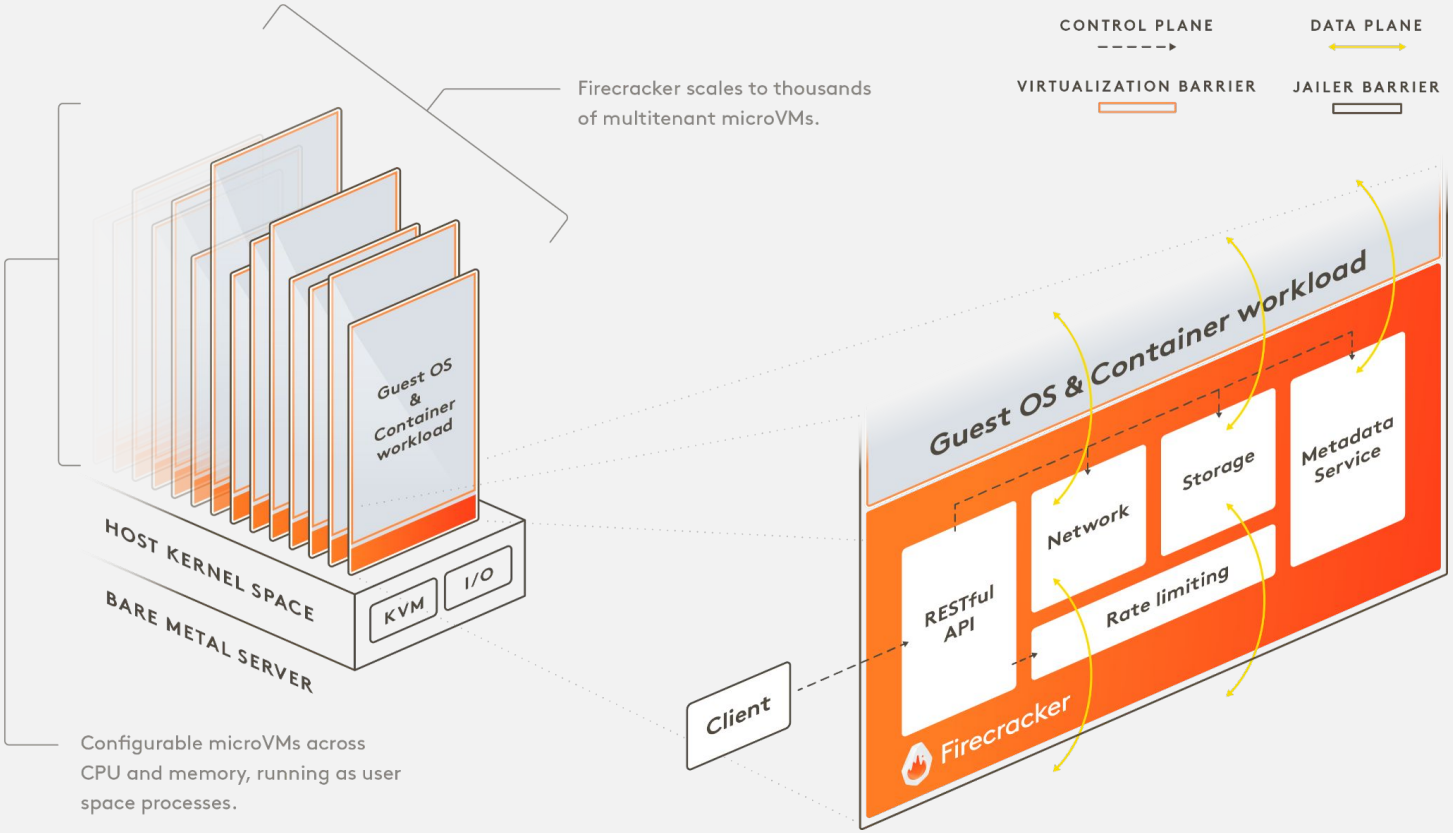
- Node.js
- Python
- Java
- C#
- Go
- etc.

AWS Lambda functions are billed in increments of 100 ms, whereas EC2 instances are billed by the hour.

Cloud runtimes

Lukas Buchner
lukas.buchner@qaware.de

Firecracker - back to VMs



Firecracker

- microVMs instead of containers
- minimal overhead, significantly better isolation
- fast startup times
- used by Amazon for AWS Lambda and Fargate

Hashicorp Nomad als Orchestrator

Advantages:

- Easy-to-use workload orchestrator
  - Containers
  - VMs
  - Binaries
- Simpler structure compared to Kubernetes
- Only one binary needs to be deployed — the same binary is used for both server and client

Disadvantages:

- Ecosystem is significantly weaker
  - No Helm
  - No GitOps, etc.
- Requires additional products for features like service discovery and secrets management (e.g., Consul and Vault)
- Features for larger enterprises require licenses

Webassembly - the runtime of the future?

QA|WARE

WEBASSEMBLY

Webassembly - Overview

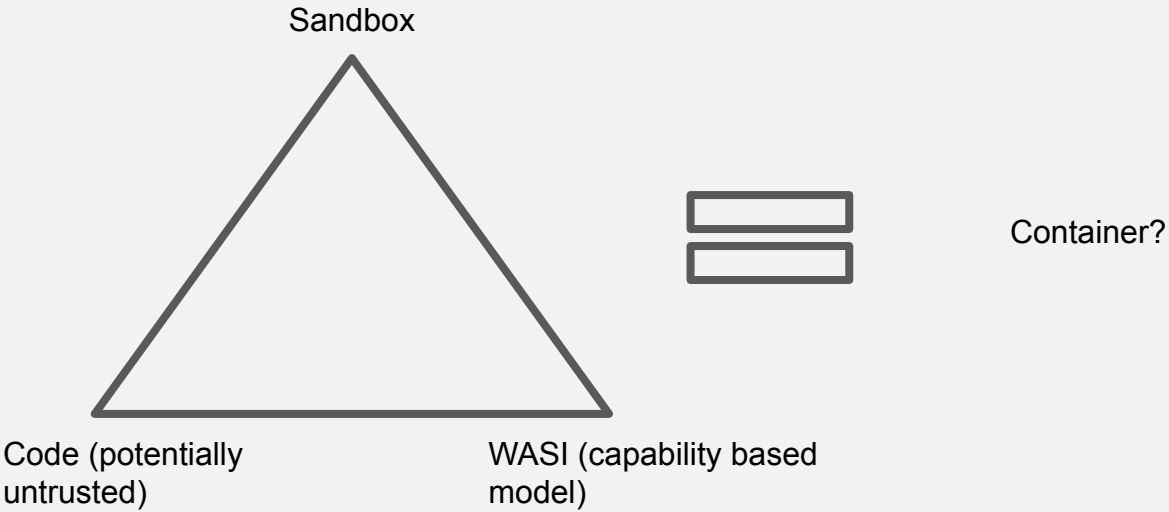
WebAssembly is a collection of standards:

- **Core Specification**: Describes the bytecode format, etc.
- **Embedding Interfaces**: Describes how WebAssembly can be embedded into various technologies.
- **WASI (WebAssembly System Interface)**: Provides a system interface description. WebAssembly programs run in a sandbox, and with WASI, access to system APIs (e.g., file or network access) can be finely controlled.

Originally developed for the web and is executable in browsers.
WebAssembly is available as a compilation target for various languages.

By default, it runs in a sandbox — a significant difference from previous attempts at generalized platforms like CLR or JVM.

Webassembly - in the Backend?



Webassembly on Kubernetes

- container runtimes do now support webassembly directly, allowing you to launch container with webassembly support in k8s
- there’s also products that build platforms around webassembly e.g. wasmcloud

Further information about webassembly on k8s:

https://wasmedge.org/docs/develop/deploy/kubernetes/kind/

https://www.cncf.io/blog/2024/03/12/webassembly-on-kubernetes-from-containers-to-wasm-part-01/