Site Reliability Engineering – Tipps für Ihre Praxis

> SRE is what happens when you ask a software engineer to design an operations team.
>
> Ben Treynor, VP of Engineering at Google

consol
Wir unternehmen IT

Was versteht man unter Site Reliability Engineering?

Site Reliability Engineering ist ein DevOps-Ansatz von Google, der IT-Operations als Software-Problem betrachtet und mit Software Engineering löst.

- ✓ Software-Entwickler werden in die Betriebsverantwortung eingebunden.
- ✓ Prozesse und Systeme werden so gestaltet, dass sie mit Fehlern umgehen können.
- ✓ Continuous Delivery ermöglicht das regelmäßige Ausrollen vieler kleiner Releases, das Risiko jedes Einzelnen Releases wird so reduziert.
- ✓ Wiederkehrende Aufgaben werden automatisiert.
- ✓ Aufgaben werden so geplant, dass Zeit zur Automatisierung und Verbesserung bleibt.

Wie organisiert Google sein SRE-Team?

Ops-Aufgaben (Toil)

- Max 50% der Zeit für Ops-Aufgaben (Tickets bearbeiten, On-Call Alerts, manuelle Tasks, etc.)
- Sonst: Delegation von Ops-Aufgaben an Entwicklerteam des Services

On-Call Duty

- Max. 2 Alerts pro Schicht (8 oder 12 Stunden), Zeit für Post Mortem Analyse
- Min. 1 Alert pro Schicht, damit sich die Bereitschaft lohnt

Error Budget

- Bei SLO von 99,99% Verfügbarkeit darf das Entwicklerteam 0,01% als Error Budget ausgeben.
- Ziel: SREs müssen nicht immer „nein“ zum Entwicklerteam sagen

Was macht ein Site Reliability Engineering Team?

Ein SRE-Team ist ein Team von Software Engineers, das sich um den Betrieb von Services in Produktion kümmert.

Warum Software Engineers statt Sysadmins?

Betriebsaufgaben sollen mit Software und nicht manuell gelöst werden. Im klassischen Betrieb steigt die Arbeitslast linear O(n) mit Anzahl / Größe der Services. Gerade in modernen Microservice-Architekturen ist der klassische Betriebsansatz nicht mehr praktikabel. Ziel von Site Reliability Engineering ist, dass die Arbeitslast mit zunehmendem Einsatz von Software-Lösungen sinkt.

Wie ist ein SRE-Team organisiert?

Es gibt verschiedene Möglichkeiten, SRE-Teams zu organisieren. Google setzt auf drei Säulen: Der Anteil an Zeit, die SREs mit manuellen Aufgaben verbringen, wird begrenzt, so dass Zeit für die Entwicklung von SRE-Tooling bleibt. Bereitschaftsdienste werden so organisiert, dass im Fehlerfall genügend Zeit für eine gründliche Post-Mortem-Analyse bleibt. Error-Budgets sorgen dafür, dass die Entwickler eines Service und das SRE-Team am selben Strang ziehen, wenn es um die Risiko-Abwägung beim Go-Live neuer Features geht.

Welche Verantwortungsbereiche hat ein SRE-Team?

Ein SRE-Team ist verantwortlich für Verfügbarkeit, Latenz, Performance, Effizienz, Deployment, Monitoring, Emergency Response und Kapazitätsplanung der Services.

Wie organisiert Google sein SRE-Team?

Ops-Aufgaben (Toil)

- Max 50% der Zeit für Ops-Aufgaben (Tickets bearbeiten, On-Call Alerts, manuelle Tasks, etc.)
- Sonst: Delegation von Ops-Aufgaben an Entwicklerteam des Services

On-Call Duty

- Max. 2 Alerts pro Schicht (8 oder 12 Stunden), Zeit für Post Mortem Analyse
- Min. 1 Alert pro Schicht, damit sich die Bereitschaft lohnt.

Error Budget

- Bei SLO von 99,99% Verfügbarkeit darf das Entwicklerteam 0.01% als Error Budget ausgeben.
- Ziel: SREs müssen nicht immer „nein“ zum Entwicklerteam sagen

SRE Know-how bei ConSol

Kompetenz in den Kern-Verantwortungsbereichen beim Betrieb von Services: Verfügbarkeit, Latenz, Performance, Effizienz, Deployment, Monitoring, Emergency Response, Kapazitätsplanung

Software Engineering: Software-Engineers bringen ihre langjährige Projekterfahrung bei der Implementierung moderner Microservice-Architekturen in dynamischen Cloud-Umgebungen ein.

OpenShift und Cloud Beratung: Unsere Experten für Cloud-Infrastruktur und Container-Runtimes bringen ihr Know-how zu Konfigurationsmanagement, Service Discovery, Deployment Patterns, Auto-Scaling und Container-Betrieb ein.

Monitoring: High-End-Beratung beim Monitoring ist seit Jahren fester Bestandteil des ConSol-Portfolios. Unser Monitoring-Team unterstützt bei Definition und Betrieb von Metriken und Alerting in hoch-dynamischen Microservice-Landschaften.

IT-Operations: Mit ihrer langjährigen Erfahrung im klassischen Server- und Applikationsbetrieb unterstützt unser IT-Operations-Team mit viel Linux/Unix Know-how und Erfahrung beim Troubleshooting.

Tipps aus der Praxis

1) Health Checks: Endpunkte für Liveness- und Readiness-Probes sind oft sehr einfach implementiert: Sie antworten mit 200 OK sobald die Applikation gestartet ist. Wir haben in Projekten die Erfahrung gemacht, dass das nicht ausreicht. Deshalb sind wir dazu übergegangen, in Health Checks die Erreichbarkeit aller angrenzenden Systeme und Message Queues zu testen. Damit können Probleme bereits beim Deployment erkannt und im Idealfall automatisiert behoben werden.

2) EJB Timer Services: In einem Projekt hatten wir ein Problem mit EJB Timer Services, das dazu geführt hat, dass nach jedem Lauf die Transaktion zurückgerollt wurde. Das Zurückrollen der Transaktion an sich ist unproblematisch, sofern einer der nächsten Läufe erfolgreich ist. Um herauszufinden, ob es sich um erwartete Rollbacks oder um echte Fehler handelt, haben wir schließlich eine Metrik implementiert, die die Zeit seit dem letzten erfolgreichen Lauf misst. Dadurch konnten wir temporäre Fehler von dauerhaften Fehlern unterscheiden.

3) Thread Dumps: Bei Java Applikationen lohnt es sich, regelmäßig Thread Dumps zu ziehen. Die Thread Dumps helfen bei der Post Mortem Analyse und beim Profiling. Zum Beispiel kann man an der Entwicklung der Thread Dumps gut sehen, wenn ein externes System blockiert ist und dadurch ständig neue Threads mit blockierten Aufrufen gestartet werden. Insbesondere ist es eine gute Idee, zwei bis drei Thread Dumps im Stopp-Script zu ziehen, damit man nach Neustart der Applikation analysieren kann, wie der Zustand vor dem Neustart war.

4) Logging: Logging Frameworks bieten mit dem Mapped Diagnostic Context (MDC) die Möglichkeit, Informationen wie z.B. den Usernamen standardmäßig mit zu loggen. Dadurch lässt sich bei der Loganalyse nachvollziehen, welche Logzeilen zusammengehören. Die MDC-Daten sind jedoch nicht immer verfügbar, z.B. wenn der User noch nicht ermittelt ist. Deshalb lohnt es sich, zusätzlich den Thread ins Logformat aufzunehmen. Der Thread bietet eine sichere und einfache Möglichkeit nachzuvollziehen, welche Logzeilen zum selben Request gehören.

Tipps aus der Praxis

5) Service-übergreifendes Throttling:
In einem Projekt aus der Telekommunikationsbranche hatten wir die Herausforderung, dass viele Microservices denselben Endpunkt aufrufen, aber die Gesamtzahl der Aufrufe einen bestimmten Schwellwert pro Sekunde nicht überschreiten durfte. Wir haben das gelöst, indem wir Zookeeper zur Koordination der Aufrufe eingesetzt haben. Dadurch konnten wir vermeiden, ein zentrales Koordinationssystem als Single Point of Failure zu schaffen.

6) Automate Everything: Manuelle Schritte beim Build und Deployment sind eine häufige Fehlerquelle. Hier lohnt es sich, alles zu automatisieren. Moderne CI/CD-Pipelines verringern nicht nur das Fehlerrisiko, sondern nehmen den SREs auch lästige wiederkehrende Aufgaben ab.

7) Lasttests: Die größte Herausforderung bei Lasttests ist das Erzeugen realistischer Testdaten. Dazu gehört nicht nur der Inhalt der Daten. In einem großen Migrationsprojekt haben wir die Erfahrung gemacht, dass die Art der Fragmentierung von Daten in einer Datenbank erhebliche Auswirkungen auf die Performance haben kann. Für den Projekterfolg war entscheidend, das bereits in der Lasttestphase festzustellen.

8) Instrument Everything: Je mehr Messpunkte eine Applikation hat, um so besser. Das hilft nicht nur im Betrieb. Zum Beispiel ist der Wert von Lasttests wesentlich höher, wenn man nicht nur sagen kann, ob ein Service die SLOs einhält, sondern wenn man gleichzeitig lernt, wo die potentiellen Bottlenecks sind.

9) Fail fast: Moderne Design Patterns wie Circuit Breaker werden gerne von Software Engineers verwendet. Man sollte jedoch die klassischen Konfigurationsmöglichkeiten nicht aus den Augen verlieren, um Überlast zu vermeiden. Zum Beispiel sollten Poolgrößen in Java Application Servern so ausgelegt werden, dass bei einer unerwarteten Lastspitze möglichst weit "vorn" ein Anschlag erreicht wird und nachgelagerte Komponenten nicht überlastet werden.

10) Know Your System: Site Reliability Engineers müssen sich mit dem Normalverhalten ihrer Services vertraut machen und regelmäßig in ihre Logs schauen. Sonst wird im Fehlerfall viel Zeit verschwendet, um Merkwürdigkeiten nachzugehen, die mit der akuten Störung gar nichts zu tun haben.

ConSol: Spezialisten. Umsetzer. Innovationstreiber.

ConSol steht für passgenaue IT-Lösungen im gesamten Software Lifecycle: High-End-IT-Beratung, agile Entwicklung, Betrieb & Support.

Individuelle Business-Anforderungen smart umgesetzt

Hochkarätige Beratungsleistungen nach Maß sind unsere Spezialität – ob beim Kunden oder in unseren Headquarter in München. So schaffen wir Ihnen Freiräume für Ihr Kerngeschäft und Innovationen.

Geballtes Know-how – erprobt und herstellerunabhängig

Dank kontinuierlichem R&D und gezielten Weiterbildungen sind wir am Puls der technologischen Entwicklungen. Wir bieten praxistaugliche Lösungen – immer Ihre Unternehmensziele fest im Fokus.

ConSol: Über 30 Jahre Expertise

Unser Portfolio

IT Consulting

Beratung und Unterstützung bei der Gestaltung von IT- und Veränderungsprojekten

IaaS, PaaS und SaaS: Bedarfsgerechte Services für On-Premise-, hybride oder Public-Cloud-Lösungen

Software und Services

Kundenindividuelle Softwareentwicklung und -architektur

7x24-Betrieb Ihrer Infrastruktur und Applikationen

DevOps: Maßgeschneiderte Konzepte zur Integration von Development und Operations

Open Source

Eigenentwickelte Open-Source-Lösungen

Herstellerneutrale Beratung und Einsatz von Open-Source-Komponenten

Monitoring-Systeme für IT-Infrastruktur & Applikationen: Implementierung und Betrieb

ConSol CM

Unsere eigene Software-Lösung zur individuellen Umsetzung von Unternehmensprozessen:

ConSol CM bildet aufbauend auf einer Low Code Plattform BPM-, CRM- und Case-Management-Funktionalitäten ab.

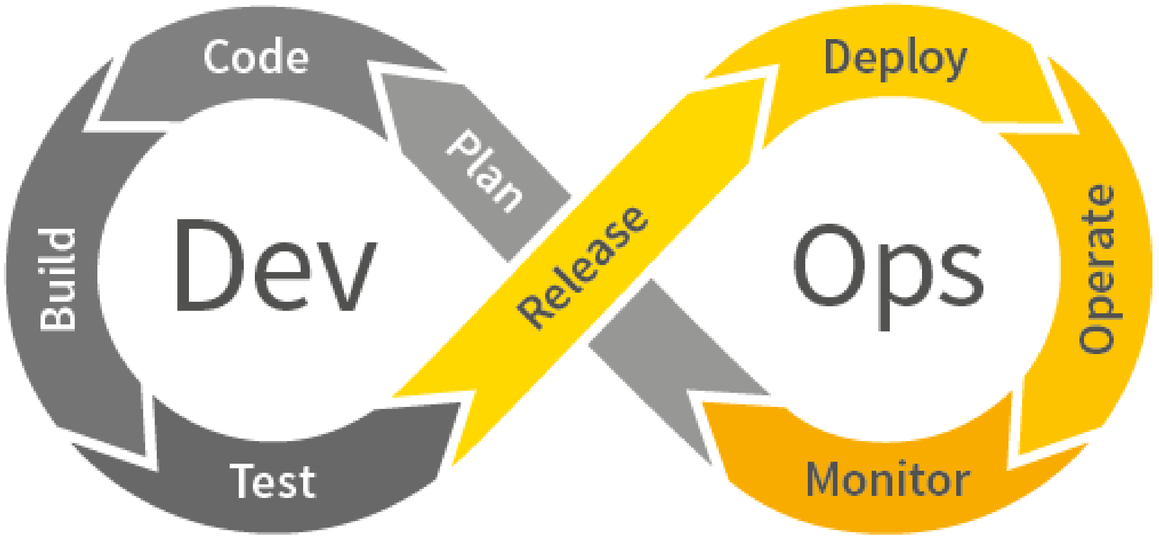
consol

Unser Anspruch: Agile Software-Entwicklung

Gemeinsam mit unseren Kunden entwickeln wir Lösungen, die passgenau abgestimmt sind - durch einen partnerschaftlichen Ansatz, kontinuierliche Lieferung und Zielüberprüfung.

Unser Technologie-Schwerpunkte:

- ✓ Moderne Web-Applikationen
- ✓ Komplexe Integrationsprojekte
- ✓ Moderne Softwarearchitektur
- ✓ Cloud Native
- ✓ Site Reliability Engineering
- ✓ Testautomatisierung
- ✓ Open Source Monitoring



Unser Weg: Scrum, Continuous Integration, Continuous Deployment, Kanban

ConSol
Consulting & Solutions Software GmbH

Franziskanerstr. 38
D-81669 München
Tel.: +49-89-45841-100
info@consol.de
www.consol.de
Twitter: @consol_de