CLOUD NATIVE COMPUTING

CLOUD-NATIVE

Unit:

Container-Orchestrierung

(2) Orchestrierung in Abgrenzung zum Scheduling

PROF. DR.
NANE KRATZKE

creative commons

CLOUD NATIVE COMPUTING

Urheberrechtshinweise

Diese Folien werden zum Zwecke einer praktikablen und pragmatischen Nutzbarkeit im Rahmen der **CCo 1.0 Lizenz** bereitgestellt.

Sie dürfen die Inhalte also kopieren, verändern, verbreiten, mit eigenen Inhalten mixen, auch zu kommerziellen Zwecken, und ohne um weitere Erlaubnis bitten zu müssen.

Eine Nennung des Autors ist nicht erforderlich (aber natürlich gern gesehen, wenn problemlos möglich).

Diese Folien sind insb. für die Lehre an Hochschulen konzipiert und machen daher vom **§51 UrhG (Zitate)** Gebrauch.

Die CCo Lizenz überträgt sich nicht auf zitierte Quellen. Hier sind bei der Nutzung natürlich die Bedingungen der entsprechenden Quellen zu beachten.

Die Quellenangaben finden sich auf den entsprechenden Folien.

KAPITEL 9

Container-Plattformen

9.1 Scheduling

- Heterogenität von Workloads
- Scheduling-Algorithmen
- Scheduling-Architekturen

9.2 Orchestrierung

- Definition von Betriebszuständen
- Regelkreis: Desired vs Current State

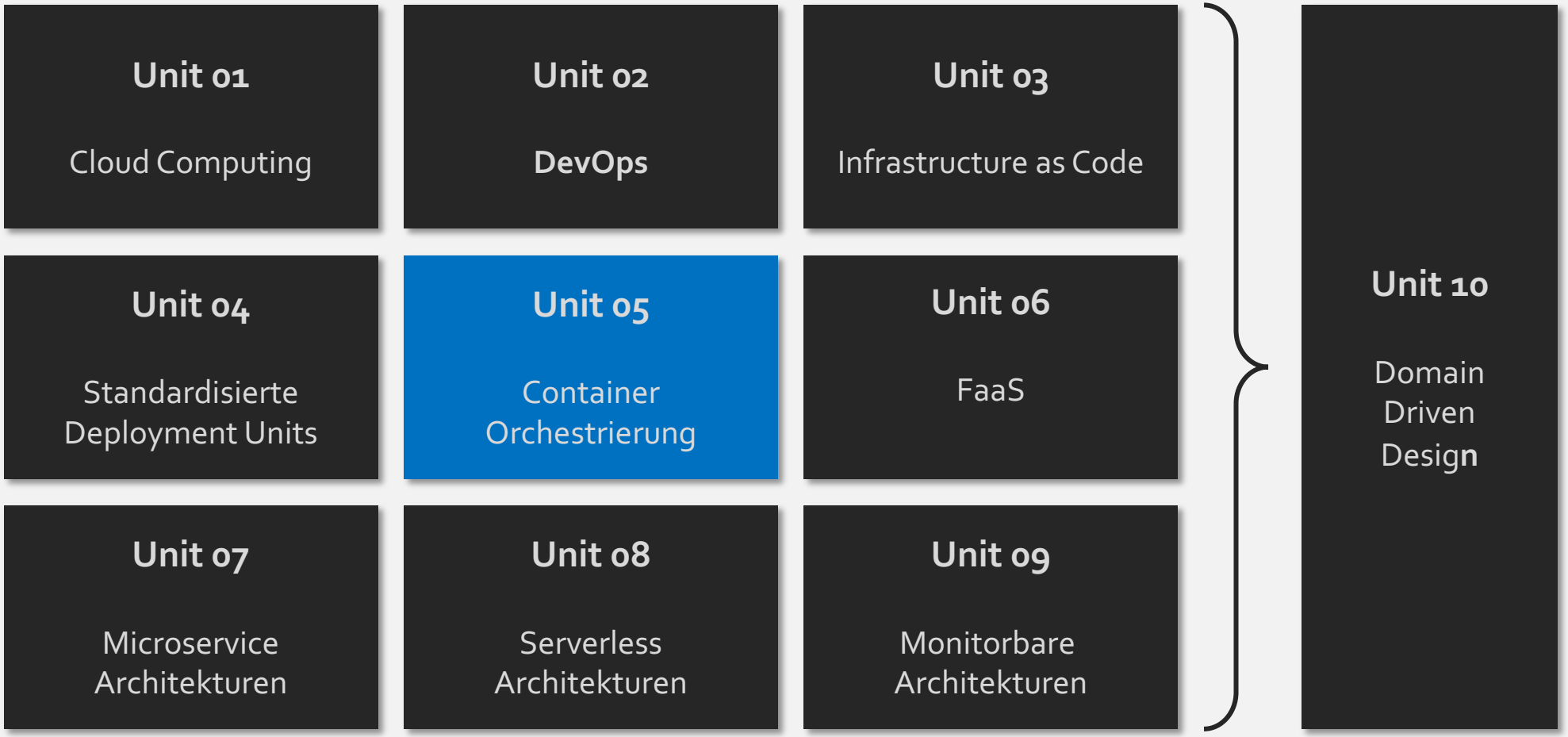
9.3 Inside Kubernetes

- Kubernetes-Architektur und Ressourcen
- Workloadarten
- Scheduling Constraints
- Automatische Skalierung von Workloads
- Exponierung von Services
- Health Checking
- Persistenz
- Isolation von Workloads

9.4 Zusammenfassung

INHALTSVERZEICHNIS

Überblick über Units und Themen dieses Moduls



INHALTE

Scheduling

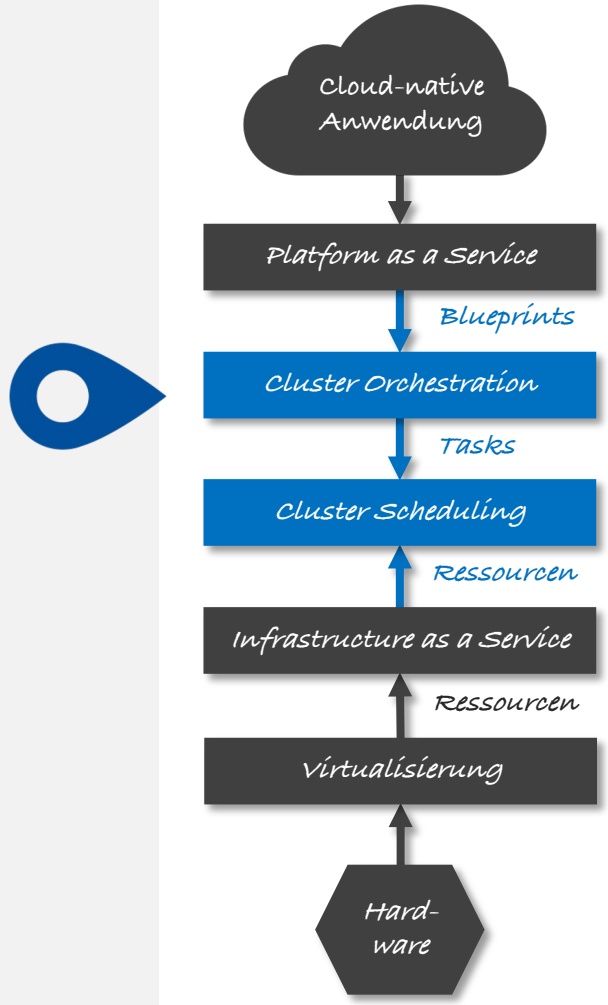
- Scheduling Problem Definition
- Scheduling Algorithmen
- Scheduler Architekturen
- Beispiele von Cluster Schedulern: Mesos, Swarm

Orchestrierung

- Was ist Orchestrierung (in Abgrenzung zum Scheduling)?
- Was sind Blueprints?
- Wie funktionieren Regelkreise in der Orchestrierung?
- Überblick über bestehende Orchestrierungslösungen

Inside Kubernetes (Typ-Vertreter)

- K8S-Architektur
- K8S-Ressourcen
- Workloads, Persistenz, Isolation und Exponieren von Services



ORCHESTRIERUNG

Was versteht man unter Orchestrierung im Cloud-native Computing?

Eine Anwendung, die in mehrere Betriebskomponenten aufgeteilt ist, auf mehreren Knoten laufen lassen.

Führt Abstraktionen zur Ausführung von Anwendungen mit ihren Services in einem (großen) Cluster ein.

Orchestrierung ist keine statische, einmalige Aktivität wie die Provisionierung, sondern eine dynamische, **kontinuierliche Aktivität**.

Orchestrierung hat den Anspruch, alle Standard-Betriebsprozeduren einer Anwendung zu automatisieren.

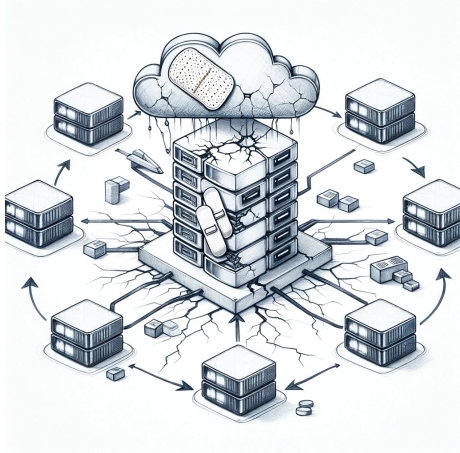
Blaupause (Blueprint) der Anwendung

Gewünschter Betriebszustand der Anwendung. Diese beschreibt die Betriebskomponenten (Container), deren Betriebsanforderungen sowie die angebotenen und benötigten Schnittstellen.

Cluster-Orchestrierer

Steuerungsaktivitäten im Cluster

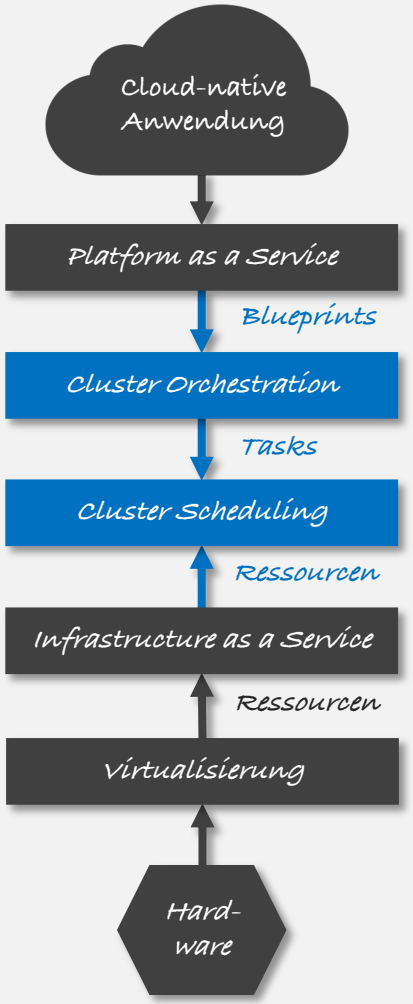
- Start von Containern auf Knoten (→ Scheduler)
- Verknüpfung von Container
- Replizierung/Skalierung von Containern
- Behebung von Fehlerzuständen
- …



Quelle: QAware GmbH
https://github.com/qaware/cloudcomputing

BACK TO BIG PICTURE

Orchestrierung



NGINX Webserver + vorgelagerter HAproxy (ausfallsicher)

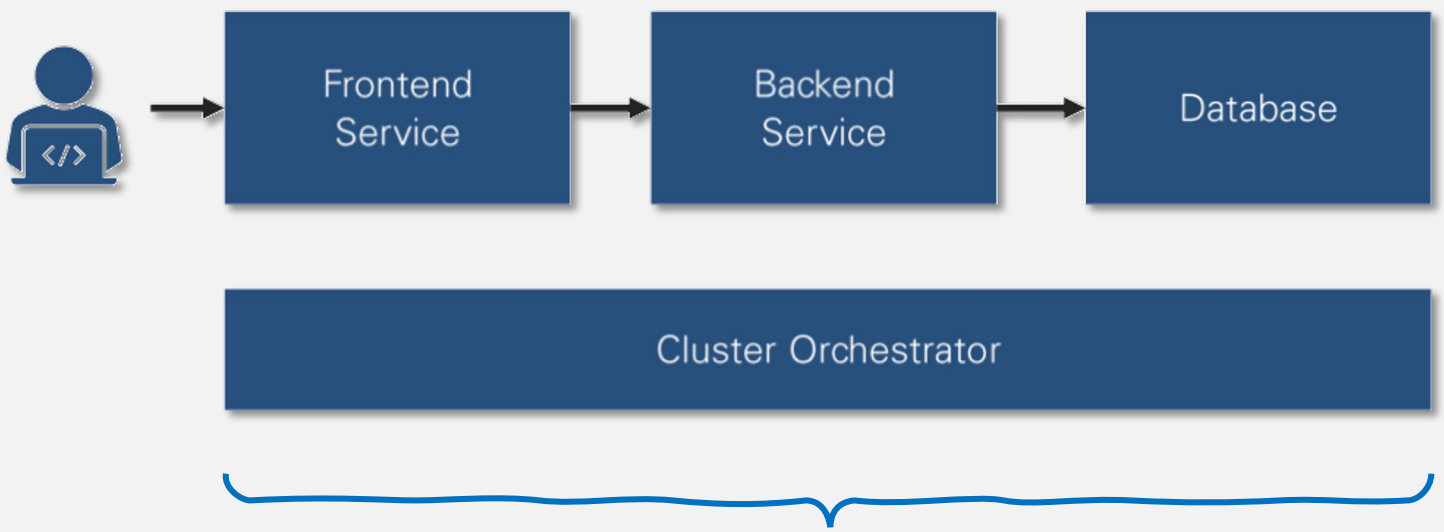
3 Instanzen NGINX, mind. 2 Instanzen HAproxy mit Liveness Probes

Kelsey Hightower

„How would you design your infrastructure if you couldn't login. Never!"

ORCHESTRIERUNG

Unterschiedliche DevOps-Perspektiven



Dev-Perspektive

Anwendungen
Betriebssystem
Abhängigkeiten

Ops-Perspektive

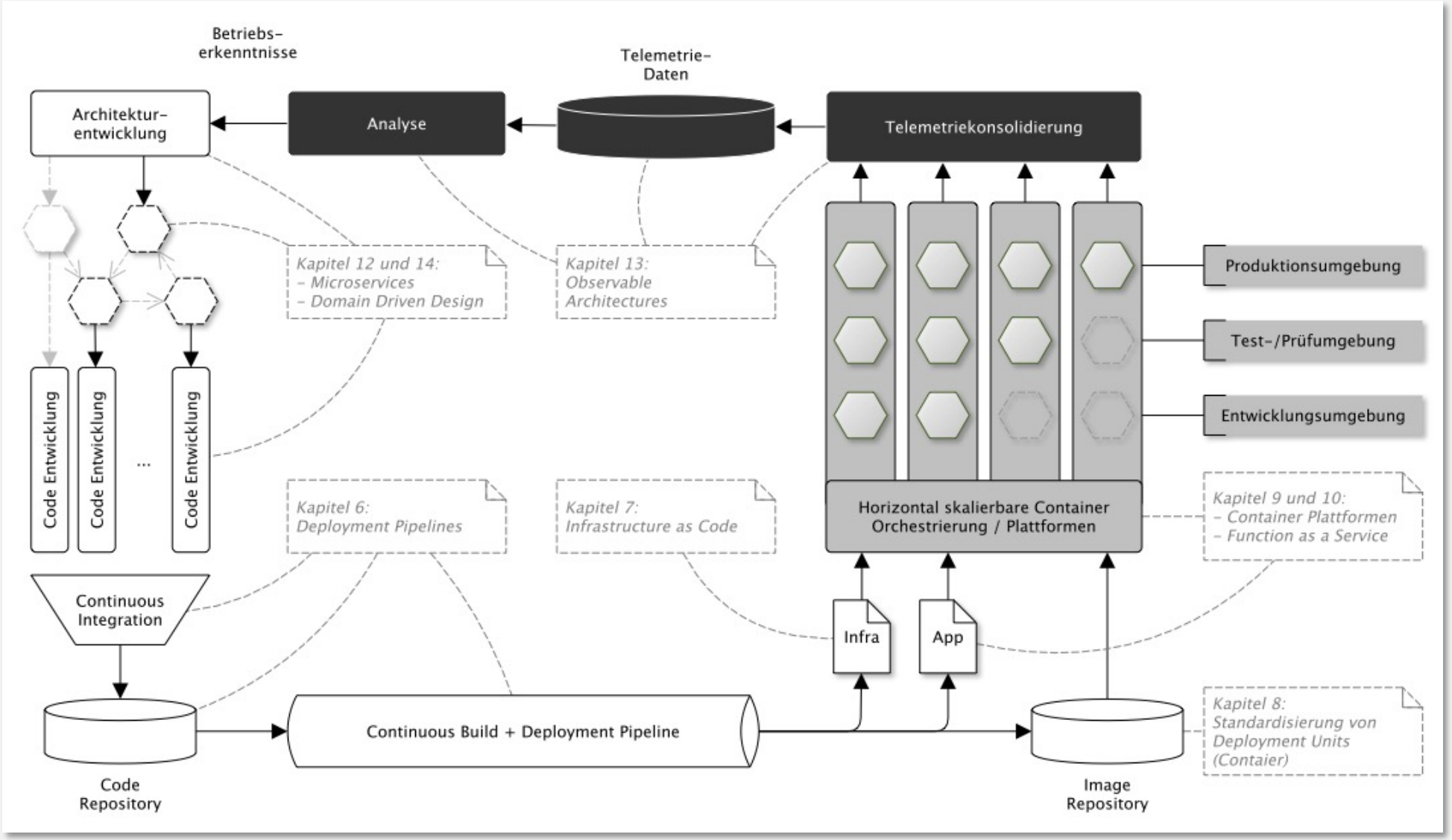
Deployment
Networking
Service-Discovery (auch Dev)
Rescheduling

Scaling
Load Balancing
Logging + Analytics
Failure Recovery

Quelle: QAware GmbH
https://github.com/qaware/cloudcomputing

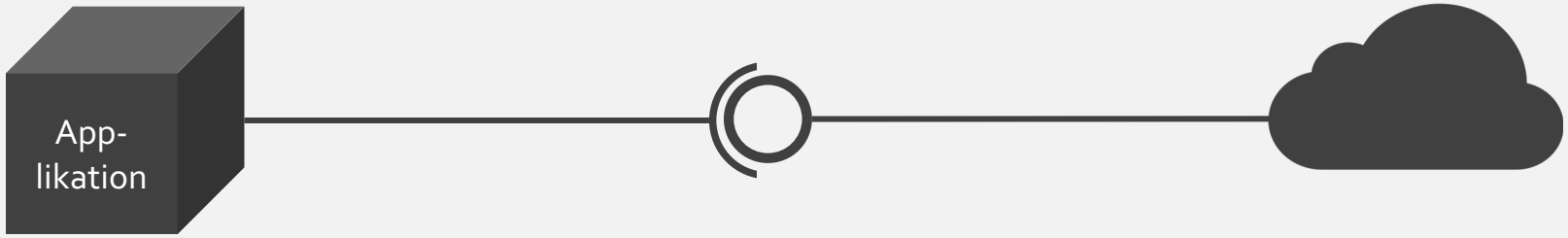
CLUSTER - ORCHESTRIERER

Schnittstelle zwischen Betrieb und Entwicklung



CLUSTER - ORCHESTRIERER

Schnittstelle zwischen Betrieb und Entwicklung



Entwicklung (Dev)

DevOps-Schnittstelle

- deploy()
- rollback()
- configure()
- scale()

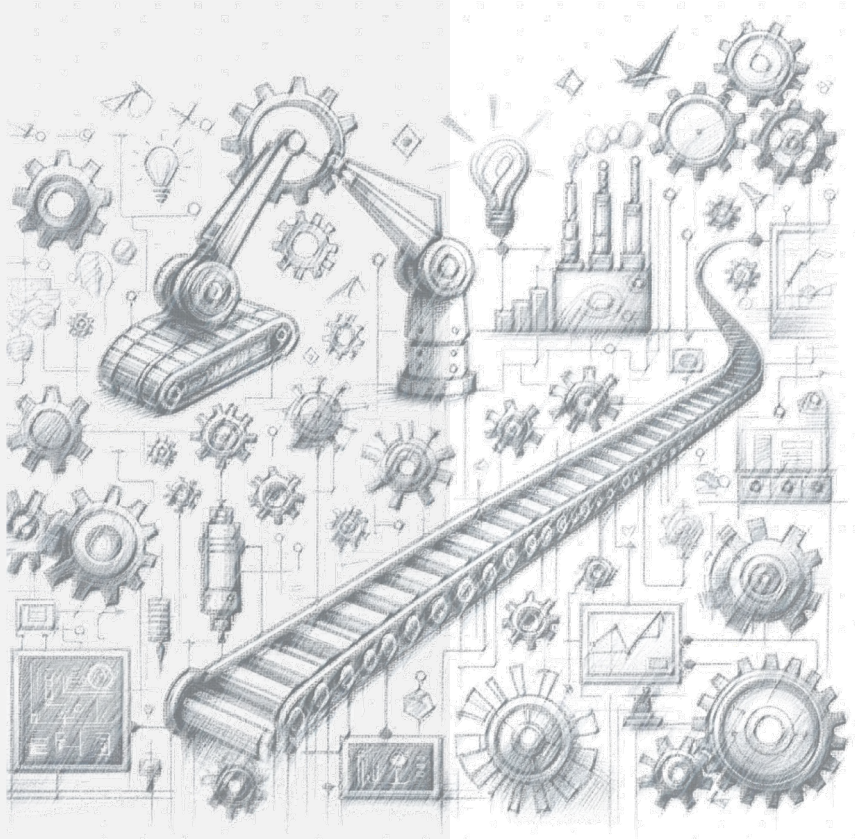
Betrieb (Ops)

Quelle: QAware GmbH
https://github.com/qaware/cloudcomputing

CLUSTER - ORCHESTRIERER

Beispiele von Betriebsaufgaben in einem Cluster

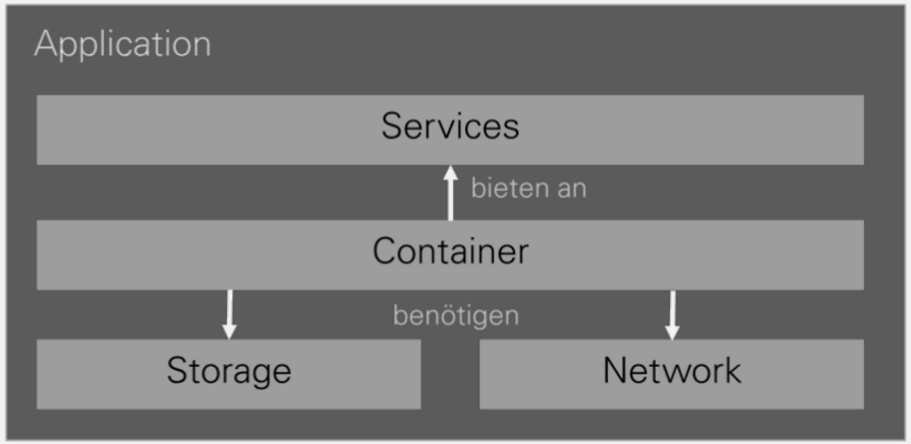
- Scheduling von Containern mit applikationsspezifischen Constraints (z.B. Affinities und Anti-Affinities)
- Aufbau von notwendigen Netzwerk-Verbindungen zwischen Containern.
- Bereitstellung von persistenten Speichern für zustandsbehaftete Container.
- (Auto-)Skalierung von Containern.
- Re-Scheduling von Containern im Fehlerfall (Auto-Healing) oder zur Ressourcen-Optimierung (→ Scheduling)
- Container-Logistik: Verwaltung, Bereitstellung und Cacheing von Container-Images.
- Management von Services: Service Discovery, Naming, Load Balancing.
- Automatismen für Rollout-Workflows wie z.B. Canary Rollout.
- Monitoring und Diagnose von Containern und Services.
- ...



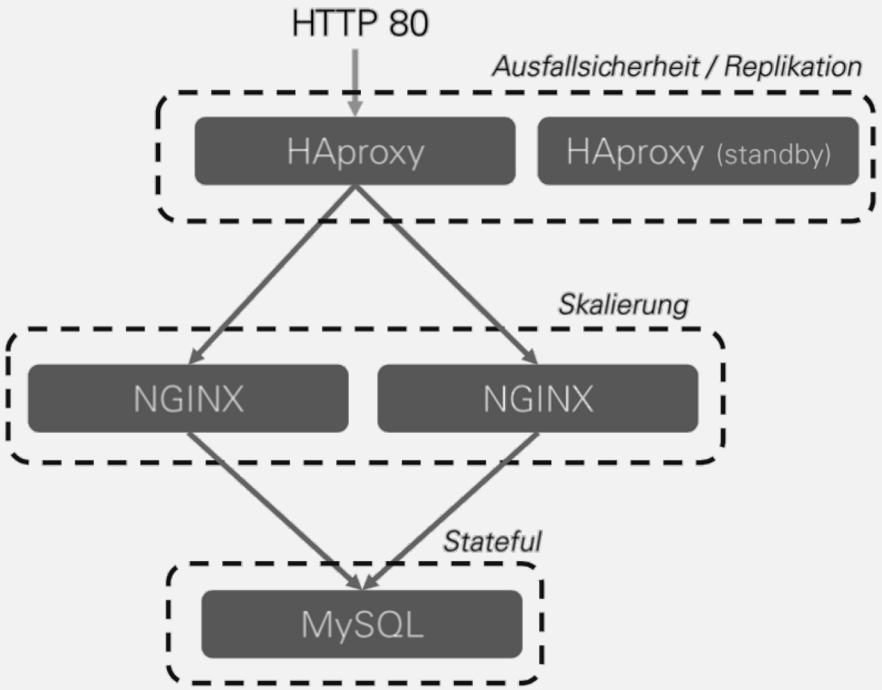
Quelle: QAware GmbH
https://github.com/qaware/cloudcomputing

BLUEPRINTS

„Konstruktionspläne" von Anwendungen (Multi-Komponenten Anwendungen)



Metamodell



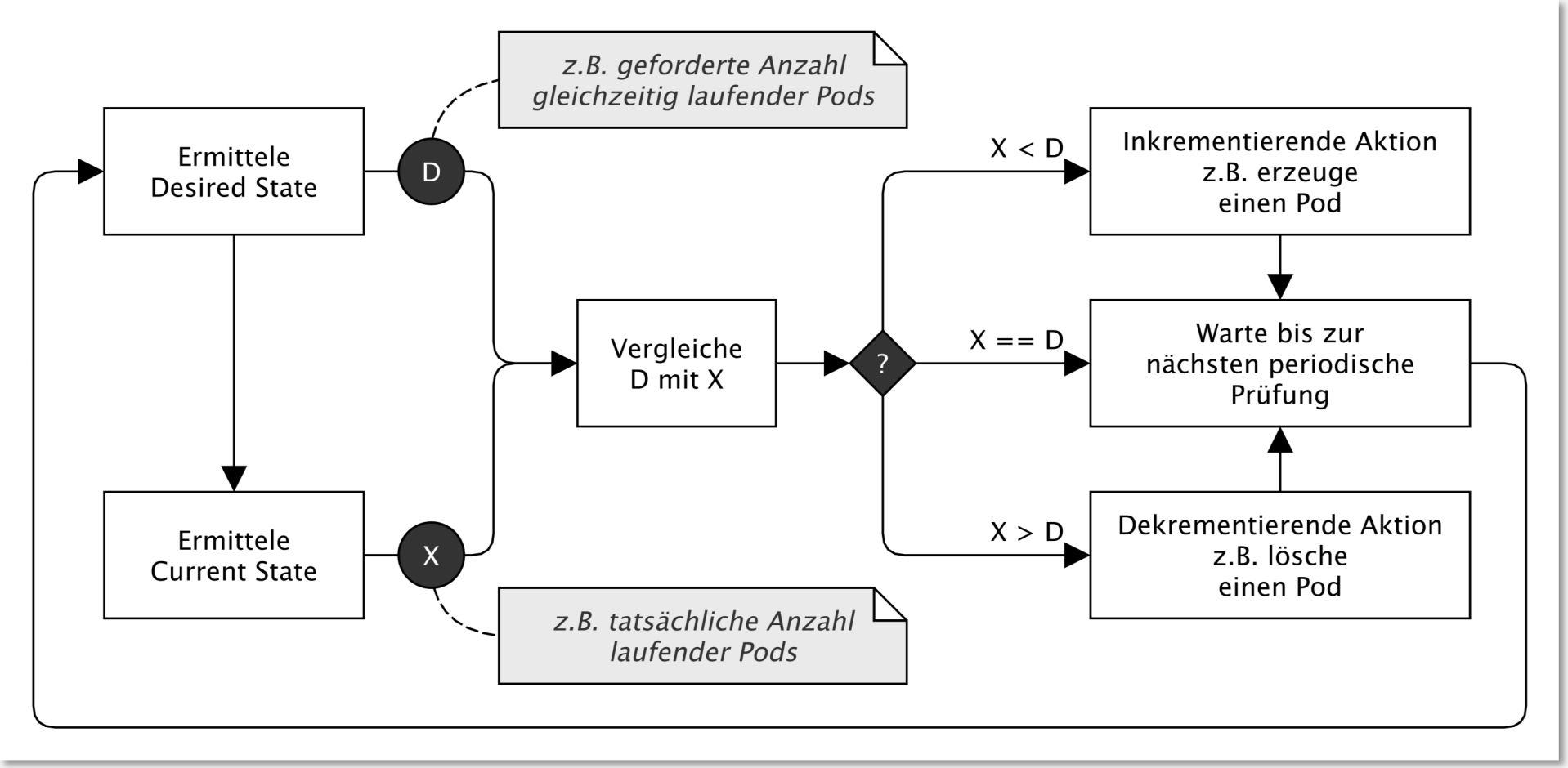
Modell

Vereinfachte Darstellung

Quelle: QAware GmbH
https://github.com/qaware/cloudcomputing

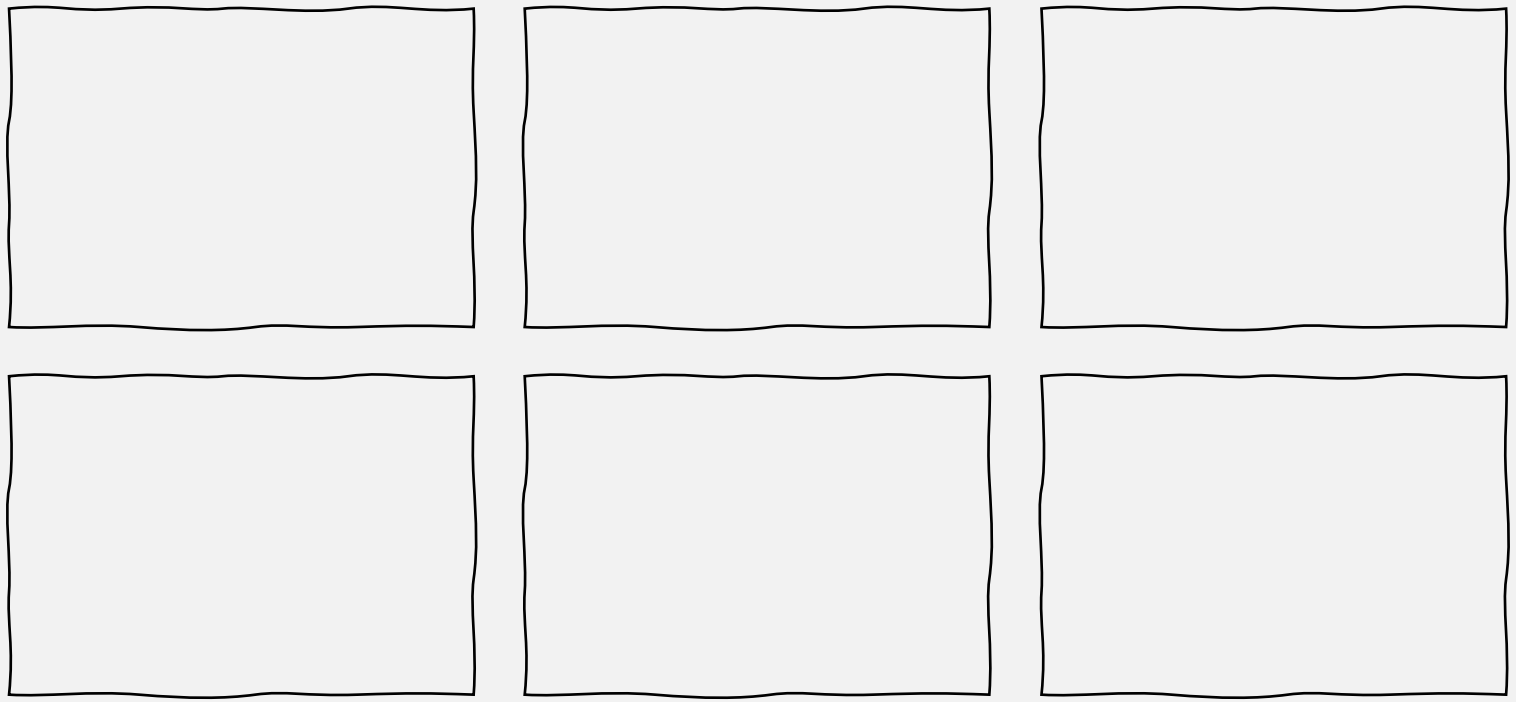
REGELKREIS

Desired vs. Current State



REGELKREIS

Desired vs. Current State



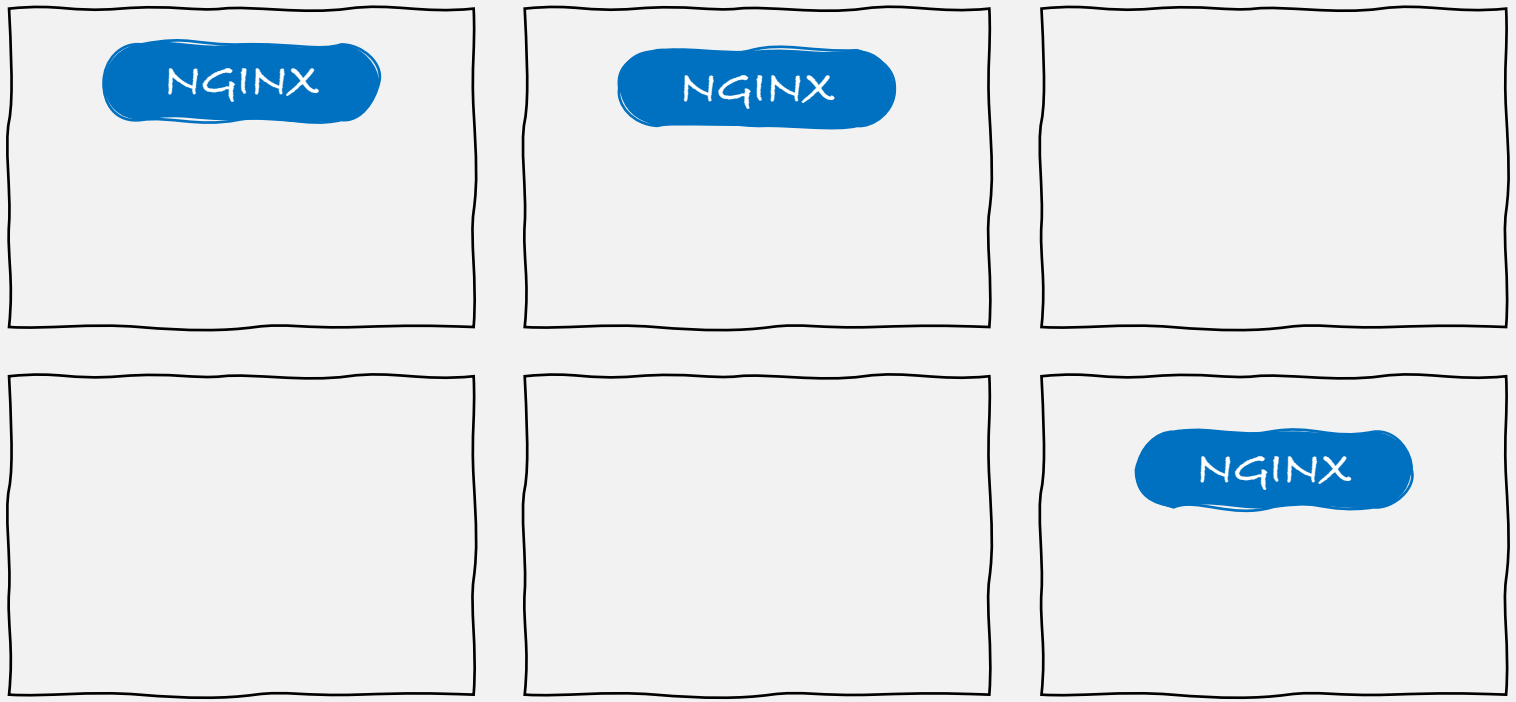
Desired State:
Betreibe 3 Instanzen von NGINX

Current State:
Es laufen 0 Instanzen von NGINX

Maßnahme:
Starte 3 Instanzen von NGINX

REGELKREIS

Desired vs. Current State



Desired State wieder erreicht !!!

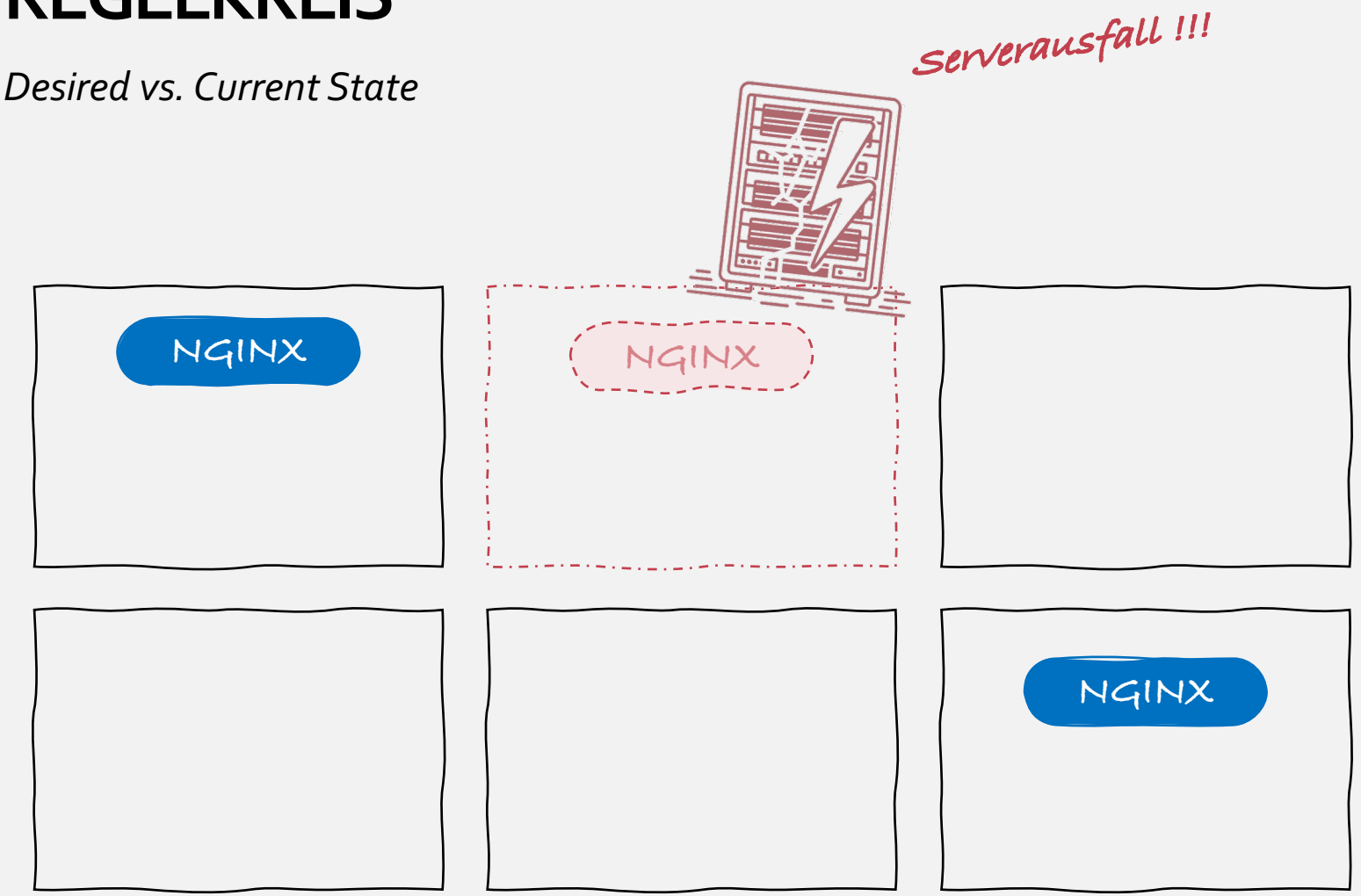
Desired State:
Betreibe 3 Instanzen von NGINX

Current State:
Es laufen 3 Instanzen von NGINX

Maßnahme:
keine

REGELKREIS

Desired vs. Current State



Externes Ereignis mit Einfluss auf den Desired State.

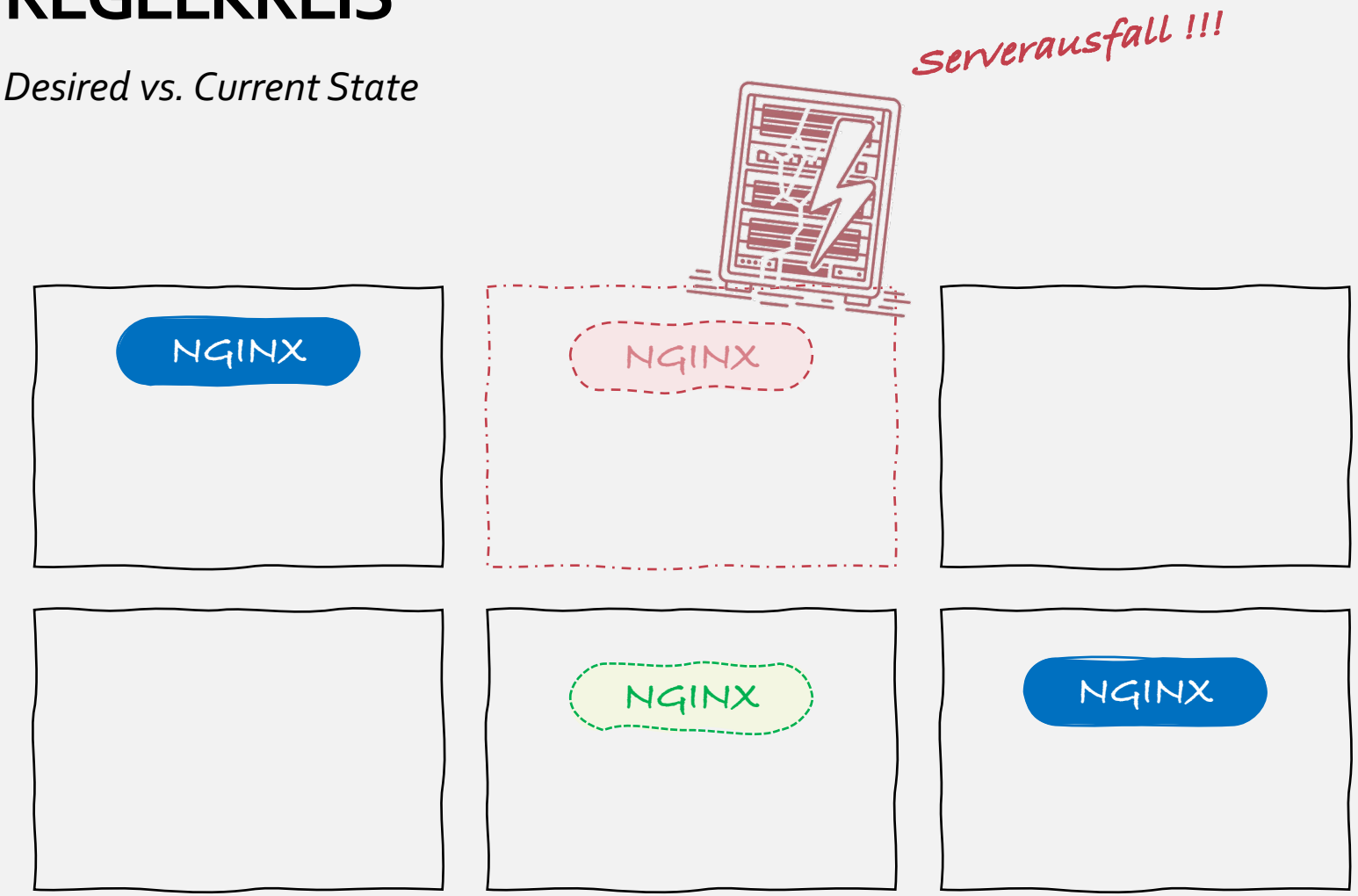
Desired State:
Betreibe 3 Instanzen von NGINX

Current State:
Es laufen 2 Instanzen von NGINX

Maßnahme:
Starte eine Instanz von NGINX

REGELKREIS

Desired vs. Current State



Planung einer Kompensationsmaßnahme.

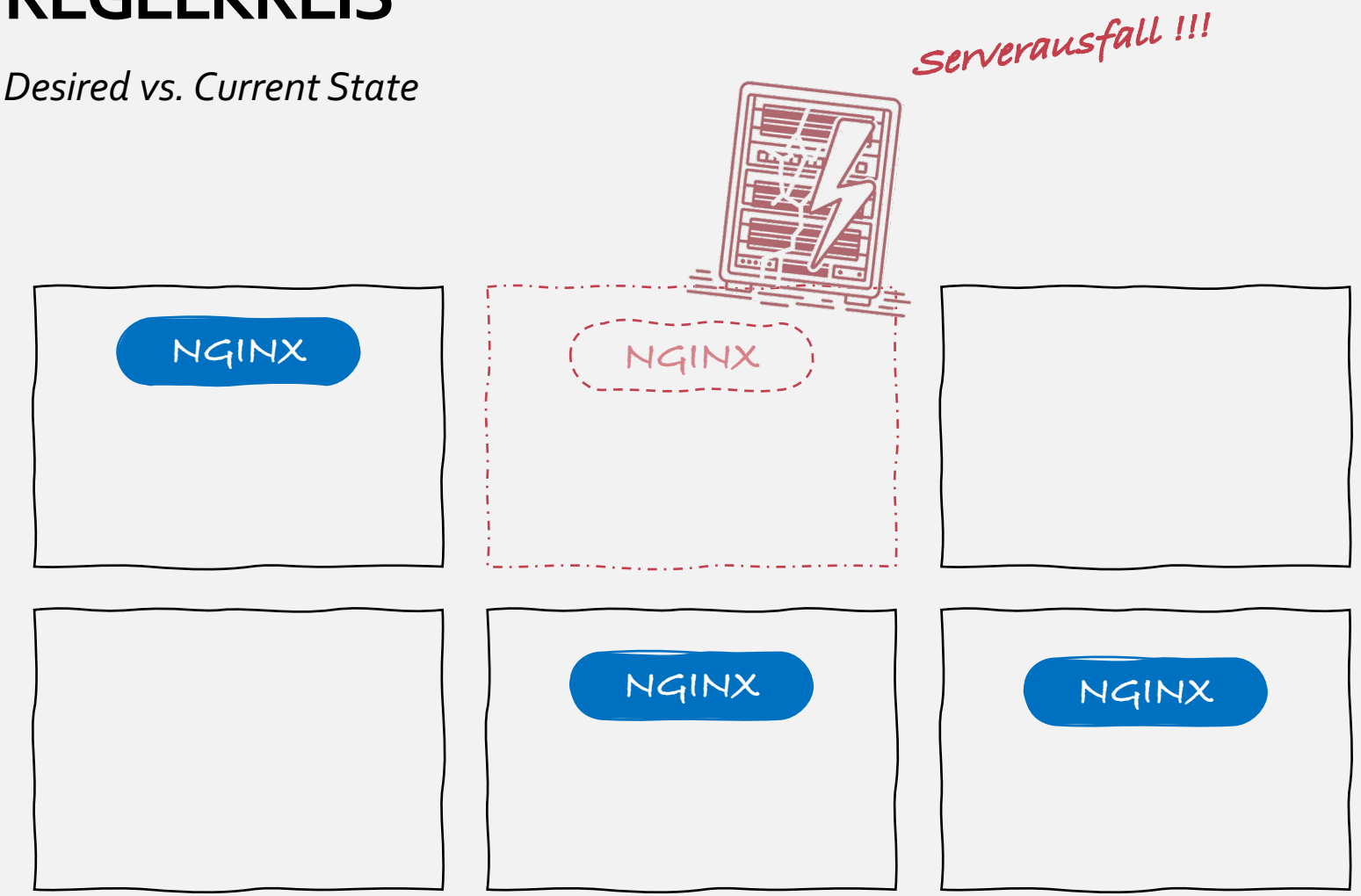
Desired State:
Betreibe 3 Instanzen von NGINX

Current State:
Es laufen 2 Instanzen von NGINX

Maßnahme:
Starte eine Instanz von NGINX

REGELKREIS

Desired vs. Current State



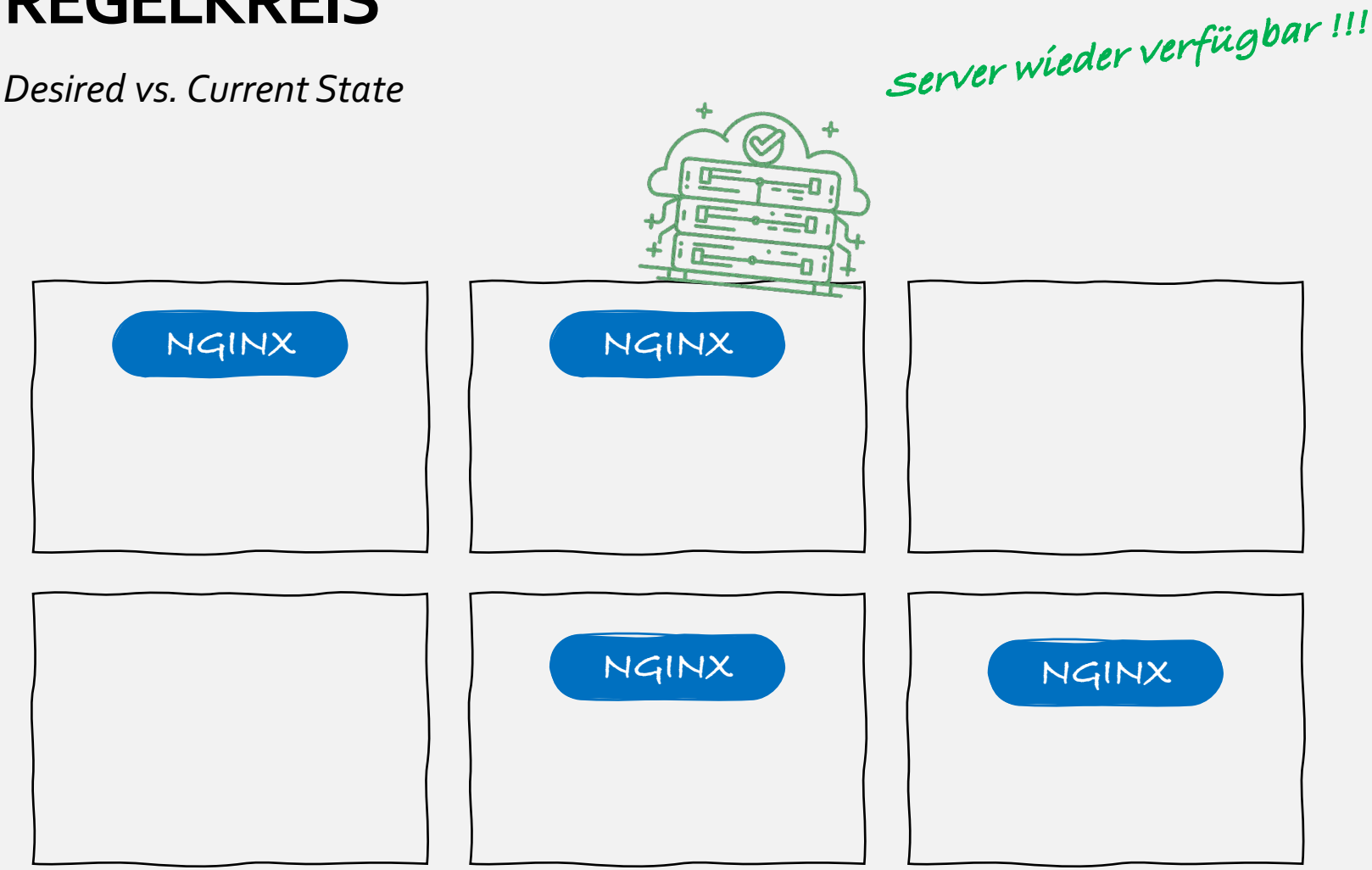
Desired State:
Betreibe 3 Instanzen von NGINX

Current State:
Es laufen 3 Instanzen von NGINX

Maßnahme:
keine

REGELKREIS

Desired vs. Current State



Erneutes Ereignis mit Einfluss auf den Desired State.

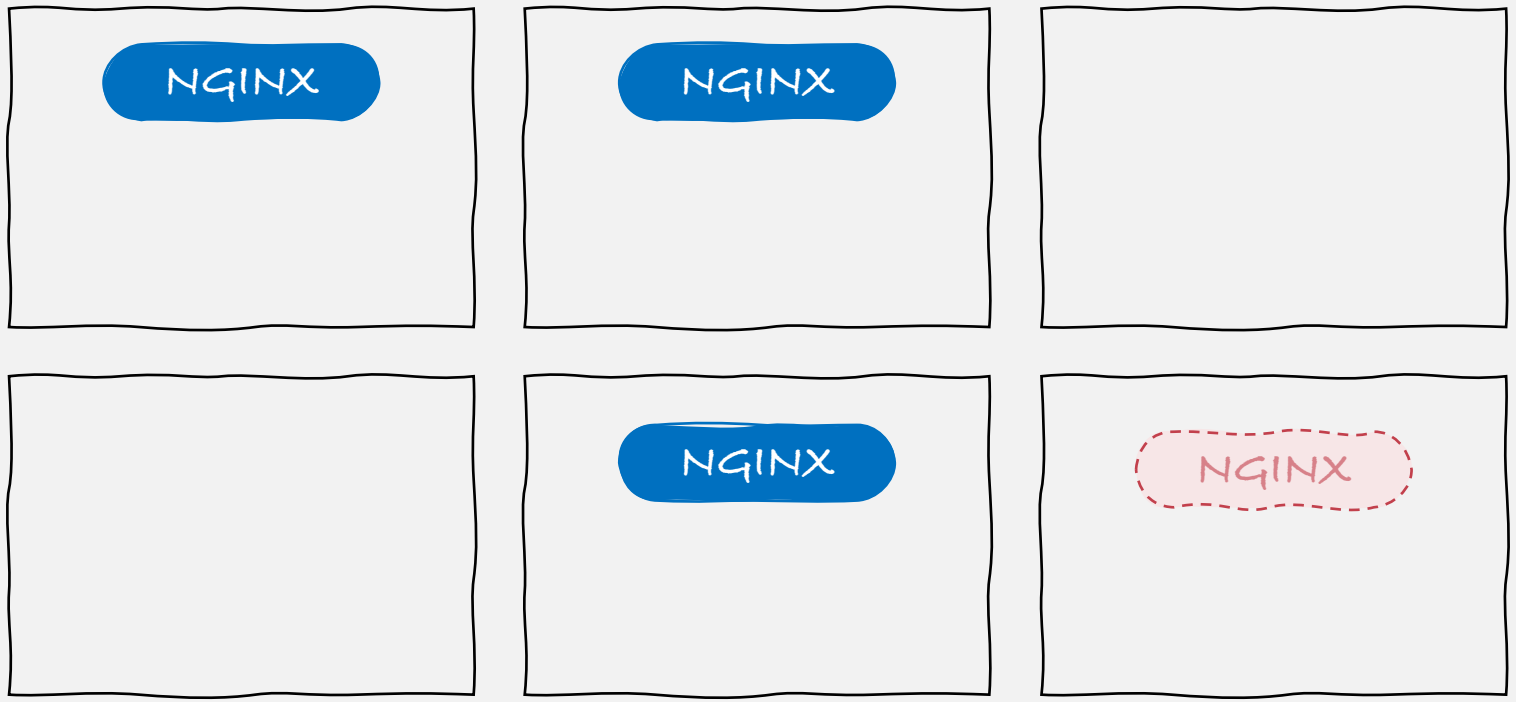
Desired State:
Betreibe 3 Instanzen von NGINX

Current State:
Es laufen 4 Instanzen von NGINX

Maßnahme:
Lösche eine NGINX Instanz

REGELKREIS

Desired vs. Current State



Planung einer Kompensationsmaßnahme.

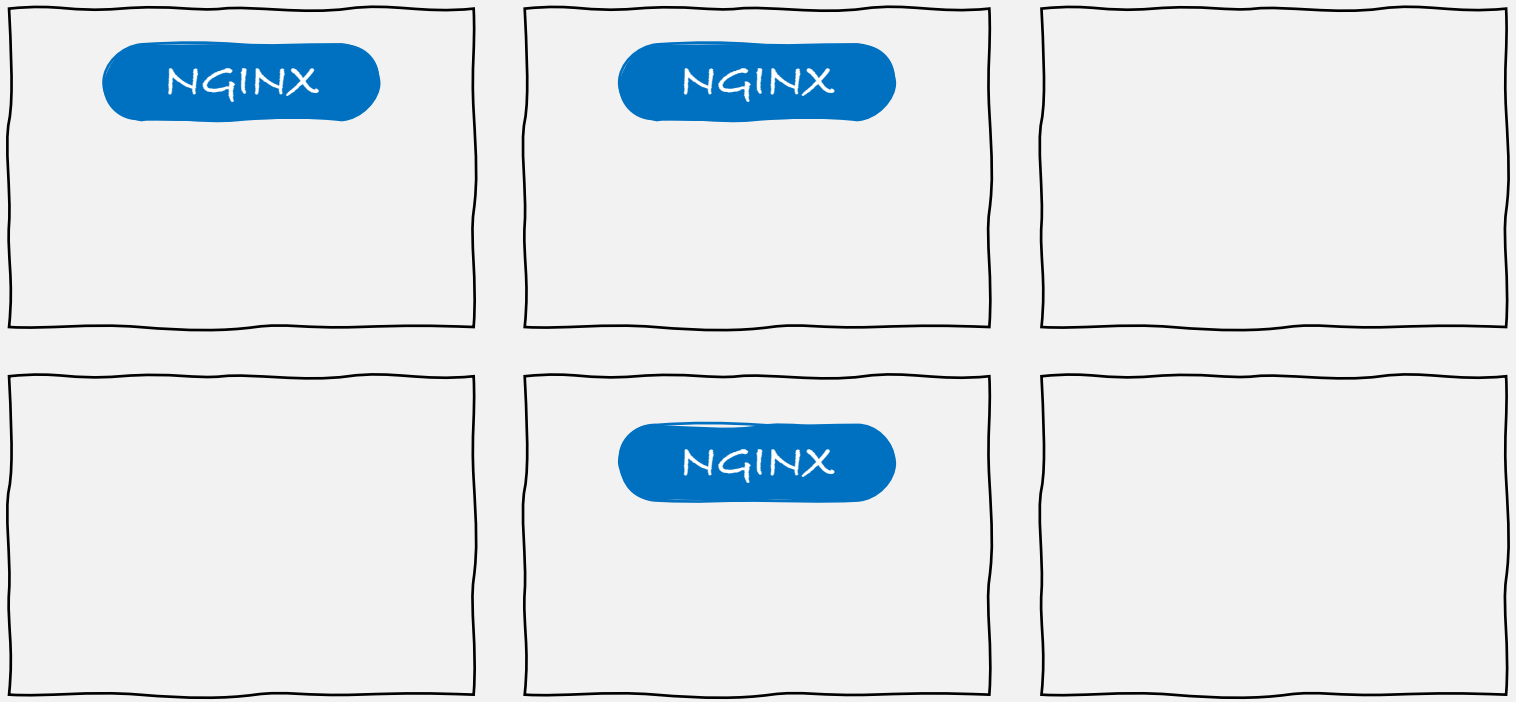
Desired State:
Betreibe 3 Instanzen von NGINX

Current State:
Es laufen 4 Instanzen von NGINX

Maßnahme:
Lösche eine NGINX Instanz

REGELKREIS

Desired vs. Current State



Desired State wieder erreicht !!!

Desired State:
Betreibe 3 Instanzen von NGINX

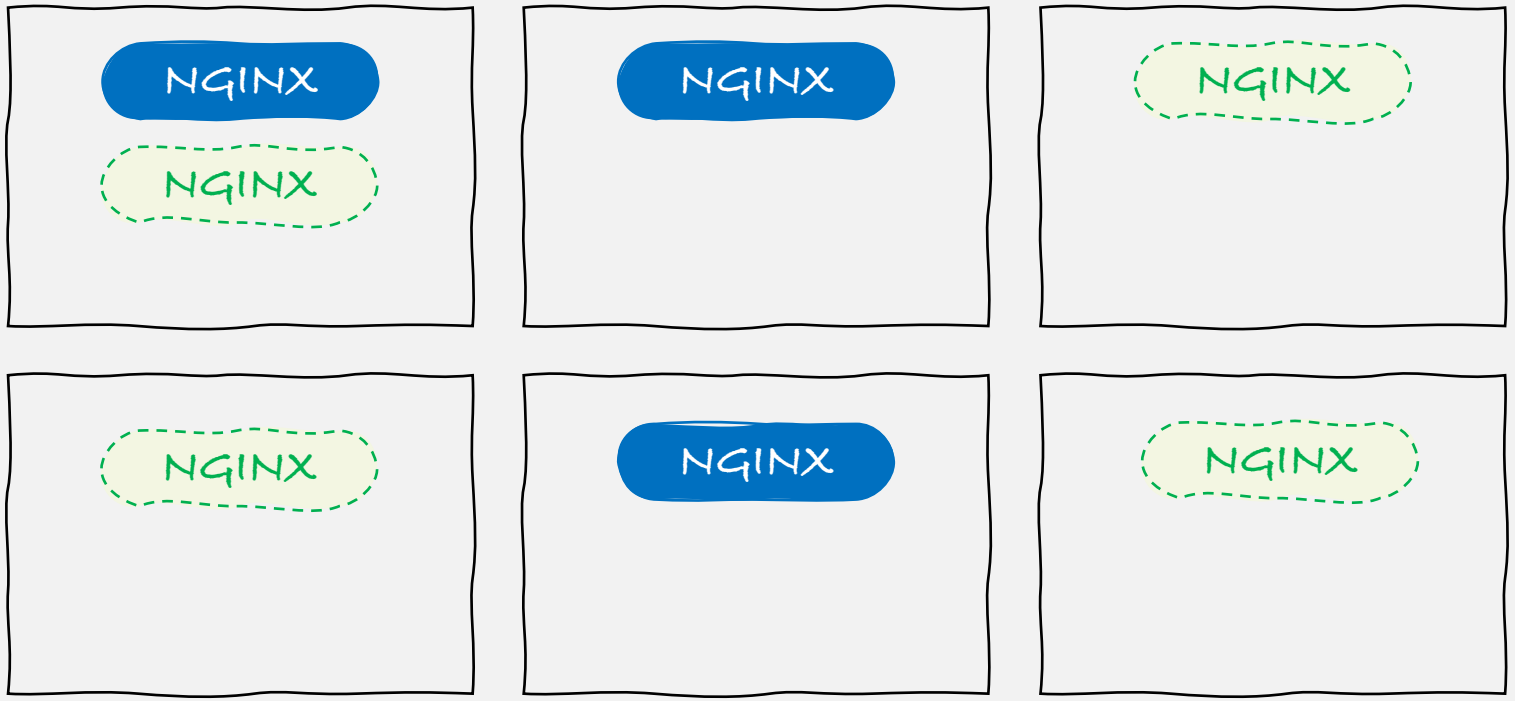
Current State:
Es laufen 3 Instanzen von NGINX

Maßnahme:
keine

REGELKREIS

Desired vs. Current State

Lastanstieg !!!



Änderung des gewünschten Betriebszustands.

Desired State:
Betreibe 7 Instanzen von NGINX

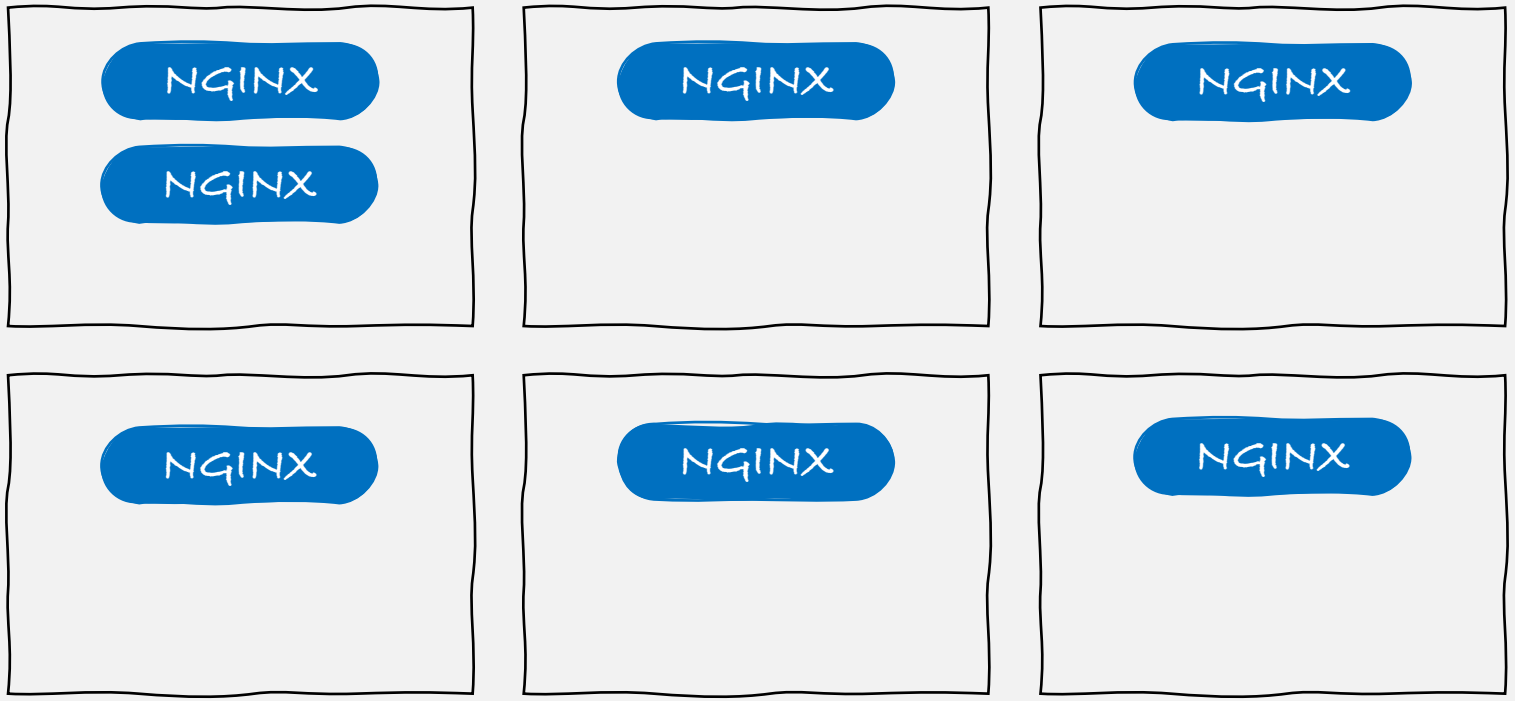
Current State:
Es laufen 3 Instanzen von NGINX

Maßnahme:
Starte 4 weitere Instanzen.

REGELKREIS

Desired vs. Current State

Lastanstieg !!!



Desired State wieder erreicht !!!

Desired State:
Betreibe 7 Instanzen von NGINX

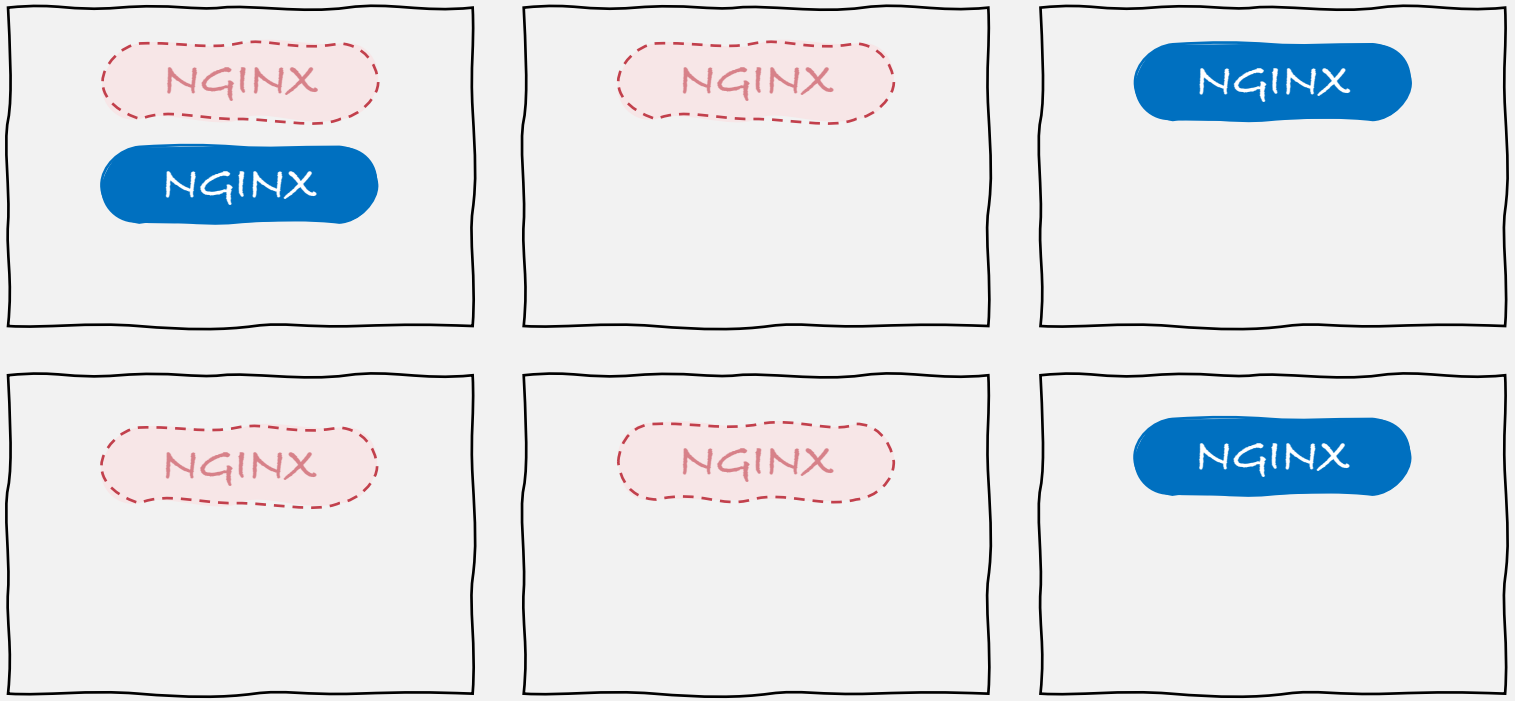
Current State:
Es laufen 7 Instanzen von NGINX

Maßnahme:
Keine

REGELKREIS

Desired vs. Current State

Lastabfall !!!



Erneute Änderung des gewünschten Betriebszustands.

Desired State:
Betreibe 3 Instanzen von NGINX

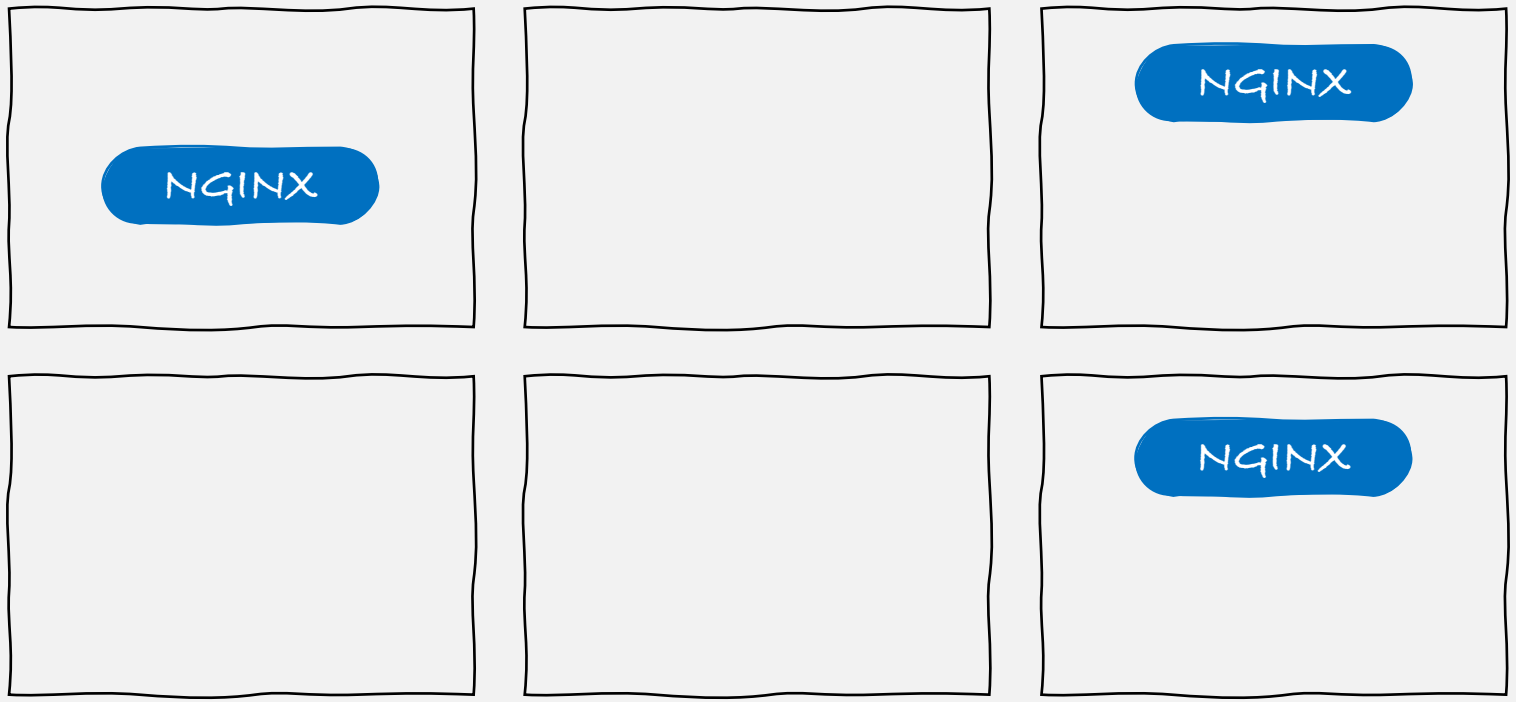
Current State:
Es laufen 7 Instanzen von NGINX

Maßnahme:
Stoppe 4 der 7 Instanzen.

CLOUD NATIVE COMPUTING

REGELKREIS

Desired vs. Current State



Desired State wieder erreicht !!!

Desired State:
Betreibe 3 Instanzen von NGINX

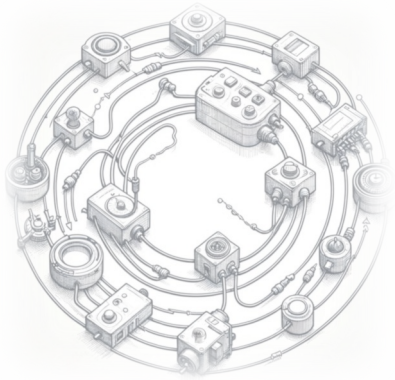
Current State:
Es laufen 3 Instanzen von NGINX

Maßnahme:
keine

REGELKREIS

Desired vs. Current State

- Regelkreise streben also von einem gegeben Ist-Zustand (I) einen Ziel-Zustand (Z) an.
- Dabei ist es egal, wieso der Ist-Zustand vom Ziel-Zustand abweicht.
  - ***Bewusst gesetzter neuer Ziel-Zustand*** (mehr Instanzen aufgrund höherer Last)
  - ***Ungewollt neuer Ist-Zustand*** (weniger Instanzen aufgrund von Ereignissen wie Serverabstürzen)
- Es werden in beiden Fällen nur Kompensationsmaßnahmen abgeleitet ($K = Z - I$), um den Ziel-Zustand zu erreichen.
- Der Ziel-Zustand ist erreicht, wenn $K = Z - I = 0$ gilt.
- Solche Regelkreise dienen sowohl dazu Ausfälle zu kompensieren (Resilienz, Self-Healing) als auch neue intendierte Zustände einzunehmen.



Merke:
Dieses einfache Prinzip ist die Basis vieler Container-fokussierter Orchestrierungs-lösungen (insb. Kubernetes).

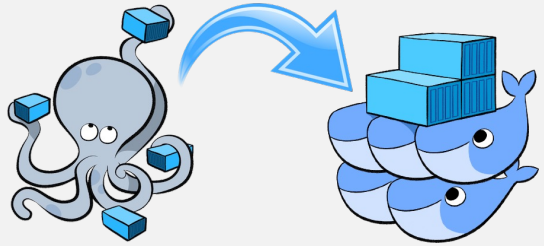
CLUSTER - ORCHESTRIERER

Überblick

Apache Mesos
Marathon + Chronos

Für grobe Rechenzentren

Docker
Compose + Swarm



Lightweight

Kubernetes

De-facto Standard

Eher Ops-fokussiert (DC, Data Center)

Mehr Dev-fokussiert

Anmerkung:
Die Systeme sind vom Prinzip her vergleichbar. Die Zielgruppen und damit die Feature-Schwerpunkte sind nur unterschiedlich.

Aus Gründen der Praktikabilität fokussieren wir im folgenden den De-facto Standard Typvertreter Kubernetes.

AUSBLICK

Scheduling

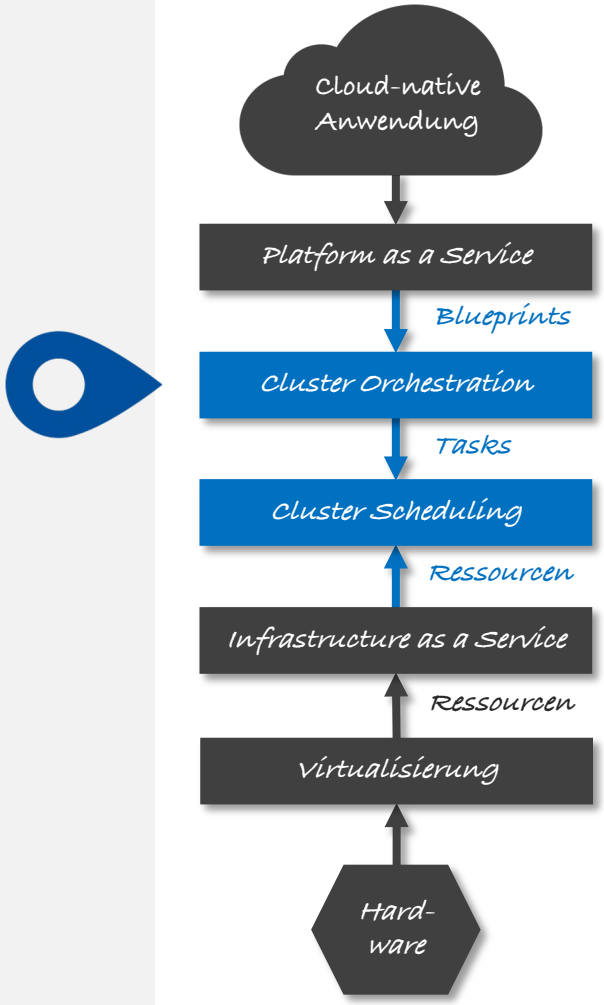
- Scheduling Problem Definition
- Scheduling Algorithmen
- Scheduler Architekturen
- Beispiele von Cluster Schedulern: Mesos, Swarm

Orchestrierung

- Was ist Orchestrierung (in Abgrenzung zum Scheduling)?
- Was sind Blueprints?
- Überblick über bestehende Orchestrierungslösungen

Inside Kubernetes (Typ-Vertreter)

- K8S-Architektur
- K8S-Ressourcen
- Workloads, Persistenz, Isolation und Exponieren von Services



KONTAKT

Disclaimer

Nane Kratzke

+49 451 300-5549

nane.kratzke@th-luebeck.de

kratzke.mylab.th-luebeck.de