DevOps – so geht es richtig

Best Practices zur Überwindung von Erfolgshindernissen

Inhaltsverzeichnis

Einführung

Ihr Team hat DevOps eingeführt. Sie entwickeln neue Verfahren, nutzen neue Tools und schaffen ein Umfeld, in dem funktionsübergreifende Zusammenarbeit gepflegt wird. Doch Sie arbeiten noch nicht mit maximaler Geschwindigkeit. Eine Sache fehlt noch, damit aus Ihrem Unternehmen eine echte DevOps-Hochleistungsmaschine wird.

Dieses fehlende Teil ist häufig das Messen von Daten. Obwohl das Messen eine der fünf Säulen des vom DevOps-Experten Jez Humble formulierten CALMS-Frameworks (Culture, Automation, Lean, Measurement, Sharing) ist, wird es von DevOps-Teams in ihrem Streben nach mehr Geschwindigkeit und Autonomie häufig vernachlässigt. Das kann jedoch zu enormen Problemen führen, da präzise Daten für die Funktion und den Erfolg eines DevOps-Teams – die effektive Vorfallreaktion, die Unterhaltung komplexer Mikroservices und vieles mehr – entscheidend sind.

Dieses E-Book ist für alle Teams und Organisationen gedacht, die ihre ersten Schritte mit DevOps getan haben und jetzt voll einsteigen möchten. Und für diejenigen, die noch nicht so recht wissen, wie sie mit DevOps zu einer wirklichen digitalen Transformation gelangen sollen.

Mit unseren praktischen Erfahrungen und den Lektionen, die wir hier bei New Relic gelernt haben, möchten wir Ihnen helfen, die verbleibenden Hürden auf Ihrem Weg zum DevOps-Erfolg abzubauen. Vom Festlegen von Zuverlässigkeitszielen bis zum Bewältigen der einzigartigen Kommunikations- und Entwicklungsanforderungen Ihres Microservice-Ansatzes haben wir für Sie bewährte Best Practices für schnelleres und effektiveres Arbeiten zusammengestellt.

NEW RELIC DAMALS UND HEUTE: DER WEG ZU DEVOPS

Ruby-Monolith	200+ Microservices
Isolierte Teams	50+ Engineering-Teams mit integrierten Site Reliability Engineers
Seltene Releases	Bis zu 70 Bereitstellungen pro Tag
Reaktive Maßnahmen	Proaktive Überwachung und Maßnahmen

Verarbeitet heute 1,8+ Mrd. Ereignisse und Metriken pro Minute

KAPITEL 1

SLOs und rasche Anwendungsbereitstellung

SLOs und rasche Anwendungsbereitstellung

Der nächste Schritt auf dem Weg zu höherer Zuverlässigkeit und schnelleren Bereitstellungen besteht darin sicherzustellen, dass Ihre Organisation auftretende Softwareprobleme jederzeit – Tag und Nacht – rasch und effektiv beheben kann. Dazu brauchen Sie Bereitschaftsrichtlinien.Halt … blättern Sie noch nicht zum nächsten Kapitel weiter. Wir wissen, dass viele auf den Begriff „Bereitschaft" allergisch reagieren. Aber das liegt hauptsächlich daran, dass viele Organisationen das Konzept der Rufbereitschaftsrotation nicht richtig umsetzen. Und das führt nicht nur zu Stress und negativer Aufmerksamkeit für die Nichteinhaltung Ihrer SLAs gegenüber den Kunden, sondern auch zu einem unproduktiven und unangenehmen Arbeitsumfeld in einem Team erschöpfter und frustrierter Techniker.

> *„Im Grunde [ist SRE] das, was geschieht, wenn Sie einen Softwareingenieur bitten, eine Operationsfunktion zu entwickeln."*
>
> Ben Treynor Sloss, Vice President of Engineering, Google

Lassen Sie uns zunächst einige wichtige Begriffe abklären:

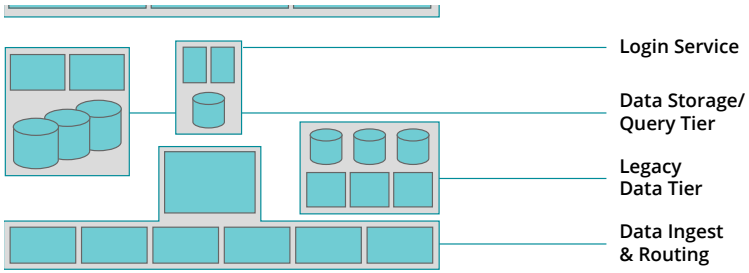
BEGRIFF	DEFINITION	BEISPIEL
Service Level Indicator (SLI)	Der SLI ist Ihr wichtigster Performance-Maßstab.	„Kunden können sich einloggen und ihre Daten anzeigen …"
Service Level Objective (SLO)	SLOs sind die Sollwerte oder Performanceziele Ihres Systems. SLOs stellen eine dauerhafte Verpflichtung dar.	„99,9 % der Zeit …"
Service Level Agreement (SLA)	Im SLA ist festgelegt, was geschieht, wenn Sie Ihre SLI/SLO-Verpflichtungen nicht einhalten.	„… oder können eine Erstattung der ihnen aufgrund der Nichtverfügbarkeit des Service entstandenen Verluste anfordern."

Mehr über SRE erfahren Sie in unserem E-Book Site Reliability Engineering: Philosophien, Gewohnheiten und Tools für den SRE-Erfolg

Festlegen geeigneter SLIs und SLOs

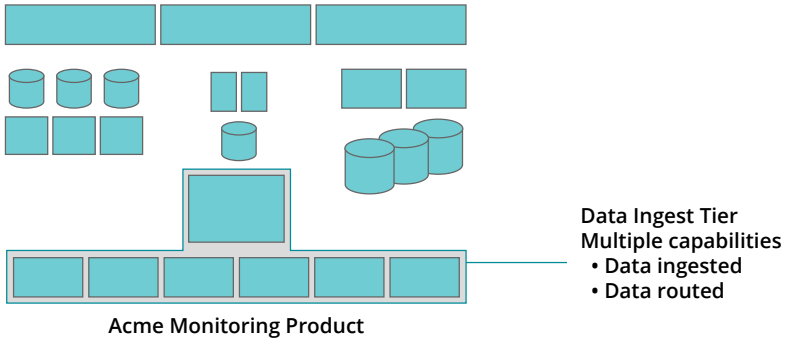
Gemäß den Best Practices der Branche im Hinblick auf SRE sollten Sie für jeden bereitgestellten Service konkrete SLIs und SLOs festlegen und anwenden. Das ist nicht immer ganz einfach, vor allem, wenn Sie es noch nie zuvor getan haben. Bei New Relic verwenden wir für das Festlegen von SLIs und SLOs ein Verfahren, das die folgenden sieben Schritte umfasst:

1. **Systemgrenzen bestimmen:** Eine Systemgrenze verläuft dort, wo eine oder mehrere Komponenten eine oder mehrere Funktionen für externe Kunden bereitstellen. Ihre Plattform kann viele bewegliche Teile – Service-Knoten, Datenbank, Load Balancer usw. – aufweisen, die nicht als Systemgrenzen gelten, weil ihre Funktionen nicht unmittelbar von Kunden genutzt werden. Vielmehr bilden sie gemeinsam mit anderen Komponenten ein Gesamtsystem, auf dessen Funktionen die Kunden zugreifen können. So besteht zum Beispiel ein Login-Service aus einer logischen Gruppe von Komponenten, die als System zusammenarbeiten und eine API für die Authentifizierung von Benutzeranmeldedaten bereitstellen. Bevor Sie Ihre SLIs festlegen, sollten Sie deshalb die Elemente Ihrer Plattform zu Systemen zusammenfassen und deren Grenzen definieren. In den weiteren Schritten werden Sie sich dann auf die Systeme konzentrieren, an deren Grenzen SLIs und SLOs den größten Nutzen bringen.



Systeme und Grenzen innerhalb einer Plattform

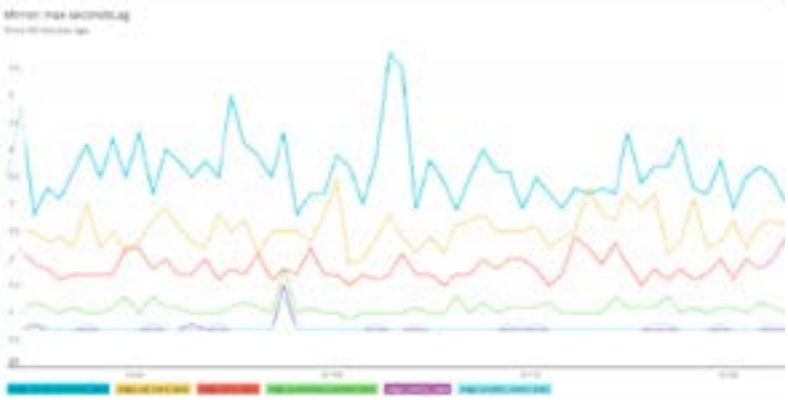
2. **Die von den einzelnen Systemen bereitgestellten Funktionen definieren:** Fassen Sie nun die Komponenten der Plattform zu logischen Einheiten zusammen (z. B. UI/API-Schicht, Login-Service, Datenspeicher-/Abfrageschicht, Legacy-Datenschicht, Datenerfassung & Routing). Bei New Relic decken sich unsere Systemgrenzen mit den Grenzen unseres Engineering-Teams. Anhand dieser Gruppierungen benennen Sie die an den jeweiligen Systemgrenzen bereitgestellten Funktionen.



An einer Systemgrenze definierte Funktionen

3. **Für jede Funktion eine eindeutige Definition von „Verfügbarkeit" formulieren:** Die Verfügbarkeit einer Datenroutingfunktion könnte beispielsweise mit „Zustellung von Nachrichten an den richtigen Empfänger" beschrieben werden. Beugen Sie Missverständnissen vor, indem Sie die erwartete Verfügbarkeit in klaren und einfachen Worten beschreiben und technische Fachbegriffe vermeiden, mit denen nicht jedermann vertraut ist.
4. **Entsprechende technische SLIs festlegen:** Jetzt ist es an der-Zeit, SLIs festzulegen, an denen Sie die für die einzelnen Funktionen definierte Verfügbarkeit messen können. In unserem Beispiel könnte ein SLI für die Datenroutingfunktion etwa „Dauer der Zustellung einer Nachricht an den richtigen Empfänger" lauten.

5. **Baselines bestimmen:** Ob Sie Ihre Verfügbarkeitsziele erreichen oder nicht, erkennen Sie natürlich mittels Monitoring. Sammeln Sie mit Ihrem Monitoring-Tool Baseline-Daten für jeden SLI, bevor Sie Ihre SLOs festlegen.
6. **SLO-Ziele festlegen (je SLI/Funktion):** Nachdem Sie nun über die erforderlichen Daten verfügen, fragen Sie Ihre Kunden nach ihren Erwartungen und überlegen Sie, wie Sie Ihre SLOs auf diese Erwartungen abstimmen können. Legen Sie anschließend auf der Grundlage Ihrer Baselines, der Angaben Ihrer Kunden, der Leistungsbereitschaft Ihres Teams und Ihrer technischen Möglichkeiten erreichbare SLO-Ziele fest. In unserem SLI-Beispiel könnte das SLO für das Datenrouting etwa „99,5 % der Nachrichten in weniger als 5 Sekunden zugestellt" lauten. Vergessen Sie nicht, in Ihrer Monitoring-Anwendung einen Alarmtrigger mit einer Warnschwelle für die von Ihnen definierten SLOs zu konfigurieren.



SLI-Beispiel für die Datenroutingfunktion

7. **Iterieren und Tunen:** Betrachten Sie das Festlegen Ihrer SLIs und SLOs nicht als einmalige Aufgabe. Gehen Sie vielmehr davon aus, dass Sie sie im Laufe der Zeit an die Entwicklung Ihrer Services und der Bedürfnisse Ihrer Kunden anpassen müssen.

Weitere Tipps zu SLIs und SLOs

- **Legen Sie für jede logische Instanz eines Systems ein eigenes SLO fest:** Bei fest segmentierten (im Gegensatz zu horizontal skalierten) Systemen sollten die SLIs und SLOs beispielsweise für jedes Segment gesondert gemessen werden.
- **Bedenken Sie, dass SLIs nicht dasselbe sind wie Alarme:** Der SRE-Prozess ist kein Ersatz für umfassende Alarmierung.
- **Fassen Sie SLOs gegebenenfalls zusammen:** Unter Umständen können Sie die Ziele für mehrere SLI-Bedingungen in einem SLO zusammenfassen, das für Kunden leichter verständlich ist.
- **Legen Sie bei Bedarf kundenspezifische SLOs fest:** Nicht selten wird großen Kunden in den SLAs eine höhere Serviceverfügbarkeit zugesichert als anderen Kunden.

> *„In unserem Streben nach Operational Excellence messen wir einfach alles. Nur so können wir auch wirklich alles managen und verbessern."*
>
> Craig Vandeputte, Director of DevOps, CarRentals.com

KAPITEL 2

Faire und effektive Bereitschaftsrichtlinien

Faire und effektive Bereitschaftsrichtlinien

Der nächste Schritt auf dem Weg zu höherer Zuverlässigkeit und schnelleren Bereitstellungen besteht darin sicherzustellen, dass Ihre Organisation auftretende Softwareprobleme jederzeit – Tag und Nacht – rasch und effektiv beheben kann. Dazu brauchen Sie Bereitschaftsrichtlinien.

Halt … blättern Sie noch nicht zum nächsten Kapitel weiter. Wir wissen, dass viele auf den Begriff „Bereitschaft" allergisch reagieren. Aber das liegt hauptsächlich daran, dass viele Organisationen das Konzept der Rufbereitschaftsrotation nicht richtig umsetzen. Und das führt nicht nur zu Stress und negativer Aufmerksamkeit für die Nichteinhaltung Ihrer SLAs gegenüber den Kunden, sondern auch zu einem unproduktiven und unangenehmen Arbeitsumfeld in einem Team erschöpfter und frustrierter Techniker.

Schaffen Sie die Grundlagen

Effektive und faire Bereitschaftsrichtlinien setzen zwei wichtige Dinge voraus:

1. **Eine klare System- und Organisationsstruktur:** Effektiv auf Probleme zu reagieren ist wesentlich einfacher, wenn sowohl Ihre Systeme (Services oder Anwendungen) als auch Ihre Produktteams gut organisiert und in logische Einheiten gegliedert sind. Bei New Relic unterstützen unsere 57 Engineering-Teams beispielsweise 200 individuelle Services, wobei jedes Team für den gesamten Produktlebenszyklus von der Entwicklung über die Bereitstellung bis zur Wartung von mindestens drei Services zuständig ist.
2. **Eine Kultur der Accountability:** Jedes DevOps-Team ist voll für den von ihm bereitgestellten Code verantwortlich. Ein Team trifft naturgemäß andere Entscheidungen im Hinblick auf Änderungen und Bereitstellungen, wenn es für seinen Service und die Rufbereitschaft selbst verantwortlich ist, als wenn wie in einer herkömmlichen Umgebung jemand anderes für den Support zuständig ist, sobald der Code in Betrieb geht.

Best Practices zur Verbesserung Ihrer Rufbereitschaft

Strukturieren Sie Ihr Team und Ihre Organisation fair

Bei New Relic sind alle Techniker und Engineering-Manager in der Produktorganisation reihum auf Abruf für die Services ihres jeweiligen Teams zuständig. Jedes Team ist für mindestens drei Services verantwortlich, wobei die Anzahl der unterstützten Services von deren Komplexität und der Teamgröße abhängt. Berücksichtigen Sie bei der Organisation Ihrer Bereitschaftsrotation sowohl die Größe Ihrer Engineering-Abteilung insgesamt als auch die Größe der einzelnen Teams. Wenn ein Team beispielsweise aus sechs Technikern besteht, könnte jeder von ihnen alle sechs Wochen zur Bereitschaft eingeteilt werden.

Seien Sie flexibel und kreativ bei der Gestaltung der Rotation

Am besten lassen Sie die einzelnen Teams ihre jeweiligen Rotationsrichtlinien selbstständig planen und umsetzen. Auf diese Weise können diese ihre Rotation ungehindert so organisieren, wie es ihren individuellen Bedürfnissen am besten entspricht. Bei New Relic ist jedes Team selbst für die Planung und Umsetzung

seines eigenen Bereitschaftssystems zuständig. Eines der Teams verwendet zum Beispiel ein Skript, das die Vertretung für den Bereitschaftsdienst nach dem Zufallsprinzip einteilt.

Verfolgen Sie Metriken und überwachen Sie Vorfälle

Ein wichtiger Aspekt der Fairness und Effektivität der Rufbereitschaftsrotation ist die Überwachung und Verfolgung von Vorfallsmetriken. Bei New Relic verfolgen wir die Anzahl der Einsätze, die Anzahl der geleisteten Arbeitsstunden und die Anzahl der Bereitschaftseinsätze. Diese Metriken betrachten wir auf Techniker-, Team- und Gruppenebene. Durch die Verfolgung der Metriken lässt sich erkennen, ob ein Team über die Maßen beansprucht wird. (Wenn ein Team im Durchschnitt mehr als einen Bereitschaftseinsatz pro Woche leistet, gilt die Bereitschaftsbelastung dieses Teams als hoch.) Wenn wir diese Metriken im Auge behalten, können wir uns bei Bedarf darauf konzentrieren, Mängel in der technischen Ausstattung eines Teams zu beheben oder die Services durch verstärkten Support zu verbessern.

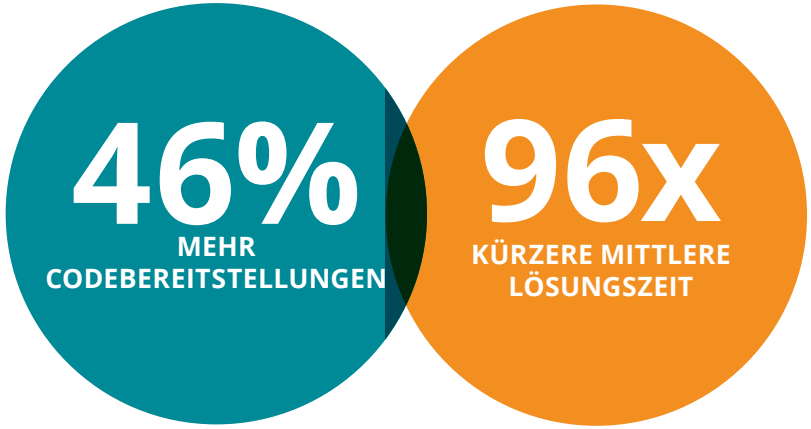
Passen Sie Ihre Richtlinien an die Situation Ihres Unternehmens an

Eine Bereitschaftsrichtlinie, die sich bei einem New Relic-Team bewährt hat, könnte für Ihr Unternehmen völlig untragbar sein. Überlegen Sie, welche weiteren Aspekte Sie berücksichtigen sollten, um Ihre Bereitschaftsrotation fair und effektiv zu gestalten. Hier einige Beispiele:

- **Wachstum:** Wie rasch wachsen Ihr Unternehmen und Ihre Engineering-Gruppe? Wie hoch ist die zu erwartende Fluktuation?
- **Geografie:** Ist Ihre Engineering-Organisation zentralisiert oder geografisch verteilt? Verfügen Sie über die nötigen Ressourcen für eine „Follow-the-Sun"-Rotation?
- **Komplexität:** Wie komplex sind Ihre Anwendungen und wie sind sie strukturiert? Wie komplex sind die Abhängigkeiten zwischen den Services?
- **Tools:** Verfügen Sie über Vorfallreaktionstools, die die Techniker automatisch mit aussagekräftigen Benachrichtigungen über auftretende Probleme informieren?
- **Kultur:** Haben Sie den Bereitschaftsdienst in Ihrer Engineering-Kultur als wichtige Aufgabe etabliert? Basiert Ihre Kultur darauf, die Ursache des Problems zu erkennen und zu beheben, statt einen Schuldigen zu suchen?

„[DevOps] waren doppelt so häufig in der Lage, ihre eigenen Ziele in den Bereichen Profitabilität, Marktanteil und Produktivität zu übertreffen."

IM VERGLEICH ZU LOW-PERFORMERN HABEN HIGH-PERFORMER IN DEVOPS[1]



1 Quelle: „2017 State of DevOps Report", Puppet and DORA

KAPITEL 3

Effektive Vorfallreaktion

Effektive Vorfallreaktion

Neben der Bereitschaftsrotation spielt auch das Vorfallmanagement eine wichtige Rolle. Was ist ein Vorfall? Eine Situation, in der sich ein System in einer unerwarteten Weise verhält, die sich nachteilig für Kunden (oder Partner oder Mitarbeiter) auswirken könnte.

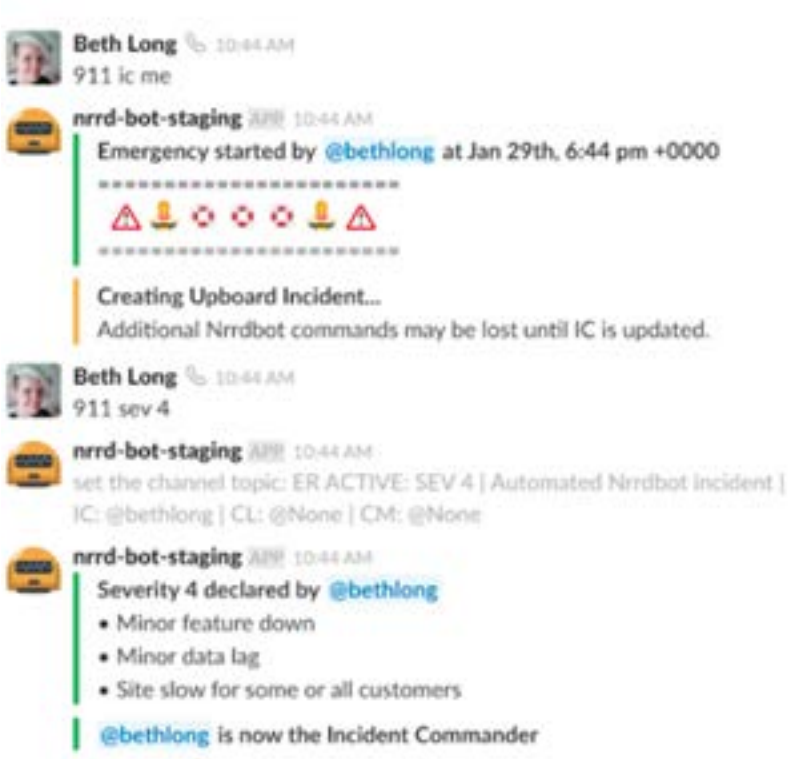
Das Vorfallmanagement ist ein entscheidender Aspekt des „You build it, you own it"-Ansatzes, auch wenn er häufig zu kurz kommt, weil die DevOps-Teams das Interesse verlieren, sobald ein Problem behoben wurde. Ohne effektives Vorfallmanagement sind die Teams im Ernstfall meist auf improvisierte Organisationen, Methoden und Kommunikationswege angewiesen. Wenn es irgendwo brennt, machen sich alle fieberhaft daran, so schnell wie möglich einen Plan für die Problemlösung auszuarbeiten.

Dabei gibt es einen wesentlich besseren Weg, mit Vorfällen umzugehen, bei dem nicht nur die Dauer und Häufigkeit von Störungen minimiert wird, sondern die verantwortlichen Ingenieure zudem den Support erhalten, den sie für eine effiziente und effektive Reaktion benötigen.

Entwickeln eines effektiven Vorfallmanagementprozesses

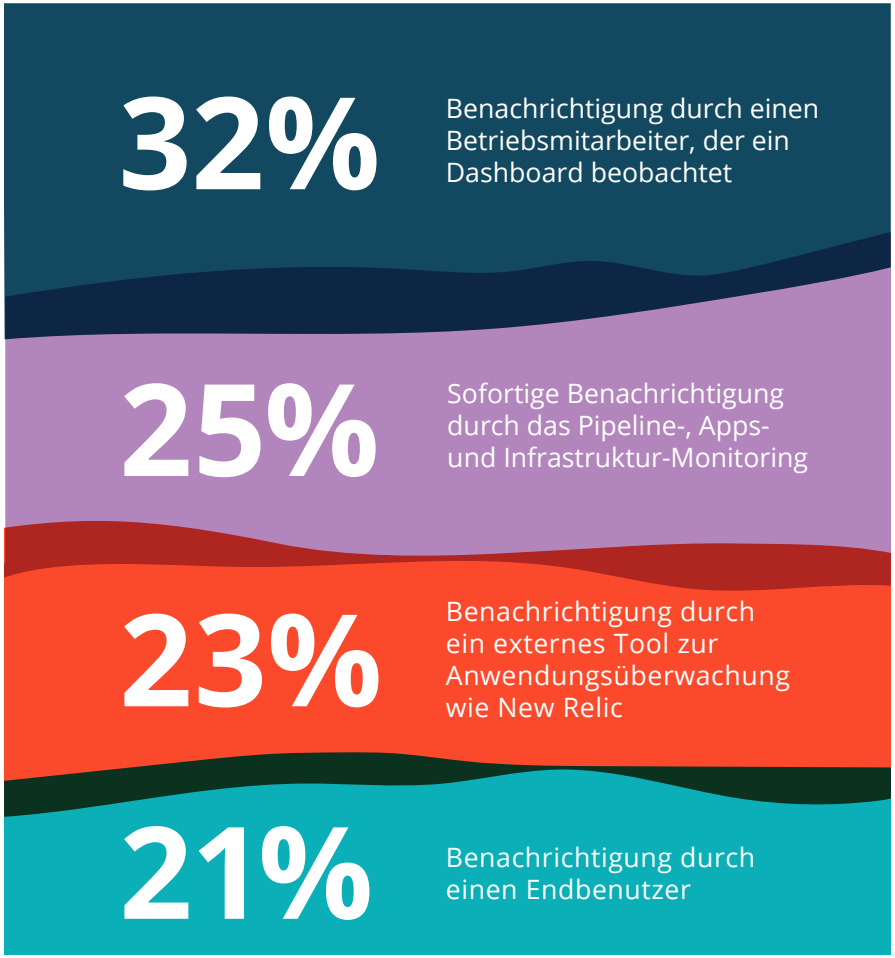
1. **Legen Sie Schweregrade fest:** Schweregrade geben die potenziellen Auswirkungen auf die Kunden an und bestimmen, wie viel Support benötigt wird. Bei New Relic verwenden wir beispielsweise eine Schweregradskala von 1 bis 5:
    - Level 5 stellt keine Beeinträchtigung der Kunden dar und kann verwendet werden, um auf ein mögliches Problem hinzuweisen
    - Level 4 umfasst kleinere Bugs oder Datenverzögerungen, die Kunden betreffen, aber nicht behindern
    - Level 3 ist für größere Datenverzögerungen oder nicht verfügbare Features vorgesehen
    - Levels 2 und 1 sind schwerwiegende Vorfälle, die Störungen verursachen
2. **Instrumentieren Sie Ihre Services:** Jeder Dienst sollte über Monitoring und Alarmierung für eine proaktive Vorfallmeldung verfügen. Das Ziel ist es, Vorfälle zu erkennen, bevor Kunden dies tun, um Worst-Case-Szenarien zu vermeiden, in denen verärgerte Kunden den Support anrufen oder Kommentare in sozialen Medien posten. Proaktive Vorfallmeldung ermöglicht Ihnen, möglichst rasch auf Vorfälle zu reagieren und sie zu beheben.
3. **Definieren Sie Responder-Funktionen:** Bei New Relic übernehmen Mitglieder des Engineering- und des Support-Teams während eines Vorfalls folgende Funktionen: Incident Commander (leitet die Vorfallbehebung), Tech Lead (diagnostiziert und behebt das Problem), Communications Lead (hält alle auf dem Laufenden), Communications Manager (koordiniert die Notfallkommunikationsstrategie), Incident Liaison (interagiert mit dem Support und dem Unternehmen für Schweregrad 1s), Emergency Commander (optional für Schweregrad 1s) und Engineering Manager (managt die Nachbearbeitung des Vorfalls).

4. **Erarbeiten Sie eine Strategie:** Sie legt die Aufgaben der einzelnen Funktionen bei allen Vorgängen während des gesamten Lebenszyklus eines Vorfalls fest, von der Vorfallmeldung über das Bestimmen des Schweregrads und des zu kontaktierenden Tech Leads, das Debuggen und Beheben des Problems, das Management des Kommunikationsflusses und das Verteilen der Zuständigkeiten bis hin zum Beenden des Vorfalls und zur Nachbesprechung.
5. **Implementieren Sie geeignete Tools und Automatisierungen zur Unterstützung des gesamten Prozesses:** Vom Monitoring über Alarme und Dashboards bis zur Verfolgung von Vorfällen sorgt die Automatisierung des Prozesses dafür, dass die betreffenden Teammitglieder stets auf dem Laufenden und bei der Sache sind und die Strategie effizient umgesetzt wird.
6. **Führen Sie Nachbesprechungen durch:** Laden Sie Ihre Teams innerhalb von ein oder zwei Tagen nach dem Vorfall zu einer Nachbesprechung. Betonen Sie, dass es bei der Nachbesprechung nicht darum geht, einen Schuldigen zu finden, sondern darum, die Ursache des Problems zu ermitteln.
7. **Implementieren Sie „Don't Repeat Incidents"-Richtlinien (DRI):** Wenn ein Serviceproblem Ihre Kunden beeinträchtigt, ist es Zeit, Mängel in der technischen Ausstattung zu erkennen und zu beheben. Eine DRI-Richtlinie besagt, dass Ihr Team alle Arbeiten an diesem Service einstellt, bis die Ursache des Problems behoben ist.



Beispielvorfall in Slack deklariert

WIE DEVOPS-TEAMS VON PROBLEMEN ERFAHREN[2]



2: Quelle: „DevOps Survey Results“, 2nd Watch, 2018.

KAPITEL 4

Überwinden der Komplexität von Microservices

Überwinden der Komplexität von Microservices

Mit Microservices und DevOps ist es wie mit Erdnussbutter und Schokolade – zusammen sind sie ein perfektes Team. Mittlerweile wissen Unternehmen, dass die Umwandlung monolithischer Anwendungen in verteilte Services die Produktivität, Geschwindigkeit, Agilität, Skalierbarkeit und Zuverlässigkeit erheblich verbessern kann.

Doch während den Teams wohl bewusst ist, was sie beim Entwickeln, Testen und Bereitstellen von Mikroservices ändern müssen, wird häufig übersehen, welche erheblichen Veränderungen im Bereich der Zusammenarbeit und Kommunikation erforderlich sind. Die Techniker von New Relic haben folgende Best Practices zur Schaffung eines kooperativen Umfelds entwickelt, in dem sich die Komplexitäten und kommunikativen Herausforderungen einer Mikroservice-Welt leichter bewältigen lassen.

Empfehlungen für die Kommunikation in einer Mikroservice-Umgebung:

- **Informieren Sie abhängige Upstream- und Downstream-Teams über wesentliche Änderungen:** Bevor Sie größere Änderungen an Ihrem Microservice vornehmen, benachrichtigen Sie die abhängigen Upstream- und Downstream-Teams, damit diese im Falle eines auftretenden Problems nicht erst lange nach der Ursache suchen müssen.

- **Kommunizieren Sie frühzeitig und häufig:** Das ist insbesondere bei der Versionierung, bei Ausmusterungen und in anderen Situationen entscheidend, in denen Sie vorübergehend für Rückwärtskompatibilität sorgen müssen.

- **Behandeln Sie interne APIs wie externe APIs:** Stellen Sie Dokumentationen, aussagekräftige Fehlermeldungen und einen Prozess für das Senden von Testdaten bereit, um Ihre API benutzerfreundlich zu gestalten.

- **Behandeln Sie Downstream-Teams wie Kunden:** Stellen Sie eine README-Datei mit einem Architekturdiagramm, einer Beschreibung, Anleitungen für die lokale Ausführung und Informationen zur Mitwirkung bereit.

- **Richten Sie einen speziellen Kanal für Ankündigungen ein:** Eine zentrale Informationsquelle für wichtige Ankündigungen ist entscheidend, damit die Teams wichtige Informationen nicht aus verschiedenen Foren, Diskussionen und E-Mails zusammentragen müssen.

- **Konzentrieren Sie sich auf die „Nachbarschaft" Ihres Service:** Ihre Upstream-, Downstream-, Infrastruktur- und Sicherheitsteams sind allesamt „Nachbarn" Ihres Service. Als guter Nachbar sollten Sie ihren Demos und Standups beiwohnen, Ihren Nachbarn Zugriff auf Ihre Service-Roadmap gewähren und eine Kontaktliste unterhalten.

Verwenden von Daten zum besseren Verständnis der Funktionsweise von Microservices

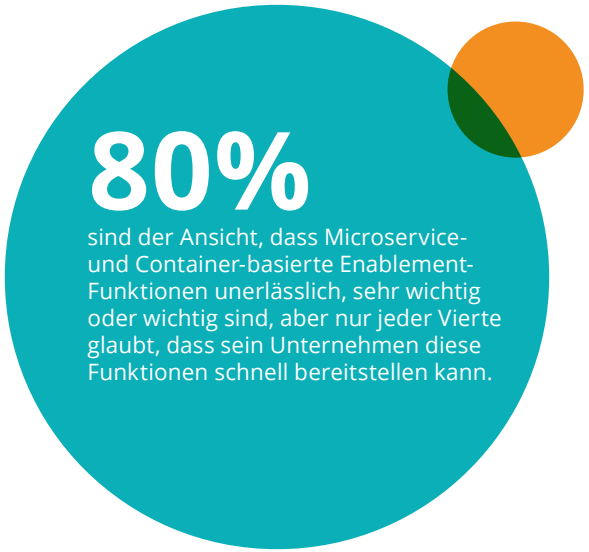
Das Aufsplitten Ihrer monolithischen Anwendungen in Microservices ist keine einfache Aufgabe. Zunächst müssen Sie ein System genau kennen, bevor Sie es in klar umrissene Services untergliedern können. Und selbst dann ist es oft schwierig, das System präzise so zu partitionieren, dass Sie echte Microservices erstellen können, die jeweils für die Bereitstellung einer

Funktion zuständig, granular und nur lose mit Anwendungen und anderen Services gekoppelt sind.

Die folgenden Metriken und Monitoring-Tipps sollen Ihnen helfen, Microservices in Ihrer Umgebung ausfindig zu machen, die noch zu viele Wechselwirkungen mit anderen Services und Anwendungen aufweisen:

1. **Bereitstellungen:** Synchronisiert Ihr Team Bereitstellungen mit Upstream- und Downstream-Teams? Wenn Sie in Ihrer Monitoring-Lösung Hinweise auf Bereitstellungen sehen, die mit mehreren Services synchronisiert sind, haben Sie Ihre Services nicht wirklich entkoppelt.
2. **Kommunikation:** Ein Microservice sollte nur eine minimale Kommunikation mit anderen Services benötigen, um seine Funktion auszuführen. Wenn Sie einen regen Austausch von Anfragen zwischen einem Service und bestimmten Downstream-Services erkennen, ist das ein deutliches Zeichen dafür, dass er nicht entkoppelt ist. Ein weiterer Marker, den Sie prüfen sollten, ist der Durchsatz. Wenn die Anzahl der Aufrufe pro Minute bei einem bestimmten Microservice erheblich höher ist als der Durchsatz der Anwendung insgesamt, ist dieser Service eindeutig nicht entkoppelt.
3. **Datenspeicher:** Jeder Microservice sollte einen eigenen Datenspeicher aufweisen, um Bereitstellungsprobleme, Datenbankkonflikte und Schemaänderungen zu vermeiden, die andere Services beeinträchtigen würden, die denselben Datenspeicher nutzen. Ein ordnungsgemäßes Monitoring kann Ihnen zeigen, ob jeder Microservice einen eigenen Datenspeicher verwendet.
4. **Skalierbarkeit:** In einer echten Mikroservice-Umgebung sollten Spitzen bei Services, die sich auf Hosts befinden, mit Spitzen beim Durchsatz einzelner Services korrespondieren. Dies deutet auf dynamische Skalierung hin, die einen der größten Vorteile von Microservices darstellt. Wenn Sie hingegen korrespondierende Spitzen bei allen Services und Hosts feststellen, gibt es gute Gründe anzunehmen, dass Ihre Services nicht entkoppelt sind.
5. **Entwickler pro Anwendung:** Wenn Ihre Microservice-Teams effektiv kommunizieren, brauchen Sie keine "Architektur-Gurus", die die einzelnen Microservices überwachen und dafür sorgen, dass sie gut funktionieren. Wenn Sie 100 Techniker und 10 Services haben und zwei Ihrer Techniker mit der Entwicklung für alle 10 Services beschäftigt sind, ist das wahrscheinlich ein Zeichen dafür, dass diese Services nicht wirklich entkoppelt sind und die Entwicklung aller Ihrer Services von den Kenntnissen und Kommunikationsfähigkeiten dieser beiden Entwickler abhängig ist.

DIE MEISTEN GLAUBEN, DASS MIKROSERVICES UND CONTAINER UNERLÄSSLICH SIND[3]



3: Quelle: „Enterprise Priorities for Hybrid Cloud Management", Ponemon Institute, Juni 2018.

KAPITEL 5

Mit Daten zu digitaler Geschwindigkeit

Mit Daten zu digitaler Geschwindigkeit

Das wichtigste Argument für DevOps ist die Geschwindigkeit – schnellere Softwarebereitstellung, schnellere Problemlösung, schnellere Innovation. Aber wie kann Ihr Unternehmen diese Geschwindigkeit erzeugen, um die sich bietenden Marktgelegenheiten zu ergreifen, die Konkurrenz bei der Innovation zu überholen und um Ihre Kunden dauerhaft zufriedenstellen zu können?

Bei New Relic wissen wir, dass Daten der Treibstoff für den DevOps-Erfolg sind, weil sie Ihnen helfen:

- die DevOps-Performance zu messen und zu verfolgen
- sofortiges Feedback zu geben, damit sich jeder auf die richtigen Dinge konzentriert
- die Softwarebereitstellung, die Performance und die Geschäftsergebnisse zu optimieren

Eine der häufigsten Fragen, die unsere Kunden zum Thema DevOps stellen, lautet: „Womit sollten wir beginnen?“ Hier sind einige Grundprinzipien für Ihren Weg zur umfassenden Einführung von DevOps:

Eliminieren Sie Datensilos

Das Messen ist ein Grundprinzip von DevOps, doch dabei sind sich viele Teams nicht bewusst, dass in einer Umgebung mit isolierten Daten jeder unter Performance etwas anderes versteht. Isolierte Performancedaten lassen Zusammenhänge zwischen Anwendungen, Infrastruktur und Benutzererfahrung meist nur schwer erkennen. Wenn Sie alle Ihre Systeme so ausstatten, dass sämtliche Daten auf einer Plattform einsehbar sind, verfügen alle über dieselben Informationen, nutzen gemeinsame Daten und Metriken und können so verschiedene Funktionen in einem Team vereinen.

Bei New Relic hatten unsere Engineering-Teams mit einer Vielzahl von Werkzeugen und konkurrierenden Prioritäten zu kämpfen. Erst als sich alle darauf geeinigt hatten, welche SLOs am wichtigsten sind, und diese dann in gemeinsamen Dashboards bereitgestellt wurden, konnten diese Teams effektiv an einem Strang ziehen.

Vereinfachen Sie den Komplex

In vielerlei Hinsicht tragen moderne Anwendungsarchitekturen dazu bei, die Entwicklung zu vereinfachen und zu beschleunigen, stellen mit ihrem modularen Aufbau aus vielen Einzelkomponenten aber selbst wiederum sehr komplexe Gebilde dar. Um diese Komplexität zu bewältigen, müssen Sie sich einen vollständigen Überblick über Ihre Architektur verschaffen, so veränderlich sie auch sein mag. Wenn Sie über die richtigen Daten verfügen, können Sie beurteilen, was funktioniert und worauf Sie Ihr Team ansetzen sollten.

Untersuchen Sie die Auswirkungen von Änderungen

Mit DevOps nimmt die Häufigkeit von Codebereitstellungen und Änderungen zu. Um mögliche Probleme zu vermeiden, muss Ihr Team diese ständig im Auge behalten. Sorgen Sie dafür, dass Ihre Reporting-Funktionen die jüngsten Bereitstellungen mit ihren Auswirkungen auf die Anwendungsperformance und das Kundenerlebnis vorher und nachher aufzeigen, darunter alle eventuell aufgetretenen Fehler. Auf diese Weise können Sie Änderungen schnell mit möglichen Auswirkungen in Verbindung bringen, sodass Ihr Team rasch reagieren und ein Release-Rollback vornehmen oder auftretende Probleme direkt beheben kann.

Zusammenfassung

DevOps erfreut sich zunehmender Beliebtheit. In vielen Branchen setzen Unternehmen DevOps-Prinzipien ein, um ihre Geschwindigkeit, Produktivität, Qualität und Innovation zu verbessern. Wie sich gezeigt hat, erzielen die High-Performer unter den DevOps-Organisationen, die optimale digitale Geschwindigkeit erreicht haben, erheblich häufigere Bereitstellungen und eine deutlich schnellere Störungsbehebung als die Low-Performer.

Der Treibstoff für einen leistungsstarken DevOps-Motor sind Daten. Best Practices, wie sie in diesem E-Book beschrieben sind, sollen dabei helfen, diese Daten effektiv und erfolgreich zu nutzen. New Relic bietet Ihnen die End-to-End-Transparenz, die Sie benötigen, um Ihre DevOps-Bemühungen zu überwachen und die Ergebnisse in jeder Phase kontinuierlich zu verbessern.

Erfolgreiches DevOps beginnt hier.
Besuchen Sie newrelic.com/devops