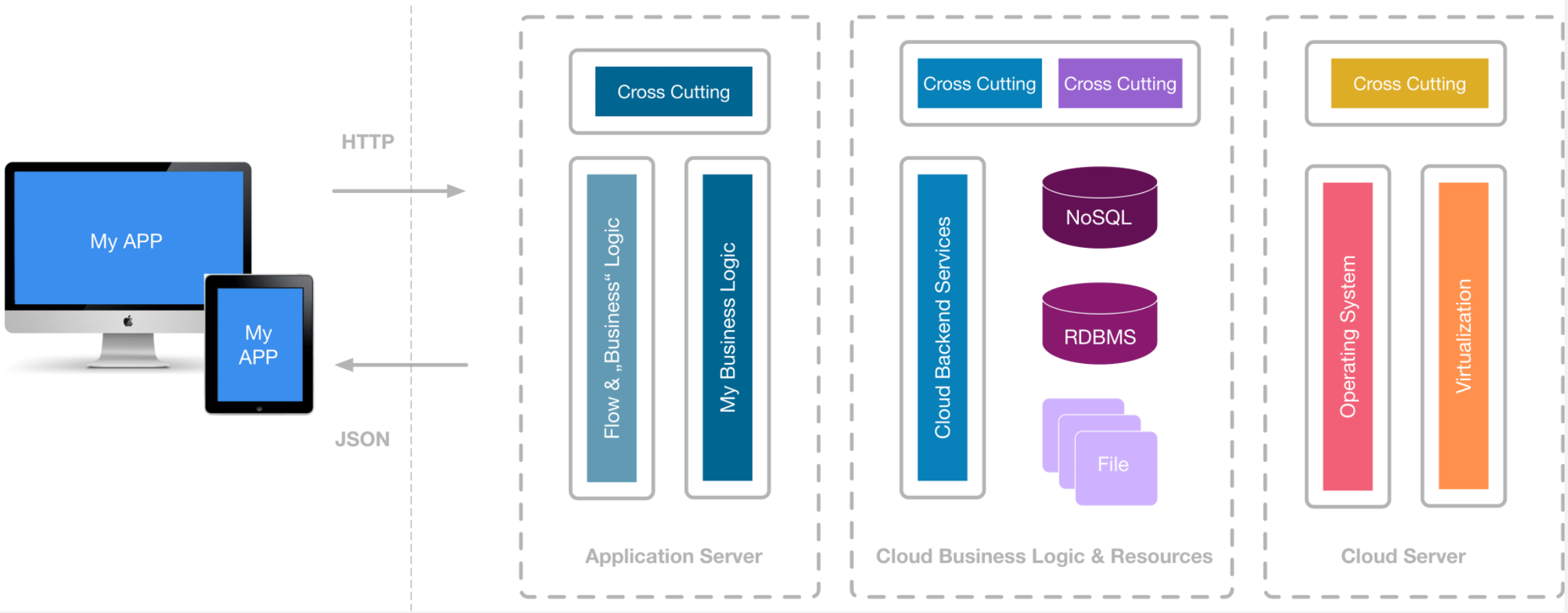
Serverless

CLOUD NATIVE SOFTWARE DEVELOPMENT IS

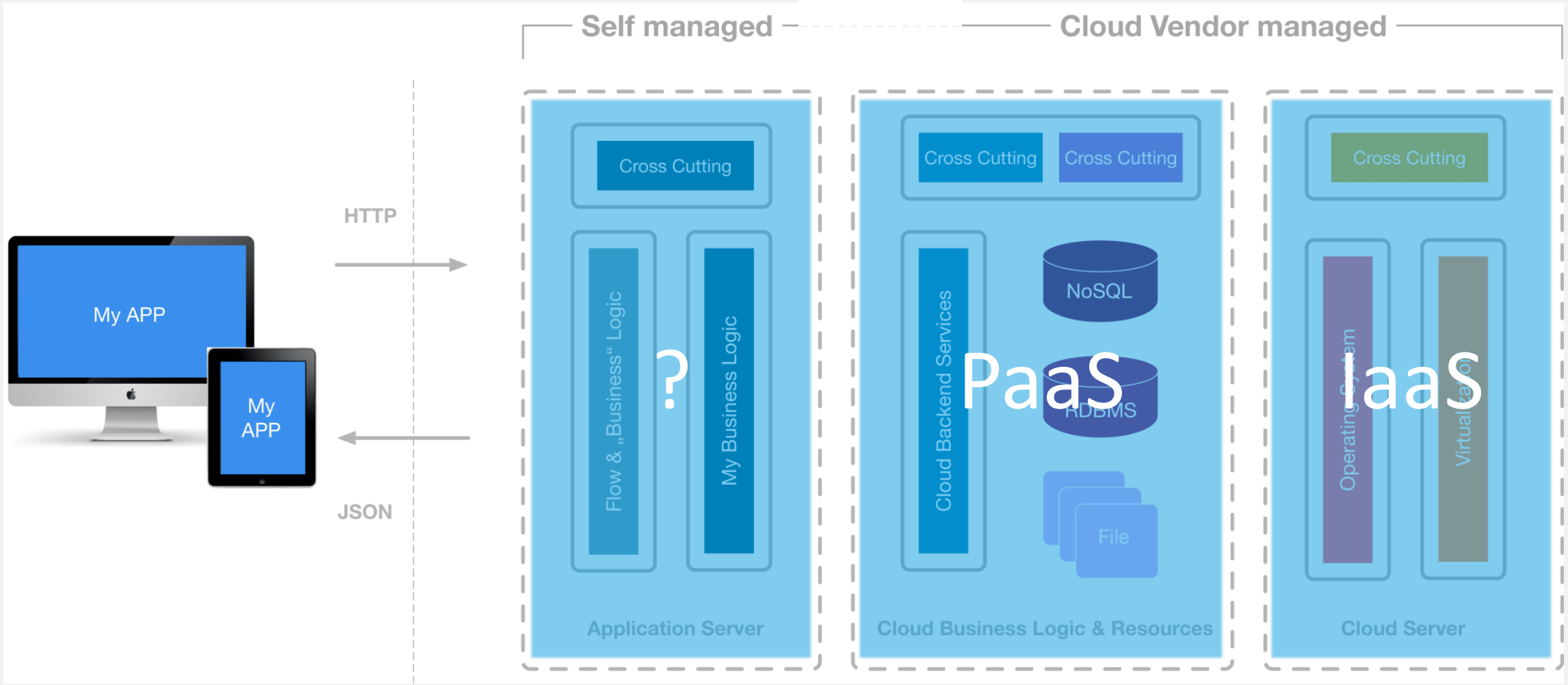
COMPLEX.

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

Traditionelle Cloud-basierte Anwendungsarchitektur



Traditionelle Cloud-basierte Anwendungsarchitektur



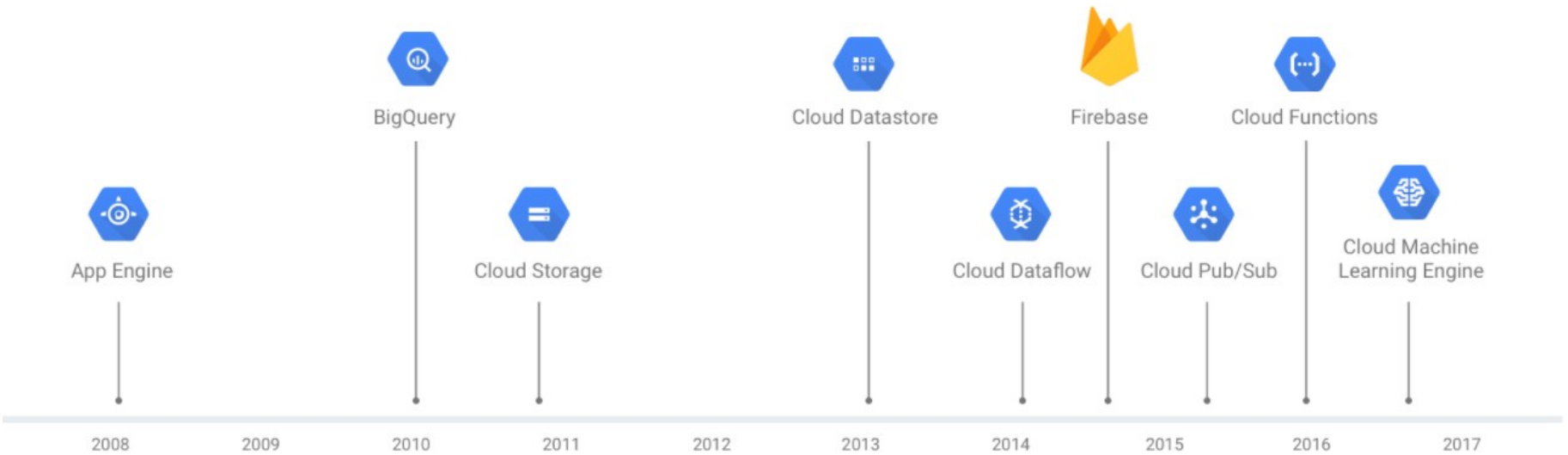
Kein Server ist einfacher zu verwalten, als kein Server!

Werner Vogels, CTO, Amazon

Bild: pavlinec – gettyimages.de

What is Serverless ?

Serverless means no upfront provisioning, no management of servers, and pay-what-you-use economics for building applications. Since the beginning of Client-Server development, IT had to manage servers for application hosting (compute) and application databases. Google Cloud has always believed in the vision of serverless by debuting with Google App Engine in 2008, the first fully serverless compute service. Since then, Google has evolved more serverless offerings in both application development and analytics.



https://cloud.google.com/serverless/whitepaper/

Serverless Computing – Überblick & Vergleich mit PaaS

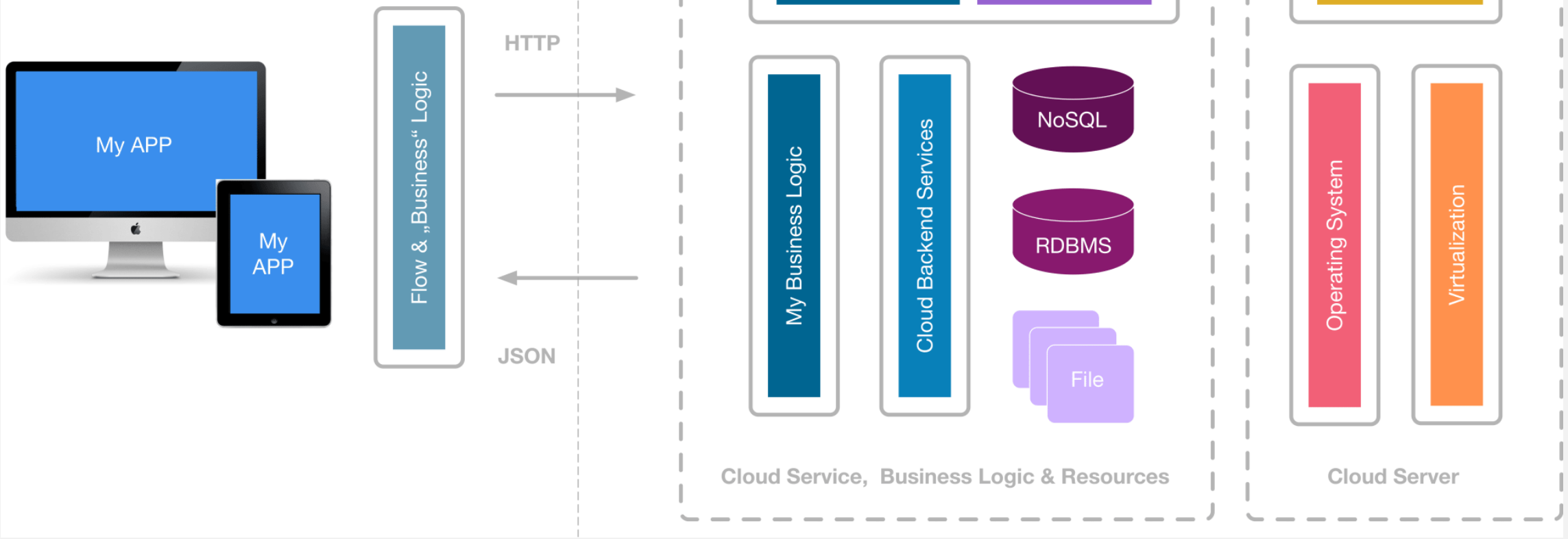
- Serverless Computing wird häufing auch als Function as a Service (**Faas**) bezeichnet
- FaaS ist eine Spezialform von PaaS
- Deployment und Betrieb wird vom Cloud Betreiber durchgeführt. Hier ähnelt eine FaaS Plattform PaaS
- Ein Unterschied zu ‚klassischen' PaaS Platformen:
  - Der Betreiber garantiert 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 Google mit Amazon mit AWS Lambda, Microsoft mit Azure Functions und Google App Engine :

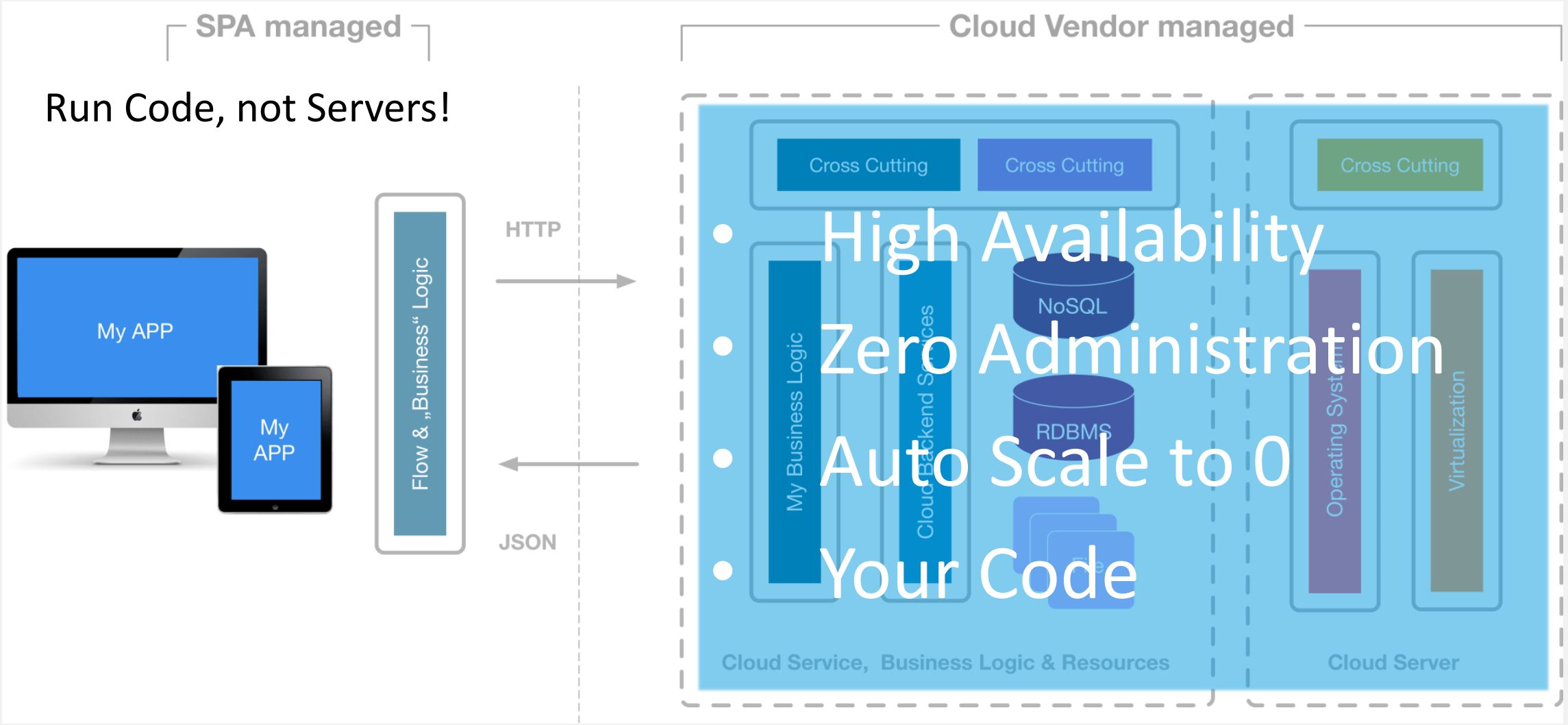


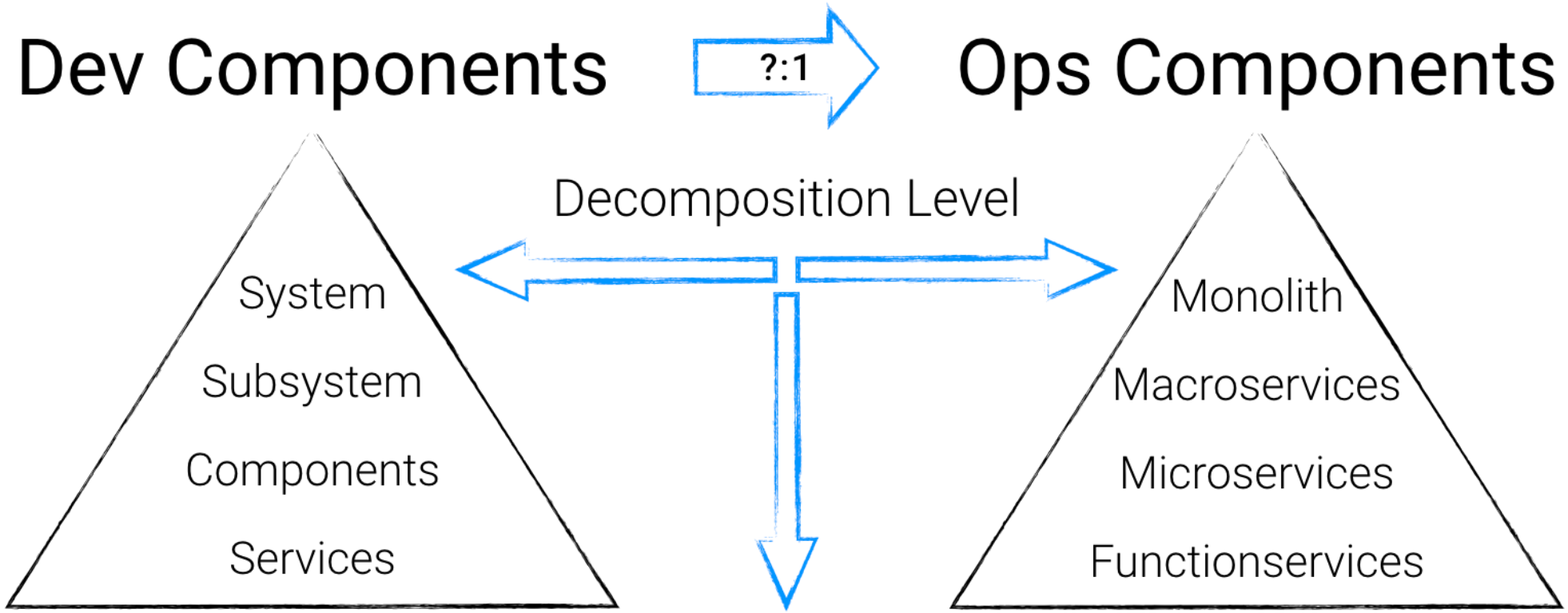
Serverless Anwendungsarchitektur

Run Code, not Servers!



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.

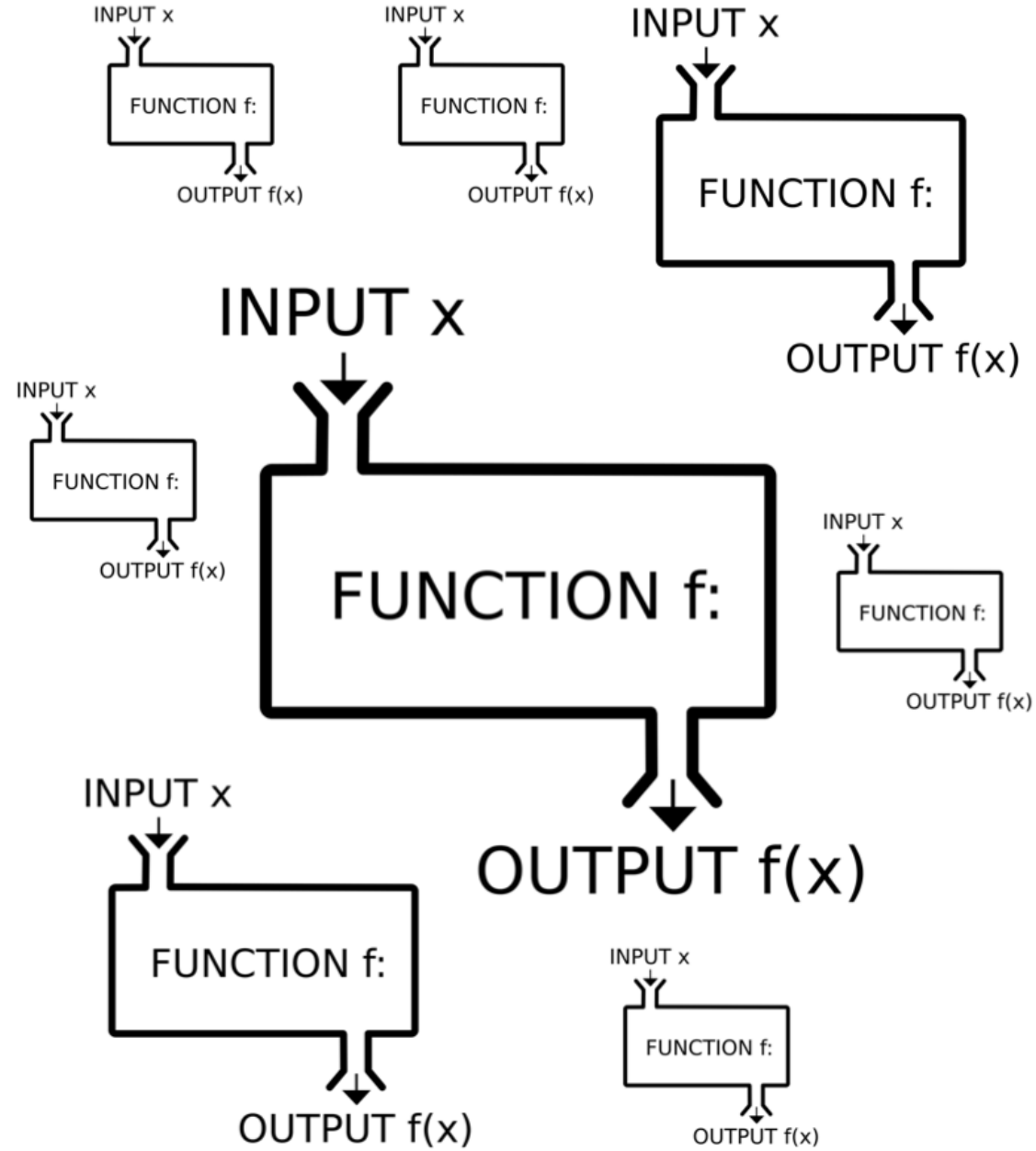
Nachteile:

- 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

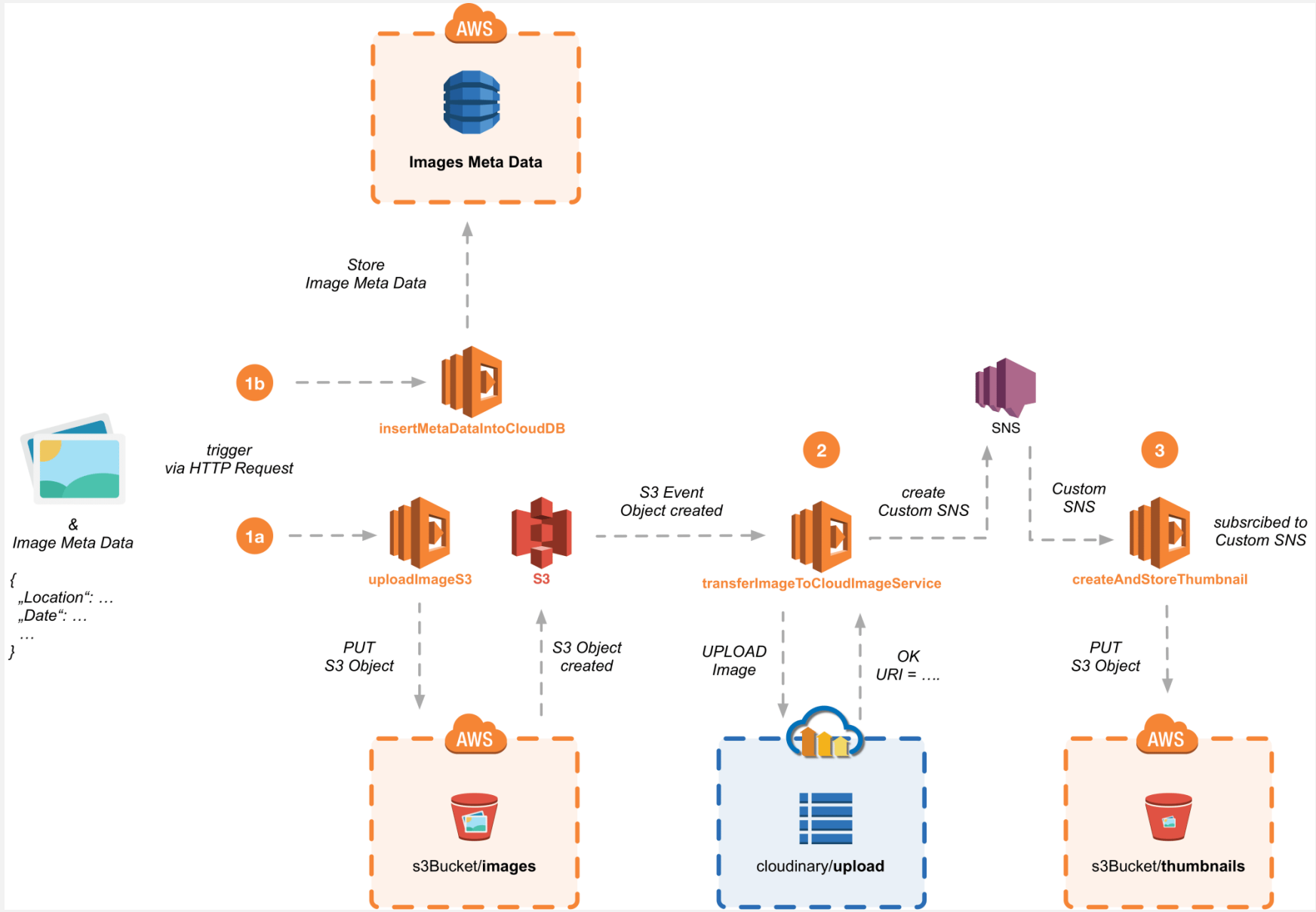
Programming Model



EVENT-DRIVEN ARCHITECTURE

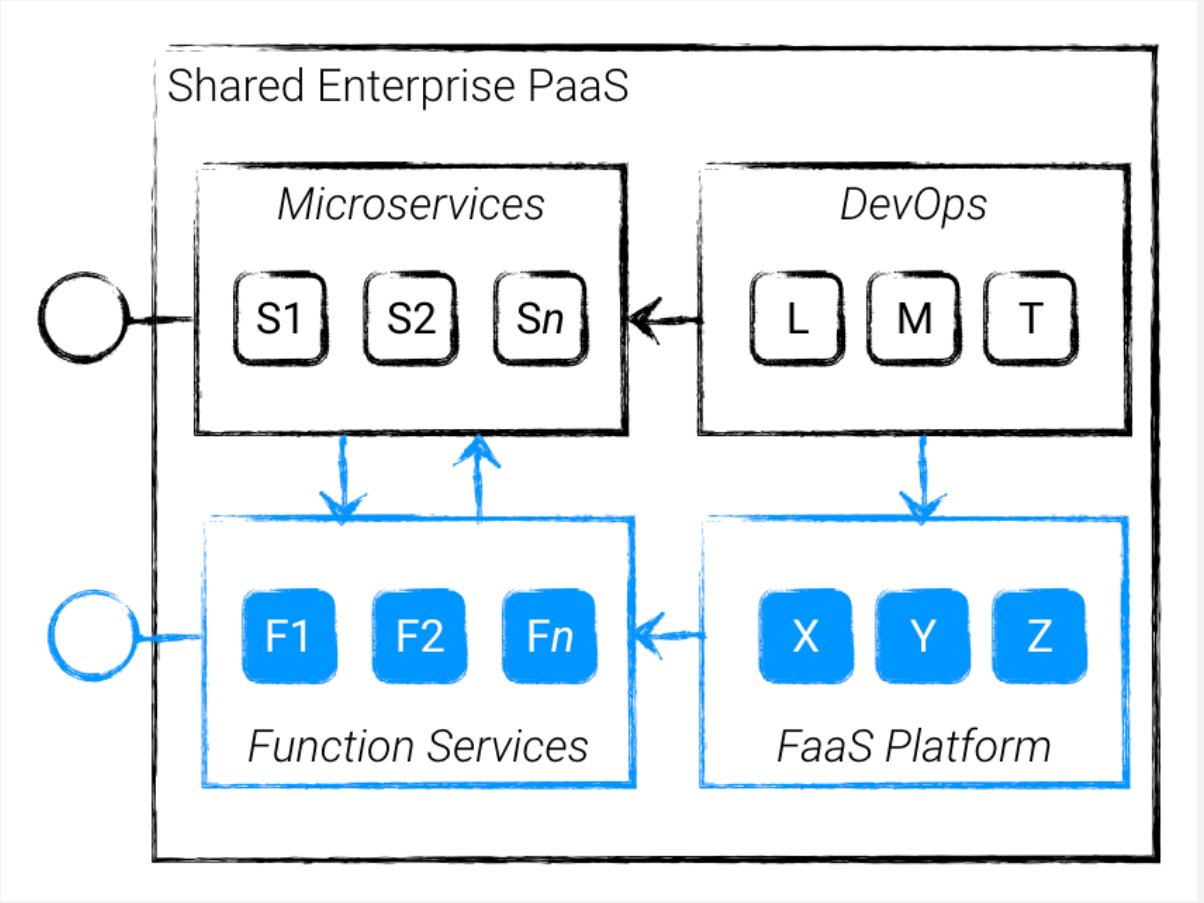
enables loosly coupled reactive software components and services.

Create Thumbnails the AWS Lambda Way

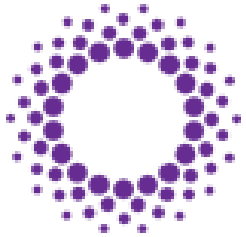


Hybrid Architectures

- 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



OPENFAAS



fission

Kubeless

nuclio

Kyma

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

Die OSS-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

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.

App Engine

Platform for building scalable web applications and mobile back ends.

28 instance hours per day

Cloud Run

A fully managed environment to run stateless containers.

2 million requests per month

Cloud Build

Fast, consistent, reliable builds on Google Cloud.

120 build-minutes per day

Operations (formerly Stackdriver)

Monitoring, logging, and diagnostics for applications on Google Cloud.

Monthly allotments for logging and monitoring

Firestore

NoSQL document database that simplifies storing, syncing, and querying data for apps.

1 GB storage

Pub/Sub

A global service for real-time and reliable messaging and streaming data.

10 GB messages per month

Cloud Functions

A serverless environment to build and connect cloud services with code.

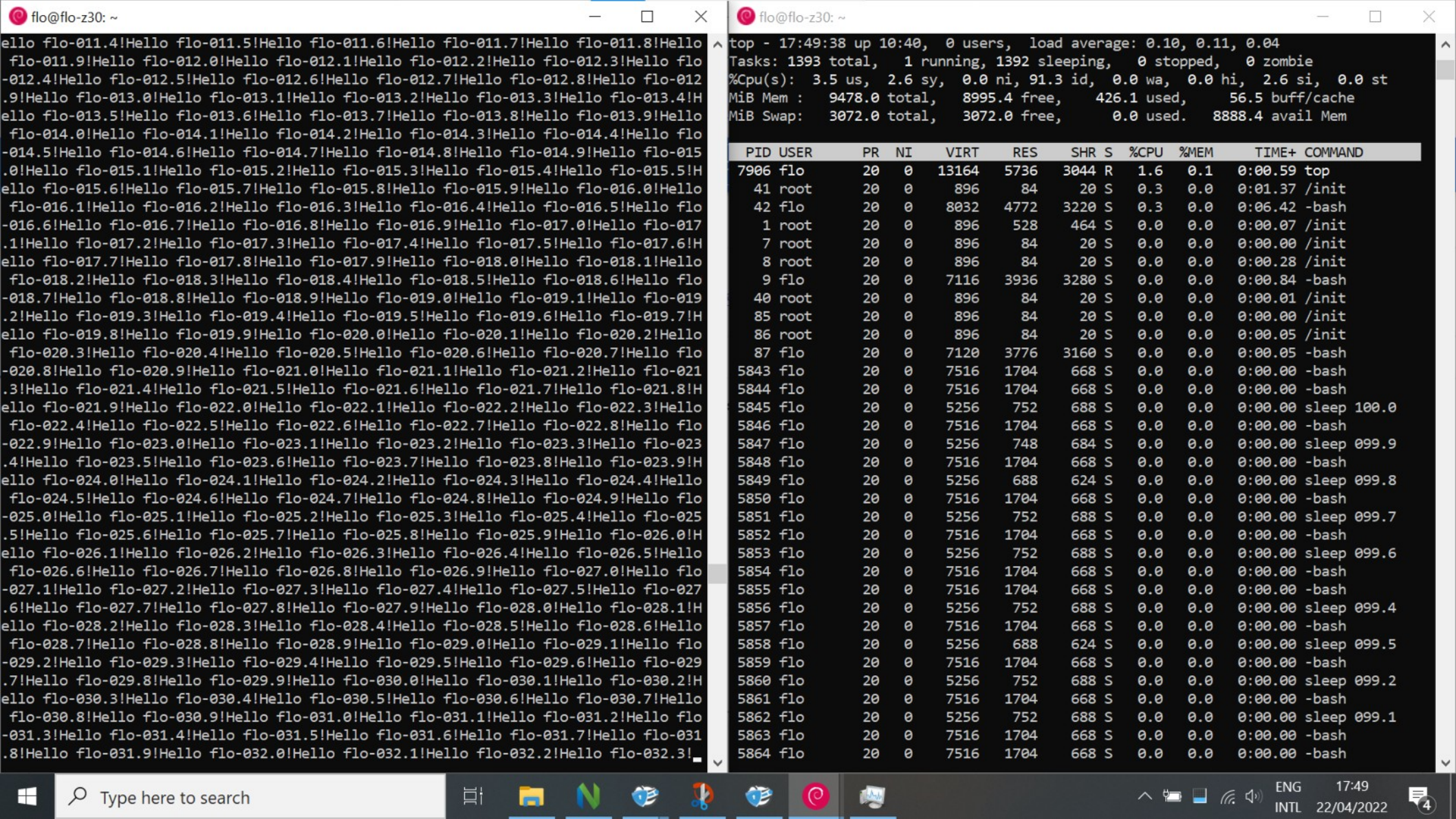
2 million invocations per month

Vision AI

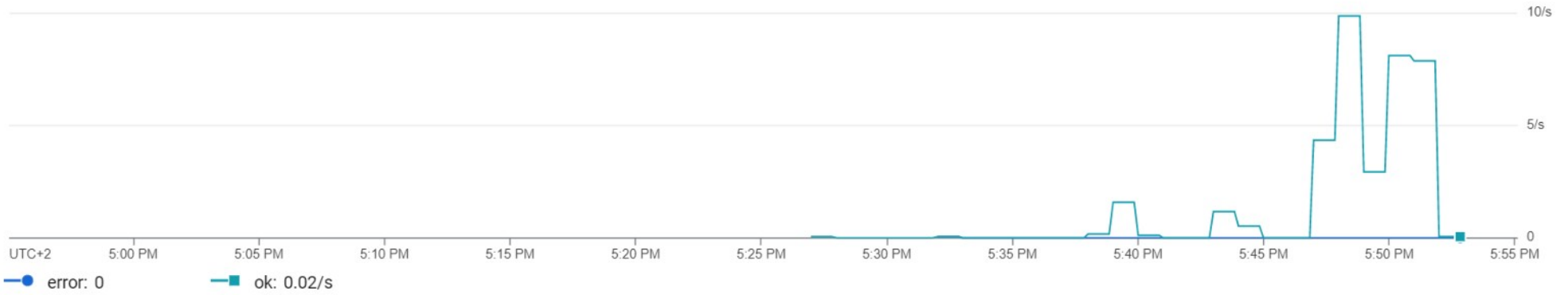
Label detection, OCR, facial detection, and more.

1,000 units per month

https://cloud.google.com/free

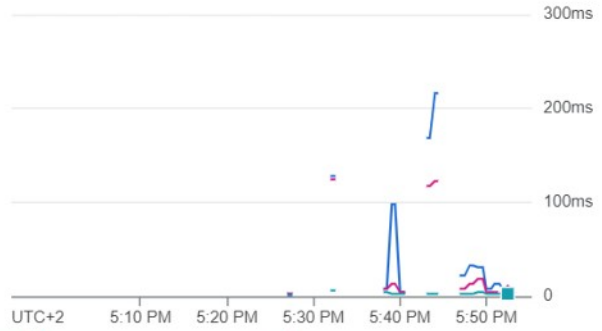

flo@flo-z30: ~
ello flo-011.4!Hello flo-011.5!Hello flo-011.6!Hello flo-011.7!Hello flo-011.8!Hello flo-011.9!Hello flo-012.0!Hello flo-012.1!Hello flo-012.2!Hello flo-012.3!Hello flo-012.4!Hello flo-012.5!Hello flo-012.6!Hello flo-012.7!Hello flo-012.8!Hello flo-012.9!Hello flo-013.0!Hello flo-013.1!Hello flo-013.2!Hello flo-013.3!Hello flo-013.4!Hello flo-013.5!Hello flo-013.6!Hello flo-013.7!Hello flo-013.8!Hello flo-013.9!Hello flo-014.0!Hello flo-014.1!Hello flo-014.2!Hello flo-014.3!Hello flo-014.4!Hello flo-014.5!Hello flo-014.6!Hello flo-014.7!Hello flo-014.8!Hello flo-014.9!Hello flo-015.0!Hello flo-015.1!Hello flo-015.2!Hello flo-015.3!Hello flo-015.4!Hello flo-015.5!Hello flo-015.6!Hello flo-015.7!Hello flo-015.8!Hello flo-015.9!Hello flo-016.0!Hello flo-016.1!Hello flo-016.2!Hello flo-016.3!Hello flo-016.4!Hello flo-016.5!Hello flo-016.6!Hello flo-016.7!Hello flo-016.8!Hello flo-016.9!Hello flo-017.0!Hello flo-017.1!Hello flo-017.2!Hello flo-017.3!Hello flo-017.4!Hello flo-017.5!Hello flo-017.6!Hello flo-017.7!Hello flo-017.8!Hello flo-017.9!Hello flo-018.0!Hello flo-018.1!Hello flo-018.2!Hello flo-018.3!Hello flo-018.4!Hello flo-018.5!Hello flo-018.6!Hello flo-018.7!Hello flo-018.8!Hello flo-018.9!Hello flo-019.0!Hello flo-019.1!Hello flo-019.2!Hello flo-019.3!Hello flo-019.4!Hello flo-019.5!Hello flo-019.6!Hello flo-019.7!Hello flo-019.8!Hello flo-019.9!Hello flo-020.0!Hello flo-020.1!Hello flo-020.2!Hello flo-020.3!Hello flo-020.4!Hello flo-020.5!Hello flo-020.6!Hello flo-020.7!Hello flo-020.8!Hello flo-020.9!Hello flo-021.0!Hello flo-021.1!Hello flo-021.2!Hello flo-021.3!Hello flo-021.4!Hello flo-021.5!Hello flo-021.6!Hello flo-021.7!Hello flo-021.8!Hello flo-021.9!Hello flo-022.0!Hello flo-022.1!Hello flo-022.2!Hello flo-022.3!Hello flo-022.4!Hello flo-022.5!Hello flo-022.6!Hello flo-022.7!Hello flo-022.8!Hello flo-022.9!Hello flo-023.0!Hello flo-023.1!Hello flo-023.2!Hello flo-023.3!Hello flo-023.4!Hello flo-023.5!Hello flo-023.6!Hello flo-023.7!Hello flo-023.8!Hello flo-023.9!Hello flo-024.0!Hello flo-024.1!Hello flo-024.2!Hello flo-024.3!Hello flo-024.4!Hello flo-024.5!Hello flo-024.6!Hello flo-024.7!Hello flo-024.8!Hello flo-024.9!Hello flo-025.0!Hello flo-025.1!Hello flo-025.2!Hello flo-025.3!Hello flo-025.4!Hello flo-025.5!Hello flo-025.6!Hello flo-025.7!Hello flo-025.8!Hello flo-025.9!Hello flo-026.0!Hello flo-026.1!Hello flo-026.2!Hello flo-026.3!Hello flo-026.4!Hello flo-026.5!Hello flo-026.6!Hello flo-026.7!Hello flo-026.8!Hello flo-026.9!Hello flo-027.0!Hello flo-027.1!Hello flo-027.2!Hello flo-027.3!Hello flo-027.4!Hello flo-027.5!Hello flo-027.6!Hello flo-027.7!Hello flo-027.8!Hello flo-027.9!Hello flo-028.0!Hello flo-028.1!Hello flo-028.2!Hello flo-028.3!Hello flo-028.4!Hello flo-028.5!Hello flo-028.6!Hello flo-028.7!Hello flo-028.8!Hello flo-028.9!Hello flo-029.0!Hello flo-029.1!Hello flo-029.2!Hello flo-029.3!Hello flo-029.4!Hello flo-029.5!Hello flo-029.6!Hello flo-029.7!Hello flo-029.8!Hello flo-029.9!Hello flo-030.0!Hello flo-030.1!Hello flo-030.2!Hello flo-030.3!Hello flo-030.4!Hello flo-030.5!Hello flo-030.6!Hello flo-030.7!Hello flo-030.8!Hello flo-030.9!Hello flo-031.0!Hello flo-031.1!Hello flo-031.2!Hello flo-031.3!Hello flo-031.4!Hello flo-031.5!Hello flo-031.6!Hello flo-031.7!Hello flo-031.8!Hello flo-031.9!Hello flo-032.0!Hello flo-032.1!Hello flo-032.2!Hello flo-032.3!
flo@flo-z30: ~
top - 17:49:38 up 10:40, 0 users, load average: 0.10, 0.11, 0.04
Tasks: 1393 total, 1 running, 1392 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.5 us, 2.6 sy, 0.0 ni, 91.3 id, 0.0 wa, 0.0 hi, 2.6 si, 0.0 st
MiB Mem : 9478.0 total, 8995.4 free, 426.1 used, 56.5 buff/cache
MiB Swap: 3072.0 total, 3072.0 free, 0.0 used. 8888.4 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
7906 flo 20 0 13164 5736 3044 R 1.6 0.1 0:00.59 top
41 root 20 0 896 84 20 S 0.3 0.0 0:01.37 /init
42 flo 20 0 8032 4772 3220 S 0.3 0.0 0:06.42 -bash
1 root 20 0 896 528 464 S 0.0 0.0 0:00.07 /init
7 root 20 0 896 84 20 S 0.0 0.0 0:00.00 /init
8 root 20 0 896 84 20 S 0.0 0.0 0:00.28 /init
9 flo 20 0 7116 3936 3280 S 0.0 0.0 0:00.84 -bash
40 root 20 0 896 84 20 S 0.0 0.0 0:00.01 /init
85 root 20 0 896 84 20 S 0.0 0.0 0:00.00 /init
86 root 20 0 896 84 20 S 0.0 0.0 0:00.05 /init
87 flo 20 0 7120 3776 3160 S 0.0 0.0 0:00.05 -bash
5843 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5844 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5845 flo 20 0 5256 752 688 S 0.0 0.0 0:00.00 sleep 100.0
5846 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5847 flo 20 0 5256 748 684 S 0.0 0.0 0:00.00 sleep 099.9
5848 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5849 flo 20 0 5256 688 624 S 0.0 0.0 0:00.00 sleep 099.8
5850 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5851 flo 20 0 5256 752 688 S 0.0 0.0 0:00.00 sleep 099.7
5852 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5853 flo 20 0 5256 752 688 S 0.0 0.0 0:00.00 sleep 099.6
5854 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5855 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5856 flo 20 0 5256 752 688 S 0.0 0.0 0:00.00 sleep 099.4
5857 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5858 flo 20 0 5256 688 624 S 0.0 0.0 0:00.00 sleep 099.5
5859 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5860 flo 20 0 5256 752 688 S 0.0 0.0 0:00.00 sleep 099.2
5861 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5862 flo 20 0 5256 752 688 S 0.0 0.0 0:00.00 sleep 099.1
5863 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
5864 flo 20 0 7516 1704 668 S 0.0 0.0 0:00.00 -bash
Type here to search
ENG
INTL
17:49
22/04/2022

Invocations/Second



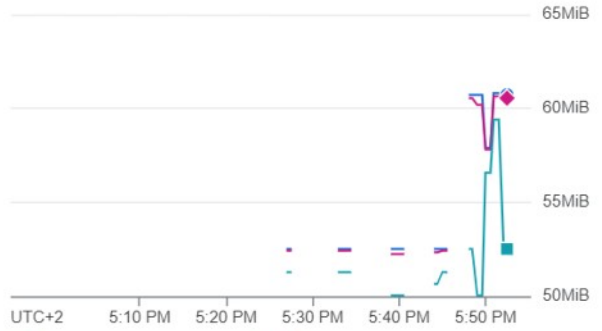
Execution time

Milliseconds/call



Memory utilization

MB/call



Active instances