Auswahl einer Git-Branching-Strategie für Umgebungen mit mehreren Konten DevOps

AWS Präskriptive Leitlinien

AWS Präskriptive Leitlinien: Auswahl einer Git-Branching-Strategie für Umgebungen mit mehreren Konten DevOps

Table of Contents

Auswahl einer Git-Branching-Strategie für Umgebungen mit mehreren Konten DevOps

Amazon Web Services (Mitwirkende)

Februar 2024 (Verlauf der Dokumente)

Die Umstellung auf einen cloudbasierten Ansatz und die Bereitstellung von Softwarelösungen AWS kann transformativ sein. Möglicherweise sind Änderungen an Ihrem Lebenszyklusprozess für die Softwareentwicklung erforderlich. In der Regel AWS-Konten werden während des Entwicklungsprozesses mehrere verwendet AWS Cloud. Die Wahl einer kompatiblen Git-Branching-Strategie zur Kombination mit Ihren DevOps Prozessen ist entscheidend für den Erfolg. Wenn Sie die richtige Git-Branching-Strategie für Ihr Unternehmen auswählen, können Sie DevOps Standards und Best Practices zwischen den Entwicklungsteams präzise kommunizieren. Git-Branching kann in einer einzigen Umgebung einfach sein, aber es kann verwirrend werden, wenn es auf mehrere Umgebungen angewendet wird, z. B. in Sandbox-, Entwicklungs-, Test-, Staging- und Produktionsumgebungen. Mehrere Umgebungen erhöhen die Komplexität der Implementierung. DevOps

Dieser Leitfaden enthält visuelle Diagramme von Git-Branching-Strategien, die zeigen, wie ein Unternehmen einen DevOps Multi-Account-Prozess implementieren kann. Visuelle Anleitungen helfen Teams zu verstehen, wie sie ihre Git-Branching-Strategien mit ihren DevOps Praktiken verbinden können. Die Verwendung eines Standard-Branching-Modells wie Gitflow, GitHub Flow oder Trunk für die Verwaltung des Quellcode-Repositorys hilft Entwicklungsteams, ihre Arbeit besser aufeinander abzustimmen. Diese Teams können auch Standard-Git-Schulungsressourcen im Internet nutzen, um diese Modelle und Strategien zu verstehen und umzusetzen.

DevOps Bewährte Verfahren dazu AWS finden Sie in den DevOpsLeitlinien in AWS Well-Architected. Gehen Sie bei der Lektüre dieses Leitfadens sorgfältig vor, um die richtige Filialstrategie für Ihr Unternehmen auszuwählen. Einige Strategien passen möglicherweise besser zu Ihrem Anwendungsfall als andere.

Ziele

Dieser Leitfaden ist Teil einer Dokumentationsreihe über die Auswahl und Implementierung von DevOps Branching-Strategien für Unternehmen mit mehreren AWS-Konten. Diese Reihe soll Ihnen

helfen, von Anfang an die Strategie anzuwenden, die Ihren Anforderungen, Zielen und bewährten Verfahren am besten entspricht, um Ihre Erfahrung in der AWS Cloud Dieses Handbuch enthält keine DevOps ausführbaren Skripts, da sie je nach der CI/CD-Engine (Continuous Integration and Continuous Delivery) und den Technologie-Frameworks, die Ihr Unternehmen verwendet, variieren.

In diesem Leitfaden werden die Unterschiede zwischen drei gängigen Git-Branching-Strategien erklärt: GitHub Flow, Gitflow und Trunk. Die Empfehlungen in diesem Leitfaden helfen Teams dabei, eine Branching-Strategie zu finden, die ihren Unternehmenszielen entspricht. Nachdem Sie diesen Leitfaden gelesen haben, sollten Sie in der Lage sein, eine Branching-Strategie für Ihr Unternehmen auszuwählen. Nachdem Sie sich für eine Strategie entschieden haben, können Sie eines der folgenden Muster verwenden, um diese Strategie in Ihren Entwicklungsteams umzusetzen:

- Implementieren Sie eine Trunk-Branching-Strategie für Umgebungen mit mehreren Konten DevOps
- Implementieren Sie eine GitHub Flow-Branching-Strategie für Umgebungen mit mehreren Konten DevOps
- Implementieren Sie eine Gitflow-Branching-Strategie für Umgebungen mit mehreren Konten DevOps

Es ist wichtig zu beachten, dass das, was für eine Organisation, ein Team oder ein Projekt funktioniert, möglicherweise nicht für andere geeignet ist. Die Wahl zwischen Git-Branching-Strategien hängt von verschiedenen Faktoren ab, wie der Teamgröße, den Projektanforderungen und dem gewünschten Gleichgewicht zwischen Zusammenarbeit, Integrationshäufigkeit und Release-Management.

Verwendung von CI/CD-Praktiken

AWS empfiehlt die Implementierung von Continuous Integration and Continuous Delivery (CI/CD). Dabei handelt es sich um den Prozess der Automatisierung des Lebenszyklus von Softwareversionen. Es automatisiert einen Großteil oder alle manuellen DevOps Prozesse, die traditionell erforderlich sind, um neuen Code von der Entwicklung in die Produktion zu bringen. Eine CI/CD-Pipeline umfasst die Sandbox-, Entwicklungs-, Test-, Staging- und Produktionsumgebungen. In jeder Umgebung stellt die CI/CD-Pipeline die gesamte Infrastruktur bereit, die zum Bereitstellen oder Testen des Codes erforderlich ist. Mithilfe von CI/CD können Entwicklungsteams Änderungen am Code vornehmen, der dann automatisch getestet und bereitgestellt wird. CI/CD-Pipelines bieten den Entwicklungsteams auch Steuerung und Leitplanken. Sie sorgen für Konsistenz, Standards, bewährte Verfahren und Mindestakzeptanzniveaus für die Akzeptanz und Bereitstellung von

Funktionen. Weitere Informationen finden Sie unter Practicing Continuous Integration and Continuous Delivery auf AWS.

Alle in diesem Leitfaden erörterten Branching-Strategien eignen sich gut für CI/CD-Praktiken. Die Komplexität der CI/CD-Pipeline nimmt mit der Komplexität der Verzweigungsstrategie zu. Gitflow ist beispielsweise die komplexeste Branching-Strategie, die in diesem Leitfaden erörtert wird. CI/CD-Pipelines für diese Strategie erfordern mehr Schritte (z. B. aus Compliance-Gründen) und müssen mehrere gleichzeitige Produktionsversionen unterstützen. Die Verwendung von CI/CD wird mit zunehmender Komplexität der Branching-Strategie ebenfalls immer wichtiger. Dies liegt daran, dass CI/CD Leitplanken und Mechanismen für Entwicklungsteams festlegt, die verhindern, dass Entwickler den definierten Prozess absichtlich oder unabsichtlich umgehen.

AWS bietet eine Reihe von Entwicklerservices, die Sie beim Aufbau von CI/CD-Pipelines unterstützen sollen. AWS CodePipelineIst beispielsweise ein vollständig verwalteter Continuous Delivery Service, der Sie bei der Automatisierung Ihrer Release-Pipelines für schnelle und zuverlässige Anwendungs- und Infrastrukturupdates unterstützt. AWS CodeCommitwurde entwickelt, um skalierbare Git-Repositorys sicher zu hosten und AWS CodeBuildkompiliert Quellcode, führt Tests durch und produziert ready-to-deploy Softwarepakete. Weitere Informationen finden Sie unter Entwicklertools unter. AWS

Die DevOps Umgebungen verstehen

Um die Branching-Strategien zu verstehen, müssen Sie den Zweck und die Aktivitäten verstehen, die in jeder Umgebung stattfinden. Durch die Einrichtung mehrerer Umgebungen können Sie die Entwicklungsaktivitäten in Phasen unterteilen, diese Aktivitäten überwachen und die unbeabsichtigte Veröffentlichung nicht genehmigter Funktionen verhindern. In jeder Umgebung können Sie eine oder mehrere AWS-Konten haben.

Die meisten Organisationen verfügen über mehrere Umgebungen, die für den Einsatz vorgesehen sind. Die Anzahl der Umgebungen kann jedoch je nach Organisation und Softwareentwicklungsrichtlinien variieren. In dieser Dokumentationsserie wird davon ausgegangen, dass Sie über die folgenden fünf gängigen Umgebungen verfügen, die sich über Ihre Entwicklungspipeline erstrecken, obwohl sie möglicherweise unterschiedliche Namen haben:

- Sandbox — Eine Umgebung, in der Entwickler Code schreiben, Fehler machen und Machbarkeitsstudien durchführen.
- Entwicklung — Eine Umgebung, in der Entwickler ihren Code integrieren, um sicherzustellen, dass alles als eine einzige, zusammenhängende Anwendung funktioniert.
- Testen — Eine Umgebung, in der QA-Teams oder Abnahmetests stattfinden. Teams führen in dieser Umgebung häufig Leistungs- oder Integrationstests durch.
- Staging — Eine Vorproduktionsumgebung, in der Sie überprüfen, ob der Code und die Infrastruktur unter produktionsäquivalenten Umständen erwartungsgemäß funktionieren. Diese Umgebung ist so konfiguriert, dass sie der Produktionsumgebung so ähnlich wie möglich ist.
- Produktion — Eine Umgebung, die den Datenverkehr Ihrer Endbenutzer und Kunden verarbeitet.

In diesem Abschnitt werden die einzelnen Umgebungen detailliert beschrieben. Außerdem werden die Build-Schritte, Bereitstellungsschritte und Exit-Kriterien für jede Umgebung beschrieben, sodass Sie mit der nächsten fortfahren können. Die folgende Abbildung zeigt diese Umgebungen nacheinander.

Themen in diesem Abschnitt:

Sandbox-Umgebung

In der Sandbox-Umgebung schreiben Entwickler Code, machen Fehler und führen Machbarkeitsstudien durch. Sie können die Bereitstellung in einer Sandbox-Umgebung von einer lokalen Arbeitsstation aus oder über ein Skript auf einer lokalen Arbeitsstation durchführen.

Zugriff

Entwickler sollten vollen Zugriff auf die Sandbox-Umgebung haben.

Schritte erstellen

Entwickler führen den Build manuell auf ihren lokalen Workstations aus, wenn sie bereit sind, Änderungen an der Sandbox-Umgebung vorzunehmen.

1. Verwenden Sie git-secrets (GitHub), um nach vertraulichen Informationen zu suchen
2. Lint den Quellcode
3. Erstellen und kompilieren Sie den Quellcode, falls zutreffend
4. Führen Sie Komponententests durch
5. Führen Sie eine Analyse der Codeabdeckung durch
6. Eine statische Codeanalyse durchführen
7. Erstellen Sie Infrastruktur als Code (IaC)
8. Führen Sie eine IaC-Sicherheitsanalyse durch
9. Extrahieren Sie Open-Source-Lizenzen
10. Veröffentlichen Sie Build-Artefakte

Schritte zur Bereitstellung

Wenn Sie die Modelle Gitflow oder Trunk verwenden, werden die Bereitstellungsschritte automatisch eingeleitet, wenn ein `feature` Branch erfolgreich in der Sandbox-Umgebung erstellt wurde. Wenn Sie das GitHub Flow-Modell verwenden, führen Sie die folgenden Bereitstellungsschritte manuell aus. Im Folgenden sind die Bereitstellungsschritte in der Sandbox-Umgebung aufgeführt:

1. Laden Sie veröffentlichte Artefakte herunter
2. Führen Sie die Datenbank-Versionierung durch
3. Führen Sie die IaC-Bereitstellung durch
4. Führen Sie Integrationstests durch

Erwartungen vor dem Übergang zur Entwicklungsumgebung

- Erfolgreicher Aufbau der `feature` Filiale in der Sandbox-Umgebung
- Ein Entwickler hat die Funktion manuell in der Sandbox-Umgebung bereitgestellt und getestet

Entwicklungsumgebung

In der Entwicklungsumgebung integrieren Entwickler ihren Code zusammen, um sicherzustellen, dass alles als eine zusammenhängende Anwendung funktioniert. In Gitflow enthält die Entwicklungsumgebung die neuesten Funktionen, die in der Merge-Anfrage enthalten sind und zur Veröffentlichung bereit sind. In GitHub Flow- und Trunk-Strategien wird die Entwicklungsumgebung als Testumgebung betrachtet, und die Codebasis ist möglicherweise instabil und für die Bereitstellung in der Produktion ungeeignet.

Zugriff

Weisen Sie Berechtigungen nach dem Prinzip der geringsten Rechte zu. Geringste Berechtigung ist die bewährte Methode, die für die Ausführung einer Aufgabe erforderlichen Mindestberechtigungen zu gewähren. Entwickler sollten weniger Zugriff auf die Entwicklungsumgebung als auf die Sandbox-Umgebung haben.

Schritte erstellen

Wenn Sie eine Merge-Anfrage für den `develop` Branch (Gitflow) oder den `main` Branch (Trunk oder GitHub Flow) erstellen, wird der Build automatisch gestartet.

1. Verwenden Sie git-secrets (GitHub), um nach vertraulichen Informationen zu suchen
2. Lint den Quellcode
3. Erstellen und kompilieren Sie den Quellcode, falls zutreffend
4. Führen Sie Komponententests durch
5. Führen Sie eine Analyse der Codeabdeckung durch
6. Eine statische Codeanalyse durchführen
7. IaC erstellen
8. Führen Sie eine IaC-Sicherheitsanalyse durch
9. Extrahieren Sie Open-Source-Lizenzen

Schritte zur Bereitstellung

Wenn Sie das Gitflow-Modell verwenden, werden die Bereitstellungsschritte automatisch eingeleitet, wenn ein `develop` Branch erfolgreich in der Entwicklungsumgebung erstellt wurde. Wenn du das GitHub Flow-Modell oder das Trunk-Modell verwendest, werden die Bereitstellungsschritte automatisch eingeleitet, wenn eine Merge-Anfrage für den `main` Branch erstellt wird. Im Folgenden sind die Bereitstellungsschritte in der Entwicklungsumgebung aufgeführt:

1. Laden Sie die veröffentlichten Artefakte aus den Build-Schritten herunter
2. Führen Sie die Datenbank-Versionierung durch
3. Führen Sie die IaC-Bereitstellung durch
4. Führen Sie Integrationstests durch

Erwartungen vor dem Übergang zur Testumgebung

- Erfolgreicher Aufbau und Bereitstellung des `develop` Branches (Gitflow) oder des `main` Branches (Trunk oder GitHub Flow) in der Entwicklungsumgebung
- Der Komponententest besteht zu 100%

- Erfolgreicher IaC-Build
- Bereitstellungsartefakte wurden erfolgreich erstellt
- Ein Entwickler hat eine manuelle Überprüfung durchgeführt, um sicherzustellen, dass die Funktion wie erwartet funktioniert

Testumgebung

Mitarbeiter der Qualitätssicherung (QA) verwenden die Testumgebung, um Funktionen zu validieren. Sie genehmigen die Änderungen, nachdem sie die Tests abgeschlossen haben. Nach der Genehmigung geht die Filiale zur nächsten Umgebung über, dem Staging. In Gitflow sind diese und andere Umgebungen darüber nur für die Bereitstellung in Branches verfügbar. `release` Ein `release` Branch basiert auf einem `develop` Branch, der die geplanten Funktionen enthält.

Zugriff

Weisen Sie Berechtigungen nach dem Prinzip der geringsten Rechte zu. Entwickler sollten weniger Zugriff auf die Testumgebung als auf die Entwicklungsumgebung haben. QA-Mitarbeiter benötigen ausreichende Berechtigungen, um die Funktion zu testen.

Schritte erstellen

Der Build-Prozess in dieser Umgebung gilt nur für Bugfixes, wenn die Gitflow-Strategie verwendet wird. Wenn Sie eine Merge-Anfrage für den `bugfix` Branch erstellen, wird der Build automatisch gestartet.

1. Verwenden Sie git-secrets (GitHub), um nach vertraulichen Informationen zu suchen
2. Lint den Quellcode
3. Erstellen und kompilieren Sie den Quellcode, falls zutreffend
4. Führen Sie Komponententests durch
5. Führen Sie eine Analyse der Codeabdeckung durch
6. Eine statische Codeanalyse durchführen
7. IaC erstellen
8. Führen Sie eine IaC-Sicherheitsanalyse durch
9. Extrahieren Sie Open-Source-Lizenzen

Schritte zur Bereitstellung

Initiieren Sie nach der `release` Bereitstellung in der Entwicklungsumgebung automatisch die Bereitstellung des `main` Branches (GitHub Gitflow) oder des Branches (Trunk oder Flow) in der Testumgebung. Im Folgenden sind die Bereitstellungsschritte in der Testumgebung aufgeführt:

1. Stellen Sie den `release` Branch (Gitflow) oder `main` Branch (Trunk oder GitHub Flow) in der Testumgebung bereit
2. Machen Sie eine Pause für die manuelle Genehmigung durch das dafür vorgesehene Personal
3. Laden Sie veröffentlichte Artefakte herunter
4. Führen Sie die Datenbank-Versionierung durch
5. Führen Sie die IaC-Bereitstellung durch
6. Führen Sie Integrationstests durch
7. Führen Sie Leistungstests durch
8. Zulassung zur Qualitätssicherung

Erwartungen vor dem Übergang zur Staging-Umgebung

- Die Entwicklungs- und QA-Teams haben ausreichend Tests durchgeführt, um die Anforderungen Ihres Unternehmens zu erfüllen.
- Das Entwicklungsteam hat alle entdeckten Fehler über eine `bugfix` Filiale behoben.

Staging-Umgebung

Die Staging-Umgebung ist so konfiguriert, dass sie der Produktionsumgebung entspricht. Beispielsweise sollte das Daten-Setup in Umfang und Größe den Produktionsworkloads ähneln. Verwenden Sie die Staging-Umgebung, um zu überprüfen, ob Code und Infrastruktur erwartungsgemäß funktionieren. Diese Umgebung ist auch die bevorzugte Wahl für geschäftliche Anwendungsfälle wie Vorschauen oder Kundenvorführungen.

Zugriff

Weisen Sie Berechtigungen nach dem Prinzip der geringsten Rechte zu. Entwickler sollten denselben Zugriff auf die Staging-Umgebung haben wie auf die Produktionsumgebung.

Schritte erstellen

Keine. Dieselben Artefakte, die in der Testumgebung verwendet wurden, werden in der Staging-Umgebung wiederverwendet.

Schritte zur Bereitstellung

Initiieren Sie nach der `release` Genehmigung und Bereitstellung in der Testumgebung automatisch die Bereitstellung des `main` Branches (GitHub Gitflow) oder des Branches (Trunk oder Flow) in der Staging-Umgebung. Im Folgenden sind die Bereitstellungsschritte in der Staging-Umgebung aufgeführt:

1. Stellen Sie den `release` Branch (Gitflow) oder `main` Branch (Trunk oder GitHub Flow) in der Staging-Umgebung bereit
2. Machen Sie eine Pause für die manuelle Genehmigung durch das dafür vorgesehene Personal
3. Laden Sie veröffentlichte Artefakte herunter
4. Führen Sie die Datenbank-Versionierung durch
5. Führen Sie die IaC-Bereitstellung durch
6. (Optional) Führen Sie Integrationstests durch
7. (Optional) Führen Sie Belastungstests durch
8. Holen Sie sich die Genehmigung der für die Entwicklung, Qualitätssicherung, Produkte oder Geschäftsbereiche zuständigen Stelle ein

Erwartungen vor dem Übergang zur Produktionsumgebung

- Eine produktionsäquivalente Version wurde erfolgreich in der Staging-Umgebung bereitgestellt
- (Optional) Integration und Belastungstests waren erfolgreich

Produktionsumgebung

Die Produktionsumgebung unterstützt das veröffentlichte Produkt und verarbeitet echte Daten von echten Kunden. Dabei handelt es sich um eine geschützte Umgebung, der der Zugriff nach den geringsten Rechten zugewiesen wird, und der Zugriff mit erhöhten Rechten nur im Rahmen eines geprüften Ausnahmeprozesses für einen begrenzten Zeitraum gewährt werden sollte.

Zugriff

In der Produktionsumgebung sollten Entwickler eingeschränkten Lesezugriff auf die AWS-Managementkonsole haben. Entwickler sollten beispielsweise auf Protokolldaten für day-to-day Operationen zugreifen können. Alle Produktionsversionen sollten vor der Bereitstellung einem Genehmigungsschritt unterliegen.

Schritte erstellen

Keine. Dieselben Artefakte, die in den Test- und Staging-Umgebungen verwendet wurden, werden in der Produktionsumgebung wiederverwendet.

Schritte zur Bereitstellung

Initiieren Sie nach der Genehmigung und Bereitstellung in der Staging-Umgebung automatisch die Bereitstellung der `release` `main` Filiale (GitHub Gitflow) oder der Filiale (Trunk oder Flow) in der Produktionsumgebung. Im Folgenden sind die Bereitstellungsschritte in der Produktionsumgebung aufgeführt:

1. Stellen Sie den `release` Branch (Gitflow) oder `main` Branch (Trunk oder GitHub Flow) in der Produktionsumgebung bereit
2. Machen Sie eine Pause für die manuelle Genehmigung durch das dafür vorgesehene Personal
3. Laden Sie veröffentlichte Artefakte herunter
4. Führen Sie die Datenbank-Versionierung durch
5. Führen Sie die IaC-Bereitstellung durch

Bewährte Methoden für die Git-basierte Entwicklung

Um die Git-basierte Entwicklung erfolgreich einzuführen, ist es wichtig, eine Reihe von Best Practices zu befolgen, die die Zusammenarbeit fördern, die Codequalität aufrechterhalten und Continuous Integration und Continuous Delivery (CI/CD) unterstützen. Lesen Sie zusätzlich zu den bewährten Methoden in diesem Leitfaden auch die AWS Well-Architected DevOps Guidance. Im Folgenden finden Sie einige der wichtigsten Best Practices für die Git-basierte Entwicklung auf AWS:

- Halten Sie die Änderungen klein und häufig — Ermutigen Sie Entwickler, kleine, schrittweise Änderungen oder Funktionen vorzunehmen. Dies reduziert das Risiko von Zusammenführungskonflikten und erleichtert die schnelle Identifizierung und Behebung von Problemen.
- Funktionsumschalter verwenden — Verwenden Sie Funktionsumschalter oder Feature-Flags, um die Veröffentlichung unvollständiger oder experimenteller Funktionen zu verwalten. Auf diese Weise können Sie bestimmte Funktionen in der Produktion ausblenden, aktivieren oder deaktivieren, ohne die Stabilität des Hauptzweigs zu beeinträchtigen.
- Sorgen Sie für eine robuste Testsuite — Eine umfassende, gut gepflegte Testsuite ist entscheidend, um Probleme frühzeitig zu erkennen und zu überprüfen, ob die Codebasis stabil bleibt. Investieren Sie in die Testautomatisierung und priorisieren Sie die Behebung fehlgeschlagener Tests.
- Setzen Sie auf kontinuierliche Integration — Verwenden Sie Tools und Methoden für die kontinuierliche Integration, um Codeänderungen automatisch zu erstellen, zu testen und in den `develop` Branch (Gitflow) oder `main` Branch (Trunk oder GitHub Flow) zu integrieren. Dies hilft Ihnen, Probleme frühzeitig zu erkennen und den Entwicklungsprozess zu optimieren.
- Führen Sie Code-Reviews durch — Fördern Sie Peer-Reviews von Code, um die Qualität aufrechtzuerhalten, Wissen catch und potenzielle Probleme zu erkennen, bevor sie in die `main` Branche integriert werden. Verwenden Sie Pull-Requests oder andere Code-Review-Tools, um diesen Prozess zu vereinfachen.
- Überwachen und reparieren Sie fehlerhafte Builds — Wenn ein Build kaputt geht oder Tests fehlschlagen, sollten Sie der schnellstmöglichen Behebung des Problems Priorität einräumen. Dadurch bleibt der `develop` Branch (Gitflow) oder `main` Branch (Trunk oder GitHub Flow) in einem Zustand, der veröffentlicht werden kann, und die Auswirkungen auf andere Entwickler werden minimiert.

- Kommunizieren und zusammenarbeiten — Fördern Sie die offene Kommunikation und Zusammenarbeit zwischen den Teammitgliedern. Stellen Sie sicher, dass Entwickler über die laufenden Arbeiten und Änderungen an der Codebasis informiert sind.
- Kontinuierliches Refaktorieren — Refaktorieren Sie die Codebasis regelmäßig, um ihre Wartbarkeit zu verbessern und technische Schulden zu reduzieren. Ermutigen Sie Entwickler, den Code in einem besseren Zustand zu belassen, als sie ihn vorgefunden haben.
- Verwenden Sie kurzlebige Branches für komplexe Aufgaben — Verwenden Sie für größere oder komplexere Aufgaben kurzlebige Branches (auch als Task-Branches bezeichnet), um an den Änderungen zu arbeiten. Achten Sie jedoch darauf, die Lebensdauer der Filialen kurz zu halten, in der Regel weniger als einen Tag. Führen Sie die Änderungen so schnell wie möglich wieder in den `develop` Branch (Gitflow) oder `main` Branch (Trunk oder GitHub Flow) ein. Kleinere und häufigere Zusammenführungen und Überprüfungen sind für ein Team einfacher zu verarbeiten und zu verarbeiten als eine große Zusammenführungsanfrage.
- Schulung und Unterstützung des Teams — Bieten Sie Schulungen und Support für Entwickler an, die noch keine Erfahrung mit der Git-basierten Entwicklung haben oder Unterstützung bei der Übernahme ihrer Best Practices benötigen.

Strategien zur Git-Verzweigung

In dieser Anleitung werden die folgenden Git-basierten Branching-Strategien, geordnet nach der geringsten bis zur komplexesten, detailliert beschrieben:

- Trunk — Bei der Trunk-basierten Entwicklung handelt es sich um eine Softwareentwicklungspraxis, bei der alle Entwickler an einem einzigen Zweig arbeiten, der üblicherweise als Oder-Zweig bezeichnet wird. `trunk` `main` Die Idee hinter diesem Ansatz besteht darin, die Codebasis in einem Zustand zu halten, der kontinuierlich veröffentlicht werden kann, indem häufig Codeänderungen integriert und auf automatisierte Tests und kontinuierliche Integration zurückgegriffen wird.
- GitHub Flow — GitHub Flow ist ein leichter, branchenbasierter Workflow, der von entwickelt wurde. GitHub Er basiert auf der Idee kurzlebiger `feature` Filialen. Wenn ein Feature fertig und bereit für die Bereitstellung ist, wird es mit dem `main` Branch zusammengeführt.
- Gitflow — Bei einem Gitflow-Ansatz wird die Entwicklung in einzelnen Feature-Branches abgeschlossen. Nach der Genehmigung fügst du `feature` Branches zu einem Integrationszweig zusammen, der normalerweise einen Namen trägt. `develop` Wenn sich in dem `develop` Zweig genügend Funktionen angesammelt haben, wird ein `release` Zweig erstellt, um die Funktionen in höheren Umgebungen bereitzustellen.

Jede Branching-Strategie hat Vor- und Nachteile. Sie verwenden zwar alle dieselben Umgebungen, aber nicht alle verwenden dieselben Branches oder manuellen Genehmigungsschritte. Sehen Sie sich in diesem Abschnitt des Leitfadens jede Branching-Strategie im Detail an, sodass Sie mit ihren Nuancen vertraut sind und beurteilen können, ob sie für den Anwendungsfall Ihres Unternehmens geeignet ist.

Themen in diesem Abschnitt:

Strategie zur Stammverzweigung

Bei der stammbasierten Entwicklung handelt es sich um eine Softwareentwicklungspraxis, bei der alle Entwickler an einem einzigen Zweig arbeiten, der üblicherweise als Oder-Zweig bezeichnet

wird. `trunk` `main` Die Idee hinter diesem Ansatz besteht darin, die Codebasis in einem Zustand zu halten, der kontinuierlich veröffentlicht werden kann, indem häufig Codeänderungen integriert und auf automatisierte Tests und kontinuierliche Integration zurückgegriffen wird.

Bei der Trunk-basierten Entwicklung übertragen Entwickler ihre Änderungen mehrmals täglich in den `main` Branch, wobei kleine, inkrementelle Updates angestrebt werden. Dies ermöglicht schnelle Feedback-Schleifen, reduziert das Risiko von Zusammenführungskonflikten und fördert die Zusammenarbeit zwischen den Teammitgliedern. In dieser Praxis wird die Bedeutung einer gut gepflegten Testsuite betont, da sie auf automatisierten Tests basiert, um potenzielle Probleme frühzeitig zu catch und sicherzustellen, dass die Codebasis stabil und veröffentlichbar bleibt.

Trunk-basierte Entwicklung wird häufig der funktionsbasierten Entwicklung (auch bekannt als Feature-Branching oder Feature-Driven Development) gegenübergestellt, bei der jedes neue Feature oder jeder Bugfix in einem eigenen, vom Hauptzweig getrennten Zweig entwickelt wird. Die Wahl zwischen Trunk-basierter Entwicklung und funktionsbasierter Entwicklung hängt von Faktoren wie Teamgröße, Projektanforderungen und dem gewünschten Gleichgewicht zwischen Zusammenarbeit, Integrationshäufigkeit und Release-Management ab.

Weitere Informationen zur Trunk-Branching-Strategie finden Sie in den folgenden Ressourcen:

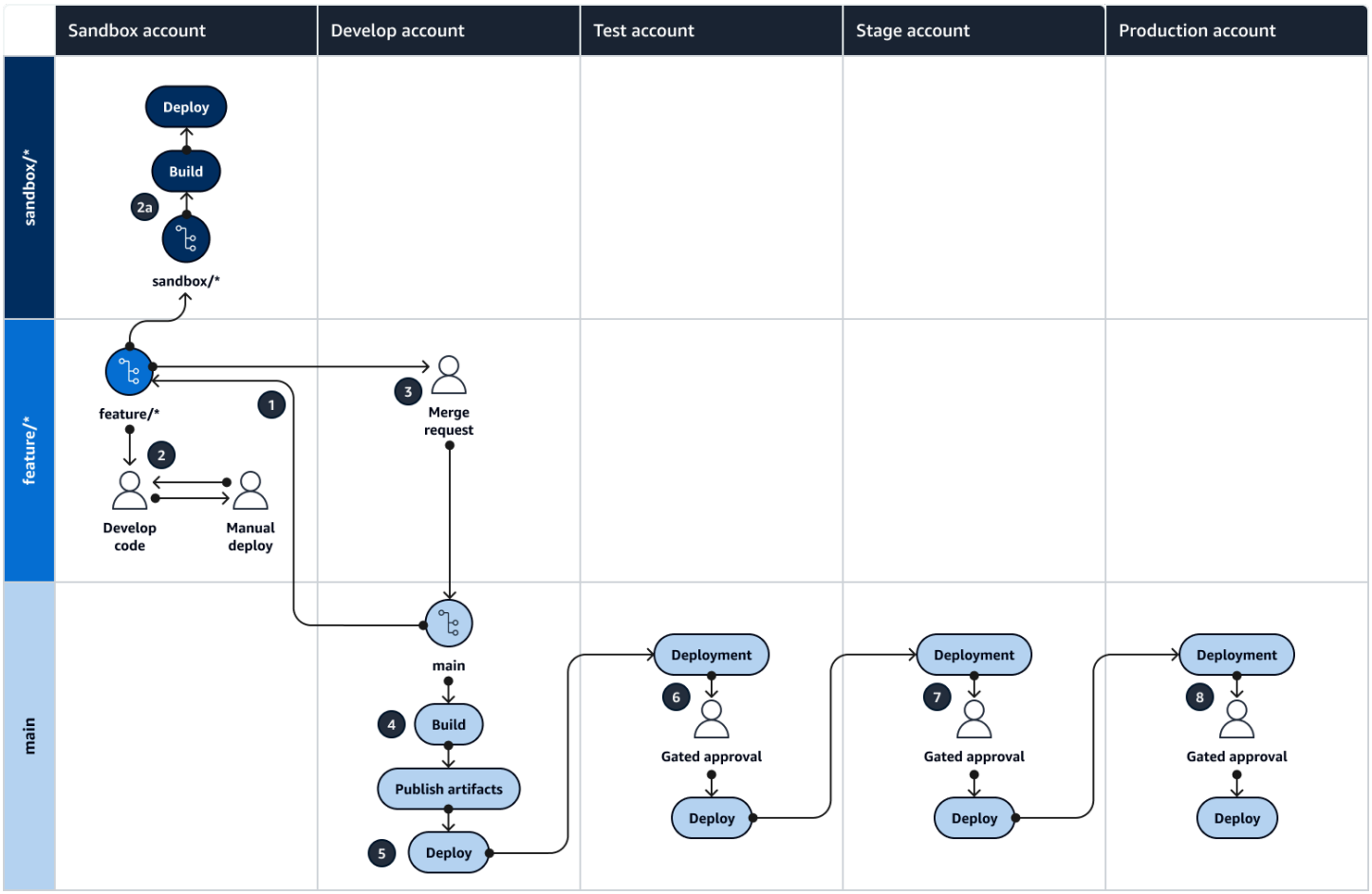
- Implementieren Sie eine Trunk-Branching-Strategie für DevOps Umgebungen mit mehreren Konten (AWS Prescriptive Guidance)
- Einführung in die Trunk-basierte Entwicklung (Website zur Trunk-basierten Entwicklung)

Themen in diesem Abschnitt:

Visueller Überblick über die Trunk-Strategie

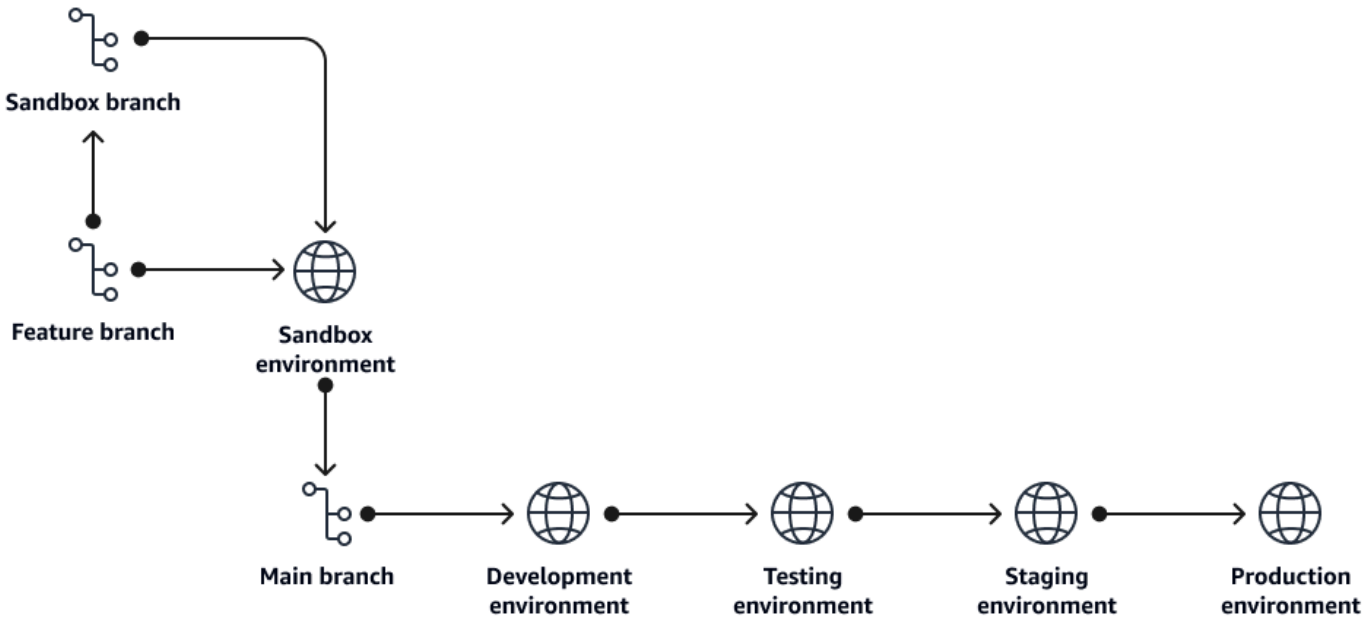
Das folgende Diagramm kann wie ein Punnett-Quadrat (Wikipedia) verwendet werden, um die Trunk-Branching-Strategie zu verstehen. Richten Sie die Zweige auf der vertikalen Achse mit den AWS Umgebungen auf der horizontalen Achse aus, um zu bestimmen, welche Aktionen in den einzelnen Szenarien ausgeführt werden sollen. Die eingekreisten Zahlen führen Sie durch die Reihenfolge der Aktionen, die im Diagramm dargestellt sind. Dieses Diagramm zeigt den Entwicklungsablauf

einer Trunk-Branching-Strategie von einem `feature` Branch in der Sandbox-Umgebung bis zur Produktionsversion des Branches. `main` Weitere Informationen zu den Aktivitäten, die in den einzelnen Umgebungen stattfinden, finden Sie in diesem DevOps Handbuch unter Umgebungen.



Filialen in einer Trunk-Strategie

Eine Trunk-Branching-Strategie umfasst üblicherweise die folgenden Verzweigungen.



Feature-Zweig

Sie entwickeln Funktionen oder erstellen einen Hotfix in einer `feature` Filiale. Um einen `feature` Zweig zu erstellen, zweigen Sie von diesem `main` Zweig ab. Entwickler iterieren, übertragen und testen Code in einem `feature` Branch. Wenn ein Feature fertig ist, bewirbt der Entwickler das Feature. Von einem `feature` Zweig aus gibt es nur zwei Pfade vorwärts:

- Mit dem `sandbox` Zweig verschmelzen
- Erstellen Sie eine Anfrage zur Zusammenführung mit der `main` Filiale

Benennungskonvention:	`feature/<story number>_<developer initials>_<descriptor>`
Beispiel für eine Namenskonvention:	`feature/123456_MS_Implement_Feature_A`

Sandbox-Zweig

Bei diesem Zweig handelt es sich um einen nicht standardmäßigen Trunk-Zweig, der jedoch für die Entwicklung von CI/CD-Pipelines nützlich ist. Der `sandbox` Zweig wird hauptsächlich für die folgenden Zwecke verwendet:

- Führen Sie mithilfe der CI/CD-Pipelines eine vollständige Bereitstellung in der Sandbox-Umgebung durch
- Entwickeln und testen Sie eine Pipeline, bevor Sie Mergeanfragen für vollständige Tests in einer niedrigeren Umgebung, z. B. Entwicklung oder Testen, einreichen.

`Sandbox`Verzweigungen sind temporärer Natur und sollen kurzlebig sein. Sie sollten nach Abschluss der spezifischen Tests gelöscht werden.

Namenskonvention:	`sandbox/<story number>_<developer initials>_<descriptor>`
Beispiel für eine Namenskonvention:	`sandbox/123456_MS_Test_Pipeline_Deploy`

Hauptzweig

Der `main` Zweig steht immer für den Code, der in der Produktion ausgeführt wird. Code wird abgezweigt`main`, entwickelt und anschließend wieder zusammengeführt. `main` Bereitstellungen von `main` können auf jede Umgebung abzielen. Um vor dem Löschen zu schützen, aktivieren Sie den Branch-Schutz für den `main` Branch.

Benennungskonvention:	`main`

Hotfix-Zweig

In einem Stamm-basierten Workflow gibt es keinen speziellen `hotfix` Branch. Hotfixes verwenden Branches. `feature`

Vor- und Nachteile der Trunk-Strategie

Die Trunk-Branching-Strategie eignet sich gut für kleinere, ausgereifte Entwicklungsteams mit ausgeprägten Kommunikationsfähigkeiten. Sie funktioniert auch gut, wenn Sie fortlaufende, fortlaufende Feature-Releases für die Anwendung haben. Es ist nicht gut geeignet, wenn Sie große oder fragmentierte Entwicklungsteams haben oder wenn Sie umfangreiche, geplante Feature-

Releases haben. Bei diesem Modell treten Zusammenführungskonflikte auf. Seien Sie sich also bewusst, dass die Lösung von Zusammenführungskonflikten eine Schlüsselkompetenz ist. Alle Teammitglieder müssen entsprechend geschult werden.

Vorteile

Die Trunk-basierte Entwicklung bietet mehrere Vorteile, die den Entwicklungsprozess verbessern, die Zusammenarbeit rationalisieren und die Gesamtqualität der Software verbessern können. Im Folgenden sind einige der wichtigsten Vorteile aufgeführt:

- Schnellere Feedback-Schleifen — Bei der Trunk-basierten Entwicklung integrieren Entwickler ihre Codeänderungen häufig, oft mehrmals täglich. Dies ermöglicht schnelleres Feedback zu potenziellen Problemen und hilft Entwicklern, Probleme schneller zu identifizieren und zu beheben, als dies bei einem funktionsbasierten Entwicklungsmodell der Fall wäre.
- Weniger Zusammenführungskonflikte — Bei der trunkbasierten Entwicklung wird das Risiko großer, komplizierter Zusammenführungskonflikte minimiert, da Änderungen kontinuierlich integriert werden. Dies trägt zur Aufrechterhaltung einer übersichtlicheren Codebasis bei und reduziert den Zeitaufwand für die Lösung von Konflikten. Das Lösen von Konflikten kann bei der funktionsbasierten Entwicklung sowohl zeitaufwändig als auch fehleranfällig sein.
- Verbesserte Zusammenarbeit — Die Trunk-basierte Entwicklung ermutigt Entwickler dazu, in derselben Branche zusammenzuarbeiten, was zu einer besseren Kommunikation und Zusammenarbeit innerhalb des Teams führt. Dies kann zu einer schnelleren Problemlösung und einer kohärenteren Teamdynamik führen.
- Einfachere Code-Reviews — Da Codeänderungen bei der Trunk-basierten Entwicklung kleiner und häufiger sind, kann es einfacher sein, gründliche Code-Reviews durchzuführen. Kleinere Änderungen sind im Allgemeinen leichter zu verstehen und zu überprüfen, was zu einer effektiveren Identifizierung potenzieller Probleme und Verbesserungen führt.
- Kontinuierliche Integration und Bereitstellung — Die Trunk-basierte Entwicklung unterstützt die Prinzipien der kontinuierlichen Integration und kontinuierlichen Bereitstellung (CI/CD). Indem die Codebasis in einem veröffentlichbaren Zustand gehalten und Änderungen häufig integriert werden, können Teams CI/CD-Praktiken leichter anwenden, was zu schnelleren Bereitstellungszyklen und verbesserter Softwarequalität führt.
- Verbesserte Codequalität — Durch häufige Integrationen, Tests und Codeüberprüfungen kann die Trunk-basierte Entwicklung zu einer insgesamt besseren Codequalität beitragen. Entwickler können Probleme schneller catch und beheben und so die Wahrscheinlichkeit verringern, dass sich im Laufe der Zeit technische Schulden ansammeln.

- Vereinfachte Branching-Strategie — Die Trunk-basierte Entwicklung vereinfacht die Branching-Strategie, indem die Anzahl langlebiger Filialen reduziert wird. Dies kann die Verwaltung und Wartung der Codebasis erleichtern, insbesondere bei großen Projekten oder Teams.

Nachteile

Die Trunk-basierte Entwicklung hat einige Nachteile, die sich auf den Entwicklungsprozess und die Teamdynamik auswirken können. Im Folgenden sind einige bemerkenswerte Nachteile aufgeführt:

- Eingeschränkte Isolierung — Da alle Entwickler in derselben Branche arbeiten, sind ihre Änderungen sofort für alle Teammitglieder sichtbar. Dies kann zu Störungen oder Konflikten führen, unbeabsichtigte Nebenwirkungen haben oder den Build beschädigen. Im Gegensatz dazu isoliert die funktionsbasierte Entwicklung Änderungen besser, sodass Entwickler unabhängiger arbeiten können.
- Erhöhter Testdruck — Die Trunk-basierte Entwicklung setzt auf kontinuierliche Integration und automatisierte Tests, um catch schnell zu erkennen. Dieser Ansatz kann jedoch großen Druck auf die Testinfrastruktur ausüben und erfordert eine gut gepflegte Testsuite. Wenn die Tests nicht umfassend oder zuverlässig sind, kann dies zu unentdeckten Problemen in der Hauptzweige führen.
- Weniger Kontrolle über Releases — Die Trunk-basierte Entwicklung zielt darauf ab, die Codebasis in einem kontinuierlich veröffentlichbaren Zustand zu halten. Dies kann zwar von Vorteil sein, eignet sich aber möglicherweise nicht immer für Projekte mit strengen Veröffentlichungszeitplänen oder für Projekte, bei denen bestimmte Funktionen gemeinsam veröffentlicht werden müssen. Die funktionsbasierte Entwicklung bietet mehr Kontrolle darüber, wann und wie Funktionen veröffentlicht werden.
- Codeabwanderung — Da Entwickler ständig Änderungen in den Hauptzweig integrieren, kann die stammbasierte Entwicklung zu einer erhöhten Codeabwanderung führen. Dies kann es Entwicklern erschweren, den Überblick über den aktuellen Stand der Codebasis zu behalten, und es kann zu Verwirrung führen, wenn versucht wird, die Auswirkungen der letzten Änderungen zu verstehen.
- Erfordert eine starke Teamkultur — Die Entwicklung auf Stammbasis erfordert ein hohes Maß an Disziplin, Kommunikation und Zusammenarbeit zwischen den Teammitgliedern. Dies kann schwierig sein, dies aufrechtzuerhalten, insbesondere in größeren Teams oder bei der Zusammenarbeit mit Entwicklern, die mit diesem Ansatz weniger Erfahrung haben.
- Herausforderungen bei der Skalierbarkeit — Mit zunehmender Größe des Entwicklungsteams kann die Anzahl der Codeänderungen, die in den Hauptzweig integriert werden, schnell zunehmen. Dies

kann zu häufigeren Build-Unterbrechungen und Testfehlern führen, was es schwierig macht, die Codebasis in einem veröffentlichbaren Zustand zu halten.

GitHub Strategie zur Flow-Branchierung

GitHub Flow ist ein leichter, branchenbasierter Workflow, der von entwickelt wurde. GitHub GitHubFlow basiert auf der Idee kurzlebiger Feature-Branches, die zum Hauptzweig zusammengeführt werden, wenn das Feature fertiggestellt und einsatzbereit ist. Die wichtigsten Prinzipien von GitHub Flow sind:

- Branching ist leichtgewichtig — Entwickler können mit nur wenigen Klicks Feature-Branches für ihre Arbeit erstellen und so die Fähigkeit zur Zusammenarbeit und zum Experimentieren verbessern, ohne dass der Hauptzweig beeinträchtigt wird.
- Kontinuierliche Bereitstellung — Änderungen werden implementiert, sobald sie in den Hauptzweig integriert sind, was schnelles Feedback und schnelle Iterationen ermöglicht.
- Anfragen zusammenführen — Entwickler verwenden Merge-Anfragen, um einen Diskussions- und Überprüfungsprozess einzuleiten, bevor sie ihre Änderungen im Hauptzweig zusammenführen.

Weitere Informationen zu GitHub Flow finden Sie in den folgenden Ressourcen:

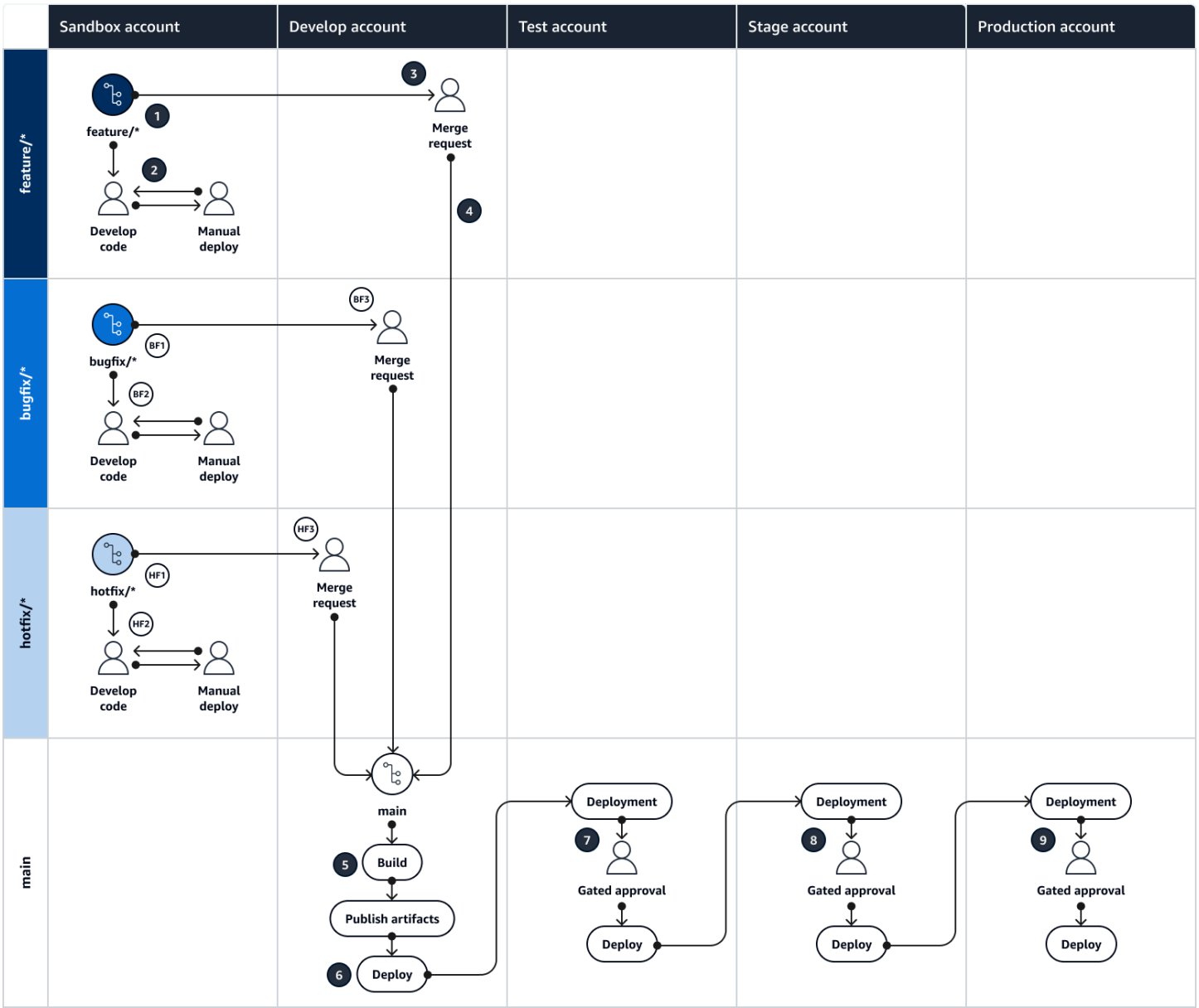
- Implementieren Sie eine GitHub Flow-Branching-Strategie für DevOps Umgebungen mit mehreren Konten (AWS Prescriptive Guidance)
- GitHub Flow Quickstart (Dokumentation) GitHub
- Warum GitHub Flow? (GitHubFlow-Webseite)

Themen in diesem Abschnitt:

Visueller Überblick über die GitHub Flow-Strategie

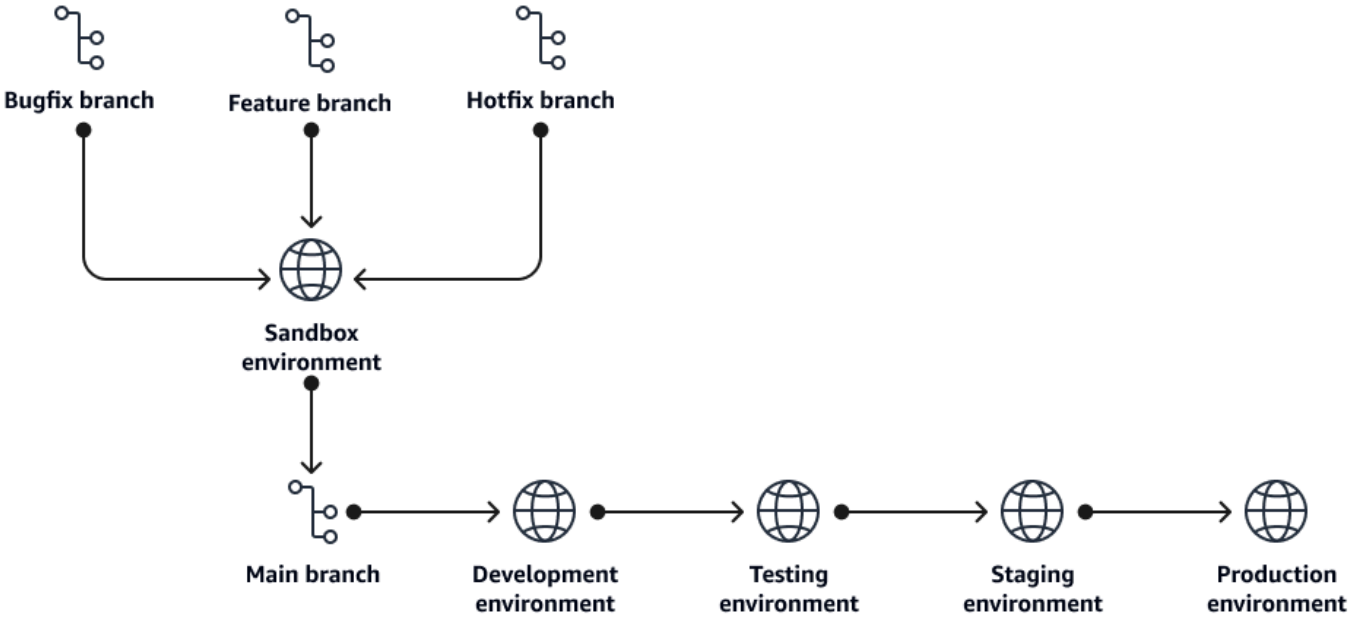
Das folgende Diagramm kann wie ein Punnett-Quadrat verwendet werden, um die GitHub Flow-Branching-Strategie zu verstehen. Richten Sie die Zweige auf der vertikalen Achse mit den AWS

Umgebungen auf der horizontalen Achse aus, um zu bestimmen, welche Aktionen in den einzelnen Szenarien ausgeführt werden sollen. Die eingekreisten Zahlen führen Sie durch die Reihenfolge der Aktionen, die im Diagramm dargestellt sind. Dieses Diagramm zeigt den GitHub Entwicklungsablauf einer Flow-Branching-Strategie von einem Feature-Branch in der Sandbox-Umgebung bis zur Produktionsversion des Hauptzweigs. Weitere Informationen zu den Aktivitäten, die in den einzelnen Umgebungen stattfinden, finden Sie in diesem DevOps Handbuch unter Umgebungen.



Filialen in einer GitHub Flow-Strategie

Eine GitHub Flow-Branching-Strategie besteht üblicherweise aus den folgenden Verzweigungen.



Feature-Zweig

Sie entwickeln Funktionen in `feature` Branchen. Um einen `feature` Zweig zu erstellen, zweigen Sie vom `main` Zweig ab. Entwickler iterieren, übernehmen und testen den Code im `feature` Branch. Wenn ein Feature fertiggestellt ist, bewirbt der Entwickler das Feature, indem er eine Merge-Anfrage an `main` erstellt.

Benennungskonvention:	`feature/<story number>_<developer initials>_<descriptor>`
Beispiel für eine Namenskonvention:	`feature/123456_MS_Implement_Feature_A`

Bugfix-Zweig

Der `bugfix` Zweig wird verwendet, um Probleme zu beheben. Diese Zweige sind von der `main` Filiale abgezweigt. Nachdem der Bugfix in der Sandbox oder einer der niedrigeren Umgebungen getestet wurde, kann er auf höhere Umgebungen hochgestuft werden, indem er mit einer Merge-Anfrage zusammengeführt wird. `main` Dies ist eine vorgeschlagene Benennungskonvention für Organisation und Nachverfolgung. Dieser Prozess könnte auch mithilfe eines Feature-Branches verwaltet werden.

Benennungskonvention:	`bugfix/<ticket number>_<developer initials>_<descriptor>`
Beispiel für eine Namenskonvention:	`bugfix/123456_MS_Fix_Problem_A`

Hotfix-Zweig

Der `hotfix` Branch wird verwendet, um kritische Probleme mit schwerwiegenden Auswirkungen mit minimaler Verzögerung zwischen dem Entwicklungsteam und dem in der Produktion bereitgestellten Code zu lösen. Diese Zweige sind von der `main` Filiale abgezweigt. Nachdem der Hotfix in einer Sandbox oder einer der niedrigeren Umgebungen getestet wurde, kann er auf höhere Umgebungen hochgestuft werden, indem er mit einer Merge-Anfrage zusammengeführt wird. `main` Dies ist eine vorgeschlagene Benennungskonvention für Organisation und Nachverfolgung. Dieser Prozess könnte auch mithilfe eines Feature-Branches verwaltet werden.

Benennungskonvention:	`hotfix/<ticket number>_<developer initials>_<descriptor>`
Beispiel für eine Namenskonvention:	`hotfix/123456_MS_Fix_Problem_A`

Hauptzweig

Der `main` Zweig steht immer für den Code, der in der Produktion ausgeführt wird. Code wird mithilfe von Merge-Anfragen aus `main` `feature` Verzweigungen mit dem Branch zusammengeführt. Um vor dem Löschen zu schützen und um zu verhindern, dass Entwickler Code direkt an den Branch `main` weiterleiten, aktivieren Sie den `main` Branch-Schutz.

Benennungskonvention:	`main`

Vor- und Nachteile der GitHub Flow-Strategie

Die Github Flow-Branching-Strategie eignet sich gut für kleinere, ausgereifte Entwicklungsteams mit ausgeprägten Kommunikationsfähigkeiten. Diese Strategie eignet sich gut für Teams, die Continuous Delivery implementieren möchten, und sie wird von gängigen CI/CD-Engines gut unterstützt.

GitHub Flow ist leichtgewichtig, hat nicht zu viele Regeln und ist in der Lage, schnelllebige Teams zu unterstützen. Es ist nicht gut geeignet, wenn Ihre Teams strenge Compliance- oder Freigabeprozesse einhalten müssen. Zusammenführungskonflikte sind in diesem Modell häufig und werden wahrscheinlich häufig auftreten. Die Lösung von Zusammenführungskonflikten ist eine Schlüsselkompetenz, und Sie müssen alle Teammitglieder entsprechend schulen.

Vorteile

GitHub Flow bietet mehrere Vorteile, die den Entwicklungsprozess verbessern, die Zusammenarbeit rationalisieren und die Gesamtqualität der Software verbessern können. Im Folgenden sind einige der wichtigsten Vorteile aufgeführt:

- Flexibel und leicht — GitHub Flow ist ein leichter und flexibler Workflow, der Entwicklern hilft, bei Softwareentwicklungsprojekten zusammenzuarbeiten. Er ermöglicht schnelle Iterationen und Experimente mit minimaler Komplexität.
- Vereinfachte Zusammenarbeit — GitHub Flow bietet einen klaren und optimierten Prozess für die Verwaltung der Funktionsentwicklung. Es fördert kleine, gezielte Änderungen, die schnell überprüft und zusammengeführt werden können, wodurch die Effizienz verbessert wird.
- Klare Versionskontrolle — Mit GitHub Flow wird jede Änderung in einem separaten Zweig vorgenommen. Dadurch wird ein klarer und nachvollziehbarer Verlauf der Versionskontrolle eingerichtet. Dies hilft Entwicklern, Änderungen nachzuverfolgen und zu verstehen, sie bei Bedarf rückgängig zu machen und eine zuverlässige Codebasis aufrechtzuerhalten.
- Nahtlose kontinuierliche Integration — GitHub Flow lässt sich in Tools für die kontinuierliche Integration integrieren. Durch die Erstellung von Pull-Requests können automatisierte Test- und Bereitstellungsprozesse eingeleitet werden. CI-Tools helfen Ihnen dabei, Änderungen gründlich zu testen, bevor sie in den `main` Branch integriert werden, wodurch das Risiko verringert wird, dass Fehler in die Codebasis aufgenommen werden.
- Schnelles Feedback und kontinuierliche Verbesserung — GitHub Flow fördert eine schnelle Feedback-Schleife, indem es häufige Codeüberprüfungen und Diskussionen durch Pull-Requests fördert. Dies erleichtert die Früherkennung von Problemen, fördert den Wissensaustausch zwischen den Teammitgliedern und führt letztendlich zu einer höheren Codequalität und einer besseren Zusammenarbeit innerhalb des Entwicklungsteams.
- Vereinfachte Rollbacks und Rückgängigmachungen — Falls eine Codeänderung zu einem unerwarteten Fehler oder Problem führt, vereinfacht GitHub Flow den Vorgang des Rollbacks oder Rückgängigmachens der Änderung. Durch eine klare Historie von Commits und Branches ist

es einfacher, problematische Änderungen zu identifizieren und rückgängig zu machen, was zur Aufrechterhaltung einer stabilen und funktionierenden Codebasis beiträgt.

- Leichte Lernkurve — GitHub Flow kann einfacher zu erlernen und anzuwenden sein als Gitflow, insbesondere für Teams, die bereits mit Git- und Versionskontrollkonzepten vertraut sind. Seine Einfachheit und sein intuitives Branching-Modell machen es für Entwickler mit unterschiedlichem Erfahrungsniveau zugänglich und reduzieren so die Lernkurve, die mit der Einführung neuer Entwicklungsworkflows verbunden ist.
- Kontinuierliche Entwicklung — GitHub Flow ermöglicht es Teams, einen kontinuierlichen Implementierungsansatz zu verfolgen, indem jede Änderung sofort implementiert wird, sobald sie in der `main` Filiale integriert ist. Dieser optimierte Prozess verhindert unnötige Verzögerungen und stellt sicher, dass die neuesten Updates und Verbesserungen den Benutzern schnell zur Verfügung gestellt werden. Dies führt zu einem agileren und reaktionsschnelleren Entwicklungszyklus.

Nachteile

GitHub Flow bietet zwar mehrere Vorteile, es ist jedoch wichtig, auch die potenziellen Nachteile zu berücksichtigen:

- Eingeschränkte Eignung für große Projekte — GitHub Flow eignet sich möglicherweise nicht so gut für Großprojekte mit komplexen Codebasen und mehreren langfristigen Funktionszweigen. In solchen Fällen könnte ein strukturierterer Workflow wie Gitflow eine bessere Kontrolle über die gleichzeitige Entwicklung und das Release-Management bieten.
- Fehlende formale Release-Struktur — GitHub Flow definiert nicht explizit einen Release-Prozess und unterstützt auch keine Funktionen wie Versionierung, Hotfixes oder Wartungszweige. Dies kann eine Einschränkung für Projekte sein, die ein striktes Release-Management erfordern oder langfristigen Support und Wartung benötigen.
- Eingeschränkte Unterstützung für langfristige Release-Planung — GitHub Flow konzentriert sich auf kurzlebige Funktionsbereiche, die möglicherweise nicht gut zu Projekten passen, die eine langfristige Release-Planung erfordern, z. B. solche mit strengen Roadmaps oder umfangreichen Feature-Abhängigkeiten. Die Verwaltung komplexer Release-Zeitpläne kann angesichts der Einschränkungen von GitHub Flow eine Herausforderung sein.
- Potenzial für häufige Zusammenführungskonflikte — Da GitHub Flow häufiges Verzweigen und Zusammenführen fördert, besteht die Möglichkeit, dass Zusammenführungskonflikte auftreten, insbesondere bei Projekten mit viel Entwicklungsaktivität. Die Lösung dieser Konflikte kann zeitaufwändig sein und zusätzliche Anstrengungen des Entwicklungsteams erfordern.

- Fehlen formalisierter Workflow-Phasen — GitHub Flow definiert keine expliziten Entwicklungsphasen, wie Alpha-, Beta- oder Release-Candidate-Phasen. Dies kann es schwieriger machen, den aktuellen Status des Projekts oder den Stabilitätsgrad verschiedener Branches oder Releases zu kommunizieren.
- Auswirkungen bahnbrechender Änderungen — Da GitHub Flow das häufige Zusammenführen von Änderungen im `main` Branch fördert, besteht ein höheres Risiko, dass bahnbrechende Änderungen eingeführt werden, die die Stabilität der Codebasis beeinträchtigen. Strenge Codeüberprüfungs- und Testpraktiken sind entscheidend, um dieses Risiko wirksam zu mindern.

Strategie zur Branchierung von Gitflow

Gitflow ist ein Verzweigungsmodell, bei dem mehrere Zweige verwendet werden, um Code von der Entwicklung in die Produktion zu übertragen. Gitflow eignet sich gut für Teams, die Release-Zyklen geplant haben und eine Sammlung von Funktionen als Release definieren müssen.
Die Entwicklung wird in einzelnen Feature-Branches abgeschlossen, die mit Genehmigung zu einem Entwicklungszweig zusammengeführt werden, der für die Integration verwendet wird. Die Funktionen in diesem Zweig gelten als produktionsbereit. Wenn sich alle geplanten Funktionen im Entwicklungsbereich angesammelt haben, wird ein Release-Zweig für Implementierungen in höheren Umgebungen erstellt. Diese Trennung verbessert die Kontrolle darüber, welche Änderungen nach einem festgelegten Zeitplan in welche benannte Umgebung übertragen werden. Bei Bedarf können Sie diesen Prozess beschleunigen und zu einem schnelleren Bereitstellungsmodell übergehen.

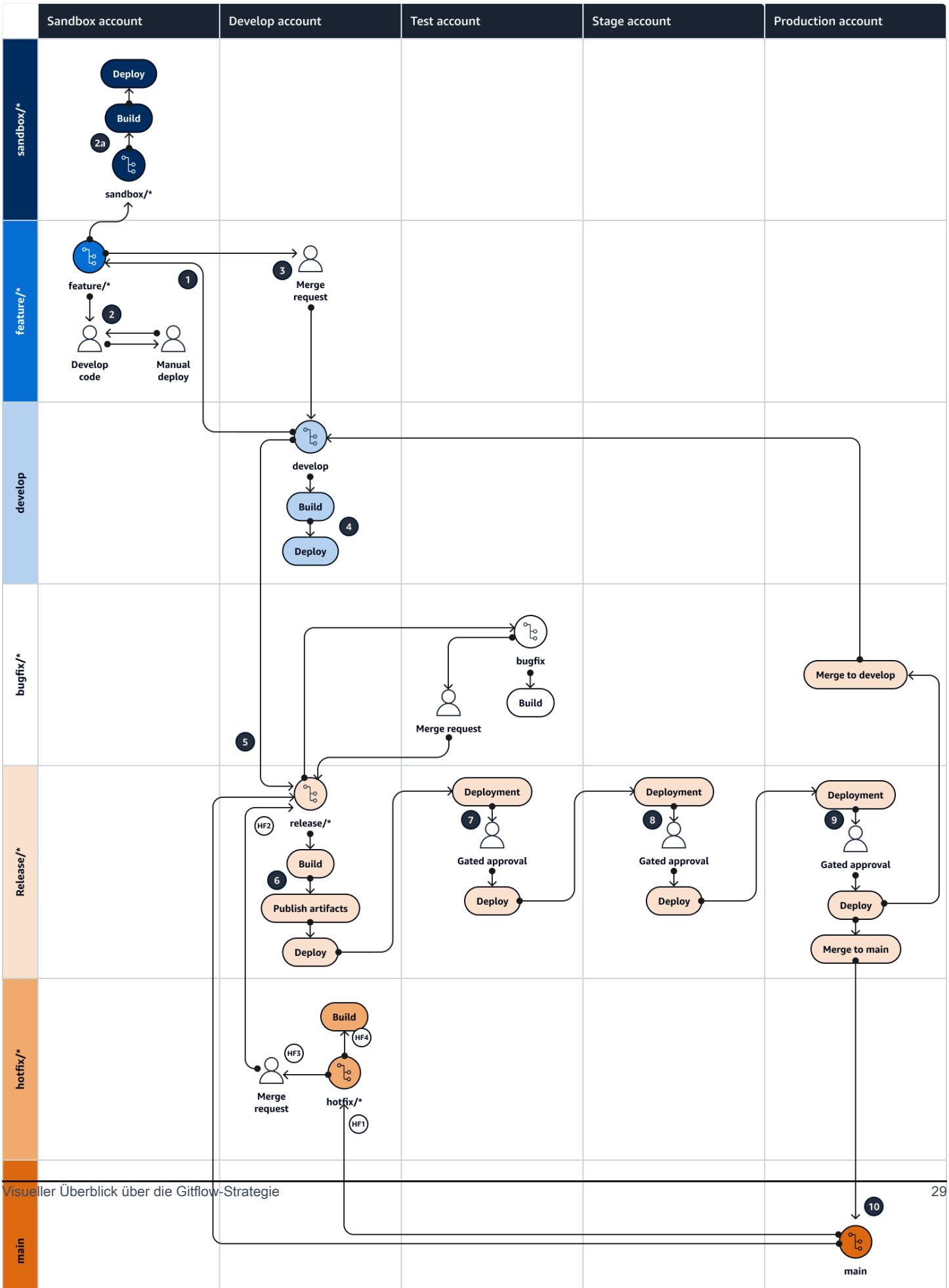
Weitere Informationen zur Gitflow-Branching-Strategie finden Sie in den folgenden Ressourcen:

- Implementieren Sie eine Gitflow-Branching-Strategie für Umgebungen mit mehreren Konten DevOps (Prescriptive Guidance)AWS
- Der ursprüngliche Gitflow-Blog (Blogbeitrag von Vincent Driessen)
- Gitflow-Workflow (Atlassian)

Themen in diesem Abschnitt:

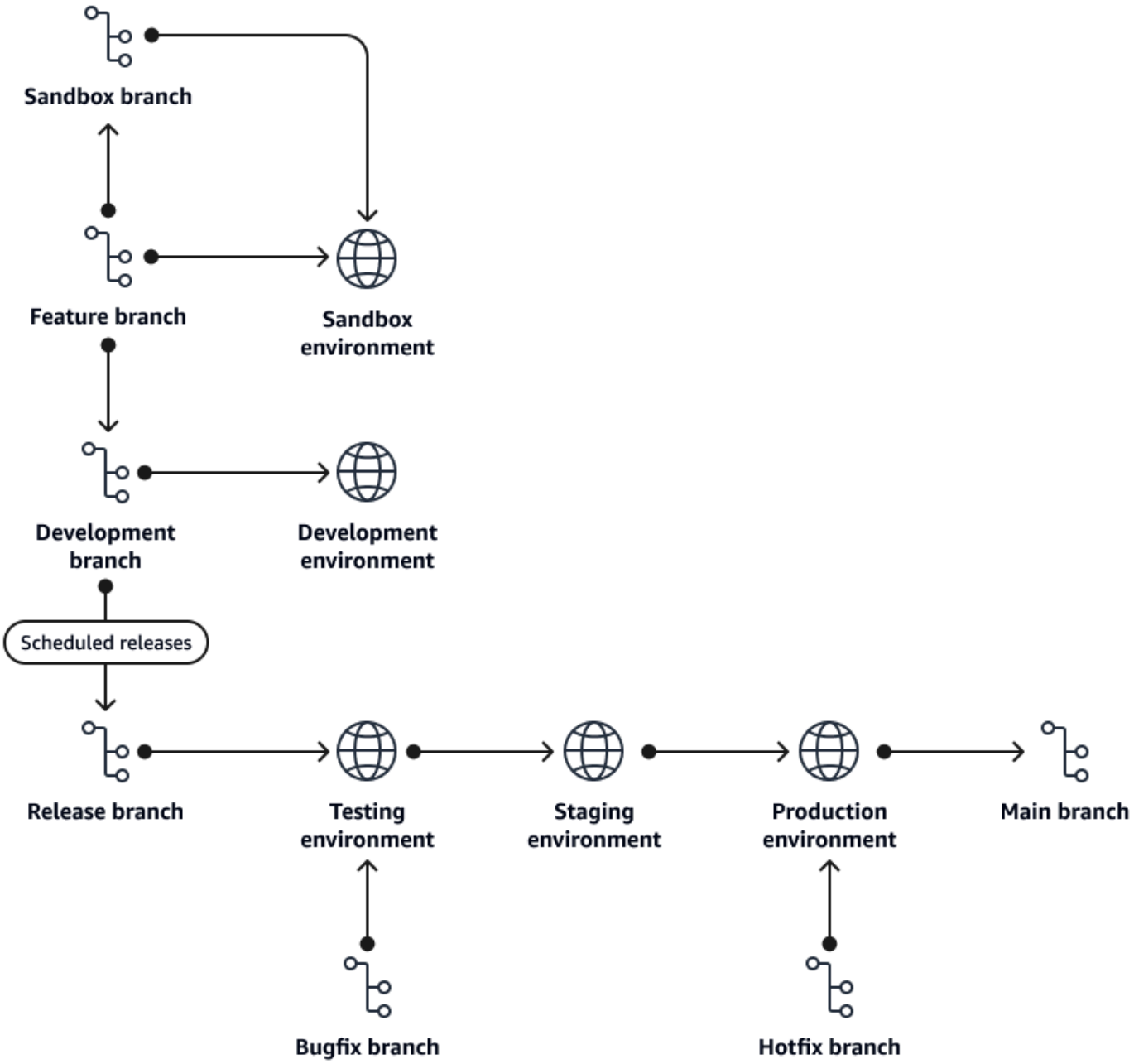
Visueller Überblick über die Gitflow-Strategie

Das folgende Diagramm kann wie ein Punnett-Quadrat verwendet werden, um die Gitflow-Verzweigungsstrategie zu verstehen. Richten Sie die Zweige auf der vertikalen Achse mit den AWS Umgebungen auf der horizontalen Achse aus, um zu bestimmen, welche Aktionen in den einzelnen Szenarien ausgeführt werden sollen. Die eingekreisten Zahlen führen Sie durch die Reihenfolge der Aktionen, die im Diagramm dargestellt sind. Weitere Informationen zu den Aktivitäten, die in den einzelnen Umgebungen stattfinden, finden Sie in diesem DevOps Handbuch unter Umgebungen.


Sandbox account
Develop account
Test account
Stage account
Production account
sandbox/*
feature/*
develop
bugfix/*
Release/*
hotfix/*
main
Deploy
Build
2a
sandbox/*
feature/*
1
2
Develop code
Manual deploy
3
Merge request
develop
Build
4
Deploy
bugfix
Build
Merge request
5
Merge to develop
HF2
release/*
Build
6
Publish artifacts
Deploy
Deployment
7
Gated approval
Deploy
Deployment
8
Gated approval
Deploy
Deployment
9
Gated approval
Deploy
Merge to main
Build
HF4
HF3
Merge request
hotfix/*
HF1
10
main

Branchen in einer Gitflow-Strategie

Eine Gitflow-Branching-Strategie besteht üblicherweise aus den folgenden Verzweigungen.



Feature-Zweig

`Feature`Branches sind kurzfristige Branches, in denen du Funktionen entwickelst. Die `feature` Verzweigung entsteht durch eine Abzweigung von der `develop` Verzweigung. Entwickler iterieren,

übertragen und testen den Code im `feature` Branch. Wenn die Funktion abgeschlossen ist, bewirbt der Entwickler die Funktion. Von einem Feature-Branch aus gibt es nur zwei Pfade vorwärts:

- Mit dem `sandbox` Zweig zusammenführen
- Erstellen Sie eine Anfrage zur Zusammenführung mit der `develop` Filiale

Benennungskonvention:	`feature/<story number>_<developer initials>_<descriptor>`
Beispiel für eine Namenskonvention:	`feature/123456_MS_Implement_Feature_A`

Sandbox-Zweig

Der `sandbox` Branch ist ein nicht standardmäßiger, kurzfristiger Branch für Gitflow. Er ist jedoch nützlich für die Entwicklung von CI/CD-Pipelines. Der `sandbox` Zweig wird hauptsächlich für folgende Zwecke verwendet:

- Führen Sie statt einer manuellen Bereitstellung eine vollständige Bereitstellung in der Sandbox-Umgebung durch, indem Sie die CI/CD-Pipelines verwenden.
- Entwickeln und testen Sie eine Pipeline, bevor Sie Zusammenführungsanfragen für vollständige Tests in einer niedrigeren Umgebung einreichen, z. B. bei der Entwicklung oder beim Testen.

`Sandbox`Zweige sind temporärer Natur und nicht für eine lange Lebensdauer konzipiert. Sie sollten gelöscht werden, nachdem die spezifischen Tests abgeschlossen sind.

Namenskonvention:	`sandbox/<story number>_<developer initials>_<descriptor>`
Beispiel für eine Namenskonvention:	`sandbox/123456_MS_Test_Pipeline_Deploy`

Zweig entwickeln

Der `develop` Zweig ist ein langlebiger Zweig, in dem Funktionen integriert, erstellt, validiert und in der Entwicklungsumgebung bereitgestellt werden. Alle `feature` Zweige sind in der `develop` Filiale zusammengeführt. Zusammenführungen mit der `develop` Filiale werden über eine Zusammenführungsanfrage abgeschlossen, für die ein erfolgreicher Build und zwei Genehmigungen durch Entwickler erforderlich sind. Um das Löschen zu verhindern, aktivieren Sie den Branch-Schutz für den `develop` Branch.

Benennungskonvention:	`develop`

Zweig freigeben

In Gitflow sind `release` Branches kurzfristige Branches. Diese Branches sind besonders, weil du sie in mehreren Umgebungen bereitstellen kannst, wobei du die Build-Once-Deploy-Many-Methode verwendest. `Release`Branches können auf Test-, Staging- oder Produktionsumgebungen abzielen. Nachdem ein Entwicklungsteam beschlossen hat, Funktionen in höheren Umgebungen einzuführen, erstellt es einen neuen `release` Branch und verwendet inkrementell die Versionsnummer der vorherigen Version. An den Eingängen in jeder Umgebung müssen Bereitstellungen manuell genehmigt werden, um fortzufahren. `Release`Für Branches sollte eine Merge-Anfrage erforderlich sein, damit sie geändert werden kann.

Nachdem der `release` Branch in Betrieb genommen wurde, sollte er wieder mit den `main` Zweigen `develop` und zusammengeführt werden, um sicherzustellen, dass alle Bugfixes oder Hotfixes wieder in future Entwicklungsbemühungen einfließen.

Benennungskonvention:	`release/v{major}.{minor}`
Beispiel für eine Namenskonvention:	`release/v1.0`

Hauptzweig

Der `main` Zweig ist ein langlebiger Zweig, der immer den Code darstellt, der in der Produktion ausgeführt wird. Nach einer erfolgreichen Bereitstellung aus der `main` Release-Pipeline wird Code automatisch aus einem Release-Branch mit dem Branch zusammengeführt. Um das Löschen zu verhindern, aktivieren Sie den Branch-Schutz für den `main` Branch.

Benennungskonvention:	`main`

Bugfix-Zweig

Der `bugfix` Branch ist ein kurzfristiger Branch, der verwendet wird, um Probleme in Release-Branches zu beheben, die noch nicht für die Produktion freigegeben wurden. Ein `bugfix` Branch sollte nur verwendet werden, um Fixes in `release` Branches in der Test-, Staging- oder Produktionsumgebung weiterzuleiten. Ein `bugfix` Zweig ist immer von einem `release` Zweig abgezweigt.

Nachdem der Bugfix getestet wurde, kann er durch eine Merge-Anfrage in den `release` Branch hochgestuft werden. Anschließend können Sie den `release` Branch weiterentwickeln, indem Sie dem Standard-Release-Prozess folgen.

Namenskonvention:	`bugfix/<ticket>_<developer initials>_<descriptor>`
Beispiel für eine Namenskonvention:	`bugfix/123456_MS_Fix_Problem_A`

Hotfix-Zweig

Der `hotfix` Zweig ist ein kurzfristiger Zweig, der zur Behebung von Problemen in der Produktion verwendet wird. Sie dient ausschließlich der Förderung von Problembehebungen, die schnellstmöglich in der Produktionsumgebung eingeführt werden müssen. Ein `hotfix` Zweig wird immer abgezweigt von. `main`

Nachdem der Hotfix getestet wurde, können Sie ihn mithilfe einer Merge-Anfrage in den `release` Branch, aus dem er erstellt wurde, zur Produktion hochstufen. `main` Zum Testen können Sie den `release` Branch dann weiterentwickeln, indem Sie dem Standard-Release-Prozess folgen.

Benennungskonvention:	`hotfix/<ticket>_<developer initials>_<descriptor>`
Beispiel für eine Namenskonvention:	`hotfix/123456_MS_Fix_Problem_A`

Vor- und Nachteile der Gitflow-Strategie

Die Gitflow-Branching-Strategie eignet sich gut für größere, stärker verteilte Teams, die strenge Freigabe- und Compliance-Anforderungen haben. Gitflow trägt zu einem vorhersehbaren Release-Zyklus für das Unternehmen bei, was häufig von größeren Organisationen bevorzugt wird. Gitflow eignet sich auch gut für Teams, die Leitplanken benötigen, um ihren Softwareentwicklungszyklus ordnungsgemäß abzuschließen. Dies liegt daran, dass in die Strategie mehrere Möglichkeiten zur Überprüfung und Qualitätssicherung integriert sind. Gitflow eignet sich auch gut für Teams, die mehrere Versionen von Produktionsversionen gleichzeitig verwalten müssen. Einige Nachteile von GitFlow bestehen darin, dass es komplexer ist als andere Branching-Modelle und die strikte Einhaltung des Musters erfordert, um erfolgreich abgeschlossen zu werden. Gitflow eignet sich aufgrund der starren Art der Verwaltung von Release-Branches nicht gut für Unternehmen, die eine kontinuierliche Bereitstellung anstreben. Gitflow-Release-Branches können langlebige Branches sein, in denen sich technische Schulden anhäufen können, wenn sie nicht rechtzeitig behoben werden.

Vorteile

Die GitFlow-basierte Entwicklung bietet mehrere Vorteile, die den Entwicklungsprozess verbessern, die Zusammenarbeit rationalisieren und die Gesamtqualität der Software verbessern können. Im Folgenden sind einige der wichtigsten Vorteile aufgeführt:

- Vorhersehbarer Release-Prozess — Gitflow folgt einem regelmäßigen und vorhersehbaren Release-Prozess. Es eignet sich gut für Teams mit regelmäßigen Entwicklungs- und Veröffentlichungsrhythmen.
- Verbesserte Zusammenarbeit — Gitflow fördert die Verwendung von `feature` und `release` Branches. Diese beiden Zweige helfen Teams dabei, parallel und mit minimalen Abhängigkeiten voneinander zu arbeiten.
- Gut geeignet für mehrere Umgebungen — Gitflow verwendet `release` Branches, bei denen es sich um langlebigere Branches handeln kann. Diese Branches ermöglichen es Teams, einzelne Releases über einen längeren Zeitraum hinweg ins Visier zu nehmen.
- Mehrere Versionen in der Produktion — Wenn Ihr Team mehrere Versionen der Software in der Produktion unterstützt, unterstützen `release` Gitflow-Branches diese Anforderung.
- Integrierte Überprüfungen der Codequalität — Gitflow verlangt und fördert die Verwendung von Codeprüfungen und Genehmigungen, bevor Code in einer anderen Umgebung veröffentlicht wird. Dieser Prozess beseitigt Reibungspunkte zwischen Entwicklern, da dieser Schritt für alle Code-Werbeaktionen erforderlich ist.

- Vorteile für die Organisation — Gitflow hat auch auf Organisationsebene Vorteile. Gitflow empfiehlt die Verwendung eines Standard-Release-Zyklus, der der Organisation hilft, den Release-Zeitplan zu verstehen und zu antizipieren. Da das Unternehmen jetzt weiß, wann neue Funktionen bereitgestellt werden können, gibt es weniger Reibungsverluste in Bezug auf Zeitpläne, da es feste Liefertermine gibt.

Nachteile

Die GitFlow-basierte Entwicklung hat einige Nachteile, die sich auf den Entwicklungsprozess und die Teamdynamik auswirken können. Im Folgenden sind einige bemerkenswerte Nachteile aufgeführt:

- Komplexität — Gitflow ist ein komplexes Muster, das neue Teams erlernen müssen, und du musst dich an die Regeln von Gitflow halten, um es erfolgreich zu nutzen.
- Kontinuierliche Bereitstellung — Gitflow passt nicht zu einem Modell, bei dem viele Implementierungen schnell für die Produktion freigegeben werden. Das liegt daran, dass Gitflow die Verwendung mehrerer Branches und einen strikten Workflow für den Branch erfordert. `release`
- Filialverwaltung — Gitflow verwendet viele Branches, deren Wartung aufwändig werden kann. Es kann schwierig sein, die verschiedenen Branches nachzuverfolgen und den veröffentlichten Code zusammenzuführen, um die Branches korrekt aufeinander abzustimmen.
- Technische Schulden — Da Gitflow-Releases in der Regel langsamer sind als die anderen Branching-Modelle, können sich bis zur Veröffentlichung mehr Funktionen ansammeln, was dazu führen kann, dass sich technische Schulden anhäufen.

Teams sollten diese Nachteile sorgfältig abwägen, wenn sie entscheiden, ob eine GitFlow-basierte Entwicklung der richtige Ansatz für ihr Projekt ist.

Nächste Schritte

In diesem Leitfaden werden die Unterschiede zwischen drei gängigen Git-Branching-Strategien erklärt: GitHub Flow, Gitflow und Trunk. Es beschreibt ihre Workflows im Detail und bietet auch die Vor- und Nachteile der einzelnen Workflows. Die nächsten Schritte bestehen darin, einen dieser Standardworkflows für Ihr Unternehmen auszuwählen. Informationen zur Implementierung einer dieser Verzweigungsstrategien finden Sie im Folgenden:

- Implementieren Sie eine Trunk-Branching-Strategie für Umgebungen mit mehreren Konten DevOps
- Implementieren Sie eine GitHub Flow-Branching-Strategie für Umgebungen mit mehreren Konten DevOps
- Implementieren Sie eine Gitflow-Branching-Strategie für Umgebungen mit mehreren Konten DevOps

Wenn du dir nicht sicher bist, wo dein Team die Einführung von Git und DevOps Prozessen beginnen soll, empfehlen wir dir, eine Standardlösung auszuwählen und sie zu testen. Die Verwendung einer Standard-Branching-Konvention hilft dem Team, auf der vorhandenen Dokumentation aufzubauen und herauszufinden, was für das Team am besten funktioniert.

Haben Sie keine Angst davor, Ihre Strategie zu ändern, wenn sie für Ihr Unternehmen oder Ihre Entwicklungsteams nicht funktioniert. Die Bedürfnisse und Anforderungen von Entwicklungsteams können sich im Laufe der Zeit ändern, und es gibt keine einzige, perfekte Lösung.

Ressourcen

In diesem Leitfaden sind keine Schulungen für Git enthalten. Es stehen jedoch viele hochwertige Ressourcen im Internet zur Verfügung, falls Sie diese Schulung benötigen. Wir empfehlen, dass Sie mit der Git-Dokumentationsseite beginnen.

Die folgenden Ressourcen können Ihnen bei Ihrer Git-Branching-Reise in der AWS Cloud helfen.

AWS Präskriptive Leitlinien

- Implementieren Sie eine Trunk-Branching-Strategie für Umgebungen mit mehreren Konten DevOps
- Implementieren Sie eine GitHub Flow-Branching-Strategie für Umgebungen mit mehreren Konten DevOps
- Implementieren Sie eine Gitflow-Branching-Strategie für Umgebungen mit mehreren Konten DevOps

Weitere Hinweise AWS

- AWS DevOps Beratung
- AWS Referenzarchitektur für die Bereitstellungspipeline
- Was ist DevOps?
- DevOpsRessourcen

Sonstige Ressourcen

- Zwölf-Faktoren-App-Methodik (12factor.net)
- Geschenk-Geheimnisse (2) GitHub
- Gitflow
  - Der ursprüngliche Gitflow-Blog (Blogbeitrag von Vincent Driessen)
  - Gitflow-Workflow (Atlassian)
  - Gitflow on GitHub: So verwendest du Git Flow-Workflows mit GitHub Based Repos (Video) YouTube

  - Beispiel für Git Flow Init (YouTube Video)
  - Der Gitflow-Release-Zweig von Anfang bis Ende (Video) YouTube
- GitHub Fluss
  - GitHub Flow Quickstart (GitHub Dokumentation)
  - Warum GitHub Flow? (GitHub Flow-Webseite)
- Kofferraum
  - Einführung in die Trunk-basierte Entwicklung (Website zur Trunk-basierten Entwicklung)

Mitwirkende

Inhaltserstellung

- Mike Stephens, leitender Architekt für Cloud-Anwendungen, AWS
- Rayjan Wilson, leitender Architekt für Cloud-Anwendungen, AWS
- Abhilash Vinod, Teamleiter, leitender Architekt für Cloud-Anwendungen, AWS

Überprüfung

- Stephen DiCato, leitender Sicherheitsberater, AWS
- Gaurav Samudra, Architekt für Cloud-Anwendungen, AWS
- Steven Guggenheimer, Teamleiter, leitender Architekt für Cloud-Anwendungen, AWS

Technisches Schreiben

- Lilly AbouHarb, leitende technische Redakteurin, AWS

Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in diesem Leitfaden beschrieben. Um Benachrichtigungen über zukünftige Aktualisierungen zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Erste Veröffentlichung	—	15. Februar 2024

AWS Glossar zu präskriptiven Leitlinien

Im Folgenden finden Sie häufig verwendete Begriffe in Strategien, Leitfäden und Mustern von AWS Prescriptive Guidance. Um Einträge vorzuschlagen, verwenden Sie bitte den Link Feedback geben am Ende des Glossars.

Zahlen

7 Rs

Sieben gängige Migrationsstrategien für die Verlagerung von Anwendungen in die Cloud. Diese Strategien bauen auf den 5 Rs auf, die Gartner 2011 identifiziert hat, und bestehen aus folgenden Elementen:

- Faktorwechsel/Architekturwechsel – Verschieben Sie eine Anwendung und ändern Sie ihre Architektur, indem Sie alle Vorteile cloudnativer Feature nutzen, um Agilität, Leistung und Skalierbarkeit zu verbessern. Dies beinhaltet in der Regel die Portierung des Betriebssystems und der Datenbank. Beispiel: Migrieren Sie Ihre lokale Oracle-Datenbank auf die Amazon Aurora PostgreSQL-kompatible Edition.
- Plattformwechsel (Lift and Reshape) – Verschieben Sie eine Anwendung in die Cloud und führen Sie ein gewisses Maß an Optimierung ein, um die Cloud-Funktionen zu nutzen. Beispiel: Migrieren Sie Ihre lokale Oracle-Datenbank zu Amazon Relational Database Service (Amazon RDS) für Oracle in der. AWS Cloud
- Neukauf (Drop and Shop) – Wechseln Sie zu einem anderen Produkt, indem Sie typischerweise von einer herkömmlichen Lizenz zu einem SaaS-Modell wechseln. Beispiel: Migrieren Sie Ihr CRM-System (Customer Relationship Management) zu Salesforce.com.
- Hostwechsel (Lift and Shift) – Verschieben Sie eine Anwendung in die Cloud, ohne Änderungen vorzunehmen, um die Cloud-Funktionen zu nutzen. Beispiel: Migrieren Sie Ihre lokale Oracle-Datenbank zu Oracle auf einer EC2-Instanz in der. AWS Cloud
- Verschieben (Lift and Shift auf Hypervisor-Ebene) – Verlagern Sie die Infrastruktur in die Cloud, ohne neue Hardware kaufen, Anwendungen umschreiben oder Ihre bestehenden Abläufe ändern zu müssen. Sie migrieren Server von einer lokalen Plattform zu einem Cloud-Dienst für dieselbe Plattform. Beispiel: Migrieren Sie eine Microsoft Hyper-V Anwendung zu AWS.
- Beibehaltung (Wiederaufgreifen) – Bewahren Sie Anwendungen in Ihrer Quellumgebung auf. Dazu können Anwendungen gehören, die einen umfangreichen Faktorwechsel erfordern und

die Sie auf einen späteren Zeitpunkt verschieben möchten, sowie ältere Anwendungen, die Sie beibehalten möchten, da es keine geschäftliche Rechtfertigung für ihre Migration gibt.

- Außerbetriebnahme – Dekommissionierung oder Entfernung von Anwendungen, die in Ihrer Quellumgebung nicht mehr benötigt werden.

A

ABAC

Siehe attributbasierte Zugriffskontrolle.

abstrahierte Dienste

Weitere Informationen finden Sie unter Managed Services.

ACID

Siehe Atomarität, Konsistenz, Isolierung und Haltbarkeit.

Aktiv-Aktiv-Migration

Eine Datenbankmigrationsmethode, bei der die Quell- und Zieldatenbanken synchron gehalten werden (mithilfe eines bidirektionalen Replikationstools oder dualer Schreibvorgänge) und beide Datenbanken Transaktionen von miteinander verbundenen Anwendungen während der Migration verarbeiten. Diese Methode unterstützt die Migration in kleinen, kontrollierten Batches, anstatt einen einmaligen Cutover zu erfordern. Es ist flexibler, erfordert aber mehr Arbeit als eine aktiv-passive Migration.

Aktiv-Passiv-Migration

Eine Datenbankmigrationsmethode, bei der die Quell- und Zieldatenbanken synchron gehalten werden, aber nur die Quelldatenbank Transaktionen von verbindenden Anwendungen verarbeitet, während Daten in die Zieldatenbank repliziert werden. Die Zieldatenbank akzeptiert während der Migration keine Transaktionen.

Aggregatfunktion

Eine SQL-Funktion, die mit einer Gruppe von Zeilen arbeitet und einen einzelnen Rückgabewert für die Gruppe berechnet. Beispiele für Aggregatfunktionen sind `SUM` und`MAX`.

AI

Siehe künstliche Intelligenz.

AIOps

Siehe Operationen mit künstlicher Intelligenz.

Anonymisierung

Der Prozess des dauerhaften Löschens personenbezogener Daten in einem Datensatz. Anonymisierung kann zum Schutz der Privatsphäre beitragen. Anonymisierte Daten gelten nicht mehr als personenbezogene Daten.

Anti-Muster

Eine häufig verwendete Lösung für ein wiederkehrendes Problem, bei dem die Lösung kontraproduktiv, ineffektiv oder weniger wirksam als eine Alternative ist.

Anwendungssteuerung

Ein Sicherheitsansatz, bei dem nur zugelassene Anwendungen verwendet werden können, um ein System vor Schadsoftware zu schützen.

Anwendungsportfolio

Eine Sammlung detaillierter Informationen zu jeder Anwendung, die von einer Organisation verwendet wird, einschließlich der Kosten für die Erstellung und Wartung der Anwendung und ihres Geschäftswerts. Diese Informationen sind entscheidend für den Prozess der Portfoliofindung und -analyse und hilft bei der Identifizierung und Priorisierung der Anwendungen, die migriert, modernisiert und optimiert werden sollen.

künstliche Intelligenz (KI)

Das Gebiet der Datenverarbeitungswissenschaft, das sich der Nutzung von Computertechnologien zur Ausführung kognitiver Funktionen widmet, die typischerweise mit Menschen in Verbindung gebracht werden, wie Lernen, Problemlösen und Erkennen von Mustern. Weitere Informationen finden Sie unter Was ist künstliche Intelligenz?

Operationen mit künstlicher Intelligenz (AIOps)

Der Prozess des Einsatzes von Techniken des Machine Learning zur Lösung betrieblicher Probleme, zur Reduzierung betrieblicher Zwischenfälle und menschlicher Eingriffe sowie zur Steigerung der Servicequalität. Weitere Informationen zur Verwendung von AIOps in der AWS -Migrationsstrategie finden Sie im Leitfaden zur Betriebsintegration.

Asymmetrische Verschlüsselung

Ein Verschlüsselungsalgorithmus, der ein Schlüsselpaar, einen öffentlichen Schlüssel für die Verschlüsselung und einen privaten Schlüssel für die Entschlüsselung verwendet. Sie können den öffentlichen Schlüssel teilen, da er nicht für die Entschlüsselung verwendet wird. Der Zugriff auf den privaten Schlüssel sollte jedoch stark eingeschränkt sein.

Atomizität, Konsistenz, Isolierung, Haltbarkeit (ACID)

Eine Reihe von Softwareeigenschaften, die die Datenvalidität und betriebliche Zuverlässigkeit einer Datenbank auch bei Fehlern, Stromausfällen oder anderen Problemen gewährleisten.

Attributbasierte Zugriffskontrolle (ABAC)

Die Praxis, detaillierte Berechtigungen auf der Grundlage von Benutzerattributen wie Abteilung, Aufgabenrolle und Teamname zu erstellen. Weitere Informationen finden Sie unter ABAC AWS in der AWS Identity and Access Management (IAM-) Dokumentation.

autoritative Datenquelle

Ein Ort, an dem Sie die primäre Version der Daten speichern, die als die zuverlässigste Informationsquelle angesehen wird. Sie können Daten aus der maßgeblichen Datenquelle an andere Speicherorte kopieren, um die Daten zu verarbeiten oder zu ändern, z. B. zu anonymisieren, zu redigieren oder zu pseudonymisieren.

Availability Zone

Ein bestimmter Standort innerhalb einer AWS-Region , der vor Ausfällen in anderen Availability Zones geschützt ist und kostengünstige Netzwerkkonnektivität mit niedriger Latenz zu anderen Availability Zones in derselben Region bietet.

AWS Framework für die Cloud-Einführung (AWS CAF)

Ein Framework mit Richtlinien und bewährten Verfahren, das Unternehmen bei der Entwicklung eines effizienten und effektiven Plans für den erfolgreichen Umstieg auf die Cloud unterstützt. AWS AWS CAF unterteilt die Leitlinien in sechs Schwerpunktbereiche, die als Perspektiven bezeichnet werden: Unternehmen, Mitarbeiter, Unternehmensführung, Plattform, Sicherheit und Betrieb. Die Perspektiven Geschäft, Mitarbeiter und Unternehmensführung konzentrieren sich auf Geschäftskompetenzen und -prozesse, während sich die Perspektiven Plattform, Sicherheit und Betriebsabläufe auf technische Fähigkeiten und Prozesse konzentrieren. Die Personalperspektive zielt beispielsweise auf Stakeholder ab, die sich mit Personalwesen (HR), Personalfunktionen und Personalmanagement befassen. Aus dieser Perspektive bietet AWS CAF Leitlinien für Personalentwicklung, Schulung und Kommunikation, um das Unternehmen auf eine erfolgreiche

Cloud-Einführung vorzubereiten. Weitere Informationen finden Sie auf der AWS -CAF-Webseite und dem AWS -CAF-Whitepaper.

AWS Workload-Qualifizierungsrahmen (AWS WQF)

Ein Tool, das Workloads bei der Datenbankmigration bewertet, Migrationsstrategien empfiehlt und Arbeitsschätzungen bereitstellt. AWS WQF ist in () enthalten. AWS Schema Conversion Tool AWS SCT Es analysiert Datenbankschemas und Codeobjekte, Anwendungscode, Abhängigkeiten und Leistungsmerkmale und stellt Bewertungsberichte bereit.

B

schlechter Bot

Ein Bot, der Einzelpersonen oder Organisationen stören oder ihnen Schaden zufügen soll.

BCP

Siehe Planung der Geschäftskontinuität.

Verhaltensdiagramm

Eine einheitliche, interaktive Ansicht des Ressourcenverhaltens und der Interaktionen im Laufe der Zeit. Sie können ein Verhaltensdiagramm mit Amazon Detective verwenden, um fehlgeschlagene Anmeldeversuche, verdächtige API-Aufrufe und ähnliche Vorgänge zu untersuchen. Weitere Informationen finden Sie unter Daten in einem Verhaltensdiagramm in der Detective-Dokumentation.

Big-Endian-System

Ein System, welches das höchstwertige Byte zuerst speichert. Siehe auch Endianness.

Binäre Klassifikation

Ein Prozess, der ein binäres Ergebnis vorhersagt (eine von zwei möglichen Klassen). Beispielsweise könnte Ihr ML-Modell möglicherweise Probleme wie „Handelt es sich bei dieser E-Mail um Spam oder nicht?“ vorhersagen müssen oder „Ist dieses Produkt ein Buch oder ein Auto?“

Bloom-Filter

Eine probabilistische, speichereffiziente Datenstruktur, mit der getestet wird, ob ein Element Teil einer Menge ist.

Blau/Grün-Bereitstellung

Eine Bereitstellungsstrategie, bei der Sie zwei separate, aber identische Umgebungen erstellen. Sie führen die aktuelle Anwendungsversion in einer Umgebung (blau) und die neue Anwendungsversion in der anderen Umgebung (grün) aus. Mit dieser Strategie können Sie schnell und mit minimalen Auswirkungen ein Rollback durchführen.

Bot

Eine Softwareanwendung, die automatisierte Aufgaben über das Internet ausführt und menschliche Aktivitäten oder Interaktionen simuliert. Manche Bots sind nützlich oder nützlich, wie z. B. Webcrawler, die Informationen im Internet indexieren. Einige andere Bots, die als bösartige Bots bezeichnet werden, sollen Einzelpersonen oder Organisationen stören oder ihnen Schaden zufügen.

Botnetz

Netzwerke von Bots, die mit Malware infiziert sind und unter der Kontrolle einer einzigen Partei stehen, die als Bot-Herder oder Bot-Operator bezeichnet wird. Botnetze sind der bekannteste Mechanismus zur Skalierung von Bots und ihrer Wirkung.

branch

Ein containerisierter Bereich eines Code-Repositorys. Der erste Zweig, der in einem Repository erstellt wurde, ist der Hauptzweig. Sie können einen neuen Zweig aus einem vorhandenen Zweig erstellen und dann Feature entwickeln oder Fehler in dem neuen Zweig beheben. Ein Zweig, den Sie erstellen, um ein Feature zu erstellen, wird allgemein als Feature-Zweig bezeichnet. Wenn das Feature zur Veröffentlichung bereit ist, führen Sie den Feature-Zweig wieder mit dem Hauptzweig zusammen. Weitere Informationen finden Sie unter Über Branches (GitHub Dokumentation).

Zugang durch Glasbruch

Unter außergewöhnlichen Umständen und im Rahmen eines genehmigten Verfahrens ist dies eine schnelle Methode für einen Benutzer, auf einen Bereich zuzugreifen AWS-Konto , für den er normalerweise keine Zugriffsrechte besitzt. Weitere Informationen finden Sie unter dem Indikator Implementation break-glass procedures in den AWS Well-Architected-Leitlinien.

Brownfield-Strategie

Die bestehende Infrastruktur in Ihrer Umgebung. Wenn Sie eine Brownfield-Strategie für eine Systemarchitektur anwenden, richten Sie sich bei der Gestaltung der Architektur nach den

Einschränkungen der aktuellen Systeme und Infrastruktur. Wenn Sie die bestehende Infrastruktur erweitern, könnten Sie Brownfield- und Greenfield-Strategien mischen.

Puffer-Cache

Der Speicherbereich, in dem die am häufigsten abgerufenen Daten gespeichert werden.

Geschäftsfähigkeit

Was ein Unternehmen tut, um Wert zu generieren (z. B. Vertrieb, Kundenservice oder Marketing). Microservices-Architekturen und Entwicklungsentscheidungen können von den Geschäftskapazitäten beeinflusst werden. Weitere Informationen finden Sie im Abschnitt Organisiert nach Geschäftskapazitäten des Whitepapers Ausführen von containerisierten Microservices in AWS.

Planung der Geschäftskontinuität (BCP)

Ein Plan, der die potenziellen Auswirkungen eines störenden Ereignisses, wie z. B. einer groß angelegten Migration, auf den Betrieb berücksichtigt und es einem Unternehmen ermöglicht, den Betrieb schnell wieder aufzunehmen.

C

CAF

Weitere Informationen finden Sie unter Framework für die AWS Cloud-Einführung.

Bereitstellung auf Kanaren

Die langsame und schrittweise Veröffentlichung einer Version für Endbenutzer. Wenn Sie sich sicher sind, stellen Sie die neue Version bereit und ersetzen die aktuelle Version vollständig.

CCoE

Weitere Informationen finden Sie im Cloud Center of Excellence.

CDC

Siehe Erfassung von Änderungsdaten.

Erfassung von Datenänderungen (CDC)

Der Prozess der Nachverfolgung von Änderungen an einer Datenquelle, z. B. einer Datenbanktabelle, und der Aufzeichnung von Metadaten zu der Änderung. Sie können CDC für

verschiedene Zwecke verwenden, z. B. für die Prüfung oder Replikation von Änderungen in einem Zielsystem, um die Synchronisation aufrechtzuerhalten.

Chaos-Technik

Absichtliches Einführen von Ausfällen oder Störungsereignissen, um die Widerstandsfähigkeit eines Systems zu testen. Sie können AWS Fault Injection Service (AWS FIS) verwenden, um Experimente durchzuführen, die Ihre AWS Workloads stress, und deren Reaktion zu bewerten.

CI/CD

Siehe Continuous Integration und Continuous Delivery.

Klassifizierung

Ein Kategorisierungsprozess, der bei der Erstellung von Vorhersagen hilft. ML-Modelle für Klassifikationsprobleme sagen einen diskreten Wert voraus. Diskrete Werte unterscheiden sich immer voneinander. Beispielsweise muss ein Modell möglicherweise auswerten, ob auf einem Bild ein Auto zu sehen ist oder nicht.

clientseitige Verschlüsselung

Lokale Verschlüsselung von Daten, bevor das Ziel sie AWS -Service empfängt.

Cloud-Kompetenzzentrum (CCoE)

Ein multidisziplinäres Team, das die Cloud-Einführung in der gesamten Organisation vorantreibt, einschließlich der Entwicklung bewährter Cloud-Methoden, der Mobilisierung von Ressourcen, der Festlegung von Migrationszeitplänen und der Begleitung der Organisation durch groß angelegte Transformationen. Weitere Informationen finden Sie in den CCoE-Beiträgen im AWS Cloud Enterprise Strategy Blog.

Cloud Computing

Die Cloud-Technologie, die typischerweise für die Ferndatenspeicherung und das IoT-Gerätemanagement verwendet wird. Cloud Computing ist häufig mit Edge-Computing-Technologie verbunden.

Cloud-Betriebsmodell

In einer IT-Organisation das Betriebsmodell, das zum Aufbau, zur Weiterentwicklung und Optimierung einer oder mehrerer Cloud-Umgebungen verwendet wird. Weitere Informationen finden Sie unter Aufbau Ihres Cloud-Betriebsmodells.

Phasen der Einführung der Cloud

Die vier Phasen, die Unternehmen bei der Migration in der Regel durchlaufen AWS Cloud:

- Projekt – Durchführung einiger Cloud-bezogener Projekte zu Machbarkeitsnachweisen und zu Lernzwecken
- Fundament – Grundlegende Investitionen tätigen, um Ihre Cloud-Einführung zu skalieren (z. B. Einrichtung einer Landing Zone, Definition eines CCoE, Einrichtung eines Betriebsmodells)
- Migration – Migrieren einzelner Anwendungen
- Neuentwicklung – Optimierung von Produkten und Services und Innovation in der Cloud

Diese Phasen wurden von Stephen Orban im Blogbeitrag The Journey Toward Cloud-First & the Stages of Adoption im AWS Cloud Enterprise Strategy-Blog definiert. Informationen darüber, wie sie mit der AWS Migrationsstrategie zusammenhängen, finden Sie im Leitfaden zur Vorbereitung der Migration.

CMDB

Siehe Datenbank für das Konfigurationsmanagement.

Code-Repository

Ein Ort, an dem Quellcode und andere Komponenten wie Dokumentation, Beispiele und Skripts gespeichert und im Rahmen von Versionskontrollprozessen aktualisiert werden. Zu den gängigen Cloud-Repositorys gehören GitHub oder AWS CodeCommit. Jede Version des Codes wird Zweig genannt. In einer Microservice-Struktur ist jedes Repository einer einzelnen Funktionalität gewidmet. Eine einzelne CI/CD-Pipeline kann mehrere Repositorien verwenden.

Kalter Cache

Ein Puffer-Cache, der leer oder nicht gut gefüllt ist oder veraltete oder irrelevante Daten enthält. Dies beeinträchtigt die Leistung, da die Datenbank-Instance aus dem Hauptspeicher oder der Festplatte lesen muss, was langsamer ist als das Lesen aus dem Puffercache.

Kalte Daten

Daten, auf die selten zugegriffen wird und die in der Regel historisch sind. Bei der Abfrage dieser Art von Daten sind langsame Abfragen in der Regel akzeptabel. Durch die Verlagerung dieser Daten auf leistungsschwächere und kostengünstigere Speicherstufen oder -klassen können Kosten gesenkt werden.

Computer Vision (CV)

Ein Bereich der KI, der maschinelles Lernen nutzt, um Informationen aus visuellen Formaten wie digitalen Bildern und Videos zu analysieren und zu extrahieren. AWS Panorama Bietet beispielsweise Geräte an, die CV zu lokalen Kameranetzwerken hinzufügen, und Amazon SageMaker stellt Bildverarbeitungsalgorithmen für CV bereit.

Drift in der Konfiguration

Bei einer Arbeitslast eine Änderung der Konfiguration gegenüber dem erwarteten Zustand. Dies kann dazu führen, dass der Workload nicht mehr richtlinienkonform wird, und zwar in der Regel schrittweise und unbeabsichtigt.

Verwaltung der Datenbankkonfiguration (CMDB)

Ein Repository, das Informationen über eine Datenbank und ihre IT-Umgebung speichert und verwaltet, inklusive Hardware- und Softwarekomponenten und deren Konfigurationen. In der Regel verwenden Sie Daten aus einer CMDB in der Phase der Portfolioerkennung und -analyse der Migration.

Konformitätspaket

Eine Sammlung von AWS Config Regeln und Abhilfemaßnahmen, die Sie zusammenstellen können, um Ihre Konformitäts- und Sicherheitsprüfungen individuell anzupassen. Mithilfe einer YAML-Vorlage können Sie ein Conformance Pack als einzelne Entität in einer AWS-Konto AND-Region oder unternehmensweit bereitstellen. Weitere Informationen finden Sie in der Dokumentation unter Conformance Packs. AWS Config

Kontinuierliche Bereitstellung und kontinuierliche Integration (CI/CD)

Der Prozess der Automatisierung der Quell-, Build-, Test-, Staging- und Produktionsphasen des Softwareveröffentlichungsprozesses. CI/CD wird allgemein als Pipeline beschrieben. CI/CD kann Ihnen helfen, Prozesse zu automatisieren, die Produktivität zu steigern, die Codequalität zu verbessern und schneller zu liefern. Weitere Informationen finden Sie unter Vorteile der kontinuierlichen Auslieferung. CD kann auch für kontinuierliche Bereitstellung stehen. Weitere Informationen finden Sie unterKontinuierliche Auslieferung im Vergleich zu kontinuierlicher Bereitstellung.

CV

Siehe Computer Vision.

D

Daten im Ruhezustand

Daten, die in Ihrem Netzwerk stationär sind, z. B. Daten, die sich im Speicher befinden.

Datenklassifizierung

Ein Prozess zur Identifizierung und Kategorisierung der Daten in Ihrem Netzwerk auf der Grundlage ihrer Kritikalität und Sensitivität. Sie ist eine wichtige Komponente jeder Strategie für das Management von Cybersecurity-Risiken, da sie Ihnen hilft, die geeigneten Schutz- und Aufbewahrungskontrollen für die Daten zu bestimmen. Die Datenklassifizierung ist ein Bestandteil der Sicherheitssäule im AWS Well-Architected Framework. Weitere Informationen finden Sie unter Datenklassifizierung.

Datendrift

Eine signifikante Abweichung zwischen den Produktionsdaten und den Daten, die zum Trainieren eines ML-Modells verwendet wurden, oder eine signifikante Änderung der Eingabedaten im Laufe der Zeit. Datendrift kann die Gesamtqualität, Genauigkeit und Fairness von ML-Modellvorhersagen beeinträchtigen.

Daten während der Übertragung

Daten, die sich aktiv durch Ihr Netzwerk bewegen, z. B. zwischen Netzwerkressourcen.

Datennetz

Ein architektonisches Framework, das verteilte, dezentrale Dateneigentum mit zentraler Verwaltung und Steuerung ermöglicht.

Datenminimierung

Das Prinzip, nur die Daten zu sammeln und zu verarbeiten, die unbedingt erforderlich sind. Durch Datenminimierung im AWS Cloud können Datenschutzrisiken, Kosten und der CO2-Fußabdruck Ihrer Analysen reduziert werden.

Datenperimeter

Eine Reihe präventiver Schutzmaßnahmen in Ihrer AWS Umgebung, die sicherstellen, dass nur vertrauenswürdige Identitäten auf vertrauenswürdige Ressourcen von erwarteten Netzwerken zugreifen. Weitere Informationen finden Sie unter Aufbau eines Datenperimeters auf. AWS

Vorverarbeitung der Daten

Rohdaten in ein Format umzuwandeln, das von Ihrem ML-Modell problemlos verarbeitet werden kann. Die Vorverarbeitung von Daten kann bedeuten, dass bestimmte Spalten oder Zeilen entfernt und fehlende, inkonsistente oder doppelte Werte behoben werden.

Herkunft der Daten

Der Prozess der Nachverfolgung des Ursprungs und der Geschichte von Daten während ihres gesamten Lebenszyklus, z. B. wie die Daten generiert, übertragen und gespeichert wurden.

betroffene Person

Eine Person, deren Daten gesammelt und verarbeitet werden.

Data Warehouse

Ein Datenverwaltungssystem, das Business Intelligence wie Analysen unterstützt. Data Warehouses enthalten in der Regel große Mengen an historischen Daten und werden in der Regel für Abfragen und Analysen verwendet.

Datenbankdefinitionssprache (DDL)

Anweisungen oder Befehle zum Erstellen oder Ändern der Struktur von Tabellen und Objekten in einer Datenbank.

Datenbankmanipulationssprache (DML)

Anweisungen oder Befehle zum Ändern (Einfügen, Aktualisieren und Löschen) von Informationen in einer Datenbank.

DDL

Siehe Datenbankdefinitionssprache.

Deep-Ensemble

Mehrere Deep-Learning-Modelle zur Vorhersage kombinieren. Sie können Deep-Ensembles verwenden, um eine genauere Vorhersage zu erhalten oder um die Unsicherheit von Vorhersagen abzuschätzen.

Deep Learning

Ein ML-Teilbereich, der mehrere Schichten künstlicher neuronaler Netzwerke verwendet, um die Zuordnung zwischen Eingabedaten und Zielvariablen von Interesse zu ermitteln.

defense-in-depth

Ein Ansatz zur Informationssicherheit, bei dem eine Reihe von Sicherheitsmechanismen und -kontrollen sorgfältig in einem Computernetzwerk verteilt werden, um die Vertraulichkeit, Integrität und Verfügbarkeit des Netzwerks und der darin enthaltenen Daten zu schützen. Wenn Sie diese Strategie anwenden AWS, fügen Sie mehrere Steuerelemente auf verschiedenen Ebenen der AWS Organizations Struktur hinzu, um die Ressourcen zu schützen. Ein defense-in-depth Ansatz könnte beispielsweise Multi-Faktor-Authentifizierung, Netzwerksegmentierung und Verschlüsselung kombinieren.

delegierter Administrator

In AWS Organizations kann ein kompatibler Dienst ein AWS Mitgliedskonto registrieren, um die Konten der Organisation und die Berechtigungen für diesen Dienst zu verwalten. Dieses Konto wird als delegierter Administrator für diesen Service bezeichnet. Weitere Informationen und eine Liste kompatibler Services finden Sie unter Services, die mit AWS Organizations funktionieren in der AWS Organizations -Dokumentation.

Bereitstellung

Der Prozess, bei dem eine Anwendung, neue Feature oder Codekorrekturen in der Zielumgebung verfügbar gemacht werden. Die Bereitstellung umfasst das Implementieren von Änderungen an einer Codebasis und das anschließende Erstellen und Ausführen dieser Codebasis in den Anwendungsumgebungen.

Entwicklungsumgebung

Siehe Umgebung.

Detektivische Kontrolle

Eine Sicherheitskontrolle, die darauf ausgelegt ist, ein Ereignis zu erkennen, zu protokollieren und zu warnen, nachdem ein Ereignis eingetreten ist. Diese Kontrollen stellen eine zweite Verteidigungslinie dar und warnen Sie vor Sicherheitsereignissen, bei denen die vorhandenen präventiven Kontrollen umgangen wurden. Weitere Informationen finden Sie unter Detektivische Kontrolle in Implementierung von Sicherheitskontrollen in AWS.

Abbildung des Wertstroms in der Entwicklung (DVSM)

Ein Prozess zur Identifizierung und Priorisierung von Einschränkungen, die sich negativ auf Geschwindigkeit und Qualität im Lebenszyklus der Softwareentwicklung auswirken. DVSM erweitert den Prozess der Wertstromanalyse, der ursprünglich für Lean-Manufacturing-Praktiken

konzipiert wurde. Es konzentriert sich auf die Schritte und Teams, die erforderlich sind, um durch den Softwareentwicklungsprozess Mehrwert zu schaffen und zu steigern.

digitaler Zwilling

Eine virtuelle Darstellung eines realen Systems, z. B. eines Gebäudes, einer Fabrik, einer Industrieanlage oder einer Produktionslinie. Digitale Zwillinge unterstützen vorausschauende Wartung, Fernüberwachung und Produktionsoptimierung.

Maßtabelle

In einem Sternschema eine kleinere Tabelle, die Datenattribute zu quantitativen Daten in einer Faktentabelle enthält. Bei Attributen von Dimensionstabellen handelt es sich in der Regel um Textfelder oder diskrete Zahlen, die sich wie Text verhalten. Diese Attribute werden häufig zum Einschränken von Abfragen, zum Filtern und zur Kennzeichnung von Ergebnismengen verwendet.

Katastrophe

Ein Ereignis, das verhindert, dass ein Workload oder ein System seine Geschäftsziele an seinem primären Einsatzort erfüllt. Diese Ereignisse können Naturkatastrophen, technische Ausfälle oder das Ergebnis menschlichen Handelns sein, z. B. unbeabsichtigte Fehlkonfigurationen oder Malware-Angriffe.

Notfallwiederherstellung (DR)

Die Strategie und der Prozess, die Sie zur Minimierung von Ausfallzeiten und Datenverlusten aufgrund einer Katastrophe anwenden. Weitere Informationen finden Sie unter Disaster Recovery von Workloads unter AWS: Wiederherstellung in der Cloud im AWS Well-Architected Framework.

DML

Siehe Sprache zur Datenbankmanipulation.

Domainorientiertes Design

Ein Ansatz zur Entwicklung eines komplexen Softwaresystems, bei dem seine Komponenten mit sich entwickelnden Domains oder Kerngeschäftszielen verknüpft werden, denen jede Komponente dient. Dieses Konzept wurde von Eric Evans in seinem Buch Domaingesteuertes Design: Bewältigen der Komplexität im Herzen der Software (Boston: Addison-Wesley Professional, 2003) vorgestellt. Informationen darüber, wie Sie domaingesteuertes Design mit dem Strangler-Fig-Muster verwenden können, finden Sie unter Schrittweises Modernisieren älterer Microsoft ASP.NET (ASMX)-Webservices mithilfe von Containern und Amazon API Gateway.

DR

Siehe Disaster Recovery.

Erkennung von Driften

Verfolgung von Abweichungen von einer Basiskonfiguration Sie können es beispielsweise verwenden, AWS CloudFormation um Abweichungen bei den Systemressourcen zu erkennen, oder Sie können AWS Control Tower damit Änderungen in Ihrer landing zone erkennen, die sich auf die Einhaltung von Governance-Anforderungen auswirken könnten.

DVSM

Siehe Abbildung des Wertstroms in der Entwicklung.

E

EDA

Siehe explorative Datenanalyse.

Edge-Computing

Die Technologie, die die Rechenleistung für intelligente Geräte an den Rändern eines IoT-Netzwerks erhöht. Im Vergleich zu Cloud Computing kann Edge Computing die Kommunikationslatenz reduzieren und die Reaktionszeit verbessern.

Verschlüsselung

Ein Rechenprozess, der Klartextdaten, die für Menschen lesbar sind, in Chiffretext umwandelt.

Verschlüsselungsschlüssel

Eine kryptografische Zeichenfolge aus zufälligen Bits, die von einem Verschlüsselungsalgorithmus generiert wird. Schlüssel können unterschiedlich lang sein, und jeder Schlüssel ist so konzipiert, dass er unvorhersehbar und einzigartig ist.

Endianismus

Die Reihenfolge, in der Bytes im Computerspeicher gespeichert werden. Big-Endian-Systeme speichern das höchstwertige Byte zuerst. Little-Endian-Systeme speichern das niedrigwertigste Byte zuerst.

Endpunkt

Siehe Service-Endpunkt.

Endpunkt-Services

Ein Service, den Sie in einer Virtual Private Cloud (VPC) hosten können, um ihn mit anderen Benutzern zu teilen. Sie können einen Endpunktdienst mit anderen AWS-Konten oder AWS Identity and Access Management (IAM AWS PrivateLink -) Prinzipalen erstellen und diesen Berechtigungen gewähren. Diese Konten oder Prinzipale können sich privat mit Ihrem Endpunktservice verbinden, indem sie Schnittstellen-VPC-Endpunkte erstellen. Weitere Informationen finden Sie unter Einen Endpunkt-Service erstellen in der Amazon Virtual Private Cloud (Amazon VPC)-Dokumentation.

Unternehmensressourcenplanung (ERP)

Ein System, das wichtige Geschäftsprozesse (wie Buchhaltung, MES und Projektmanagement) für ein Unternehmen automatisiert und verwaltet.

Envelope-Verschlüsselung

Der Prozess der Verschlüsselung eines Verschlüsselungsschlüssels mit einem anderen Verschlüsselungsschlüssel. Weitere Informationen finden Sie unter Envelope-Verschlüsselung in der AWS Key Management Service (AWS KMS) -Dokumentation.

Umgebung

Eine Instance einer laufenden Anwendung. Die folgenden Arten von Umgebungen sind beim Cloud-Computing üblich:

- Entwicklungsumgebung – Eine Instance einer laufenden Anwendung, die nur dem Kernteam zur Verfügung steht, das für die Wartung der Anwendung verantwortlich ist. Entwicklungsumgebungen werden verwendet, um Änderungen zu testen, bevor sie in höhere Umgebungen übertragen werden. Diese Art von Umgebung wird manchmal als Testumgebung bezeichnet.
- Niedrigere Umgebungen – Alle Entwicklungsumgebungen für eine Anwendung, z. B. solche, die für erste Builds und Tests verwendet wurden.
- Produktionsumgebung – Eine Instance einer laufenden Anwendung, auf die Endbenutzer zugreifen können. In einer CI/CD-Pipeline ist die Produktionsumgebung die letzte Bereitstellungsumgebung.

- Höhere Umgebungen – Alle Umgebungen, auf die auch andere Benutzer als das Kernentwicklungsteam zugreifen können. Dies kann eine Produktionsumgebung, Vorproduktionsumgebungen und Umgebungen für Benutzerakzeptanztests umfassen.

Epics

In der agilen Methodik sind dies funktionale Kategorien, die Ihnen helfen, Ihre Arbeit zu organisieren und zu priorisieren. Epics bieten eine allgemeine Beschreibung der Anforderungen und Implementierungsaufgaben. Zu den Sicherheitsthemen AWS von CAF gehören beispielsweise Identitäts- und Zugriffsmanagement, Detektivkontrollen, Infrastruktursicherheit, Datenschutz und Reaktion auf Vorfälle. Weitere Informationen zu Epics in der AWS - Migrationsstrategie finden Sie im Leitfaden zur Programm-Implementierung.

ERP

Siehe Enterprise Resource Planning.

Explorative Datenanalyse (EDA)

Der Prozess der Analyse eines Datensatzes, um seine Hauptmerkmale zu verstehen. Sie sammeln oder aggregieren Daten und führen dann erste Untersuchungen durch, um Muster zu finden, Anomalien zu erkennen und Annahmen zu überprüfen. EDA wird durchgeführt, indem zusammenfassende Statistiken berechnet und Datenvisualisierungen erstellt werden.

F

Faktentabelle

Die zentrale Tabelle in einem Sternschema. Sie speichert quantitative Daten über den Geschäftsbetrieb. In der Regel enthält eine Faktentabelle zwei Arten von Spalten: Spalten, die Kennzahlen enthalten, und Spalten, die einen Fremdschlüssel für eine Dimensionstabelle enthalten.

schnell scheitern

Eine Philosophie, die häufige und inkrementelle Tests verwendet, um den Entwicklungslebenszyklus zu verkürzen. Dies ist ein wichtiger Bestandteil eines agilen Ansatzes.

Grenze zur Fehlerisolierung

Dabei handelt es sich um eine Grenze AWS Cloud, z. B. eine Availability Zone AWS-Region, eine Steuerungsebene oder eine Datenebene, die die Auswirkungen eines Fehlers begrenzt und die

Widerstandsfähigkeit von Workloads verbessert. Weitere Informationen finden Sie unter Grenzen zur AWS Fehlerisolierung.

Feature-Zweig

Siehe Zweig.

Features

Die Eingabedaten, die Sie verwenden, um eine Vorhersage zu treffen. In einem Fertigungskontext könnten Feature beispielsweise Bilder sein, die regelmäßig von der Fertigungslinie aus aufgenommen werden.

Bedeutung der Feature

Wie wichtig ein Feature für die Vorhersagen eines Modells ist. Dies wird in der Regel als numerischer Wert ausgedrückt, der mit verschiedenen Techniken wie Shapley Additive Explanations (SHAP) und integrierten Gradienten berechnet werden kann. Weitere Informationen finden Sie unter Interpretierbarkeit von Modellen für maschinelles Lernen mit:AWS.

Featuretransformation

Daten für den ML-Prozess optimieren, einschließlich der Anreicherung von Daten mit zusätzlichen Quellen, der Skalierung von Werten oder der Extraktion mehrerer Informationssätze aus einem einzigen Datenfeld. Das ermöglicht dem ML-Modell, von den Daten profitieren. Wenn Sie beispielsweise das Datum „27.05.2021 00:15:37“ in „2021“, „Mai“, „Donnerstag“ und „15“ aufschlüsseln, können Sie dem Lernalgorithmus helfen, nuancierte Muster zu erlernen, die mit verschiedenen Datenkomponenten verknüpft sind.

FGAC

Weitere Informationen finden Sie unter detaillierter Zugriffskontrolle.

Feinkörnige Zugriffskontrolle (FGAC)

Die Verwendung mehrerer Bedingungen, um eine Zugriffsanfrage zuzulassen oder abzulehnen.

Flash-Cut-Migration

Eine Datenbankmigrationsmethode, bei der eine kontinuierliche Datenreplikation durch Erfassung von Änderungsdaten verwendet wird, um Daten in kürzester Zeit zu migrieren, anstatt einen schrittweisen Ansatz zu verwenden. Ziel ist es, Ausfallzeiten auf ein Minimum zu beschränken.

G

Geoblocking

Siehe geografische Einschränkungen.

Geografische Einschränkungen (Geoblocking)

Bei Amazon eine Option CloudFront, um zu verhindern, dass Benutzer in bestimmten Ländern auf Inhaltsverteilungen zugreifen. Sie können eine Zulassungsliste oder eine Sperrliste verwenden, um zugelassene und gesperrte Länder anzugeben. Weitere Informationen finden Sie in der Dokumentation unter Beschränkung der geografischen Verteilung Ihrer Inhalte. CloudFront

Gitflow-Workflow

Ein Ansatz, bei dem niedrigere und höhere Umgebungen unterschiedliche Zweige in einem Quellcode-Repository verwenden. Der Gitflow-Workflow gilt als veraltet, und der Trunk-basierte Workflow ist der moderne, bevorzugte Ansatz.

Greenfield-Strategie

Das Fehlen vorhandener Infrastruktur in einer neuen Umgebung. Bei der Einführung einer Neuausrichtung einer Systemarchitektur können Sie alle neuen Technologien ohne Einschränkung der Kompatibilität mit der vorhandenen Infrastruktur auswählen, auch bekannt als Brownfield. Wenn Sie die bestehende Infrastruktur erweitern, könnten Sie Brownfield- und Greenfield-Strategien mischen.

Integritätsschutz

Eine allgemeine Regel, die dabei hilft, Ressourcen, Richtlinien und die Einhaltung von Vorschriften in allen Organisationseinheiten (OUs) zu regeln. Präventiver Integritätsschutz setzt Richtlinien durch, um die Einhaltung von Standards zu gewährleisten. Sie werden mithilfe von Service-Kontrollrichtlinien und IAM-Berechtigungsgrenzen implementiert. Detektivischer Integritätsschutz erkennt Richtlinienverstöße und Compliance-Probleme und generiert Warnmeldungen zur Abhilfe. Sie werden mithilfe von AWS Config, AWS Security Hub, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector und benutzerdefinierten AWS Lambda Prüfungen implementiert.

H

HEKTAR

Siehe Hochverfügbarkeit.

Heterogene Datenbankmigration

Migrieren Sie Ihre Quelldatenbank in eine Zieldatenbank, die eine andere Datenbank-Engine verwendet (z. B. Oracle zu Amazon Aurora). Eine heterogene Migration ist in der Regel Teil einer Neuarchitektur, und die Konvertierung des Schemas kann eine komplexe Aufgabe sein. AWS bietet AWS SCT, welches bei Schemakonvertierungen hilft.

hohe Verfügbarkeit (HA)

Die Fähigkeit eines Workloads, im Falle von Herausforderungen oder Katastrophen kontinuierlich und ohne Eingreifen zu arbeiten. HA-Systeme sind so konzipiert, dass sie automatisch ein Failover durchführen, gleichbleibend hohe Leistung bieten und unterschiedliche Lasten und Ausfälle mit minimalen Leistungseinbußen bewältigen.

historische Modernisierung

Ein Ansatz zur Modernisierung und Aufrüstung von Betriebstechnologiesystemen (OT), um den Bedürfnissen der Fertigungsindustrie besser gerecht zu werden. Ein Historian ist eine Art von Datenbank, die verwendet wird, um Daten aus verschiedenen Quellen in einer Fabrik zu sammeln und zu speichern.

Homogene Datenbankmigration

Migrieren Sie Ihre Quelldatenbank zu einer Zieldatenbank, die dieselbe Datenbank-Engine verwendet (z. B. Microsoft SQL Server zu Amazon RDS für SQL Server). Eine homogene Migration ist in der Regel Teil eines Hostwechsels oder eines Plattformwechsels. Sie können native Datenbankserviceprogramme verwenden, um das Schema zu migrieren.

heiße Daten

Daten, auf die häufig zugegriffen wird, z. B. Echtzeitdaten oder aktuelle Translationsdaten. Für diese Daten ist in der Regel eine leistungsstarke Speicherebene oder -klasse erforderlich, um schnelle Abfrageantworten zu ermöglichen.

Hotfix

Eine dringende Lösung für ein kritisches Problem in einer Produktionsumgebung. Aufgrund seiner Dringlichkeit wird ein Hotfix normalerweise außerhalb des typischen DevOps Release-Workflows erstellt.

Hypercare-Phase

Unmittelbar nach dem Cutover, der Zeitraum, in dem ein Migrationsteam die migrierten Anwendungen in der Cloud verwaltet und überwacht, um etwaige Probleme zu beheben. In der Regel dauert dieser Zeitraum 1–4 Tage. Am Ende der Hypercare-Phase überträgt das Migrationsteam in der Regel die Verantwortung für die Anwendungen an das Cloud-Betriebsteam.

I

IaC

Sehen Sie sich Infrastruktur als Code an.

Identitätsbasierte Richtlinie

Eine Richtlinie, die einem oder mehreren IAM-Prinzipalen zugeordnet ist und deren Berechtigungen innerhalb der AWS Cloud Umgebung definiert.

Leerlaufanwendung

Eine Anwendung mit einer durchschnittlichen CPU- und Arbeitsspeicherauslastung zwischen 5 und 20 Prozent über einen Zeitraum von 90 Tagen. In einem Migrationsprojekt ist es üblich, diese Anwendungen außer Betrieb zu nehmen oder sie On-Premises beizubehalten.

IIoT

Siehe Industrielles Internet der Dinge.

unveränderliche Infrastruktur

Ein Modell, das eine neue Infrastruktur für Produktionsworkloads bereitstellt, anstatt die bestehende Infrastruktur zu aktualisieren, zu patchen oder zu modifizieren. Unveränderliche Infrastrukturen sind von Natur aus konsistenter, zuverlässiger und vorhersehbarer als veränderliche Infrastrukturen. Weitere Informationen finden Sie in der Best Practice Deploy using immutable infrastructure im AWS Well-Architected Framework.

Eingehende (ingress) VPC

In einer Architektur AWS mit mehreren Konten ist dies eine VPC, die Netzwerkverbindungen von außerhalb einer Anwendung akzeptiert, überprüft und weiterleitet. Die AWS -Referenzarchitektur für die Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden VPCs und Inspektions-VPCs einzurichten, um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet zu schützen.

Inkrementelle Migration

Eine Cutover-Strategie, bei der Sie Ihre Anwendung in kleinen Teilen migrieren, anstatt eine einziges vollständiges Cutover durchzuführen. Beispielsweise könnten Sie zunächst nur einige Microservices oder Benutzer auf das neue System umstellen. Nachdem Sie sich vergewissert haben, dass alles ordnungsgemäß funktioniert, können Sie weitere Microservices oder Benutzer schrittweise verschieben, bis Sie Ihr Legacy-System außer Betrieb nehmen können. Diese Strategie reduziert die mit großen Migrationen verbundenen Risiken.

Industrie 4.0

Ein Begriff, der 2016 von Klaus Schwab eingeführt wurde und sich auf die Modernisierung von Fertigungsprozessen durch Fortschritte in den Bereichen Konnektivität, Echtzeitdaten, Automatisierung, Analytik und KI/ML bezieht.

Infrastruktur

Alle Ressourcen und Komponenten, die in der Umgebung einer Anwendung enthalten sind.

Infrastructure as Code (IaC)

Der Prozess der Bereitstellung und Verwaltung der Infrastruktur einer Anwendung mithilfe einer Reihe von Konfigurationsdateien. IaC soll Ihnen helfen, das Infrastrukturmanagement zu zentralisieren, Ressourcen zu standardisieren und schnell zu skalieren, sodass neue Umgebungen wiederholbar, zuverlässig und konsistent sind.

Industrielles Internet der Dinge (IIoT)

Einsatz von mit dem Internet verbundenen Sensoren und Geräten in Industriesektoren wie Fertigung, Energie, Automobilindustrie, Gesundheitswesen, Biowissenschaften und Landwirtschaft. Mehr Informationen finden Sie unter Aufbau einer digitalen Transformationsstrategie für das industrielle Internet der Dinge (IIoT).

Inspektions-VPC

In einer Architektur AWS mit mehreren Konten eine zentralisierte VPC, die Inspektionen des Netzwerkverkehrs zwischen VPCs (in derselben oder unterschiedlichen AWS-Regionen), dem Internet und lokalen Netzwerken verwaltet. Die AWS -Referenzarchitektur für die Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden VPCs und Inspektions-VPCs einzurichten, um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet zu schützen.

Internet of Things (IoT)

Das Netzwerk verbundener physischer Objekte mit eingebetteten Sensoren oder Prozessoren, das über das Internet oder über ein lokales Kommunikationsnetzwerk mit anderen Geräten und Systemen kommuniziert. Weitere Informationen finden Sie unter Was ist IoT?

Interpretierbarkeit

Ein Merkmal eines Modells für Machine Learning, das beschreibt, inwieweit ein Mensch verstehen kann, wie die Vorhersagen des Modells von seinen Eingaben abhängen. Weitere Informationen finden Sie unter Interpretierbarkeit von Modellen für Machine Learning mit AWS.

IoT

Siehe Internet der Dinge.

IT information library (ITIL, IT-Informationsbibliothek)

Eine Reihe von bewährten Methoden für die Bereitstellung von IT-Services und die Abstimmung dieser Services auf die Geschäftsanforderungen. ITIL bietet die Grundlage für ITSM.

T service management (ITSM, IT-Servicemanagement)

Aktivitäten im Zusammenhang mit der Gestaltung, Implementierung, Verwaltung und Unterstützung von IT-Services für eine Organisation. Informationen zur Integration von Cloud-Vorgängen mit ITSM-Tools finden Sie im Leitfaden zur Betriebsintegration.

BIS

Weitere Informationen finden Sie in der IT-Informationsbibliothek.

ITSM

Siehe IT-Servicemanagement.

L

Labelbasierte Zugangskontrolle (LBAC)

Eine Implementierung der Mandatory Access Control (MAC), bei der den Benutzern und den Daten selbst jeweils explizit ein Sicherheitslabelwert zugewiesen wird. Die Schnittmenge zwischen der Benutzersicherheitsbeschriftung und der Datensicherheitsbeschriftung bestimmt, welche Zeilen und Spalten für den Benutzer sichtbar sind.

Landing Zone

Eine landing zone ist eine gut strukturierte AWS Umgebung mit mehreren Konten, die skalierbar und sicher ist. Dies ist ein Ausgangspunkt, von dem aus Ihre Organisationen Workloads und Anwendungen schnell und mit Vertrauen in ihre Sicherheits- und Infrastrukturumgebung starten und bereitstellen können. Weitere Informationen zu Landing Zones finden Sie unter Einrichtung einer sicheren und skalierbaren AWS -Umgebung mit mehreren Konten..

Große Migration

Eine Migration von 300 oder mehr Servern.

SCHWARZ

Weitere Informationen finden Sie unter Label-basierte Zugriffskontrolle.

Geringste Berechtigung

Die bewährte Sicherheitsmethode, bei der nur die für die Durchführung einer Aufgabe erforderlichen Mindestberechtigungen erteilt werden. Weitere Informationen finden Sie unter Geringste Berechtigungen anwenden in der IAM-Dokumentation.

Lift and Shift

Siehe 7 Rs.

Little-Endian-System

Ein System, welches das niedrigwertigste Byte zuerst speichert. Siehe auch Endianness.

Niedrigere Umgebungen

Siehe Umwelt.

M

Machine Learning (ML)

Eine Art künstlicher Intelligenz, die Algorithmen und Techniken zur Mustererkennung und zum Lernen verwendet. ML analysiert aufgezeichnete Daten, wie z. B. Daten aus dem Internet der Dinge (IoT), und lernt daraus, um ein statistisches Modell auf der Grundlage von Mustern zu erstellen. Weitere Informationen finden Sie unter Machine Learning.

Hauptzweig

Siehe Filiale.

Malware

Software, die entwickelt wurde, um die Computersicherheit oder den Datenschutz zu gefährden. Malware kann Computersysteme stören, vertrauliche Informationen durchsickern lassen oder sich unbefugten Zugriff verschaffen. Beispiele für Malware sind Viren, Würmer, Ransomware, Trojaner, Spyware und Keylogger.

verwaltete Dienste

AWS -Services für die die Infrastrukturebene, das Betriebssystem und die Plattformen AWS betrieben werden, und Sie greifen auf die Endgeräte zu, um Daten zu speichern und abzurufen. Amazon Simple Storage Service (Amazon S3) und Amazon DynamoDB sind Beispiele für Managed Services. Diese werden auch als abstrakte Dienste bezeichnet.

Manufacturing Execution System (MES)

Ein Softwaresystem zur Nachverfolgung, Überwachung, Dokumentation und Steuerung von Produktionsprozessen, bei denen Rohstoffe in der Fertigung zu fertigen Produkten umgewandelt werden.

MAP

Siehe Migration Acceleration Program.

Mechanismus

Ein vollständiger Prozess, bei dem Sie ein Tool erstellen, die Akzeptanz des Tools vorantreiben und anschließend die Ergebnisse überprüfen, um Anpassungen vorzunehmen. Ein Mechanismus ist ein Zyklus, der sich im Laufe seiner Tätigkeit selbst verstärkt und verbessert. Weitere Informationen finden Sie unter Aufbau von Mechanismen im AWS Well-Architected Framework.

Mitgliedskonto

Alle AWS-Konten außer dem Verwaltungskonto, die Teil einer Organisation in sind. AWS Organizations Ein Konto kann jeweils nur einer Organisation angehören.

DURCHEINANDER

Siehe Manufacturing Execution System.

Message Queuing-Telemetrietransport (MQTT)

Ein leichtes machine-to-machine (M2M) -Kommunikationsprotokoll, das auf dem Publish/Subscribe-Muster für IoT-Geräte mit beschränkten Ressourcen basiert.

Microservice

Ein kleiner, unabhängiger Service, der über klar definierte APIs kommuniziert und in der Regel kleinen, eigenständigen Teams gehört. Ein Versicherungssystem kann beispielsweise Microservices beinhalten, die Geschäftsfunktionen wie Vertrieb oder Marketing oder Subdomains wie Einkauf, Schadenersatz oder Analytik zugeordnet sind. Zu den Vorteilen von Microservices gehören Agilität, flexible Skalierung, einfache Bereitstellung, wiederverwendbarer Code und Ausfallsicherheit. Weitere Informationen finden Sie unter Integration von Microservices mithilfe serverloser Dienste. AWS

Microservices-Architekturen

Ein Ansatz zur Erstellung einer Anwendung mit unabhängigen Komponenten, die jeden Anwendungsprozess als Microservice ausführen. Diese Microservices kommunizieren über eine klar definierte Schnittstelle mithilfe einfacher APIs. Jeder Microservice in dieser Architektur kann aktualisiert, bereitgestellt und skaliert werden, um den Bedarf an bestimmten Funktionen einer Anwendung zu decken. Weitere Informationen finden Sie unter Implementierung von Microservices auf. AWS

Migration Acceleration Program (MAP)

Ein AWS Programm, das Beratung, Unterstützung, Schulungen und Services bietet, um Unternehmen dabei zu unterstützen, eine solide betriebliche Grundlage für die Umstellung auf die Cloud zu schaffen und die anfänglichen Kosten von Migrationen auszugleichen. MAP umfasst eine Migrationsmethode für die methodische Durchführung von Legacy-Migrationen sowie eine Reihe von Tools zur Automatisierung und Beschleunigung gängiger Migrationsszenarien.

Migration in großem Maßstab

Der Prozess, bei dem der Großteil des Anwendungsportfolios in Wellen in die Cloud verlagert wird, wobei in jeder Welle mehr Anwendungen schneller migriert werden. In dieser Phase werden die bewährten Verfahren und Erkenntnisse aus den früheren Phasen zur Implementierung einer Migrationsfabrik von Teams, Tools und Prozessen zur Optimierung der Migration von Workloads durch Automatisierung und agile Bereitstellung verwendet. Dies ist die dritte Phase der AWS - Migrationsstrategie.

Migrationsfabrik

Funktionsübergreifende Teams, die die Migration von Workloads durch automatisierte, agile Ansätze optimieren. Zu den Teams in der Migrationsabteilung gehören in der Regel Betriebsabläufe, Geschäftsanalysten und Eigentümer, Migrationsingenieure, Entwickler und DevOps Experten, die in Sprints arbeiten. Zwischen 20 und 50 Prozent eines Unternehmensanwendungsportfolios bestehen aus sich wiederholenden Mustern, die durch einen Fabrik-Ansatz optimiert werden können. Weitere Informationen finden Sie in Diskussion über Migrationsfabriken und den Leitfaden zur Cloud-Migration-Fabrik in diesem Inhaltssatz.

Migrationsmetadaten

Die Informationen über die Anwendung und den Server, die für den Abschluss der Migration benötigt werden. Für jedes Migrationsmuster ist ein anderer Satz von Migrationsmetadaten erforderlich. Beispiele für Migrationsmetadaten sind das Zielsubnetz, die Sicherheitsgruppe und AWS das Konto.

Migrationsmuster

Eine wiederholbare Migrationsaufgabe, in der die Migrationsstrategie, das Migrationsziel und die verwendete Migrationsanwendung oder der verwendete Migrationsservice detailliert beschrieben werden. Beispiel: Rehost-Migration zu Amazon EC2 mit AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

Ein Online-Tool, das Informationen zur Validierung des Geschäftsszenarios für die Migration auf das bereitstellt. AWS Cloud MPA bietet eine detaillierte Portfoliobewertung (richtige Servergröße, Preisgestaltung, Gesamtbetriebskostenanalyse, Migrationskostenanalyse) sowie Migrationsplanung (Anwendungsdatenanalyse und Datenerfassung, Anwendungsgruppierung, Migrationspriorisierung und Wellenplanung). Das MPA-Tool (Anmeldung erforderlich) steht allen AWS Beratern und APN-Partnerberatern kostenlos zur Verfügung.

Migration Readiness Assessment (MRA)

Der Prozess, bei dem mithilfe des AWS CAF Erkenntnisse über den Cloud-Bereitschaftsstatus eines Unternehmens gewonnen, Stärken und Schwächen identifiziert und ein Aktionsplan zur Schließung festgestellter Lücken erstellt wird. Weitere Informationen finden Sie im Benutzerhandbuch für Migration Readiness. MRA ist die erste Phase der AWS - Migrationsstrategie.

Migrationsstrategie

Der Ansatz, der verwendet wurde, um einen Workload auf den AWS Cloud zu migrieren. Weitere Informationen finden Sie im Eintrag 7 Rs in diesem Glossar und unter Mobilisieren Sie Ihr Unternehmen, um umfangreiche Migrationen zu beschleunigen.

ML

Siehe maschinelles Lernen.

Modernisierung

Umwandlung einer veralteten (veralteten oder monolithischen) Anwendung und ihrer Infrastruktur in ein agiles, elastisches und hochverfügbares System in der Cloud, um Kosten zu senken, die Effizienz zu steigern und Innovationen zu nutzen. Weitere Informationen finden Sie unter Strategie zur Modernisierung von Anwendungen in der AWS Cloud.

Bewertung der Modernisierungsfähigkeit

Eine Bewertung, anhand derer festgestellt werden kann, ob die Anwendungen einer Organisation für die Modernisierung bereit sind, Vorteile, Risiken und Abhängigkeiten identifiziert und ermittelt wird, wie gut die Organisation den zukünftigen Status dieser Anwendungen unterstützen kann. Das Ergebnis der Bewertung ist eine Vorlage der Zielarchitektur, eine Roadmap, in der die Entwicklungsphasen und Meilensteine des Modernisierungsprozesses detailliert beschrieben werden, sowie ein Aktionsplan zur Behebung festgestellter Lücken. Weitere Informationen finden Sie unter Evaluierung der Modernisierungsbereitschaft von Anwendungen in der AWS Cloud.

Monolithische Anwendungen (Monolithen)

Anwendungen, die als ein einziger Service mit eng gekoppelten Prozessen ausgeführt werden. Monolithische Anwendungen haben verschiedene Nachteile. Wenn ein Anwendungs-Feature stark nachgefragt wird, muss die gesamte Architektur skaliert werden. Das Hinzufügen oder Verbessern der Feature einer monolithischen Anwendung wird ebenfalls komplexer, wenn die Codebasis wächst. Um diese Probleme zu beheben, können Sie eine Microservices-Architektur verwenden. Weitere Informationen finden Sie unter Zerlegen von Monolithen in Microservices.

MPA

Siehe Bewertung des Migrationsportfolios.

MQTT

Siehe Message Queuing-Telemetrietransport.

Mehrklassen-Klassifizierung

Ein Prozess, der dabei hilft, Vorhersagen für mehrere Klassen zu generieren (wobei eines von mehr als zwei Ergebnissen vorhergesagt wird). Ein ML-Modell könnte beispielsweise fragen: „Ist dieses Produkt ein Buch, ein Auto oder ein Telefon?“ oder „Welche Kategorie von Produkten ist für diesen Kunden am interessantesten?“

veränderbare Infrastruktur

Ein Modell, das die bestehende Infrastruktur für Produktionsworkloads aktualisiert und modifiziert. Für eine verbesserte Konsistenz, Zuverlässigkeit und Vorhersagbarkeit empfiehlt das AWS Well-Architected Framework die Verwendung einer unveränderlichen Infrastruktur als bewährte Methode.

O

OAC

Weitere Informationen finden Sie unter Origin Access Control.

OAI

Siehe Zugriffsidentität von Origin.

COM

Siehe organisatorisches Change-Management.

Offline-Migration

Eine Migrationsmethode, bei der der Quell-Workload während des Migrationsprozesses heruntergefahren wird. Diese Methode ist mit längeren Ausfallzeiten verbunden und wird in der Regel für kleine, unkritische Workloads verwendet.

OI

Siehe Betriebsintegration.

OLA

Siehe Vereinbarung auf operativer Ebene.

Online-Migration

Eine Migrationsmethode, bei der der Quell-Workload auf das Zielsystem kopiert wird, ohne offline genommen zu werden. Anwendungen, die mit dem Workload verbunden sind, können während der Migration weiterhin funktionieren. Diese Methode beinhaltet keine bis minimale Ausfallzeit und wird in der Regel für kritische Produktionsworkloads verwendet.

OPC-UA

Siehe Open Process Communications — Unified Architecture.

Offene Prozesskommunikation — Einheitliche Architektur (OPC-UA)

Ein machine-to-machine (M2M) -Kommunikationsprotokoll für die industrielle Automatisierung. OPC-UA bietet einen Interoperabilitätsstandard mit Datenverschlüsselungs-, Authentifizierungs- und Autorisierungsschemata.

Vereinbarung auf Betriebsebene (OLA)

Eine Vereinbarung, in der klargestellt wird, welche funktionalen IT-Gruppen sich gegenseitig versprechen zu liefern, um ein Service Level Agreement (SLA) zu unterstützen.

Überprüfung der Betriebsbereitschaft (ORR)

Eine Checkliste mit Fragen und zugehörigen bewährten Methoden, die Ihnen helfen, Vorfälle und mögliche Ausfälle zu verstehen, zu bewerten, zu verhindern oder deren Umfang zu reduzieren. Weitere Informationen finden Sie unter Operational Readiness Reviews (ORR) im AWS Well-Architected Framework.

Betriebstechnologie (OT)

Hardware- und Softwaresysteme, die mit der physischen Umgebung zusammenarbeiten, um industrielle Abläufe, Ausrüstung und Infrastruktur zu steuern. In der Fertigung ist die Integration von OT- und Informationstechnologie (IT) -Systemen ein zentraler Schwerpunkt der Industrie 4.0-Transformationen.

Betriebsintegration (OI)

Der Prozess der Modernisierung von Abläufen in der Cloud, der Bereitschaftsplanung, Automatisierung und Integration umfasst. Weitere Informationen finden Sie im Leitfaden zur Betriebsintegration.

Organisationspfad

Ein Pfad, der von erstellt wird und in AWS CloudTrail dem alle Ereignisse für alle AWS-Konten in einer Organisation protokolliert werden. AWS Organizations Diese Spur wird in jedem AWS-Konto , der Teil der Organisation ist, erstellt und verfolgt die Aktivität in jedem Konto. Weitere Informationen finden Sie in der CloudTrail Dokumentation unter Erstellen eines Pfads für eine Organisation.

Organisatorisches Veränderungsmanagement (OCM)

Ein Framework für das Management wichtiger, disruptiver Geschäftstransformationen aus Sicht der Mitarbeiter, der Kultur und der Führung. OCM hilft Organisationen dabei, sich auf neue Systeme und Strategien vorzubereiten und auf diese umzustellen, indem es die Akzeptanz von Veränderungen beschleunigt, Übergangsprobleme angeht und kulturelle und organisatorische Veränderungen vorantreibt. In der AWS Migrationsstrategie wird dieses Framework aufgrund der Geschwindigkeit des Wandels, der bei Projekten zur Cloud-Einführung erforderlich ist, als Mitarbeiterbeschleunigung bezeichnet. Weitere Informationen finden Sie im OCM-Handbuch.

Ursprungszugriffskontrolle (OAC)

In CloudFront, eine erweiterte Option zur Zugriffsbeschränkung, um Ihre Amazon Simple Storage Service (Amazon S3) -Inhalte zu sichern. OAC unterstützt alle S3-Buckets insgesamt AWS-Regionen, serverseitige Verschlüsselung mit AWS KMS (SSE-KMS) sowie dynamische `PUT` und `DELETE` Anfragen an den S3-Bucket.

Ursprungszugriffsidentität (OAI)

In CloudFront, eine Option zur Zugriffsbeschränkung, um Ihre Amazon S3 S3-Inhalte zu sichern. Wenn Sie OAI verwenden, CloudFront erstellt es einen Principal, mit dem sich Amazon S3 authentifizieren kann. Authentifizierte Principals können nur über eine bestimmte Distribution auf Inhalte in einem S3-Bucket zugreifen. CloudFront Siehe auch OAC, das eine detailliertere und verbesserte Zugriffskontrolle bietet.

ODER

Siehe Überprüfung der Betriebsbereitschaft.

NICHT

Siehe Betriebstechnologie.

Ausgehende (egress) VPC

In einer Architektur AWS mit mehreren Konten eine VPC, die Netzwerkverbindungen verarbeitet, die von einer Anwendung aus initiiert werden. Die AWS -Referenzarchitektur für die Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden VPCs und Inspektions-VPCs einzurichten, um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet zu schützen.

P

Berechtigungsgrenze

Eine IAM-Verwaltungsrichtlinie, die den IAM-Prinzipalen zugeordnet ist, um die maximalen Berechtigungen festzulegen, die der Benutzer oder die Rolle haben kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitys in der IAM-Dokumentation.

persönlich identifizierbare Informationen (PII)

Informationen, die, wenn sie direkt betrachtet oder mit anderen verwandten Daten kombiniert werden, verwendet werden können, um vernünftige Rückschlüsse auf die Identität einer Person zu ziehen. Beispiele für personenbezogene Daten sind Namen, Adressen und Kontaktinformationen.

Personenbezogene Daten

Siehe persönlich identifizierbare Informationen.

Playbook

Eine Reihe vordefinierter Schritte, die die mit Migrationen verbundenen Aufgaben erfassen, z. B. die Bereitstellung zentraler Betriebsfunktionen in der Cloud. Ein Playbook kann die Form von Skripten, automatisierten Runbooks oder einer Zusammenfassung der Prozesse oder Schritte annehmen, die für den Betrieb Ihrer modernisierten Umgebung erforderlich sind.

PLC

Siehe programmierbare Logiksteuerung.

PLM

Siehe Produktlebenszyklusmanagement.

policy

Ein Objekt, das Berechtigungen definieren (siehe identitätsbasierte Richtlinie), Zugriffsbedingungen spezifizieren (siehe ressourcenbasierte Richtlinie) oder die maximalen Berechtigungen für alle Konten in einer Organisation definieren kann AWS Organizations (siehe Dienststeuerungsrichtlinie).

Polyglotte Beharrlichkeit

Unabhängige Auswahl der Datenspeichertechnologie eines Microservices auf der Grundlage von Datenzugriffsmustern und anderen Anforderungen. Wenn Ihre Microservices über dieselbe Datenspeichertechnologie verfügen, kann dies zu Implementierungsproblemen oder zu Leistungseinbußen führen. Microservices lassen sich leichter implementieren und erzielen eine bessere Leistung und Skalierbarkeit, wenn sie den Datenspeicher verwenden, der ihren Anforderungen am besten entspricht. Weitere Informationen finden Sie unter Datenpersistenz in Microservices aktivieren.

Portfoliobewertung

Ein Prozess, bei dem das Anwendungsportfolio ermittelt, analysiert und priorisiert wird, um die Migration zu planen. Weitere Informationen finden Sie in Bewerten der Migrationsbereitschaft.

predicate

Eine Abfragebedingung, die `true` oder zurückgibt`false`, was üblicherweise in einer Klausel vorkommt. `WHERE`

Prädikat Pushdown

Eine Technik zur Optimierung von Datenbankabfragen, bei der die Daten in der Abfrage vor der Übertragung gefiltert werden. Dadurch wird die Datenmenge reduziert, die aus der relationalen Datenbank abgerufen und verarbeitet werden muss, und die Abfrageleistung wird verbessert.

Präventive Kontrolle

Eine Sicherheitskontrolle, die verhindern soll, dass ein Ereignis eintritt. Diese Kontrollen stellen eine erste Verteidigungslinie dar, um unbefugten Zugriff oder unerwünschte Änderungen an Ihrem Netzwerk zu verhindern. Weitere Informationen finden Sie unter Präventive Kontrolle in Implementierung von Sicherheitskontrollen in AWS.

Prinzipal

Eine Entität AWS , die Aktionen ausführen und auf Ressourcen zugreifen kann. Bei dieser Entität handelt es sich in der Regel um einen Root-Benutzer für eine AWS-Konto, eine IAM-Rolle oder

einen Benutzer. Weitere Informationen finden Sie unter Prinzipal in Rollenbegriffe und -konzepte in der IAM-Dokumentation.

Datenschutz durch Design

Ein Ansatz in der Systemtechnik, der den Datenschutz während des gesamten Engineering-Prozesses berücksichtigt.

Privat gehostete Zonen

Ein Container, der Informationen darüber enthält, wie Amazon Route 53 auf DNS-Abfragen für eine Domain und ihre Subdomains innerhalb einer oder mehrerer VPCs reagieren soll. Weitere Informationen finden Sie unter Arbeiten mit privat gehosteten Zonen in der Route-53-Dokumentation.

proaktive Steuerung

Eine Sicherheitskontrolle, die den Einsatz nicht richtlinienkonformer Ressourcen verhindern soll. Mit diesen Steuerelementen werden Ressourcen gescannt, bevor sie bereitgestellt werden. Wenn die Ressource nicht mit der Steuerung konform ist, wird sie nicht bereitgestellt. Weitere Informationen finden Sie im Referenzhandbuch zu Kontrollen in der AWS Control Tower Dokumentation und unter Proaktive Kontrollen unter Implementierung von Sicherheitskontrollen am AWS.

Produktlebenszyklusmanagement (PLM)

Das Management von Daten und Prozessen für ein Produkt während seines gesamten Lebenszyklus, vom Design, der Entwicklung und Markteinführung über Wachstum und Reife bis hin zur Markteinführung und Markteinführung.

Produktionsumgebung

Siehe Umgebung.

Speicherprogrammierbare Steuerung (SPS)

In der Fertigung ein äußerst zuverlässiger, anpassungsfähiger Computer, der Maschinen überwacht und Fertigungsprozesse automatisiert.

Pseudonymisierung

Der Prozess, bei dem persönliche Identifikatoren in einem Datensatz durch Platzhalterwerte ersetzt werden. Pseudonymisierung kann zum Schutz der Privatsphäre beitragen. Pseudonymisierte Daten gelten weiterhin als personenbezogene Daten.

veröffentlichen/abonnieren (pub/sub)

Ein Muster, das asynchrone Kommunikation zwischen Microservices ermöglicht, um die Skalierbarkeit und Reaktionsfähigkeit zu verbessern. In einem auf Microservices basierenden MES kann ein Microservice beispielsweise Ereignismeldungen in einem Kanal veröffentlichen, den andere Microservices abonnieren können. Das System kann neue Microservices hinzufügen, ohne den Veröffentlichungsservice zu ändern.

Q

Abfrageplan

Eine Reihe von Schritten, wie Anweisungen, die für den Zugriff auf die Daten in einem relationalen SQL-Datenbanksystem verwendet werden.

Abfrageplanregression

Wenn ein Datenbankserviceoptimierer einen weniger optimalen Plan wählt als vor einer bestimmten Änderung der Datenbankumgebung. Dies kann durch Änderungen an Statistiken, Beschränkungen, Umgebungseinstellungen, Abfrageparameter-Bindungen und Aktualisierungen der Datenbank-Engine verursacht werden.

R

RACI-Matrix

Siehe verantwortlich, rechenschaftspflichtig, konsultiert, informiert (RACI).

Ransomware

Eine bösartige Software, die entwickelt wurde, um den Zugriff auf ein Computersystem oder Daten zu blockieren, bis eine Zahlung erfolgt ist.

RASCI-Matrix

Siehe verantwortlich, rechenschaftspflichtig, konsultiert, informiert (RACI).

RCAC

Siehe Zugriffskontrolle für Zeilen und Spalten.

Read Replica

Eine Kopie einer Datenbank, die nur für Lesezwecke verwendet wird. Sie können Abfragen an das Lesereplikat weiterleiten, um die Belastung auf Ihrer Primärdatenbank zu reduzieren.

neu strukturieren

Siehe 7 Rs.

Recovery Point Objective (RPO)

Die maximal zulässige Zeitspanne seit dem letzten Datenwiederherstellungspunkt. Dies bestimmt, was als akzeptabler Datenverlust zwischen dem letzten Wiederherstellungspunkt und der Betriebsunterbrechung angesehen wird.

Wiederherstellungszeitziel (RTO)

Die maximal zulässige Verzögerung zwischen der Betriebsunterbrechung und der Wiederherstellung des Dienstes.

Refaktorierung

Siehe 7 Rs.

Region

Eine Sammlung von AWS Ressourcen in einem geografischen Gebiet. Jeder AWS-Region ist isoliert und unabhängig von den anderen, um Fehlertoleranz, Stabilität und Belastbarkeit zu gewährleisten. Weitere Informationen finden Sie unter Geben Sie an, was AWS-Regionen Ihr Konto verwenden kann.

Regression

Eine ML-Technik, die einen numerischen Wert vorhersagt. Zum Beispiel, um das Problem „Zu welchem Preis wird dieses Haus verkauft werden?“ zu lösen Ein ML-Modell könnte ein lineares Regressionsmodell verwenden, um den Verkaufspreis eines Hauses auf der Grundlage bekannter Fakten über das Haus (z. B. die Quadratmeterzahl) vorherzusagen.

rehosten

Siehe 7 Rs.

Veröffentlichung

In einem Bereitstellungsprozess der Akt der Förderung von Änderungen an einer Produktionsumgebung.

umziehen

Siehe 7 Rs.

neue Plattform

Siehe 7 Rs.

Rückkauf

Siehe 7 Rs.

Ausfallsicherheit

Die Fähigkeit einer Anwendung, Störungen zu widerstehen oder sich von ihnen zu erholen. Hochverfügbarkeit und Notfallwiederherstellung sind häufig Überlegungen bei der Planung der Ausfallsicherheit in der. AWS Cloud Weitere Informationen finden Sie unter AWS Cloud Resilienz.

Ressourcenbasierte Richtlinie

Eine mit einer Ressource verknüpfte Richtlinie, z. B. ein Amazon-S3-Bucket, ein Endpunkt oder ein Verschlüsselungsschlüssel. Diese Art von Richtlinie legt fest, welchen Prinzipalen der Zugriff gewährt wird, welche Aktionen unterstützt werden und welche anderen Bedingungen erfüllt sein müssen.

RACI-Matrix (verantwortlich, rechenschaftspflichtig, konsultiert, informiert)

Eine Matrix, die die Rollen und Verantwortlichkeiten aller an Migrationsaktivitäten und Cloud-Operationen beteiligten Parteien definiert. Der Matrixname leitet sich von den in der Matrix definierten Zuständigkeitstypen ab: verantwortlich (R), rechenschaftspflichtig (A), konsultiert (C) und informiert (I). Der Unterstützungstyp (S) ist optional. Wenn Sie Unterstützung einbeziehen, wird die Matrix als RASCI-Matrix bezeichnet, und wenn Sie sie ausschließen, wird sie als RACI-Matrix bezeichnet.

Reaktive Kontrolle

Eine Sicherheitskontrolle, die darauf ausgelegt ist, die Behebung unerwünschter Ereignisse oder Abweichungen von Ihren Sicherheitsstandards voranzutreiben. Weitere Informationen finden Sie unter Reaktive Kontrolle in Implementieren von Sicherheitskontrollen in AWS.

Beibehaltung

Siehe 7 Rs.

zurückziehen

Siehe 7 Rs.

Drehung

Der Vorgang, bei dem ein Geheimnis regelmäßig aktualisiert wird, um es einem Angreifer zu erschweren, auf die Anmeldeinformationen zuzugreifen.

Zugriffskontrolle für Zeilen und Spalten (RCAC)

Die Verwendung einfacher, flexibler SQL-Ausdrücke mit definierten Zugriffsregeln. RCAC besteht aus Zeilenberechtigungen und Spaltenmasken.

RPO

Siehe Recovery Point Objective.

RTO

Siehe Ziel der Wiederherstellungszeit.

Runbook

Eine Reihe manueller oder automatisierter Verfahren, die zur Ausführung einer bestimmten Aufgabe erforderlich sind. Diese sind in der Regel darauf ausgelegt, sich wiederholende Operationen oder Verfahren mit hohen Fehlerquoten zu rationalisieren.

S

SAML 2.0

Ein offener Standard, den viele Identitätsanbieter (IdPs) verwenden. Diese Funktion ermöglicht föderiertes Single Sign-On (SSO), sodass sich Benutzer bei den API-Vorgängen anmelden AWS Management Console oder die AWS API-Operationen aufrufen können, ohne dass Sie einen Benutzer in IAM für alle in Ihrer Organisation erstellen müssen. Weitere Informationen zum SAML-2.0.-basierten Verbund finden Sie unter Über den SAML-2.0-basierten Verbund in der IAM-Dokumentation.

SCADA

Siehe Aufsichtskontrolle und Datenerfassung.

SCP

Siehe Richtlinie zur Dienstkontrolle.

Secret

Interne AWS Secrets Manager, vertrauliche oder eingeschränkte Informationen, wie z. B. ein Passwort oder Benutzeranmeldeinformationen, die Sie in verschlüsselter Form speichern. Es besteht aus dem geheimen Wert und seinen Metadaten. Der geheime Wert kann binär, eine einzelne Zeichenfolge oder mehrere Zeichenketten sein. Weitere Informationen finden Sie unter Was ist in einem Secrets Manager Manager-Geheimnis? in der Secrets Manager Manager-Dokumentation.

Sicherheitskontrolle

Ein technischer oder administrativer Integritätsschutz, der die Fähigkeit eines Bedrohungsakteurs, eine Schwachstelle auszunutzen, verhindert, erkennt oder einschränkt. Es gibt vier Haupttypen von Sicherheitskontrollen: präventiv, detektiv, reaktionsschnell und proaktiv.

Härtung der Sicherheit

Der Prozess, bei dem die Angriffsfläche reduziert wird, um sie widerstandsfähiger gegen Angriffe zu machen. Dies kann Aktionen wie das Entfernen von Ressourcen, die nicht mehr benötigt werden, die Implementierung der bewährten Sicherheitsmethode der Gewährung geringster Berechtigungen oder die Deaktivierung unnötiger Feature in Konfigurationsdateien umfassen.

System zur Verwaltung von Sicherheitsinformationen und Ereignissen (security information and event management – SIEM)

Tools und Services, die Systeme für das Sicherheitsinformationsmanagement (SIM) und das Management von Sicherheitsereignissen (SEM) kombinieren. Ein SIEM-System sammelt, überwacht und analysiert Daten von Servern, Netzwerken, Geräten und anderen Quellen, um Bedrohungen und Sicherheitsverletzungen zu erkennen und Warnmeldungen zu generieren.

Automatisierung von Sicherheitsreaktionen

Eine vordefinierte und programmierte Aktion, die darauf ausgelegt ist, automatisch auf ein Sicherheitsereignis zu reagieren oder es zu beheben. Diese Automatisierungen dienen als detektive oder reaktionsschnelle Sicherheitskontrollen, die Sie bei der Implementierung bewährter AWS Sicherheitsmethoden unterstützen. Beispiele für automatisierte Antwortaktionen sind das Ändern einer VPC-Sicherheitsgruppe, das Patchen einer Amazon EC2 EC2-Instance oder das Rotieren von Anmeldeinformationen.

Serverseitige Verschlüsselung

Verschlüsselung von Daten am Zielort durch denjenigen AWS -Service , der sie empfängt.

Service-Kontrollrichtlinie (SCP)

Eine Richtlinie, die eine zentrale Kontrolle über die Berechtigungen für alle Konten in einer Organisation in AWS Organizations ermöglicht. SCPs definieren Integritätsschutz oder legen Grenzwerte für Aktionen fest, die ein Administrator an Benutzer oder Rollen delegieren kann. Sie können SCPs als Zulassungs- oder Ablehnungslisten verwenden, um festzulegen, welche Services oder Aktionen zulässig oder verboten sind. Weitere Informationen finden Sie in der AWS Organizations Dokumentation unter Richtlinien zur Dienststeuerung.

Service-Endpunkt

Die URL des Einstiegspunkts für einen AWS -Service. Sie können den Endpunkt verwenden, um programmgesteuert eine Verbindung zum Zielservice herzustellen. Weitere Informationen finden Sie unter AWS -Service -Endpunkte in der Allgemeine AWS-Referenz.

Service Level Agreement (SLA)

Eine Vereinbarung, in der klargestellt wird, was ein IT-Team seinen Kunden zu bieten verspricht, z. B. in Bezug auf Verfügbarkeit und Leistung der Services.

Service-Level-Indikator (SLI)

Eine Messung eines Leistungsaspekts eines Dienstes, z. B. seiner Fehlerrate, Verfügbarkeit oder Durchsatz.

Service-Level-Ziel (SLO)

Eine Zielkennzahl, die den Zustand eines Dienstes darstellt, gemessen anhand eines Service-Level-Indikators.

Modell der geteilten Verantwortung

Ein Modell, das die Verantwortung beschreibt, mit der Sie gemeinsam AWS für Cloud-Sicherheit und Compliance verantwortlich sind. AWS ist für die Sicherheit der Cloud verantwortlich, wohingegen Sie für die Sicherheit in der Cloud verantwortlich sind. Weitere Informationen finden Sie unter Modell der geteilten Verantwortung.

SIEM

Siehe Sicherheitsinformations- und Event-Management-System.

Single Point of Failure (SPOF)

Ein Fehler in einer einzelnen, kritischen Komponente einer Anwendung, der das System stören kann.

SLA

Siehe Service Level Agreement.

SLI

Siehe Service-Level-Indikator.

ALSO

Siehe Service-Level-Ziel.

split-and-seed Modell

Ein Muster für die Skalierung und Beschleunigung von Modernisierungsprojekten. Sobald neue Features und Produktversionen definiert werden, teilt sich das Kernteam auf, um neue Produktteams zu bilden. Dies trägt zur Skalierung der Fähigkeiten und Services Ihrer Organisation bei, verbessert die Produktivität der Entwickler und unterstützt schnelle Innovationen. Weitere Informationen finden Sie unter Schrittweiser Ansatz zur Modernisierung von Anwendungen in der AWS Cloud.

SPOTTEN

Siehe Single Point of Failure.

Sternschema

Eine Datenbank-Organisationsstruktur, die eine große Faktentabelle zum Speichern von Transaktions- oder Messdaten und eine oder mehrere kleinere dimensionale Tabellen zum Speichern von Datenattributen verwendet. Diese Struktur ist für die Verwendung in einem Data Warehouse oder für Business Intelligence-Zwecke konzipiert.

Strangler-Fig-Muster

Ein Ansatz zur Modernisierung monolithischer Systeme, bei dem die Systemfunktionen schrittweise umgeschrieben und ersetzt werden, bis das Legacy-System außer Betrieb genommen werden kann. Dieses Muster verwendet die Analogie einer Feigenrebe, die zu einem etablierten Baum heranwächst und schließlich ihren Wirt überwindet und ersetzt. Das Muster wurde eingeführt von Martin Fowler als Möglichkeit, Risiken beim Umschreiben

monolithischer Systeme zu managen. Ein Beispiel für die Anwendung dieses Musters finden Sie unter Schrittweises Modernisieren älterer Microsoft ASP.NET (ASMX)-Webservices mithilfe von Containern und Amazon API Gateway.

Subnetz

Ein Bereich von IP-Adressen in Ihrer VPC. Ein Subnetz muss sich in einer einzigen Availability Zone befinden.

Aufsichtskontrolle und Datenerfassung (SCADA)

In der Fertigung ein System, das Hardware und Software zur Überwachung von Sachanlagen und Produktionsabläufen verwendet.

Symmetrische Verschlüsselung

Ein Verschlüsselungsalgorithmus, der denselben Schlüssel zum Verschlüsseln und Entschlüsseln der Daten verwendet.

synthetisches Testen

Testen eines Systems auf eine Weise, die Benutzerinteraktionen simuliert, um potenzielle Probleme zu erkennen oder die Leistung zu überwachen. Sie können Amazon CloudWatch Synthetics verwenden, um diese Tests zu erstellen.

T

tags

Schlüssel-Wert-Paare, die als Metadaten für die Organisation Ihrer Ressourcen dienen. AWS Mit Tags können Sie Ressourcen verwalten, identifizieren, organisieren, suchen und filtern. Weitere Informationen finden Sie unter Markieren Ihrer AWS -Ressourcen.

Zielvariable

Der Wert, den Sie in überwachtem ML vorhersagen möchten. Dies wird auch als Ergebnisvariable bezeichnet. In einer Fertigungsumgebung könnte die Zielvariable beispielsweise ein Produktfehler sein.

Aufgabenliste

Ein Tool, das verwendet wird, um den Fortschritt anhand eines Runbooks zu verfolgen. Eine Aufgabenliste enthält eine Übersicht über das Runbook und eine Liste mit allgemeinen Aufgaben,

die erledigt werden müssen. Für jede allgemeine Aufgabe werden der geschätzte Zeitaufwand, der Eigentümer und der Fortschritt angegeben.

Testumgebungen

Siehe Umgebung.

Training

Daten für Ihr ML-Modell bereitstellen, aus denen es lernen kann. Die Trainingsdaten müssen die richtige Antwort enthalten. Der Lernalgorithmus findet Muster in den Trainingsdaten, die die Attribute der Input-Daten dem Ziel (die Antwort, die Sie voraussagen möchten) zuordnen. Es gibt ein ML-Modell aus, das diese Muster erfasst. Sie können dann das ML-Modell verwenden, um Voraussagen für neue Daten zu erhalten, bei denen Sie das Ziel nicht kennen.

Transit-Gateway

Ein Transit-Gateway ist ein Netzwerk-Transit-Hub, mit dem Sie Ihre VPCs und On-Premises-Netzwerke miteinander verbinden können. Weitere Informationen finden Sie in der AWS Transit Gateway Dokumentation unter Was ist ein Transit-Gateway.

Stammbasierter Workflow

Ein Ansatz, bei dem Entwickler Feature lokal in einem Feature-Zweig erstellen und testen und diese Änderungen dann im Hauptzweig zusammenführen. Der Hauptzweig wird dann sequentiell für die Entwicklungs-, Vorproduktions- und Produktionsumgebungen erstellt.

Vertrauenswürdiger Zugriff

Gewährung von Berechtigungen für einen Dienst, den Sie angeben, um Aufgaben in Ihrer Organisation AWS Organizations und in deren Konten in Ihrem Namen auszuführen. Der vertrauenswürdige Service erstellt in jedem Konto eine mit dem Service verknüpfte Rolle, wenn diese Rolle benötigt wird, um Verwaltungsaufgaben für Sie auszuführen. Weitere Informationen finden Sie in der AWS Organizations Dokumentation unter Verwendung AWS Organizations mit anderen AWS Diensten.

Optimieren

Aspekte Ihres Trainingsprozesses ändern, um die Genauigkeit des ML-Modells zu verbessern. Sie können das ML-Modell z. B. trainieren, indem Sie einen Beschriftungssatz generieren, Beschriftungen hinzufügen und diese Schritte dann mehrmals unter verschiedenen Einstellungen wiederholen, um das Modell zu optimieren.

Zwei-Pizzen-Team

Ein kleines DevOps Team, das Sie mit zwei Pizzen ernähren können. Eine Teamgröße von zwei Pizzen gewährleistet die bestmögliche Gelegenheit zur Zusammenarbeit bei der Softwareentwicklung.

U

Unsicherheit

Ein Konzept, das sich auf ungenaue, unvollständige oder unbekannte Informationen bezieht, die die Zuverlässigkeit von prädiktiven ML-Modellen untergraben können. Es gibt zwei Arten von Unsicherheit: Epistemische Unsicherheit wird durch begrenzte, unvollständige Daten verursacht, wohingegen aleatorische Unsicherheit durch Rauschen und Randomisierung verursacht wird, die in den Daten liegt. Weitere Informationen finden Sie im Leitfaden Quantifizieren der Unsicherheit in Deep-Learning-Systemen.

undifferenzierte Aufgaben

Diese Arbeit wird auch als Schwerstarbeit bezeichnet. Dabei handelt es sich um Arbeiten, die zwar für die Erstellung und den Betrieb einer Anwendung erforderlich sind, aber dem Endbenutzer keinen direkten Mehrwert bieten oder keinen Wettbewerbsvorteil bieten. Beispiele für undifferenzierte Aufgaben sind Beschaffung, Wartung und Kapazitätsplanung.

höhere Umgebungen

Siehe Umgebung.

V

Vacuuming

Ein Vorgang zur Datenbankwartung, bei dem die Datenbank nach inkrementellen Aktualisierungen bereinigt wird, um Speicherplatz zurückzugewinnen und die Leistung zu verbessern.

Versionskontrolle

Prozesse und Tools zur Nachverfolgung von Änderungen, z. B. Änderungen am Quellcode in einem Repository.

VPC-Peering

Eine Verbindung zwischen zwei VPCs, mit der Sie den Datenverkehr mithilfe von privaten IP-Adressen weiterleiten können. Weitere Informationen finden Sie unter Was ist VPC-Peering? in der Amazon-VPC-Dokumentation.

Schwachstelle

Ein Software- oder Hardwarefehler, der die Sicherheit des Systems gefährdet.

W

Warmer Cache

Ein Puffer-Cache, der aktuelle, relevante Daten enthält, auf die häufig zugegriffen wird. Die Datenbank-Instance kann aus dem Puffer-Cache lesen, was schneller ist als das Lesen aus dem Hauptspeicher oder von der Festplatte.

warme Daten

Daten, auf die selten zugegriffen wird. Bei der Abfrage dieser Art von Daten sind mäßig langsame Abfragen in der Regel akzeptabel.

Fensterfunktion

Eine SQL-Funktion, die eine Berechnung für eine Gruppe von Zeilen durchführt, die sich in irgendeiner Weise auf den aktuellen Datensatz beziehen. Fensterfunktionen sind nützlich für die Verarbeitung von Aufgaben wie die Berechnung eines gleitenden Durchschnitts oder für den Zugriff auf den Wert von Zeilen auf der Grundlage der relativen Position der aktuellen Zeile.

Workload

Ein Workload ist eine Sammlung von Ressourcen und Code, die einen Unternehmenswert bietet, wie z. B. eine kundenorientierte Anwendung oder ein Backend-Prozess.

Workstream

Funktionsgruppen in einem Migrationsprojekt, die für eine bestimmte Reihe von Aufgaben verantwortlich sind. Jeder Workstream ist unabhängig, unterstützt aber die anderen Workstreams im Projekt. Der Portfolio-Workstream ist beispielsweise für die Priorisierung von Anwendungen, die Wellenplanung und die Erfassung von Migrationsmetadaten verantwortlich. Der Portfolio-Workstream liefert diese Komponenten an den Migrations-Workstream, der dann die Server und Anwendungen migriert.

WURM

Sehen Sie einmal schreiben, viele lesen.

WQF

Weitere Informationen finden Sie unter AWS Workload Qualification Framework.

einmal schreiben, viele lesen (WORM)

Ein Speichermodell, das Daten ein einziges Mal schreibt und verhindert, dass die Daten gelöscht oder geändert werden. Autorisierte Benutzer können die Daten so oft wie nötig lesen, aber sie können sie nicht ändern. Diese Datenspeicherinfrastruktur gilt als unveränderlich.

Z

Zero-Day-Exploit

Ein Angriff, in der Regel Malware, der eine Zero-Day-Sicherheitslücke ausnutzt.

Zero-Day-Sicherheitslücke

Ein unfehlbarer Fehler oder eine Sicherheitslücke in einem Produktionssystem. Bedrohungsakteure können diese Art von Sicherheitslücke nutzen, um das System anzugreifen. Entwickler werden aufgrund des Angriffs häufig auf die Sicherheitsanfälligkeit aufmerksam.

Zombie-Anwendung

Eine Anwendung, deren durchschnittliche CPU- und Arbeitsspeichernutzung unter 5 Prozent liegt. In einem Migrationsprojekt ist es üblich, diese Anwendungen außer Betrieb zu nehmen.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.