LetsChangeTogether

FRAMEWORK MCR

Application Operations als Verantwortungs- und Kontrollsystem in marktkritischen Umgebungen

Ein architektur- und beobachtbarkeitsbasierter Ansatz zur Risikobegrenzung

Executive Summary

Daniel Siebert

Eigenentwicklung · Generalisiert · Version 1.0 · ohne Institutionsbezug

1. Ausgangslage und Kernproblem

In marktkritischen Systemen (wie ECC) ist der Betrieb nicht nur ein technischer, sondern ein geschäftskritischer und regulatorischer Faktor. Die Systeme müssen während definierter Marktzeiten mit höchster Verfügbarkeit und Nachvollziehbarkeit laufen. Gleichzeitig werden sie über einheitliche Delivery-Pipelines (CI/CD) ausgerollt, die jedoch nicht automatisch zu einheitlichen Risikoprofilen führen.

Das zentrale Problem liegt darin, dass viele Organisationen Delivery Architecture und Application Operations nicht ausreichend voneinander trennen. Pipelines sorgen für Reproduzierbarkeit des Deployments, können aber nicht die betriebliche Verantwortung, die Risikobewertung im laufenden Betrieb und die Eskalationsfähigkeit bei Störungen ersetzen. Besonders in regulierten, zeitkritischen Umgebungen führt diese Vermischung zu unklaren Verantwortlichkeiten und unzureichender operativer Kontrolle.

2. Kern des Ansatzes: Trennung von Delivery und Operations

Der zentrale Gedanke des Dokuments ist die konsequente Trennung von Delivery Architecture und Application Operations als eigenständige Verantwortungsbereiche.

Delivery Architecture definiert, wie Software gebaut, getestet und ausgerollt wird. Sie schafft Reproduzierbarkeit und reduziert bestimmte Klassen von Fehlern. Application Operations ist dagegen für den laufenden Betrieb, die Beobachtung des tatsächlichen Systemverhaltens und die Entscheidung über Eingriffe verantwortlich.

Diese Trennung ist nicht nur organisatorisch, sondern auch technisch relevant. Kubernetes-Plattformen stellen Schnittstellen bereit (z. B. durch Namespaces, RBAC, Policy Controller), die es ermöglichen, Verantwortungsbereiche klar abzugrenzen. Eine einheitliche Pipeline kann trotzdem zu sehr unterschiedlichen Risikoprofilen in verschiedenen Umgebungen führen – je nachdem, wie die Plattform und die Anwendung betrieben werden.

3. Observability als operatives Kontrollsystem

Ein weiterer zentraler Beitrag des Dokuments ist die Positionierung von Observability als operatives Kontrollsystem und nicht als reines Reporting- oder Monitoring-Tool.

Observability wird hier als die Fähigkeit verstanden, den tatsächlichen Zustand eines Systems während des Betriebs zu rekonstruieren und daraus steuerungsrelevante Entscheidungen abzuleiten. Dabei wird zwischen verschiedenen Signal-Ebenen unterschieden:

- Technische Signale (Golden Signals: Latency, Traffic, Errors, Saturation)
- Anwendungsspezifische Signale
- Geschäftsrelevante und regulatorische Signale

Besonders wichtig ist die Erkenntnis, dass in marktkritischen Systemen nicht einzelne Metriken, sondern das Zusammenspiel mehrerer Signale über die Zeit entscheidend ist. Application Operations muss in der Lage sein, aus diesem Zusammenspiel schnell zu erkennen, ob eine Abweichung nur technischer Natur ist oder bereits geschäftliche oder regulatorische Auswirkungen hat.

4. Konkretes technisches Beispiel: Konfigurationsänderung und Eskalation

4.1 Ausgangssituation

Eine Konfigurationsänderung wird über die bestehende Pipeline ausgerollt. Die Änderung ist technisch minimal und auditierbar dokumentiert. Nach dem Rollout zeigt das System zunächst keine offensichtlichen Fehler.

4.2 Beobachtung durch Application Operations

Application Operations beobachtet parallel mehrere Signale:

- Erhöhte Latenz bei bestimmten Transaktionen
- Leichter Anstieg von Fehlerraten (nicht kritisch, aber auffällig)
- Veränderung im Sättigungsverhalten einzelner Komponenten
- Abweichungen in anwendungsspezifischen Metriken, die nicht direkt aus den Golden Signals ableitbar sind

Während klassische Monitoring-Systeme möglicherweise nur einzelne Schwellwerte melden würden, erkennt Application Operations durch die gleichzeitige Betrachtung mehrerer Signale und deren zeitliche Entwicklung, dass hier ein systematisches Problem vorliegen könnte.

4.3 Eskalation und Verantwortung

Weil Application Operations als eigenständiger Verantwortungsbereich definiert ist, kann es frühzeitig eine qualifizierte Einschätzung abgeben, ob es sich um ein vorübergehendes Phänomen, ein Plattformproblem oder ein anwendungsspezifisches Risiko handelt. Die Entscheidung über weitere Maßnahmen (Rollback, Scale-up, manuelle Intervention) erfolgt auf Basis dieser operativen Bewertung und nicht allein auf Basis von Pipeline- oder Plattform-Logs.

Dieses Beispiel zeigt deutlich: Die Qualität der operativen Reaktion hängt nicht primär von der Qualität der Pipeline ab, sondern von der Fähigkeit von Application Operations, den tatsächlichen Systemzustand zu interpretieren und daraus steuerungsrelevante Schlüsse zu ziehen.

5. Wesentliche Stärken des Ansatzes

Der Ansatz des Dokuments zeichnet sich durch folgende Stärken aus:

- Klare Trennung von Delivery und Operations als eigenständige technische und organisatorische Disziplinen — reduziert Verantwortungsdiffusion und verbessert die Eskalationsfähigkeit
- Observability als aktives Kontrollsystem, nicht als passive Überwachung — Fokus auf Interpretation von Signalen im operativen Kontext
- Starke Betonung von Verantwortungsgrenzen (Separation of Duties) auf technischer Ebene, insbesondere durch Plattform-Schnittstellen (z. B. Kubernetes)
- Praxisnahe Fallbetrachtung, die zeigt, wie theoretische Prinzipien in realen, marktkritischen Störungssituationen angewendet werden
- Fokus auf Risikobegrenzung statt reiner Automatisierung — operative Urteilskraft bleibt in regulierten, zeitkritischen Systemen notwendig

6. Praktischer Nutzen

Der Ansatz bietet Organisationen, die marktkritische Systeme betreiben, einen klaren Rahmen, um Verantwortlichkeiten zwischen Delivery und Operations sauber zu trennen und Observability als operatives Steuerungsinstrument zu etablieren. Dadurch wird die Fähigkeit verbessert, in Störungssituationen schnell und fundiert zu reagieren, Risiken besser einzugrenzen und regulatorischen sowie betrieblichen Anforderungen gleichzeitig gerecht zu werden.

PRINCIPAL

Hinter der Marke

Daniel Siebert

Senior Finance & Cloud FinOps Consultant

Daniel Siebert ist Cloud FinOps & Financial Operations Consultant mit über 12 Jahren Erfahrung in Finance, SAP und IT-Management. Unter der Marke LetsChangeTogether entwickelt er finanzsteuerbare Betriebsmodelle für regulierte Hybrid-IT — von Architektur und Datenmodell bis zur governancefähigen Umsetzung. Schwerpunkte: Inform Layer, Kostenallokation, Tagging-Governance und die Verbindung von Cloud-Verbrauchsdaten mit Controlling und Management-Reporting.

LetsChangeTogether

Framework MCR · Application Operations in marktkritischen Systemen

daniel@siebert.cv · https://siebert.cv