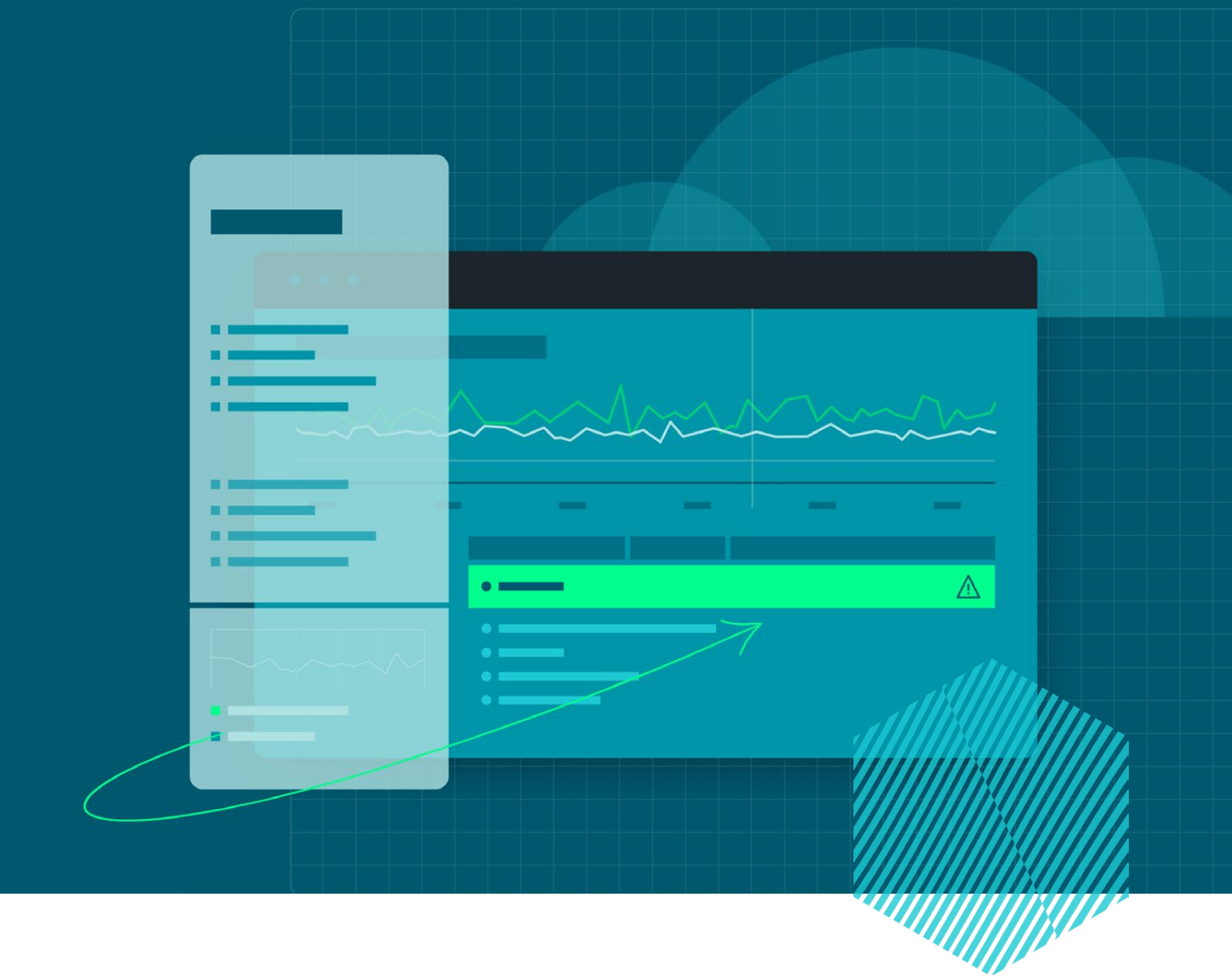
new relic



Best Practices für Ihr Log-Management

Log-Methodik als starke Triebfeder für Full-Stack Observability

Whitepaper / August 2022

Inhalt

Einleitung

Log-Management als Monitoring-Disziplin hat sich enorm weiterentwickelt: Vorbei sind die Zeiten, als Incidents mit einem wenig kuratierten Export an Rohdaten in Form von Anwendungs- und Infrastruktur-Logs adressiert wurden. Und so ist Logging nunmehr auch ein wichtiges Informationsbindeglied für operatives Geschäft, Business Intelligence und Marketing geworden. Nicht zuletzt sind Logs aber auch veritable Observability-Katalysatoren, bilden so dynamisch wie kompakt alle wichtigen Systemabläufe ab und sind dabei sogar von präventivem Nutzen.

Für echte Observability mit effektivem Logging ist mehr vonnöten, als einfach nur enorme Mengen unzureichend formatierter Logs in einer Datenbank oder einer Datei abzuladen. Hier darf sich nun auch die Frage stellen: Wie gelingt es Ihnen, Ihre Logging-Methodik so zu adaptieren, dass Ihre Logs Ihnen Korrelationsmöglichkeiten in Echtzeit über Anwendungen und Infrastruktur hinweg vermitteln? Dies vor allem, ohne sich mühsam durch verschiedene Anwendungen klicken zu müssen? Wie gelingt Ihnen bessere End-to-End-Observability? Und schließlich: Wie münzen Sie diese auch wirklich gewinnbringend auf Ihre geschäftlichen Prioritäten um?

Auch die Logging-Methodik lässt sich hierbei adaptieren, Logs in einen Optimierungsfaktor für Full-Stack Observability konvertieren. In diesem Whitepaper gehen wir auf Best Practices ein, die dieses Ziel im Zusammenhang moderner Unternehmensstrukturen konkret umsetzbar machen.

Klassisches Logging

An dieser Stelle soll nun ein Vergleich mit klassischem Logging im Datensilo angestrengt werden. In der Vergangenheit stützte sich Observability insbesondere auf Application Performance Monitoring (APM) und Infrastruktur-Monitoring. Monitoring als solches mag eine hervorragende Grundlage bieten, enthüllt aber längst nicht alle wichtigen Zusammenhänge zwischen den Logs der Anwendungen und Infrastruktur-Geräte. Viele der komplett separat betriebenen Monitoring- und Logging-Tools sind zudem sehr stark anwendungsbezogen, fokussieren sich also speziell auf einen Stack-Bereich. So bleiben stets wichtige relationale Faktoren über dessen Grenzen hinaus außen vor. Ohne die hieraus ableitbaren Informationen lassen sich aber natürlich auch keine positiven Effekte für Time to Market und Incident Response herbeiführen oder Einblicke in Kundenverhalten generieren.

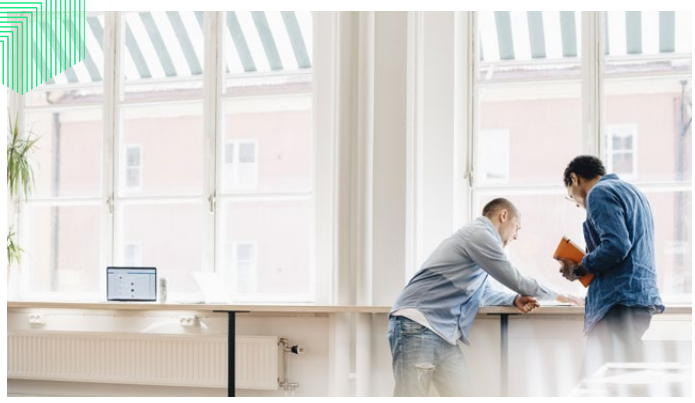
Man steht dabei konsequent vor der Wahl: entweder keine präzisen Log-Details und somit auch Unklarheit zur Ursache von Fehlern zu haben. Oder aber ein Stückwerk aus separaten Tools betreiben und dann Log-Daten irgendwie Fehlern und Traces zuordnen. Detaillierte Logs über Silostrukturen hinweg ergeben dabei jedoch nie ein kohärentes Gesamtbild und steigern so Kosten, verlängern die Time to Market für neue Lösungen und Services. Parallel machen sie zudem auch das Kundenerlebnis in seinen Facetten nur schwer erkennbar, Fehler nur langsam adressierbar.

Full-Stack Observability

Besteht hingegen Transparenz im gesamten Stack und dies für alle Zusammenhänge und Details, die auf die UX Einfluss nehmen könnten, spricht man von Full-Stack Observability bzw. End-to-End-Observability. Vonnöten ist dabei eine konsistente 360°-Ansicht aller Telemetriedaten (Metrics, Events, Logs und Traces).

Kurz gesagt generiert Full-Stack Observability zentral über eine Lösung umfassende Performance-Transparenz für komplexe Anwendungen und Systeme. Sie koppelt hier die Schnittstellen zur Problembehebung, verhilft zu einer geringeren MTTR und präziserer Klarheit beim Kundenerlebnis.

Engineering- und Dev-Teams müssen mit ihr kein Daten-Sampling mehr betreiben oder die Ergebnisse mehrerer Datensilos zusammenflicken, sondern erhalten direkt vollständigen Einblick in ihren Stack. So können sie sich mit Observability endlich auf Coding-Themen konzentrieren, die sie intellektuell fordern und mit denen sie strategisch etwas bewegen.



Logging mit Full-Stack Observability im Visier

Logs für den gesamten Stack zu generieren erweist sich oft als Herkulesaufgabe. Was genau soll erfasst werden? Welche Details sind wichtig? Führen zu viele Daten auch zu exorbitanten Kosten? Unzählige Unternehmen zahlen für die Zentralisierung ihres Log-Managements in einer separaten Plattform einen hohen Preis. So sehen sie sich letztlich gezwungen, ihre Log-Daten performance- und preisbasiert einzugrenzen – und Visibility wie auch Business-Mehrwert damit ebenso. Full-Stack Observability erfordert ein Fundament konsequent umgesetzter Best Practices, auf die wir im Folgenden eingehen.

Fokus auf die richtigen Elemente

Um einen Log zu generieren, muss lediglich Text in ein Standard Output oder eine Datei geschrieben werden. Die wichtigste Entscheidung ist dabei natürlich der Inhalt. Es sollten so viele Metadaten wie erforderlich enthalten sein, um Events und Fehlerursachen genau ausmachen zu können. Dabei kann es sich um Elemente wie Fehlermeldungen oder Stack-Traces und zugehörige Werte, Metrics oder Events handeln.

Alle tatsächlich erfassten Elemente sollten aber auch Sinn und Zweck haben. Ob Nutzungsdaten, Nutzer-Events oder Anwendungsfehler und -ausnahmen: Der Mehrwert zählt, und so sollten Informationen in Log-Daten:

- unmittelbar greifbaren Mehrwert liefern
- die für Klarheit zu Fehlerursachen und Entscheidungsfindung notwendigen Details liefern

Gängige Szenarien antizipieren

Logs dienen nicht nur der Incident Response, können vielmehr auch anderen geschäftlichen Aspekten enorm nützlich sein. Mit der Erstellung von Performance-Profilen oder der Beantwortung von Statistikfragen seien hier nur zwei Einsatzbereiche genannt.

Wird eine Log-Methodik vor allem auch anhand gängiger Szenarien konzipiert, lassen sich auch viel direkter Brücken zwischen Theorie und tatsächlichem Geschäftswert schlagen. Logs zur Nutzerinteraktion etwa liefern wichtige UX-Einblicke, über System-Logs können Probleme oder Hardware-Ausfälle abgebildet werden. Detaillierte Anwendungslogs ermöglichen also Performance-Einblicke und Informationen zu potenziellen Problemen wie Speicherlecks. Diese Faktoren können für Geschäftsentscheidungen außerordentlich wertvoll sein.

Relevanz, Relevanz, Relevanz

Log-Meldungen sind nur so nützlich wie die Informationen und der Kontext, den sie bieten. Anreicherung um adäquate Details und Lesbarkeit macht die Logs dann auch effektiv nutzbar. Von externen Anbietern entwickelte Infrastruktur erfasst diese spezifische Informationsebene zumeist. Inhouse-Anwendungen erfordern hingegen nach wie vor die Fragestellung: „Erfasse ich hier wirklich genau die Details, die ich zur Diagnose benötige und um meinen Benutzern zu den richtigen Entscheidungen zu verhelfen?“

Bei Anwendungsfehlern muss die Meldung stichhaltig vermitteln, was genau in der Code-Zeile vor sich geht. Eine Fehlermeldung `Transaktion fehlgeschlagen` ist bei weitem nicht so hilfreich wie die Beschreibung `Transaktion fehlgeschlagen: Benutzer konnte nicht erstellt werden ${path/to/file:line-number}`. Enthält der Log nun eben auch Daten zur Transaktion, erschließt sich den Entwickler:innen leichter, warum sie fehlgeschlagen ist.

Fehler- oder Status-Codes geben dabei für gewöhnlich die Problemart der Anwendung an. Statt nur einen Fehlercode auszugeben, bietet sich also eine Kurzbeschreibung im Log an. Diese spart dann wiederum anderen Entwickler:innen wertvolle Zeit bei der Fehlerbehebung.

Logs sollten Ihrem Unternehmen wichtige Informationen mit relevantem Geschäftswert liefern. Kryptische Meldungen ohne Hintergrunddetails, die nur wenige Teammitglieder verstehen, tun dies nicht.

Einfach und kompakt in der Struktur bleiben

Sicher sollte eine Meldung ausreichende Informationen enthalten. Eine Überladung an Details gilt es aber zu vermeiden. Überschüssige Hinweise ohne echten Mehrwert treiben Speicheranforderungen und Kosten schnell in die Höhe und verlangsamen Suchabläufe. Obendrein lenken sie zudem vom Kernproblem ab, stehen der Fehlerbehebung somit erst recht im Wege.

Halten Sie Ihre Logs daher vor allem auch kompakt strukturiert. Zentral und essenziell bleibt natürlich immer auch das „Warum", ohne unnötiges Datenrauschen zu verursachen.

Tritt ein Fehler auf, müssen die Informationen zur Fehlerursache nicht jedes noch so kleine Umgebungsdetail enthalten. Ein Beispiel: Bei einer Anwendung ist der Verbindungsaufbau fehlgeschlagen. Es konnten somit keine Daten über eine interne API abgerufen werden. In diesem Fall kann es etwa nützlich sein, Fehlermeldungen aus der API oder zum Netzwerkstatus für die Umgebung zu erfassen. Wie viel Speicher die Anwendung benötigt, ist hingegen eher sekundär, genauso wie die Anzahl der ausgeführten Anwendungen.

Die Zeit zählt mit

Auch auf den Timestamp kommt es an. Das mag zunächst so selbstverständlich klingen, dass es geradezu wie eine Banalität anmutet. Doch wer es gewohnt ist, Logs in einer Datenbank zu speichern, die automatisch Datum und Uhrzeit ergänzt, für den ist das Hinzufügen dieser Daten in Log-Meldungen sicher nicht vorrangig intuitiv. Es bietet sich hier an, die genaueste brauchbare Datenebene zu verwenden und in den Logs auszugeben. Aufgaben mit hoher Frequenz weisen die Zeit womöglich auf die Millisekunde genau aus, solche mit niedrigerer Frequenz unter Umständen nur auf die Minute oder auf den Tag. Entscheidend ist dabei nicht nur die Datengenauigkeit, sondern ein unternehmensweit konsequent umgesetzter Standard.

Ebenfalls wird vordergründig die Empfehlung nicht überraschen, alle Systeme zur gleichen Zeit zu synchronisieren. Denn hierdurch wird es Ihrer Observability-Plattform möglich, über den Timestamp Log-Events mit anderen Telemetriedaten zu korrelieren.

Parsingfreundliches Log-Format verwenden

Kann Ihre Observability-Plattform keine Details aus Ihren Log-Daten extrahieren, wird sie Ihnen auch keine große Hilfe sein. Ein parsingfreundliches Log-Format und eine konsistente Log-Struktur helfen bei Erfassung und Aggregierung ungemein. Mit Log-Management in New Relic können Custom-Regeln für Log-Parsing ganz einfach definiert werden.[1] Verständlich und leicht nutzbar müssen Ihre Log-Daten aber natürlich bleiben.

Ein Beispiel für ein Log-Format, das Ihnen in punkto Parsing wenig weiterhelfen wird, sind NGINX Access Logs mit unstrukturiertem Text, die über Suchen hinaus von sehr geringem Wert sind. Zur Beantwortung ihrer Fragen werden Ihre Entwickler:innen bei derartigen Formaten im Gros der Fälle eine Volltextsuche durchführen müssen. Hier ein typisches Inhaltsbeispiel:

```
127.180.71.3 - - [10/May/2022:08:05:32 +0000]
"GET /downloads/product_1 HTTP/1.1" 304 0 "-"
"Debian APT-HTTP/1.3 (0.8.16~exp12ubuntu10.21)"
```

Nach dem Parsing wird der Log in Attribute aufgelöst, so etwa `Antwort-Code` und `Abfrage-URL`. Im Folgenden nun ein Beispiel der gleichen Log-Details in einem parsingfreundlichen Format:

```
{
  "remote_addr":"93.180.71.3",
  "time":"1586514731",
  "method":"GET",
  "path":"/downloads/product_1",
  "version":"HTTP/1.1",
  "response":"304",
  "bytesSent": 0,
  "user_agent": "Debian APT-HTTP/1.3
  (0.8.16~exp12ubuntu10.21)"
}
```

Handelt es sich in seiner Gänze um ein Custom-Format, löst die Definition des Log-Typs individuell definierte Parsing-Regeln aus.

Falls Sie mehrere Anwendungen mit demselben Zweck nutzen, ist ein standardisiertes Log-Format für alle empfehlenswert. So lassen sich Daten in Ihre Observability-Plattform leichter integrieren. Dies gilt auch dann, wenn das Team hinter einer App Einblicke für andere Attribute gewinnen möchte.

[1] (New Relic, Inc., n.d.)

Log-Formate im Detail

Logs verfügen über drei grundlegende Strukturvarianten. Jede von ihnen geht mit eigenen Auswirkungen auf die Nutzbarkeit erfasster Daten einher. Bei den Varianten handelt es sich um die folgenden:

- **Strukturiert.** Eines der gängigsten strukturierten Log-Formate ist JSON. Es zeichnet sich durch Flexibilität und Kompaktheit aus, und viele Tools können es leicht parsen. Idealerweise liegen alle Logs in einem strukturierten Format vor. JSON ist besonders nützlich bei der Organisation hierarchischer Daten. Andere gängige strukturierte Formate nutzen Komma- (CSV) bzw. Spaltentrennung (TSV).
- **Allgemein.** Ein allgemeines Format ist nicht strukturiert, aber bekannt, definiert und konsistent. Das allgemeine Log-Format von Apache für Access Logs bietet sich als Beispiel an. Der Vorteil eines allgemeinen Formats besteht darin, dass viele Tools die Daten quasi out of the box parsen können.
- **Custom.** Nutzt eine Anwendung zum Logging weder ein strukturiertes noch ein allgemeines Format, liegt eine Custom-Variante vor. Beim Log-Forwarding ist ggf. Parsing notwendig, um Anfang und Ende einer Log-Zeile zu erfassen. Custom definierte Parsing-Regeln machen die Log-Daten in diesem Zuge nützlicher und besser verwertbar.

Logs kategorisieren und gruppieren

Die Konkretisierung eines Datenmodells für Ihre Logs vereinfacht Suchen und macht sie effektiver. Definieren und inkludieren Sie hierzu Attribute wann immer möglich, um Ihre Logs entsprechend zu kategorisieren und zu gruppieren.

Die OpenTelemetry-Standards für Logs mehrerer Branchenführer wie New Relic decken dabei eine Vielzahl an Faktoren wie Benennungsregeln und Feldwert-Definitionen ab.[2] Nicht jedes Framework unterstützt nativ Logs, die nach Maßgabe genau dieser Standards formatiert sind. Nützlich als strukturelle Richtschnur sind sie aber allemal.

Zu den in einem Log-Datenmodell empfehlenswerten Attributen gehören Ressourcen, Logs in Context sowie Log-Levels.

Ressourcen
Über Ressourcen wird definiert, wann und von wo die Logs abgerufen wurden, etwa anhand folgender Elemente:

- Datum und Uhrzeit
- Gerät oder Hostname oder Identifikator
- Anwendungs- oder Servicename

In Logs aus klassischen, hostbasierten Anwendungen mit benannten Umgebungen kann der Hostname sich als wertvolle Information erweisen. Eine Pod- oder Container-ID organisiert Logs aus containerisierten oder orchestrierten Umgebungen besser.

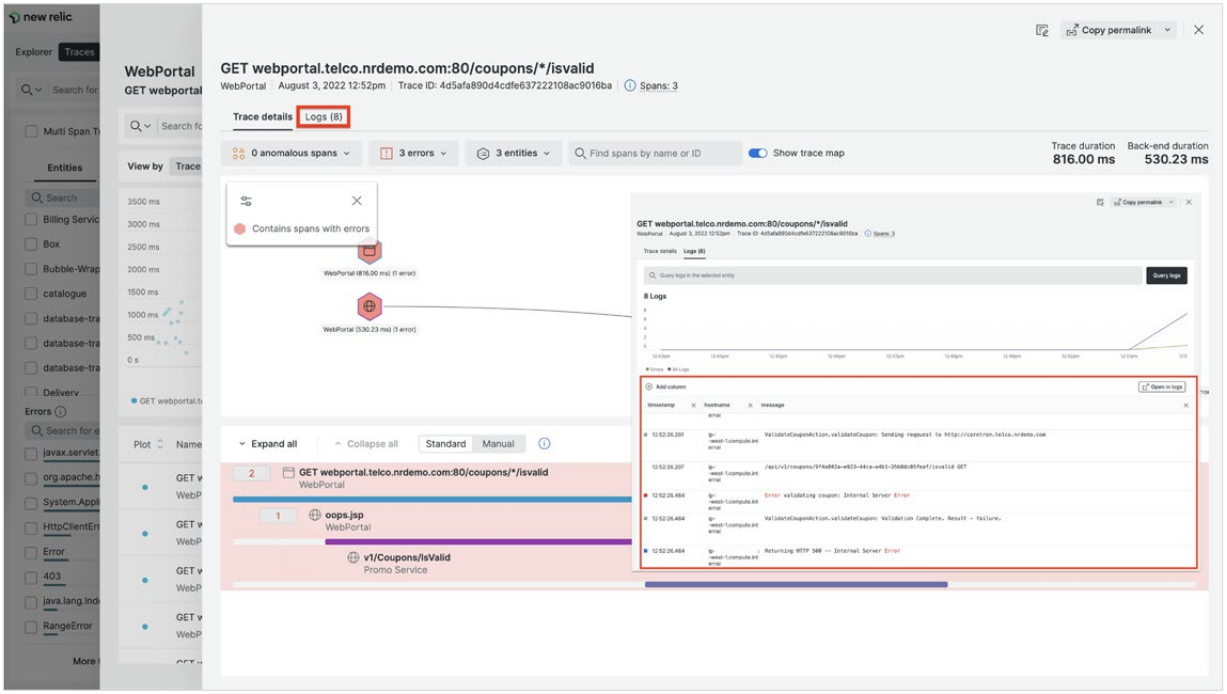
Orchestrierte bzw. PaaS-Umgebungen füllen Logs häufig automatisch mit einer Vielzahl an Metadaten. Hilfreich ist das allemal. Die Inhalte müssen aber noch zusätzlich um Faktoren ergänzt werden, die dem System nicht bekannt sein können. Angaben zur Produktversion etwa oder zu Staging- bzw. Produktionsumgebungen, Testbereichen oder A/B-Tests. Hierbei steht zu beachten, dass bei einer Aggregation ja Logs aus verschiedenen Quellen im selben System erfasst werden. Ohne die richtigen Metadaten wird es nicht möglich sein, einen echten Fehlerlog in der Produktion von einer Transaktion zu unterscheiden, die bei einem Testlauf nicht durchgeführt werden konnte.

Auch mit Log-Forwarding kann die Log-Quelle gut identifiziert werden. Das Gros der entsprechenden, von New Relic automatisch bereitgestellten Lösungen ergänzt die Daten automatisch um das für die Auslieferung der Daten genutzte Tool mit Versionierung.

[2] (OpenTelemetry, n.d.)

Logs in Context

Für Entwickler:innen kann es enorm hilfreich sein, Logs im Zusammenhang von Problemen in Anwendungen und Hosts einzusehen. Bei Logs in Context handelt es sich um ein Feature von New Relic, mit dem Logs automatisch um Anwendungsinformationen erweitert werden können. Die entsprechenden Daten werden dabei vom New Relic APM-Agent an Ihr Logging-Framework übermittelt und in Ihren Anwendungslogs ausgegeben. APM-Fehler und Distributed Traces werden direkt mit den Logs assoziiert, die in derselben Transaktion wie der Fehler oder die Trace generiert wurden. Umgesetzt wird dies durch Ergänzung der Log-Meldungen um Span- und Trace-ID sowie den jeweiligen Anwendungsnamen. So lassen sich Anwendungs- und Log-Daten zusammenführen und Probleme viel rascher beheben.



Log-Filterung zur Abbildung von Fehlern im Trace-Kontext in der New Relic Observability-Plattform

Log-Levels
Wo für Entwickler:innen, DevOps-Anwender:innen und Manager häufig die Rede von Schweregraden ist, beziehen sich Log-Levels ganz ähnlich auf die Wichtigkeit eines Events sowie die Detaildichte aus dem Logging-Framework. Sie stehen dabei im Zusammenhang mit Begriffen rund um Fehlerbehebung, Warnungen, Fehlern etc. Ein Schwereattribut macht es einfacher, weniger wichtige Informationen rascher herauszufiltern oder gänzlich zu eliminieren.

Effektiv eingesetzt halten Log-Levels so im Falle von zentralisiertem Log-Management auch Datenmengen und Kosten im Zaum und sorgen für dauerhaft zügige Suchabläufe. In einigen Fällen ist eine präzise Kontrolle über die Log-Generierung nicht konkret umsetzbar, und so lassen sich hiermit auch unerwünschte Daten erfolgreich beseitigen. Über New Relic können Ausreißer mit maschinellem Lernen basierend auf dem Log-Level als solche erfasst werden. Farbcodierungen vermitteln zudem einen zusätzlichen visuellen Indikator, um die nächsten Schritte entsprechend priorisiert zu lenken.

Log-Levels sollten dabei aber auch umsichtig eingesetzt werden, speziell bei der Fehlerbehebung. Debugging kann sich im Zusammenhang mit sehr ausführlichen Meldungen und spezifisch eingrenzbarem Verhalten als sehr nützlich erweisen. Kommt es aber ungezielt zum Einsatz, entstehen dabei auch unnötige Log-Volumina. Erfassungs- und Suchfunktionen werden verlangsamt, ohne dass parallel zusätzlicher Nutzen generiert wird. Für größere Teams und Projekte ratsam ist daher ein gemeinsamer Standard für Log-Levels mit konsistenten Methoden zu Gruppierung, Kategorisierung und Logging.

Logging-Tools und -Frameworks

Statt also Zeit und Ressourcen auf die Entwicklung und Implementierung einer komplett neuen Logging-Lösung aufzuwenden, bieten sich ein etabliertes, konsequent getestetes Tool und Framework meist eher an. Die New Relic APM Language-Agents etwa erweitern Logs um die notwendigen Metadaten, um so für automatischen Zugriff auf das Logs in Context-Feature der Plattform zu gewähren. Das Log-Forwarding läuft zudem ganz ohne Implementierung oder Verwaltung externer Software ab.

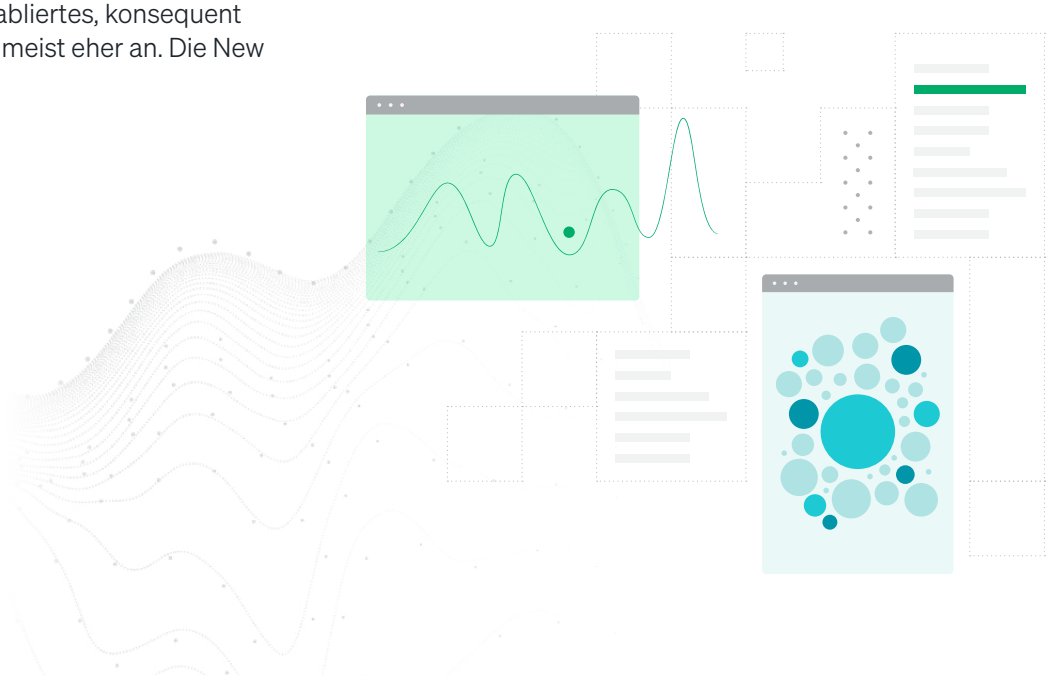
Ein konsistentes Logging-Framework vereinfacht die Umsetzung seitens der Engineering-Teams, standardisiert Log-Output und macht Logs in Context umfassend einsatzfähig. Bei der Einführung von Logging-Frameworks ist Umsicht geboten. Ihre Performance-Auswirkungen wollen genauso getestet werden wie neuer Code.

Hohe Werte sind Referenzpunkte, nicht Inhaltsfaktoren

In bestimmten Fällen kann ein größerer Datensatz vonnöten sein, um umfassenderen Kontext zu liefern. Ein Speicher-Export etwa, mehrere Dateien oder Bilder. Dabei empfiehlt es sich zumeist, die Dateien separat zu speichern oder auf einen eigenen Server hochzuladen, um dann im Log auf diesen zu verweisen und nicht etwa seine gesamte Adresse in selbigem zu speichern. Ihre Logs sollten so kompakt wie möglich bleiben und separat auf diese Daten zugreifen.

Nützliche Ansichten, Abfragen und Alerts weitergeben

Erstellen und teilen Sie Standard-Visualisierungen, -Abfragen und -Alerts für die Logs Ihres Teams. Dies vermittelt Ihrer Abteilung und Ihrem Unternehmen generell einen klareren Status Quo, informiert teamübergreifend und verbessert die Kommunikation. Und es zeigt eine beispielhafte Benefit-Kette von Full-Stack Observability auf.



Was nicht Teil Ihrer Logging-Methodik sein sollte

Was Nutzen und Mehrwert generieren kann, darf und soll seinen Weg schon in Ihre Logs finden. Wie bei vielem im Leben gibt es aber auch hier Ausnahmen und Fallstricke, die es zu vermeiden gilt.

Sensible Daten

Behandeln Sie sensible Daten mit größter Vorsicht. Regulierte Informationen wie personenbezogene Daten und Kreditkarten-Elemente müssen umfassend geschützt bleiben. Gesetze wie die Datenschutz-Grundverordnung (DSGVO) auf EU-Ebene[3] sowie der Health Insurance Portability and Accountability Act (HIPAA) der USA[4] seien hier als besonders prominente Beispiele genannt.

Die Logging-Richtlinien des Open Web Application Security Project (OWASP) geben an, welche Elemente nicht ihren Weg in Logs finden sollen. Hierzu gehören unter anderem Zugangs-Tokens, Passwörter, sensible Informationen und generell Inhalte, die – gerade auch in den Augen anderer – privat bleiben sollten.[5]

Werden Logs auf einem privaten Server oder in einer Privatdatenbank gespeichert, kann es leicht unbeabsichtigt zur Erfassung von personenbezogenen Daten wie Namen und E-Mail-Adressen kommen. Ist das Tracking der Abläufe und Events eines Benutzers aber unabdingbar, sind anonyme Identifikatoren hingegen sehr sinnvoll. Ihre Log-Daten sind in einer Observability-Plattform wie der von New Relic zwar sicher. Doch eine Übertragung von personenbezogenen Informationen über Unternehmensgrenzen hinaus will freilich weiterhin tunlichst vermieden werden.

Quellcode und proprietäre Daten

Neben regulierten und compliance-relevanten Informationen gibt es womöglich noch weitere Daten, die Sie nicht in Ihren Logs speichern möchten. Hierbei kann es sich um Quellcode aus Anwendungen oder um geschützte Daten in Ihrem Unternehmen handeln.

Neben der sicheren Speicherung von Logs ist auch der Schutz auf Access-Ebene von größter Bedeutung. Ebenso sollten auch keine Handelsgeheimnisse oder noch nicht veröffentlichte Lösungen oder Features ihren Weg in Logs finden, vor allem falls Sie Ihre Logs in einer externen Lösung speichern.

Informationsdoubletten

Doppelte Informationen in Ihren Logs richten per se keinen Schaden an, und zu viel ist typischerweise immer noch besser als zu wenig. Viele doppelte Informationspunkte führen jedoch zu unnötigen Logs und höheren Kosten, ohne dabei überhaupt einen Zweck zu erfüllen.

[3] (Europäische Kommission, n.d.)
[4] (Ministerium für Gesundheitspflege und Soziale Dienste der USA, n.d.)
[5] (Open Web Application Security Project (OWASP), n.d.)

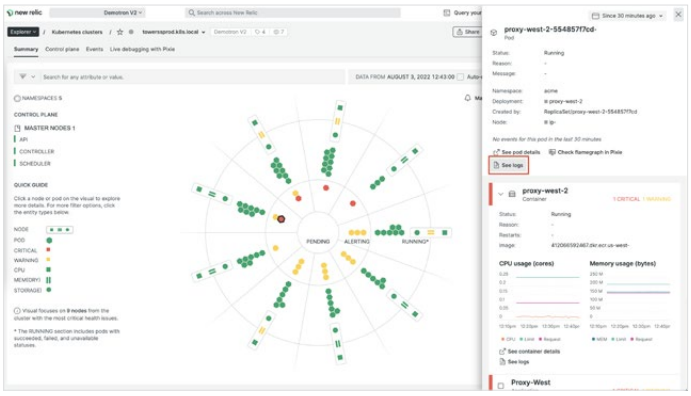
Fazit

Mit Logs erhält Full-Stack Observability einen weiteren Stärke-Faktor, werden wichtige Business-Entscheidungen in Echtzeit unterstützt. Ebenso wie Ihre Engineers, die weniger Zeit mit der Fehlerbehebung und Incident Response verbringen und dafür mehr mit der Entwicklung innovativer Technologien.

Diese Best Practices vermitteln Ihnen dabei die Details, die Sie für mehrere essenzielle Ziele benötigen: ein nahtloses, effizientes Kundenerlebnis, Transparenz für Ihren Gesamt-Stack, um Probleme rascher zu beheben, und schnellere Entwicklungs-Zyklen.

New Relic: Die Observability-Plattform

Mit New Relic erhalten Sie eine zentrale Plattform für alle Telemetriedaten und damit auch für Ihr Log-Management. Die New Relic Observability-Plattform führt verschiedene wichtige Feature-Bereiche konsolidiert zusammen: Log-Management, APM, Infrastruktur-Monitoring, Serverless-Monitoring, Mobile Monitoring, Browser- und Synthetic Monitoring, Distributed Tracing sowie Kubernetes-Monitoring. Möglich werden so konzertierte Visualisierung, Analyse und Fehlerbehebung Ihres gesamten Software-Stacks. New Relic Log-Management macht in diesem Zuge auch die Kombination von Log- mit Anwendungsdaten und Infrastruktur-Monitoring direkt umsetzbar – und eine in allen Belangen präzise Observability-Plattform Realität.



APM, Infrastruktur, Events und Log-Zugriff zentral konsolidiert

Metrics, Events, Logs und Traces aus dem gesamten Stack werden in New Relic zentral erfasst und im Rahmen von AIOps mit KI-Features erweitert nutzbar. Logs lassen sich so schneller und kosteneffizienter durchsuchen als mit fragmentierten Legacy-Lösungen. Statt nun also mit einem wahren Log-Flickenteppich anzusetzen, erhalten Ihre Entwickler:innen so ansatzlos alle Details zu einem spezifischen Fehler.

Auch Probleme in punkto Geschwindigkeit und Skalierbarkeit sorgen bei traditionellen Lösungen für mühsame Abfragesequenzen, die sich über Minuten oder gar Stunden hinziehen können. Suchabfragen für Log-Management mit New Relic nehmen hingegen nur Sekunden in Anspruch, machen es so schneller als je zuvor möglich, Incidents in Ihrem gesamten Stack zu adressieren.

Die New Relic Observability-Plattform bietet Features für Log-Management, eine kostenlose Produktvariante für Kund:innen mit geringeren Datenvolumina sowie ein attraktives Kostenmodell mit Abrechnung auf GB-Basis, das die Erfassung aller relevanten Logs sicherstellt.

Log-Management in New Relic können Sie noch heute zu nutzen: Das Einstiegskonto ist komplett kostenlos – dies mit 100 GB zur Datenerfassung, einer Komplettlizenz und unbegrenzten Basic-Lizenzen.

Jetzt registrieren

Bibliographie

Europäische Kommission. „EU data protection rules“. Europäische Kommission. Letzter Zugriff: 19. Juli 2022. https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_de.

New Relic, Inc. „Parsing log data“. Dokumentation von New Relic. Letzter Zugriff: 19. Juli 2022. https://docs.newrelic.com/docs/logs/ui-data/parsing/#custom-parsing.

Open Telemetry. „Open Telemetry Logging Overview“. Open Telemetry. Letzter Zugriff: 18. Juli 2022. https://opentelemetry.io/docs/reference/specification/logs/overview/.

Open Web Application Security Project (OWASP). „OWASP Logging Guide“. https://owasp.org/www-pdf-archive/OWASP_Logging_Guide.pdf.

Ministerium für Gesundheitspflege und Soziale Dienste der USA. „Summary of the HIPAA Security Rule“. HHS.gov. Letzter Zugriff: 19. Juli 2022. https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html.

new relic.