Schneller mit kontinuierlicher Lieferung

Mark Mansour

Schneller mit kontinuierlicher Lieferung

Kontinuierliche Verbesserung und Software-Automatisierung

Vor über 10 Jahren haben wir bei Amazon ein Projekt durchgeführt, um zu verstehen, wie schnell unsere Teams Ideen in hochwertige Produktionssysteme verwandeln. Dies hat uns dazu veranlasst, unseren Softwaredurchsatz zu messen, um die Ausführungsgeschwindigkeit zu verbessern. Wir stellten fest, dass es vom Einchecken des Codes bis zur Produktion durchschnittlich 16 Tage dauerte. Bei Amazon begannen die Teams mit einer Idee und brauchten dann in der Regel anderthalb Tage, um Code zu schreiben, um die Idee zum Leben zu erwecken. Wir haben weniger als eine Stunde gebraucht, um den neuen Code zu erstellen und bereitzustellen. Den Rest der Zeit, fast 14 Tage, haben wir darauf gewartet, dass die Teammitglieder einen Build starten, Bereitstellungen durchführen und Tests durchführen. Am Ende des Projekts empfahlen wir, unsere Post-Check-in-Prozesse zu automatisieren, um die Ausführungsgeschwindigkeit zu verbessern. Ziel war es, Verzögerungen zu vermeiden und die Qualität zu erhalten oder sogar zu verbessern.

Kern dieser Empfehlung war ein kontinuierliches Verbesserungsprogramm, um die Ausführungsgeschwindigkeit zu erhöhen. Wir haben unsere Entscheidung, unsere Ausführungsgeschwindigkeit zu verbessern, auf das Führungsprinzip des Beharrens auf den höchsten Standards gestützt. Bei diesem Prinzip geht es darum, unablässig hohe Standards zu setzen, die Messlatte immer höher zu legen und qualitativ hochwertige Produkte, Dienstleistungen und Prozesse bereitzustellen. Unsere Führungsprinzipien beschreiben, wie Amazon Geschäfte tätigt, wie Führungskräfte führen und wie wir den Kunden im Mittelpunkt unserer Entscheidungen halten.

Amazon hatte bereits Tools für die Softwareentwicklung entwickelt, um unsere Softwareentwickler produktiver zu machen. Wir haben unser eigenes zentralisiertes und gehostetes Build-System, Brazil, erstellt, das eine Reihe von Befehlen auf einem Server ausführt, um ein Artefakt zu generieren, das bereitgestellt werden kann. Zu diesem Zeitpunkt hörte Brasilien nicht auf Quellcode-Änderungen. Eine Person musste einen Build initiieren. Wir hatten auch unser eigenes Bereitstellungssystem, Apollo, für das ein Build-Artefakt hochgeladen werden musste, um eine Bereitstellung zu initiieren. Angespornt durch das Interesse an einer kontinuierlichen Lieferung in der Branche, haben wir unser eigenes System, Pipelines, gebaut, um den Software-Lieferprozess zwischen Brasilien und Apollo zu automatisieren.

Pipelines: Unser kontinuierliches Bereitstellungstool

Wir haben ein Pilotprogramm gestartet, um den Software-Bereitstellungsprozess für eine kleine Anzahl von Teams zu automatisieren. Bis wir fertig waren, konnte das Headline-Pilotenteam die Gesamtzeit vom Einchecken bis zur Produktion um 90 Prozent verkürzen.

Das Projekt validierte das Konzept einer Pipeline als Möglichkeit für unsere Teams, alle Schritte zu definieren, die erforderlich sind, um Software für Kunden freizugeben. Der erste Schritt in einer Pipeline ist das Erstellen eines Artefakts. Anschließend wird die Pipeline ausgeführt, die

das Artefakt in mehreren Schritten erstellt, bis das Artefakt für alle Kunden freigegeben wird. Wir verwenden Pipelines, um das Risiko zu verringern, dass sich eine neue Codeänderung negativ auf unsere Kunden auswirkt. Jeder Schritt in der Pipeline sollte unser Vertrauen stärken, dass das Build-Artefakt keine Fehler enthält. Wenn ein Defekt die Produktion erreicht, wollen wir die Produktion so schnell wie möglich wieder in einen gesunden Zustand versetzen.

Als wir Pipelines gestartet haben, konnte nur ein Release-Prozess pro Anwendung modelliert werden. Diese Einschränkung führte zu Konsistenz, Standardisierung und Vereinfachung der Freigabeprozesse eines Teams. Dies führte dazu, dass die Zahl der Mängel reduziert werden konnte. Bevor wir Pipelines verwendeten, war es für Teams üblich, unterschiedliche Release-Prozesse für Fehlerkorrekturen und wichtige Feature-Releases zu verwenden. Als andere Teams den Erfolg der Teams sahen, die die automatisierte Zustellung pilotiert hatten, begannen sie damit, ihre von Hand verwalteten Freigabeprozesse in Pipelines zu migrieren, um auch die Konsistenz zu verbessern. Teams, die früher verschiedene Freigabeprozesse hatten, verfügten nun über einen standardisierten Prozess, den alle verwendeten. Wenn sie ihre Freigabeprozesse in ein Tool verlagerten, überprüften die Teammitglieder häufig ihren Ansatz und fanden Wege, um ihren Prozess zu vereinfachen.

Das Pipelines-Team hatte sich zum Ziel gesetzt, die Nutzung durch „verführerische Adoption“ zu steigern. Mit anderen Worten, sie mussten das Produkt so gut machen, dass die Menschen verlangen werden, es zu verwenden. Wir haben die Anzahl der Teams gemessen, die mithilfe einer Pipeline ihre Software für die Produktion bereitgestellt haben, und wir haben Pipelines nach ihrem Automatisierungsgrad klassifiziert. Wir haben gesehen, wie Teams sich zum Ziel gesetzt haben, mithilfe einer Pipeline Software freizugeben und zu vollautomatisierten Releases überzugehen. Wir haben jedoch festgestellt, dass in einigen Organisationen die Art und Weise, wie wir die Qualität messen, dazu führen kann, dass Teams ihren Freigabeprozess automatisieren, ohne Tests durchführen zu müssen.

Die Antwort auf die Frage: „Wie viel Testen ist genug?“ Ist ein wertendes Urteil. Ein Team muss den Kontext verstehen, in dem es tätig ist. Um mit dieser Situation fertig zu werden, haben wir ein anderes Führungsprinzip angewendet, die Eigentümerschaft. Bei diesem Prinzip geht es darum, langfristig zu denken und nicht den langfristigen Wert für kurzfristige Ergebnisse zu opfern. Die Softwareteams von Amazon haben eine hohe Messlatte für das Testen und geben sich viel Mühe, da der Besitz eines Produkts auch die Folgen etwaiger Fehler in diesem Produkt mit sich bringt. Wenn ein Problem Auswirkungen auf Kunden hat, sind es Mitglieder des kleinen Single-Threaded-Softwareteams, die dieses Problem beheben und in Echtzeit beheben. Die Spannung zwischen der Erhöhung der Ausführungsgeschwindigkeit und der Reaktion auf Probleme in der Produktion bedeutet, dass die Teams motiviert sind, angemessene Tests durchzuführen. Wenn wir jedoch zu viel in Tests investieren, haben wir möglicherweise keinen Erfolg, weil andere sich schneller bewegt haben als wir. Wir sind stets bemüht, unsere Software-Release-Prozesse zu verbessern, ohne das Geschäft zu blockieren.

Ein weiteres Problem, mit dem wir konfrontiert waren, bestand darin, dass die Teams nicht lernten, wie sich Best Practices für Softwareversionen voneinander unterscheiden. Single-Threaded-Teams werden ermutigt, autonom zu arbeiten, und das bedeutete, dass die Ingenieure ihre Bereitstellungsprobleme unabhängig voneinander lösten. Wenn sie eine Lösung fanden, die ihren Softwareversionsanforderungen entsprach, wurden sie über

Mailinglisten, Betriebsbesprechungen und andere Kommunikationskanäle für die Technik anderer Ingenieure werben. Bei dieser Art der Kommunikation gab es zwei Probleme. Erstens handelt es sich bei diesen Kanälen um einen Kommunikationskanal, bei dem nicht alle von den neuen Techniken erfahren haben. Zweitens hatten Führungskräfte, die ihre Teams ermutigten, die neuen Best Practices zu übernehmen, keine Möglichkeit zu verstehen, ob ihre Teams die Arbeit geleistet hatten, die zur tatsächlichen Übernahme der Best Practices erforderlich war. Wir haben erkannt, dass wir allen Ingenieuren helfen müssen, Zugang zu den bewährten Methoden zu erhalten, die wir gelernt haben, und den Führungskräften die Möglichkeit geben müssen, Pipelines zu identifizieren, die besondere Aufmerksamkeit erfordern.

Unsere Lösung bestand darin, unser Lernen zu mechanisieren, indem wir den Tools, die wir zum Erstellen und Freigeben von Software verwenden, Überprüfungen auf Best Practices hinzufügen. Wir waren uns der Tatsache bewusst, dass eine bewährte Methode für eine Organisation möglicherweise keine bewährte Methode für eine andere Organisation ist. Daher haben wir zugelassen, dass diese Überprüfungen für jede Organisation einzeln konfiguriert werden. Durch Best-Practice-Prüfungen auf Organisationsebene konnten Führungskräfte ihre Freigabeprozesse an die Anforderungen ihres Unternehmens anpassen. Führungskräfte, die eine neue Best Practice fördern oder durchsetzen wollten, konnten zunächst eine Warnung aus den Tools heraus bereitstellen, die die Ingenieure täglich verwendeten. Durch das Platzieren von Nachrichten in den Tools wurde beinahe garantiert, dass die Teammitglieder mehr über die bewährten Methoden erfahren und wann diese bewährten Methoden wirksam werden. Wir stellten fest, dass ein Unternehmen die Möglichkeit hatte, seine Best-Practice-Prüfungen zu wiederholen und zu verbessern, indem es den Teams Zeit gab, sich mit neuen Best-Practice-Methoden vertraut zu machen und darüber zu debattieren. Dies führte letztendlich zu einer Verbesserung der Qualität der Best Practices und zu einem besseren Buy-In von Seiten der Ingenieure.

Wir haben systematisch die anzuwendenden Best Practices ermittelt. Eine Gruppe unserer erfahrensten Ingenieure katalogisierte häufige Gründe dafür, dass ein Release nicht funktionierte. Sie identifizierten die Schritte, die das Release zum Laufen gebracht hätten. Dann haben wir diese Liste verwendet, um unsere Best-Practice-Prüfungen zu erstellen. Durch diesen Prozess wurde uns klar, dass wir, obwohl wir wollten, dass neue Software-Revisionen die Kunden sofort, ohne Aufwand und ohne Einschränkung der Verfügbarkeit erreichen, zuerst die Verfügbarkeit priorisieren, dann die Geschwindigkeit und dann die Verfügbarkeit für unsere Ingenieure vereinfachen.

Reduzierung des Risikos, dass ein Defekt den Kunden erreicht

Es wird erwartet, dass alle Ingenieure irgendwann einen Defekt in eines unserer Systeme einführen. Unsere Freigabeprozesse, einschließlich unserer Pipelines und Bereitstellungssysteme, müssen so ausgelegt sein, dass diese Fehler so schnell wie möglich erkannt werden und sich nicht auf unsere Kunden auswirken. Wir müssen sicherstellen, dass unsere Freigabeprozesse korrekt konfiguriert sind und dass unser Build-Artefakt wie vorgesehen funktioniert.

Bereitstellungshygiene: Die grundlegendste Form von Bereitstellungstests stellt sicher, dass das neu bereitgestellte Artefakt gestartet werden und auf die Arbeit reagieren kann. Im Rahmen des Workflows nach der Bereitstellung führen wir schnelle Überprüfungen durch, um sicherzustellen, dass das neu implementierte Artefakt gestartet wurde und den Datenverkehr bedient. Beispielsweise verwenden wir Lifecycle-Ereignis-Hooks in der AWS CodeDeploy AppSpec-Datei, um einfache Skripts zum Stoppen, Starten und Überprüfen der Bereitstellung auszulösen. Wir überprüfen auch, ob wir über genügend Kapazität verfügen, um den Kundenverkehr zu bedienen. Wir haben Techniken wie ein Minimum an gesunden Hosts in CodeDeploy entwickelt, um sicherzustellen, dass wir immer über genügend Kapazität verfügen, um unsere Kunden zu bedienen. Wenn die Implementierungsengine einen Fehler erkennt, sollte sie die Änderung zurücksetzen, um die Zeit zu minimieren, in der Kunden einen Fehler bemerken.

Testen vor der Produktion: Eine der Best Practices von Amazon ist die Automatisierung von Unit-, Integrations- und Pre-Production-Tests und das Hinzufügen dieser Tests in unsere Pipeline. Wir bestehen darauf, Last- und Sicherheitstests durchzuführen, um diese Tests in unsere Pipelines aufzunehmen. Mit Unit-Tests sind alle Tests gemeint, die Sie möglicherweise auf Ihrem Build-Computer ausführen möchten, einschließlich Stilprüfungen, Codeabdeckung, Codekomplexität und mehr. Unter Integrationstests verstehen wir alle Off-Box-Tests wie Fehlerinjektion, automatisierte Browsertests und dergleichen. Es gibt viele großartige Artikel zu Unit- und Integrationstests, daher werde ich hier nicht näher darauf eingehen.

Mit unseren Unit- und Integrationstests möchten wir sicherstellen, dass das Verhalten unserer Build-Artefakte funktionell korrekt ist. Je mehr Validierungen wir durchführen, desto geringer ist das Risiko, dass ein Defekt unseren Kunden angezeigt wird. Um die Zeit zu verkürzen, die erforderlich ist, um ein Produkt in die Hände unserer Kunden zu bekommen, versuchen wir, einen Defekt so früh wie möglich im Freigabeprozess zu erkennen. Im Allgemeinen bedeutet dies, dass Sie bei kleineren und schnelleren Tests bei Problemen mit Ihren Änderungen schneller eine Rückmeldung erhalten.

Bei Amazon verwenden wir auch eine Technik, die wir *Tests vor der Produktion* nennen. Eine Vorproduktionsumgebung ist der letzte Ort, an dem Tests durchgeführt werden, bevor wir unsere Änderungen in der Produktion implementieren. Bei einem Umgebungstest vor der Produktion wird die Produktionskonfiguration des Systems verwendet, sodass sie genau wie ein Produktionssystem funktioniert. Dieser Ansatz hat zwei Vorteile. Zunächst testen Vorproduktionsumgebungen die Produktionskonfiguration, um sicherzustellen, dass der Dienst eine ordnungsgemäße Verbindung zu allen Produktionsressourcen, einschließlich Produktionsdatenspeichern, herstellen kann. Zweitens wird sichergestellt, dass das System korrekt mit den APIs der Produktionsservices interagiert, von denen es abhängt. Vorproduktionsumgebungen werden nur von dem Team verwendet, das diesen Service besitzt, und es wird kein Datenverkehr von Kunden gesendet. Das Ausführen von Tests vor der Produktion erhöht unser Vertrauen, dass derselbe Code und dieselbe Konfiguration in der Produktion funktionieren.

Validierung in der Produktion: Wenn wir unseren Kunden Code zur Verfügung stellen, tun wir nicht alles auf einmal. Der Umfang der Auswirkung der gleichzeitigen Mängelfreigabe an alle Kunden ist zu groß. Stattdessen stellen wir Zellen bereit - eine völlig unabhängige Instanz eines

Dienstes. Wenn wir Änderungen an unseren ersten Kunden in unserer ersten Zelle vornehmen, sind wir äußerst vorsichtig. Wir lassen nur eine kleine Anzahl von Kunden die neue Änderung sehen und sammeln Feedback, ob der neue Code funktioniert. Wir überwachen die Anzahl der Fehler, die unsere Services nach einem kanarischen Einsatz verursachen. Wenn die Fehlerrate steigt, wird die Änderung automatisch zurückgesetzt. Beispielsweise können wir auf 3.000 positive Datenpunkte ohne negative Datenpunkte warten, bevor wir mit der Bereitstellung fortfahren.

Eine Komplikation kann entstehen, wenn Ihre automatisierten Tests einen Anwendungsfall verpassen. Wir bemühen uns, alle Fehler mit unseren strukturierten und wiederholbaren Tests, ob automatisiert oder manuell, abzufangen. Aber auch wenn wir uns anstrengen, kann ein Defekt durchrutschen. Um unsere Tests zu testen, lassen wir die neue Änderung in der Produktion für einen festen Zeitraum, um zu sehen, ob ein Nicht-Teammitglied ein Problem findet. Wir haben viel Zeit damit verbracht, zu überlegen, ob wir Änderungen nur in der Produktion lassen oder wie lange wir nach einer kanarischen Bereitstellung warten sollen, bevor wir sie für den Rest der Bereitstellungsgruppe bereitstellen. Viele unserer Teams haben beschlossen, zusätzlich zur Erfassung positiver Datenpunkte eine bestimmte Zeitspanne abzuwarten, bevor sie unsere Bereitstellungsroutine durchlaufen. Die Wartezeit einer Pipeline hängt stark vom Team ab. Einige Teams warten Stunden, andere Minuten. Je höher die Auswirkung und je länger die Zeit zur Behebung des Problems ist, desto langsamer ist der Freigabeprozess.

Nachdem wir das Vertrauen in die erste Zelle gewonnen haben, werden wir den neuen Code mehr und mehr Kunden zur Verfügung stellen, bis er vollständig freigegeben ist. Genau wie beim kanarischen Einsatz warten wir, bis wir das Vertrauen in den Einsatz in der ersten neuen Zelle haben, bevor wir mit der nächsten Zelle fortfahren. Je mehr Vertrauen wir in das Build-Artefakt gewinnen, desto weniger Zeit wird für die Überprüfung der Codeänderung aufgewendet. Dies führt zu einem Muster, bei dem wir versuchen, so schnell wie möglich vom Check-in zu unserem ersten Produktionskunden zu gelangen. Nachdem wir jedoch in der Produktion sind, geben wir den neuen Code langsam an die Kunden weiter, um zusätzliches Vertrauen zu gewinnen, während wir den Rest unserer Bereitstellungen schrittweise beschleunigen.

Um sicherzustellen, dass unsere Produktionssysteme weiterhin Kundenanforderungen erfüllen, generieren wir synthetischen Datenverkehr auf unseren Systemen. Wir möchten schnelles Feedback, wenn unser Service nicht ordnungsgemäß funktioniert. Daher führen wir unsere synthetischen Tests mindestens jede Minute durch. Wir entwerfen synthetische Tests, um sicherzustellen, dass unsere laufenden Prozesse fehlerfrei sind und alle Abhängigkeiten getestet werden. Dazu gehört häufig das Testen aller öffentlich zugänglichen APIs.

Kontrollieren, wann Software veröffentlicht wird: Um die Sicherheit unserer Softwareversionen zu kontrollieren, haben wir Mechanismen entwickelt, mit denen wir die Geschwindigkeit steuern können, mit der sich Änderungen durch unsere Pipeline bewegen. Wir verwenden Metriken, Zeitfenster und Sicherheitsprüfungen, um zu steuern, wann unsere Software freigegeben wird.

Pipelines können konfiguriert werden, um eine Bereitstellung zu verhindern, wenn ein Alarm aufgrund einer Änderung der Metriken ausgelöst wird. Wir verwenden Metriken, und wir haben Alarme bezüglich des Zustands unserer Systeme, des Zustands unserer Zellen, der Availability Zones und Regionen und fast alles, was Ihnen sonst noch einfällt. Wir konfigurieren unsere Pipelines so, dass die Bereitstellung von Code gestoppt wird, wenn eine wichtige Metrik einen Alarm ausgelöst hat. Manchmal muss ein Team jedoch einen Fix bereitstellen, damit der Alarm des Systems behoben wird. In diesem Szenario können Teams die Alarme außer Kraft setzen, um zu verhindern, dass Änderungen über eine Pipeline übertragen werden.

Unsere Pipelines können ein Zeitfenster festlegen, in dem Änderungen in einer Pipeline vorgenommen werden können. Teams können ihre eigenen Zeitfenster finden, um einzuschränken, wann Änderungen Kunden erreichen. AWS-Teams ziehen es vor, Software freizugeben, wenn es viele Leute gibt, die schnell auf ein Problem reagieren und es abschwächen können, das durch eine Bereitstellung verursacht wird. Um dies zu verwirklichen, legen die Teams ihre Zeitfenster im Allgemeinen so fest, dass sie nur während der Geschäftszeiten bereitgestellt werden. Andere Teams bei Amazon ziehen es vor, Software freizugeben, wenn der Kundenverkehr gering ist. Diese Zeitfenster können bei Bedarf überschrieben werden.

Wir haben auch die Möglichkeit, eine Pipeline basierend auf dem Inhalt des Build-Artefakts anzuhalten. Beispielsweise können wir ein Build-Artefakt blockieren, das ein bekanntermaßen fehlerhaftes Paket oder eine bestimmte Git-Referenz enthält. Wir haben diese Funktion verwendet, als wir festgestellt haben, dass eine Änderung an einem Paket eine Leistungsregression enthält. Wenn wir das Paket nur aus unserem Paket-Repository entfernen würden, würden Pipelines, die das fehlerhafte Paket bereits enthalten, diese fehlerhafte Änderung weiterhin für Kunden bereitstellen.

Wie wir uns der Geschwindigkeit unserer Ausführung zugewandt haben

Wir haben festgestellt, dass die Teams die Automatisierung sehr zu schätzen wissen. Wir sind alle hochmotiviert, Funktionen für Kunden zu entwickeln und freizugeben, die ihr Leben verbessern, und eine kontinuierliche Lieferung macht dies nachhaltig. Wir haben gesehen, dass die Automatisierung den Ingenieuren Zeit zurückgibt, indem sie frustrierende, fehleranfällige und mühsame manuelle Arbeit beseitigt. Wir haben gezeigt, dass sich eine kontinuierliche Bereitstellung positiv auf die Qualität auswirkt. Wir haben festgestellt, dass Teams durch die Automatisierung häufig Änderungen vornehmen können, um Regressionen leichter identifizieren zu können.

Wenn Systeme neu sind, ist die zu testende Oberfläche für die meisten Teammitglieder verständlich, was einige manuelle Tests nachvollziehbar macht. Wenn jedoch die Systeme komplexer werden und sich die Teammitglieder ändern, steigt der Wert der Automatisierung. Wir automatisieren gerne unsere Systeme, damit wir uns auf die Steigerung des Kundennutzens konzentrieren können, anstatt den Prozess der Herausgabe dieser Änderungen an die Kunden von Hand zu steuern.

Seit vielen Jahren führt Amazon kontinuierliche Verbesserungsprogramme durch, die sich auf die Geschwindigkeit, mit der wir Software für Kunden veröffentlichen, und die Sicherheit dieser Releases konzentrieren. Wir haben nicht mit allen Risikoprüfungen und Tests begonnen, über die ich in diesem Artikel geschrieben habe. Im Laufe der Zeit haben wir Wege gefunden, um Risiken zu identifizieren und zu mindern.

Die Programme zur kontinuierlichen Verbesserung werden von Führungskräften auf verschiedenen Ebenen der Organisation durchgeführt. Auf diese Weise kann jeder Unternehmensleiter seinen Software-Release-Prozess an die Risiken und Auswirkungen auf sein Unternehmen anpassen. Einige unserer kontinuierlichen Verbesserungsprogramme werden in großen Teilen von Amazon ausgeführt, und manchmal führen Führungskräfte kleinerer Organisationen ihre eigenen Programme aus. Wir wissen, dass es immer Ausnahmen von der Regel gibt. Unsere Systeme verfügen über Opt-out-Mechanismen, damit Teams, die eine dauerhafte oder vorübergehende Ausnahme benötigen, nicht verlangsamt werden. Letztendlich besitzen unsere Teams das Verhalten ihrer Software und sind dafür verantwortlich, angemessen in ihren Software-Release-Prozess zu investieren.

Wir begannen damit, zu ermitteln, wo unser Schmerz war, ihn anzugehen und zu iterieren. Um diese Arbeit nachhaltig zu gestalten, mussten wir sie schrittweise durchführen und die Verbesserungen im Laufe der Zeit feiern. Als Amazon zum ersten Mal Pipelines einsetzte, waren sich viele Teams nicht sicher, ob eine kontinuierliche Bereitstellung funktionieren würde. Um die Teams zum Laufen zu bringen, haben wir sie ermutigt, ihren aktuellen Veröffentlichungsprozess, manuelle Schritte und alles in einer Pipeline zu kodieren. Für viele Teams fungierte ihre Pipeline als visuelle Schnittstelle zu ihrem Veröffentlichungsprozess, ohne dass Build-Artefakte durch einen Veröffentlichungsprozess automatisch gefördert wurden. Mit zunehmendem Vertrauen schalteten sie die Automatisierung in verschiedenen Phasen ihrer Pipeline schrittweise ein, bis sie keinen Schritt ihrer Pipeline mehr manuell auslösen mussten.

Schneller Vorlauf bis heute. Amazon ist jetzt an dem Punkt angelangt, an dem Teams beim Schreiben von neuem Code eine vollständige Automatisierung anstreben. Automatisierung ist für uns die einzige Möglichkeit, unser Geschäft weiter auszubauen.