WEISSENBERG

Was verbirgt sich hinter dem Begriff DevOps?

DevOps als Brückenbauer zwischen Entwicklung und Betrieb

DevOps ist eine Philosophie, die darauf abzielt, zwischen Entwicklung und Betrieb eine veränderte Denkweise und eine bessere Zusammenarbeit zu implementieren, um historisches Silodenken zu überwinden. Der Begriff DevOps ist eine Kombination der Wörter „Development" und „Operations".

Was ist DevOps?

DevOps ist ein Ansatz, der die Prozesse zwischen Softwareentwicklung und operationalen IT-Teams automatisiert und optimiert, damit Software schneller und zuverlässiger erstellt, getestet und freigegeben werden kann. Bei DevOps geht es darum, die Barrieren zwischen traditionell isolierten Entwicklungs- und operationellen IT-Teams zu beseitigen. Bei einem DevOps-Modell arbeiten Entwicklungs- und Betriebsteams über den gesamten Lebenszyklus von Softwareanwendungen – von der Entwicklung und dem Test über die Bereitstellung bis hin zum Betrieb – zusammen.
Die DevOps-Bewegung konstituierte sich zwischen 2007 und 2008, als sich Softwareentwickler und IT-Experten über die fatale Funktionsstörung bewusst wurden, die dem traditionellen Softwareentwicklungsmodell innewohnte: die organisatorische und funktionale Trennung derjenigen, die den Code schreiben, von denen, die diesen Code bereitstellen und unterstützen. In unterschiedlichen Abteilungen mit unterschiedlichen Leistungsindikatoren, anhand derer sie beurteilt wurden, arbeiteten sie auf separaten Etagen oder in separaten Gebäuden.
Zu den Gründervätern der DevOps-Bewegung gehören Patrick Debois, Andrew Shafer und John Allspaw, die bis 2011 die DevOps-Bewegung alleine vorantrieben. 2011 begann sich dann die Bewegung zu etablieren und erregte die Aufmerksamkeit von Analysten wie Cameron Haight von Gartner. 2012 entwickelte sich DevOps immer weiter, was 2013 dann mehrere Autoren dazu inspirierte, Bücher zu diesem Thema zu schreiben. Bemerkenswerte Beispiele sind das Phoenix-Projekt von Gene Kim, Kevin Behr und George Spafford sowie die Implementierung einer schlanken Softwareentwicklung von Mary und Tom Poppendiek. 2014 fand DevOps Eingang in die ersten großen Unternehmen wie Nordstrom und LEGO.

Wie funktioniert der DevOps-Ansatz?

DevOps ist eine Methode zur Verbesserung der Arbeit während des gesamten Lebenszyklus der Softwareentwicklung. Der DevOps-Prozess gleicht einem „Endless Loop", der von der Softwareplanung über Code-, Build-, Test- und Release-Phasen über die Bereitstellung, den Betrieb, die laufende Überwachung und das Feedback wieder zur Planung zurückführt. Im Idealfall bedeutet DevOps, dass ein Entwickler-Team Software schreibt, die die Benutzeranforderungen perfekt erfüllt, ohne Zeitverschwendung bereitgestellt wird und beim ersten Einsatz optimal ausgeführt wird.
Eine grundlegende DevOps-Praxis besteht darin, sehr häufige, aber kleine Updates durchzuführen. Diese Updates sind normalerweise inkrementeller als die Updates, die unter herkömmlichen Release-Praktiken durchgeführt werden. Unternehmen, die ein DevOps-Modell verwenden, stellen Updates viel häufiger bereit als Unternehmen, die herkömmliche Softwareentwicklungsmethoden verwenden.
Um Wartezeiten zu vermeiden, verwenden IT-Teams sogenannte Continuous Integration/Continuous Delivery (CI/CD) -Pipelines, um Code von einem Entwicklungs- und Bereitstellungsschritt in einen anderen zu verschieben. Eine CI/CD-Pipeline umfasst eine Reihe von Schritten, die durchgeführt werden müssen, um eine neue Softwareversion auszuliefern. Dabei handelt es sich um eine Praxis, die sich auf die Verbesserung der Softwarebereitstellung konzentriert.

Die wichtigsten DevOps-Methoden und Prinzipien

Zu den gängigen und beliebten DevOps-Methoden zählen Scrum, Kanban und Agile:

Scrum

Scrum definiert, wie Mitglieder eines Teams zusammenarbeiten sollen, um Entwicklungs- und QS-Projekte zu beschleunigen. Zu den Scrum-Praktiken gehören wichtige Workflows und spezifische Begriffe (Sprints, Zeitfenster, tägliches Scrum-Meeting) sowie bestimmte Rollen (Scrum Master, Product Owner).

Kanban

Kanban entstand aus Effizienzgewinnen, die in der Toyota-Fabrik erzielt wurden. Kanban schreibt vor, dass der Status der laufenden Softwareprojekte auf einer Kanban-Karte verfolgt wird. Dabei ist die Anzahl paralleler Arbeiten, der Work in Progress (WiP), begrenzt. Auf diese Weise sollen kürzere Durchlaufzeiten erreicht und Probleme – insbesondere Engpässe – schnell sichtbar gemacht werden.

Agile

Frühere agile Softwareentwicklungsmethoden haben weiterhin großen Einfluss auf die DevOps-Praktiken und -Tools. Viele DevOps-Methoden, einschließlich Scrum und Kanban, enthalten Elemente der agilen Programmierung, die sich durch eine größere Reaktionsfähigkeit auf sich ändernde Anforderungen auszeichnet, indem tägliche Stand-ups durchgeführt und kontinuierliches Kundenfeedback einbezogen werden. Agile schreibt auch kürzere Softwareentwicklungszyklen anstelle langwieriger traditioneller „Waterfall" -Entwicklungsmethoden vor.

DevOps-Praktiken

Die DevOps-Praktiken spiegeln die Idee einer kontinuierlichen Verbesserung und Automatisierung wider. Viele Praktiken konzentrieren sich auf eine oder mehrere Entwicklungszyklusphasen.
Diese Praktiken umfassen:

Fortlaufende Entwicklung

Diese Vorgehensweise umfasst die Planungs- und Codierungsphasen des DevOps-Lebenszyklus. Teilweise sind Versionskontrollmechanismen beteiligt.

Kontinuierliche Prüfung

Diese Vorgehensweise umfasst automatisierte, vorab geplante, fortgesetzte Codetests, während der Anwendungscode geschrieben oder aktualisiert wird. Solche Tests können die Lieferung von Code an die Produktion beschleunigen.

Kontinuierliche Integration

Diese Vorgehensweise kombiniert CM-Tools (Configuration Management) mit anderen Test- und Entwicklungstools, um zu verfolgen, wie viel des zu entwickelnden Codes für die Produktion bereit ist. Sie beinhaltet ein schnelles Feedback zwischen Test und Entwicklung, um Codeprobleme schnell zu identifizieren und zu lösen. Das Herzstück von CI ist ein zentrales Versionsverwaltungssystem, dessen Hauptzweck darin besteht, den Teams dabei zu helfen, den Code zu organisieren, Änderungen zu verfolgen und automatisierte Tests zu ermöglichen. Im Gegensatz zur herkömmlichen Vorgehensweise sollen bei einer kontinuierlichen Integration die Entwickler ihren Code täglich einreichen, um so Fehler schneller zu erkennen und letztendlich weniger Zeit damit zu verbringen, sie zu beheben. Wenn ein Entwickler neuen Code in das gemeinsam genutzte Code-Repository überträgt, wird eine Automatisierung aktiviert, um den neuen und vorhandenen Code in einem Build zu kompilieren. Wenn der Erstellungsprozess fehlschlägt, erhalten die Entwickler eine Warnung, die sie darüber informiert, welche Codezeilen überarbeitet werden müssen.

Kontinuierliche Lieferung

In einem typischen DevOps-Szenario verschieben Entwickler ihren Code zunächst in eine Vorproduktions- oder Staging-Umgebung, um zu bewerten, wie er sich verhält. An diesem Punkt in der Pipeline muss dann entschieden werden, ob das Build in die Produktion verlagert oder zur weiteren Bewertung zurückgehalten wird. Code-Updates sollten so oft wie möglich bereitgestellt werden, um die kontinuierliche Bereitstellung optimal nutzen zu können. Die Veröffentlichungshäufigkeit hängt vom Workflow ab, folgt aber normalerweise einem täglichen, wöchentlichen oder monatlichen Rhythmus. Durch die Freigabe von Code in kleineren Blöcken werden Engpässe vermieden und ein stetiger, kontinuierlicher Integrations-Fluss gewährleistet.

Kontinuierliche Bereitstellung

Ähnlich wie bei der kontinuierlichen Lieferung automatisiert diese Vorgehensweise die Freigabe von neuem oder geändertem Code in der Produktion. Ein Unternehmen, das eine kontinuierliche Bereitstellung durchführt, veröffentlicht möglicherweise mehrmals täglich Code- oder Funktionsänderungen. Die Verwendung von Containertechnologien wie Docker und Kubernetes kann eine kontinuierliche Bereitstellung ermöglichen, indem die Konsistenz des Codes über verschiedene Bereitstellungsplattformen und -umgebungen hinweg gewährleistet wird. Um das volle Potenzial einer kontinuierlichen Bereitstellung auszuschöpfen, müssen robuste Test-Frameworks vorhanden sein, die sicherstellen, dass der neue Code wirklich fehlerfrei ist und sofort für die Produktion bereitgestellt werden kann.

Kontinuierliche Überwachung

Diese Vorgehensweise umfasst die fortlaufende Überwachung sowohl des sich in Betrieb befindlichen Codes als auch der zugrunde liegenden Infrastruktur, die ihn unterstützt. Eine Rückkopplungsschleife, die über Fehler oder Probleme berichtet, führt dann zur Entwicklung zurück.

Infrastruktur als Code

Diese Vorgehensweise kann in verschiedenen DevOps-Phasen verwendet werden, um die Bereitstellung der für eine Softwareversion erforderlichen Infrastruktur zu automatisieren. Entwickler fügen Infrastruktur-Code aus ihren vorhandenen Entwicklungstools hinzu. Beispielsweise können Entwickler bei Bedarf ein Speichervolumen von Docker, Kubernetes oder OpenShift erstellen. Diese Vorgehensweise ermöglicht es Betriebsteams auch, Umgebungskonfigurationen zu überwachen, Änderungen zu verfolgen und das Zurücksetzen von Konfigurationen zu vereinfachen.

Was ist eine Pipeline bei DevOps?

Die Zeiten, in denen Entwickler jahrelang neue Softwareprodukte entwickelten, bevor sie veröffentlicht werden konnten, sind lange vorbei. Softwareunternehmen setzen heute auf effektive DevOps-Pipelines, um mit den Anforderungen der Kunden Schritt zu halten.

Eine DevOps-Pipeline besteht aus einer Reihe von Methoden, die die Entwicklungsteams und Operationsteams implementieren, um Software schneller und einfacher zu erstellen, zu testen und bereitzustellen. Einer der Hauptzwecke einer Pipeline besteht darin, den Softwareentwicklungsprozess organisiert und fokussiert zu halten.
Eine DevOps-Pipeline muss man sich wie eine Montagelinie in einer Autofabrik vorstellen. Bevor der Hersteller ein Auto der Öffentlichkeit vorstellt, durchläuft es zahlreiche Montagephasen, Tests und Qualitätsprüfungen. Das Chassis wird zusammengebaut, Motor, Räder, Türen angebracht, die Elektronik eingebaut und das Auto lackiert, um es für die Kunden attraktiv zu machen. Die DevOps-Pipelines funktionieren ähnlich.
Bevor eine App oder eine neue Funktion für Benutzer freigegeben wird, muss zuerst der Code geschrieben werden. Sodann wird sichergestellt, dass keine schwerwiegenden Fehler auftreten, die zum Absturz der App führen können. Um ein solches Szenario zu vermeiden, müssen verschiedene Tests durchgeführt werden, um Fehler herauszufischen. Sobald alles wie beabsichtigt funktioniert, kann der Code für Benutzer freigegeben werden. Damit wird klar, dass eine DevOps-Pipeline aus den Phasen Entwickeln, Erstellen, Testen und Bereitstellen besteht.

Phase 1: Entwickeln

In der Entwicklungsphase beginnen Entwickler mit dem Codieren. Abhängig von der Programmiersprache werden auf den lokalen Computern geeignete IDEs, Code-Editoren und andere Technologien installiert, um maximale Produktivität zu erzielen. In den meisten Fällen müssen Entwickler bestimmte Codierungsstile und -standards befolgen, um ein einheitliches Codierungsmuster sicherzustellen. Dies erleichtert jedem Teammitglied das Lesen und Verstehen des Codes. Ist der Code erstellt, wird eine Pull-Anforderung an das gemeinsam genutzte Quellcode-Repository gestellt. Der neu übermittelte Code kann dann manuell überprüft und mit dem Hauptzweig zusammengeführt werden, indem die Pull-Anforderung genehmigt wird.

Phase 2: Erstellen

Die Erstellungsphase einer DevOps-Pipeline ist von entscheidender Bedeutung, da Entwickler Fehler im Code erkennen können, bevor sie eine größere Katastrophe verursachen. Nachdem der neu geschriebene Code mit dem freigegebenen Repository zusammengeführt wurde, führen Entwickler eine Reihe automatisierter Tests durch. In einem typischen Szenario initiiert die Pull-Anforderung einen automatisierten Prozess, der den Code zu einem Build kompiliert – einem bereitstellbaren Paket oder einer ausführbaren Datei. Wenn es ein Problem mit dem Code gibt, schlägt der Build fehl und der Entwickler wird über die Probleme informiert. In diesem Fall schlägt auch die Pull-Anforderung fehl. Dieser Vorgang wird jedes Mal wiederholt, wenn Code an das gemeinsam genutzte Repository gesendet wird, um sicherzustellen, dass nur fehlerfreier Code in der Pipeline weiterverarbeitet wird.

Phase 3: Testen

Wenn der Build erfolgreich ist, geht er in die Testphase über. Dort führen Entwickler manuelle und automatisierte Tests durch, um die Integrität des Codes weiter zu überprüfen. In den meisten Fällen wird ein Benutzerakzeptanztest durchgeführt. Die Benutzer interagieren mit der App als Endbenutzer, um festzustellen, ob für den Code zusätzliche Änderungen erforderlich sind, bevor sie an die Produktion gesendet werden. In dieser Phase ist es auch üblich, Sicherheits-, Leistungs- und Lasttests durchzuführen.

Phase 4: Bereitstellen

Wenn der Build die Bereitstellungsphase erreicht, kann die Software in die Produktion übertragen werden. Eine automatisierte Bereitstellungsmethode wird verwendet, wenn der Code nur geringfügige Änderungen benötigt. Wenn die Anwendung jedoch einer umfassenden Überarbeitung unterzogen wurde, wird der Build zunächst in einer produktionsähnlichen Umgebung bereitgestellt, um das Verhalten des neu hinzugefügten Codes zu überwachen.
Die Implementierung einer blaugrünen Bereitstellungsstrategie ist auch bei der Veröffentlichung wichtiger Updates üblich. Eine blaugrüne Bereitstellung bedeutet, dass zwei identische Produktionsumgebungen vorhanden sind, in denen eine Umgebung die aktuelle Anwendung und die andere Umgebung die aktualisierte Version hostet. Um die Änderungen für den Endbenutzer freizugeben, können Entwickler einfach alle Anforderungen an die entsprechenden Server weiterleiten. Wenn es Probleme gibt, können Entwickler einfach zur vorherigen Produktionsumgebung zurückkehren, ohne Serviceunterbrechungen zu verursachen.
Das ultimative Ziel einer DevOps-Pipeline ist die Schaffung eines wiederholbaren Systems, das die Automatisierung nutzt und kontinuierliche Verbesserungen ermöglicht, um qualitativ hochwertige Produkte schneller und einfacher zu liefern.

DevOps und Container

Was sind Container?

Container bieten Entwicklern die Möglichkeit, Quellcode, Konfigurationsdateien, Bibliotheken, Binärdateien und alle Abhängigkeiten in einem einzelnen Paket zusammenzufassen, um eine Anwendung bei einem Wechsel von einer Computerumgebung zur anderen schnell und zuverlässig ausführen zu können. Durch die Containerisierung der Anwendungsplattform und ihrer Abhängigkeiten werden Unterschiede in der Betriebssystemverteilung und der zugrunde liegenden Infrastruktur beseitigt. Mit einem einzigen Container können kleine Microservices, aber auch große Softwareanwendungen ausgeführt werden. In Bezug auf die Bereitstellung umfangreicherer Anwendungen werden mehrere Container als ein oder mehrere Containercluster bereitgestellt. Diese Cluster werden von einem Container-Orchestrator wie Kubernetes gesteuert und verwaltet.

Warum Container in DevOps verwenden?

Wenn die Anwendung von einer Computerumgebung in eine andere verschoben wird, z. B. vom Laptop eines Entwicklers zu einer virtuellen Testumgebung zum Testen oder von einer Staging-Umgebung zur Produktionsumgebung, kann es zu Probleme kommen, wenn in beiden Umgebungen die Softwarekonfiguration, die Sicherheitsrichtlinien, der Speichertyp und die Netzwerktopologie unterschiedlich sind. Um dieses Problem zu lösen, kommen Container ins Spiel, für die Betriebssysteme und Infrastruktur keine Rolle spielen. Container können in jeder Umgebung reibungslos betrieben werden.

Vorteile von Containern in DevOps

Container bieten die Möglichkeit, Software zu entwerfen, zu testen und in einer Produktionsumgebung oder in verschiedenen Cloud-Umgebungen bereitzustellen. Container bieten eine größere Flexibilität, konsistentere Operationen, eine höhere Produktion und einen geringeren Overhead.

1. Verbesserte Flexibilität

Für viele verschiedene Betriebssysteme und Hardwareplattformen können Programme, die in Containern ausgeführt werden, schnell bereitgestellt werden.

2. Die Operation ist konsistenter

Systeme in Containern können unabhängig von ihrem Einsatzort immer gleich ausgeführt werden.

3. Mehr Produktion

Container ermöglichen eine schnellere Bereitstellung von Patches und die Skalierung von Anwendungen.

4. Der Overhead ist geringer

Container erfordern weniger Systemressourcen als herkömmliche Maschinenumgebungen oder die Hardware für virtuelle Maschinen, da sie keine Images vom Betriebssystem enthalten.
Mit einem CaaS-Content-Management-System können Unternehmen App-Inhalte dynamisch aktualisieren.

Was sind die Vorteile von DevOps?

Teams, die nach dem DevOps-Ansatz vorgehen, arbeiten schneller, stellen Anwendungen häufiger und in kürzeren Zeitabständen bereit, produzieren qualitativ hochwertige Software, reagieren auf Zwischenfälle schneller und verbessern generell die Zusammenarbeit und Kommunikation zwischen den Teams. Die Zusammenarbeit, die gemeinsame Verantwortung und Problemlösung, die Transparenz und schnelles Feedback sind die wichtigsten Elemente einer erfolgreichen DevOps-Kultur. Teams, die den DevOps-Ansatz verfolgen, veröffentlichen Ergebnisse häufiger mit höherer Qualität und Stabilität.
Mit Tools, die die Automatisierung und neue Prozesse vorantreiben, können DevOps-Teams die Produktivität steigern und häufiger auch problemloser veröffentlichen. Volle Transparenz und nahtlose Kommunikation ermöglichen es DevOps-Teams, Ausfallzeiten zu minimieren und Probleme schneller zu lösen. Mit etablierten Prozessen und einer klaren Priorisierung können Entwicklungs- und Betriebsteams besser mit ungeplanten Arbeiten umgehen und sich weiterhin auf geplante Arbeiten konzentrieren. Durch erhöhte Sichtbarkeit und proaktive Rückschau können Teams ungeplante Arbeiten jedoch besser antizipieren und teilen.
Teams, die die DevOps-Möglichkeiten voll ausnutzen, arbeiten intelligenter und schneller und bieten ihren Kunden eine bessere Qualität. Der vermehrte Einsatz von Automatisierung und funktionsübergreifender Zusammenarbeit reduziert Komplexität und Fehler, was wiederum die mittlere Reparaturzeit bis zur Wiederherstellung (Mean Time to Recovery, MTTR) bei Systemvorfällen und -ausfällen verbessert.

Warum setzen Unternehmen DevOps ein?

In traditionellen Organisationen wird der Erfolg von Entwicklungs- und Betriebsteams an unterschiedlichen Ergebnissen gemessen. Während die Anzahl und Qualität der bereitgestellten Updates für Entwickler von entscheidender Bedeutung ist, legen die Betriebsteams mehr Wert auf die Aufrechterhaltung des Systemzustands. Dieser weitverbreitete Zustand ist für den Erfolg eines Unternehmens aber kontraproduktiv, da Unternehmen sowohl neue Funktionen bereitstellen als auch Stabilität gewährleisten müssen. Und hier kommt DevOps ins Spiel.
DevOps bietet Unternehmen viele greifbare Vorteile.

1. Technische Vorteile

Da es bei DevOps darum geht, die Zusammenarbeit zwischen Entwicklungs- und Betriebsteams zu verbessern, führt dies direkt zu kürzeren Entwicklungszyklen, wodurch die Häufigkeit der Freigabe von Code für die Produktion erhöht wird.
Traditionelle Organisationen benötigen 3 bis 6 Monate von der Produktanforderung bis zum Release. Durch die Einführung von DevOps kann derselbe Zyklus auf einen täglichen oder sogar stündlichen Release-Build-Zyklus verkürzt werden. Diese Art der kontinuierlichen Entwicklung und Bereitstellung schafft einen Wettbewerbsvorteil für Unternehmen und erhöht den Wert der IT für das Unternehmen.
Die agile Methodik ist das Rückgrat von DevOps. Durch die Verbesserung der Zusammenarbeit, die Förderung der iterativen Entwicklung und der modularen Programmierung sowie die Aufteilung großer Codebasen in kleinere Teile ist DevOps ein Ansatz, der den Entwicklungs- und Betriebsteams das Leben erleichtert.

2. Kulturvorteile

Ein weiterer Bereich, in dem DevOps handfeste Vorteile bringt, ist die Unternehmenskultur. Die verstärkte Kommunikation und Zusammenarbeit zwischen den Entwicklungs- und Betriebsteams bedeutet, dass sie nicht mehr in Silos arbeiten, sondern frei miteinander kommunizieren können, indem sie Wissen und Best Practices austauschen, um einen robusten Prozess aufzubauen. DevOps fördert eine Kultur des Vertrauens zwischen Teammitgliedern und der Risikoteilung. Es ermutigt die Teams, kontinuierlich zu experimentieren, um die Produkte und Dienstleistungen des Unternehmens zu verbessern. Auf diese Weise können sowohl Entwicklungs- als auch Betriebsteams neue Kundenbedürfnisse eruieren und Innovationen entwickeln, um diese zu adressieren. Schließlich bringt DevOps eine dauerhafte Qualität in die Unternehmenskultur ein. Anstelle traditioneller regelbasierter oder machtbasierter Kulturen zielt DevOps darauf ab, die Bürokratie zu reduzieren, die Teil des Prozesses ist, was zu einer zufriedeneren und daher produktiveren Belegschaft führt. Dies wirkt sich direkt auf die Leistung des gesamten Unternehmens aus.

3. Geschäftsvorteile

Einer der wichtigsten Vorteile des DevOps-Modells ist seine hohe Geschwindigkeit. Unternehmen verschaffen sich einen Wettbewerbsvorteil, indem sie sich an veränderte Märkte anpassen, schneller Innovationen umsetzen und ihre Geschäftsziele effizienter erreichen können. Da DevOps auf Lean-Prinzipien basiert, ist die Reduzierung von Ineffizienz für jede DevOps-Implementierung von entscheidender Bedeutung. DevOps ist ein Ansatz, der sich für Unternehmen eignet, die durch die kontinuierliche Bereitstellung von Produkten und Dienstleistungen, die den Anforderungen und Vorlieben der Endbenutzer entsprechen, agiler werden möchten.

Wie DevOps die IT beschleunigen kann

Während Entwicklungs- und Betriebsteams traditionell als getrennte Einheiten operieren, ermutigt DevOps die Teams, über die gesamte Anwendung hinweg zusammenzuarbeiten. Die Teams können dann Bereiche automatisieren, die in der Vergangenheit ineffizient und repetitiv waren.

Ein DevOps-Modell ermöglicht normalerweise häufigere Aktualisierungen als herkömmliche Entwicklungsmodelle, jedoch mit inkrementelleren Änderungen. Dies bedeutet, dass Teams Probleme schneller beheben können und jede Bereitstellung für sich genommen ein weitaus geringeres Risiko birgt. Im Bereich der Microservices hilft DevOps bei häufigen kleineren Updates, da Anwendungen in Dienste unterteilt werden, die für eine einzelne Funktion ausgeführt werden. Ein Element kann geändert werden, ohne das Ganze zu beeinflussen.

Ein entscheidender Vorteil bei der Arbeit mit DevOps-Pipelines und -Strategien ist die Geschwindigkeit. Innovationen lassen sich einfacher realisieren, wenn Änderungen häufiger und mit weniger Risiko bereitgestellt werden und Teams die Verantwortung für ihre eigenen Dienste übernehmen können, ohne sich Sorgen machen zu müssen, dass dies Auswirkungen auf andere Bereiche der Anwendung hat.

MILAD SAFAR

Managing Partner und Autor zahlreicher Veröffentlichungen zum Themenfeld Digitalisierung