Cloud Architektur

24.11.2021

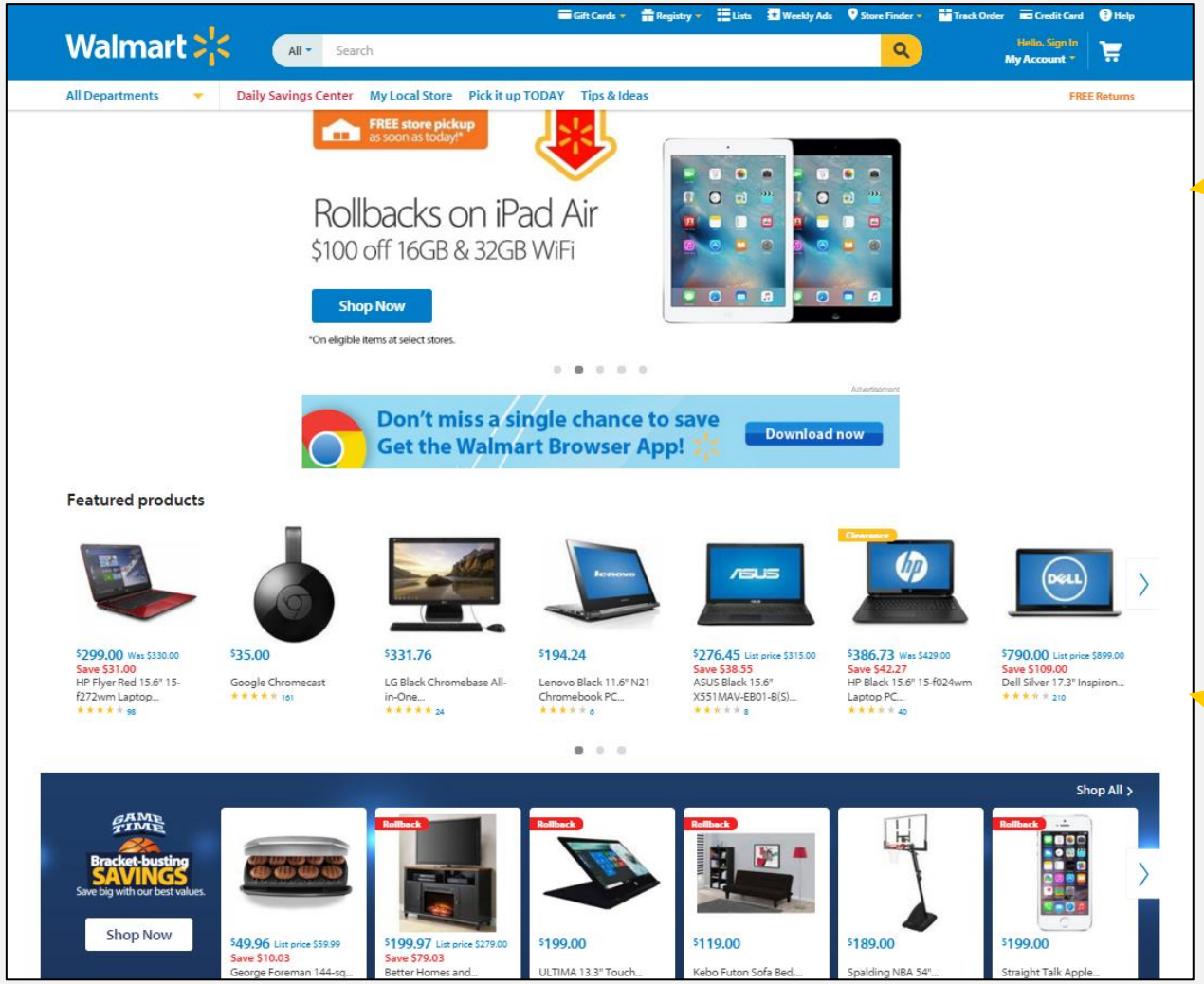
WHAT DID THEY DO?

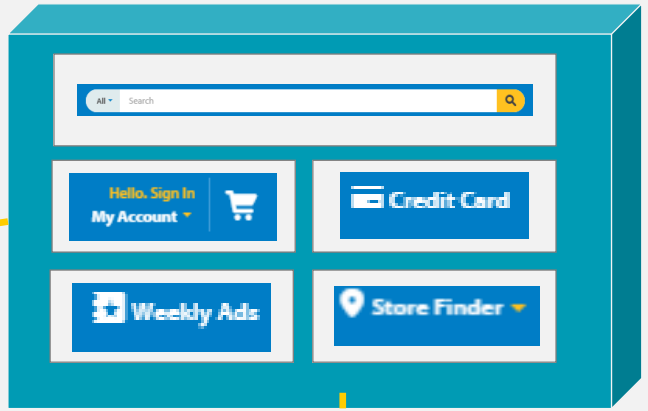
Betriebsmonolithen

Knoten

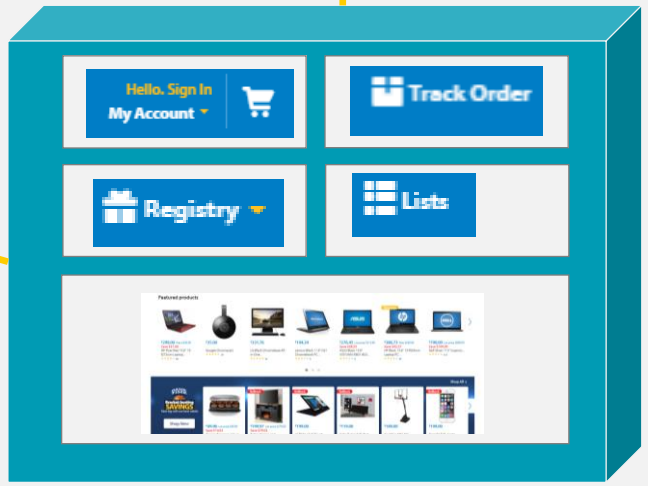
Deployment-Einheit

Betriebskomponenten





Skalieren durch
Verteilung und Klone

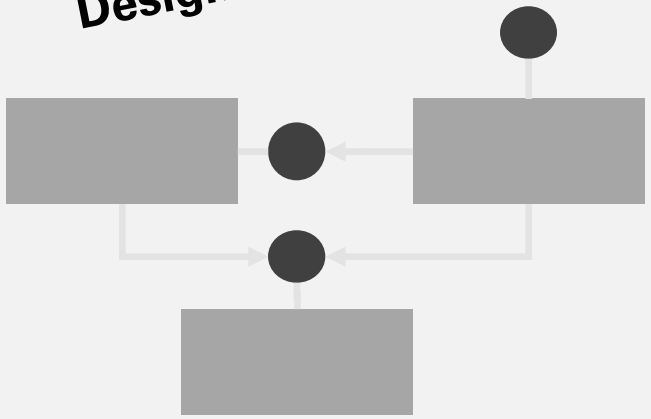


Cloud Native Application Development: Components All Along the Software Lifecycle.

DESIGN → BUILD → RUN

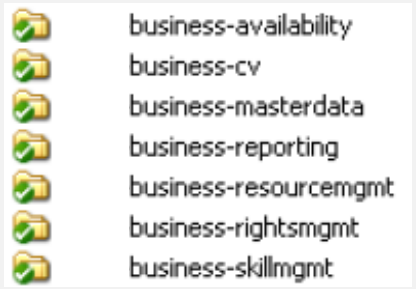
NEW!

Design Components



1:1

Dev Components



n:1

Ops Components

- Complexity unit
- Data integrity unit
- Coherent and cohesive features unit
- Decoupled unit

- Planning unit
- Team assignment unit
- Knowledge unit
- Development unit
- Integration unit

- Release unit
- Deployment unit
- Runtime unit (crash, slow-down, access)
- Scaling unit

Die Anatomie einer Betriebs-Komponente.

Diagnose Interface

Internet Protocol

Container

Application

Starting Inteface

Internet Protocol

Regel 1 für den Betrieb in der Cloud.

"Everything fails all the time."

— Werner Vogels, CTO of Amazon

Regel 2 für den Betrieb in der Cloud.

Soll nur der Himmel die Grenze sein, dann funktioniert nur horizontale Skalierung.

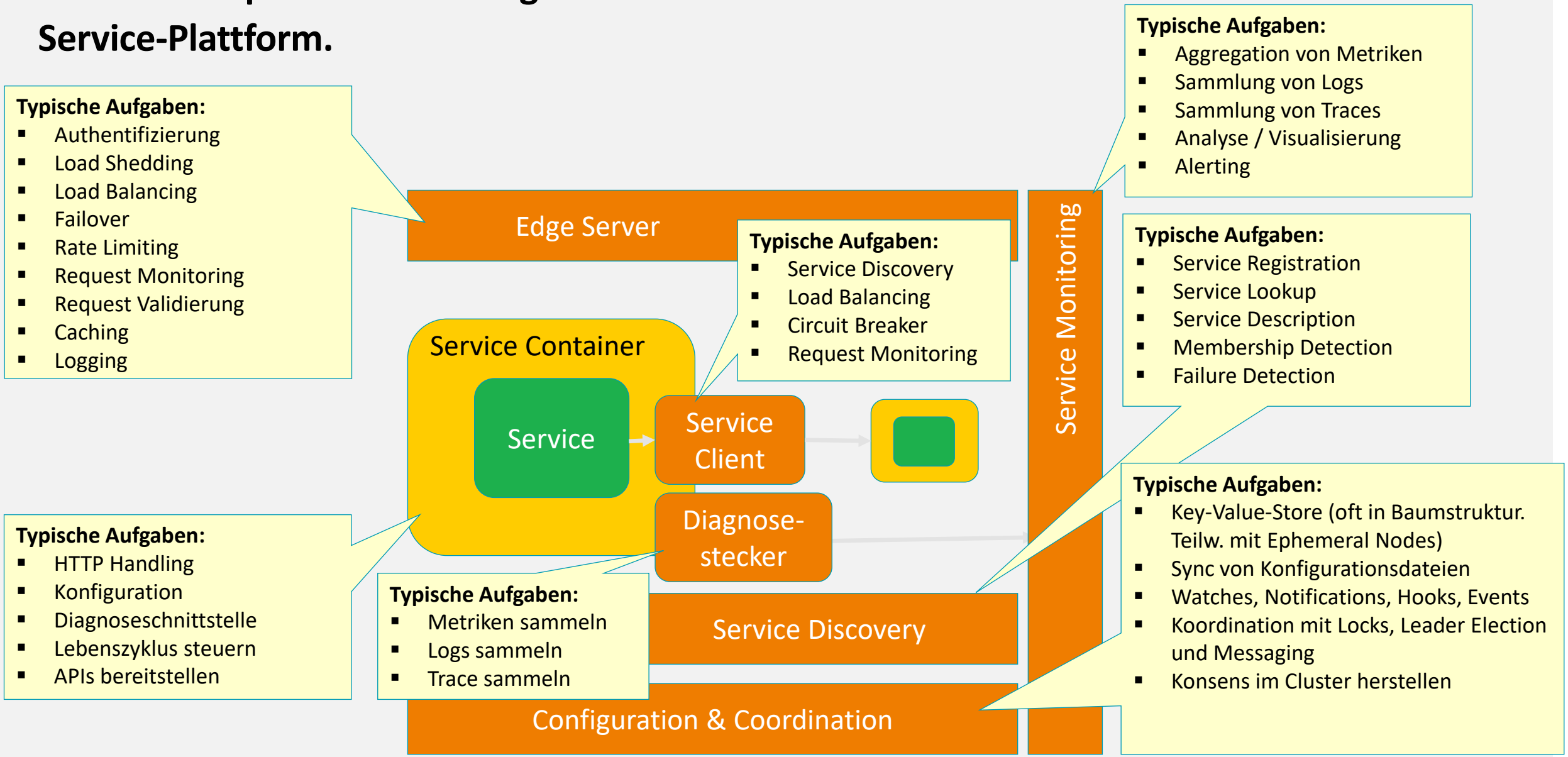
Regel 3 für den Betrieb in der Cloud.

Wer in die Cloud will, der sollte Cloud sprechen.

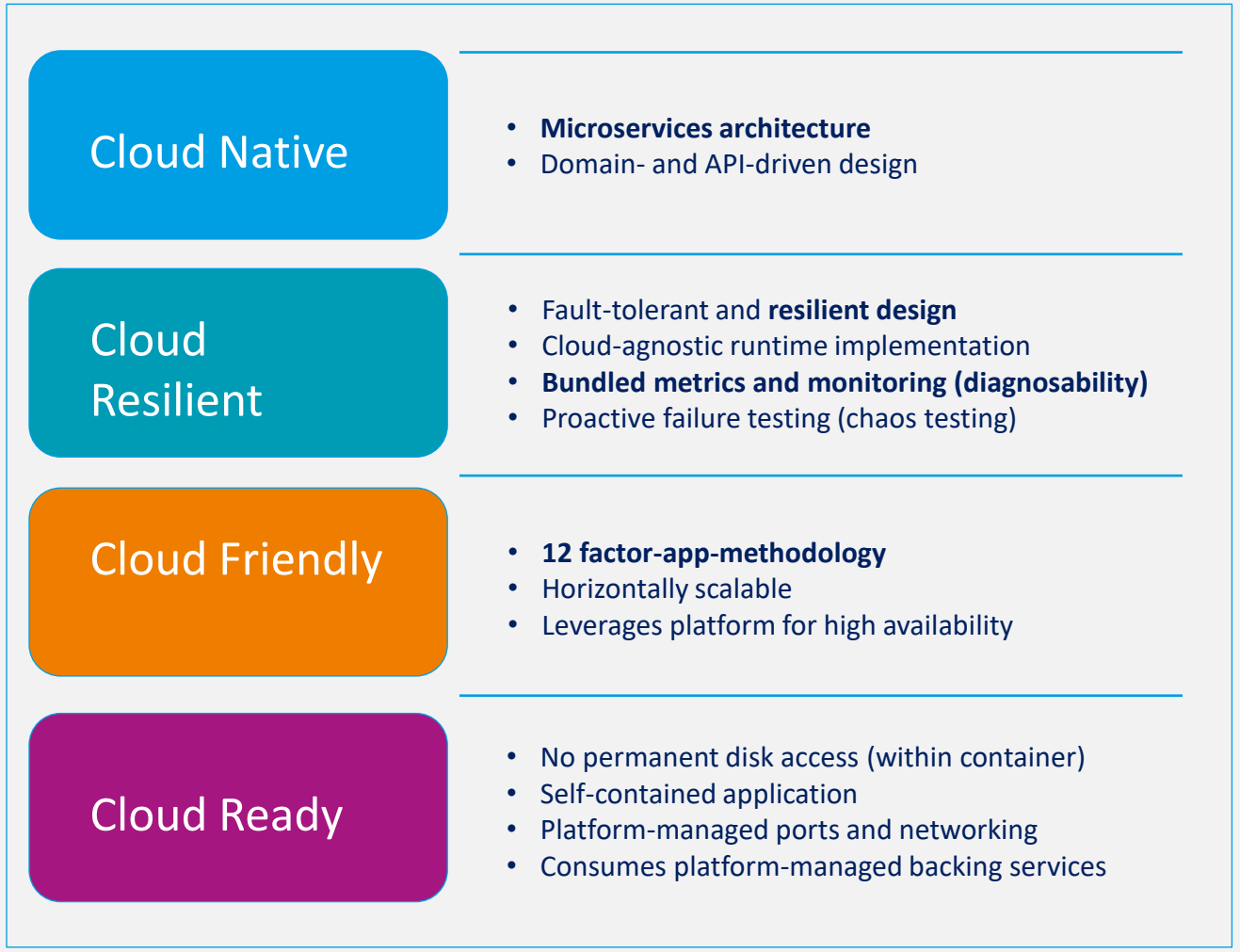
Cloud-Architektur aus Sicht der Softwarearchitektur:
Design for Failure.

1. Jede Komponente läuft eigenständig und isoliert ➔ *Betriebskomponenten*
2. Die Betriebskomponenten kommunizieren untereinander über Internet-Protokolle ➔ *HTTP, UDP, …*
3. Jede Betriebskomponente kann in mehreren Instanzen laufen und bietet damit Redundanz. Es gibt keinen „Common Point of Failure“. ➔ *Cluster Orchestrator*
4. Jede Betriebskomponente besitzt Diagnoseschnittstellen um ein fehlerhaftes Verhalten erkennen zu können
5. Jeder Microservice kann zu jeder Zeit neu gestartet und auf einem anderen Knoten in Betrieb genommen werden. Er besitzt keinen eigenen Zustand.
6. Die Implementierung hinter einem jeden Microservice kann ausgetauscht werden, ohne dass die Nutzer davon etwas bemerken.

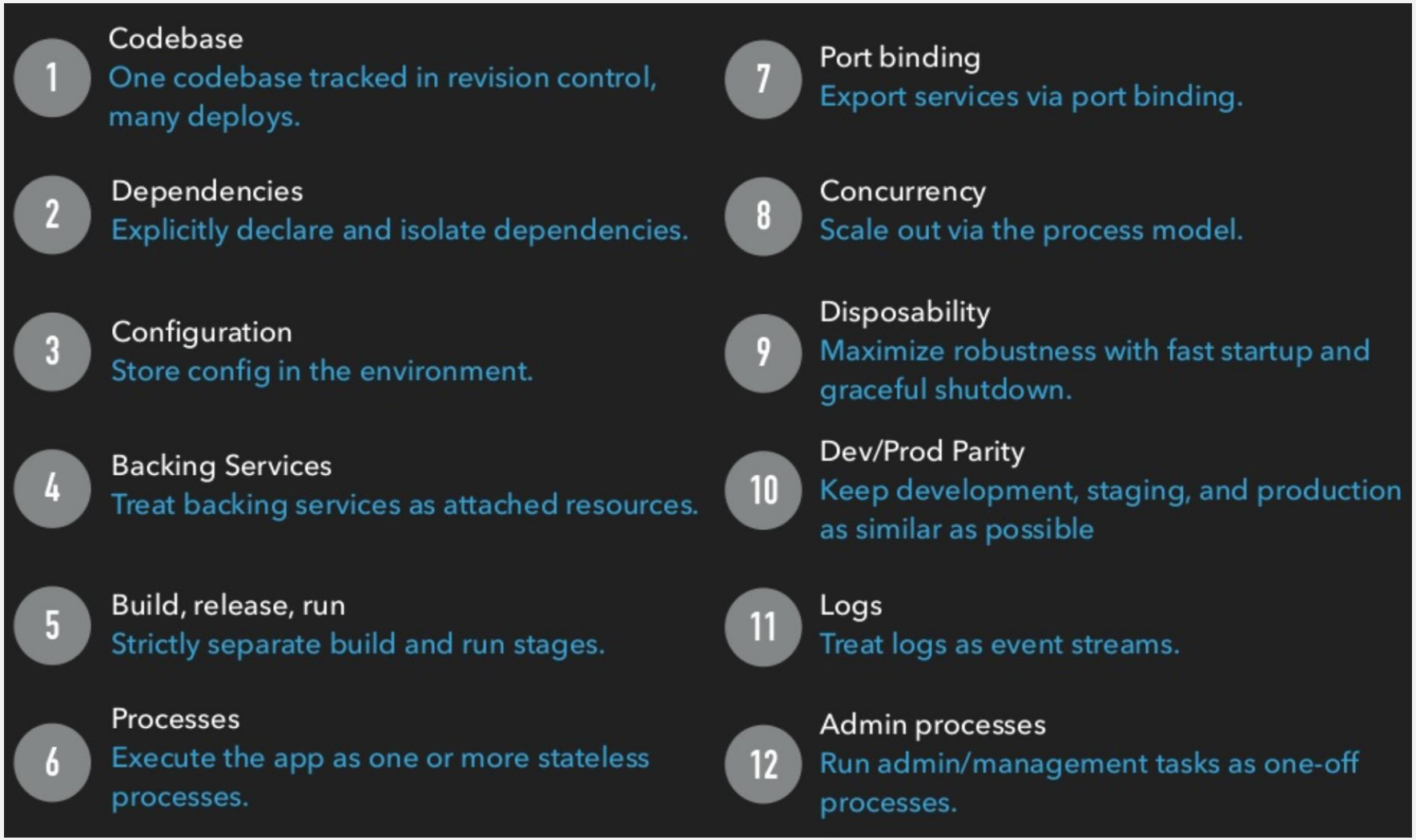
Betriebskomponenten benötigen eine Infrastruktur um sie herum: Eine Micro-Service-Plattform.



Das Cloud Native Application Reifegradmodell



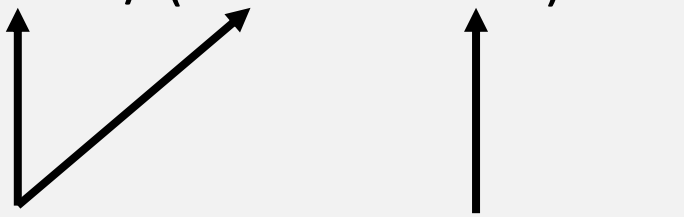
12 Factor App



https://12factor.net/de
https://www.slideshare.net/Alicanakku1/12-factor-apps

Resilienz

Verfügbarkeit = MTTF / (MTTF + MTTR)



Zuverlässigkeit steigern

Resilienz & Diagnostizierbarkeit

Resilienz: Die Fähigkeit eines Systems mit unerwarteten und fehlerhaften Situationen umzugehen

- Ohne dass es der Nutzer merkt (Bestfall)
- Mit ein einer „graceful degradation“ des Services (schlechtester Fall)

Dignostizierbarkeit