Leitfaden

AWS CodePipeline

API-Version 2015-07-09

AWS CodePipeline: Leitfaden

Table of Contents

Was ist AWS CodePipeline?

AWS CodePipeline ist ein Continuous-Delivery-Service, mit dem Sie die zur Veröffentlichung Ihrer Software erforderlichen Schritte modellieren, visualisieren und automatisieren können. Sie können die verschiedenen Phasen eines Softwareveröffentlichungsprozesses schnell modellieren und konfigurieren. CodePipeline automatisiert die Schritte, die erforderlich sind, um Ihre Softwareänderungen kontinuierlich zu veröffentlichen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

Themen

Kontinuierliche Bereitstellung und kontinuierliche Integration

CodePipeline ist ein Service mit kontinuierlicher Bereitstellung, der die Entwicklung, das Testen und die Bereitstellung Ihrer Software in der Produktion automatisiert.

Kontinuierliche Bereitstellung ist eine Methode für die Bereitstellung von Software, bei der der Veröffentlichungsprozess automatisiert ist. Jede Änderung der Software wird automatisch entwickelt, getestet und für die Produktion bereitgestellt. Vor der endgültigen Bereitstellung für die Produktion wird von einer Person, einem automatisierten Test oder einer Geschäftsregel entschieden, wann die endgültige Bereitstellung erfolgen soll. Auch wenn bei einer kontinuierlichen Bereitstellung jede erfolgreiche Software-Änderung sofort für die Produktion bereitgestellt werden kann, müssen nicht alle Änderungen sofort bereitgestellt werden.

Kontinuierliche Integration ist eine Softwareentwicklungspraxis, bei der Mitglieder eines Teams ein Versionskontrollsystem verwenden und ihre Arbeit häufig am selben Ort, z. B. in einer Hauptniederlassung, integrieren. Jede Änderung wird erstellt und überprüft, um Integrationsfehler so schnell wie möglich zu entdecken. Die kontinuierliche Integration konzentriert sich auf die automatische Erstellung und dem automatischen Testen von Code, während bei der kontinuierlichen Bereitstellung der gesamte Prozess für die Software-Veröffentlichung bis zur Produktionseinführung automatisiert wird.

Weitere Informationen finden Sie unter Practicing Continuous Integration and Continuous Delivery unter AWS: Beschleunigte Softwarebereitstellung mit DevOps.

Sie können die CodePipeline Konsole, die AWS Command Line Interface (AWS CLI), die AWS SDKs oder eine beliebige Kombination davon verwenden, um Ihre Pipelines zu erstellen und zu verwalten.

Was kann ich damit machen? CodePipeline

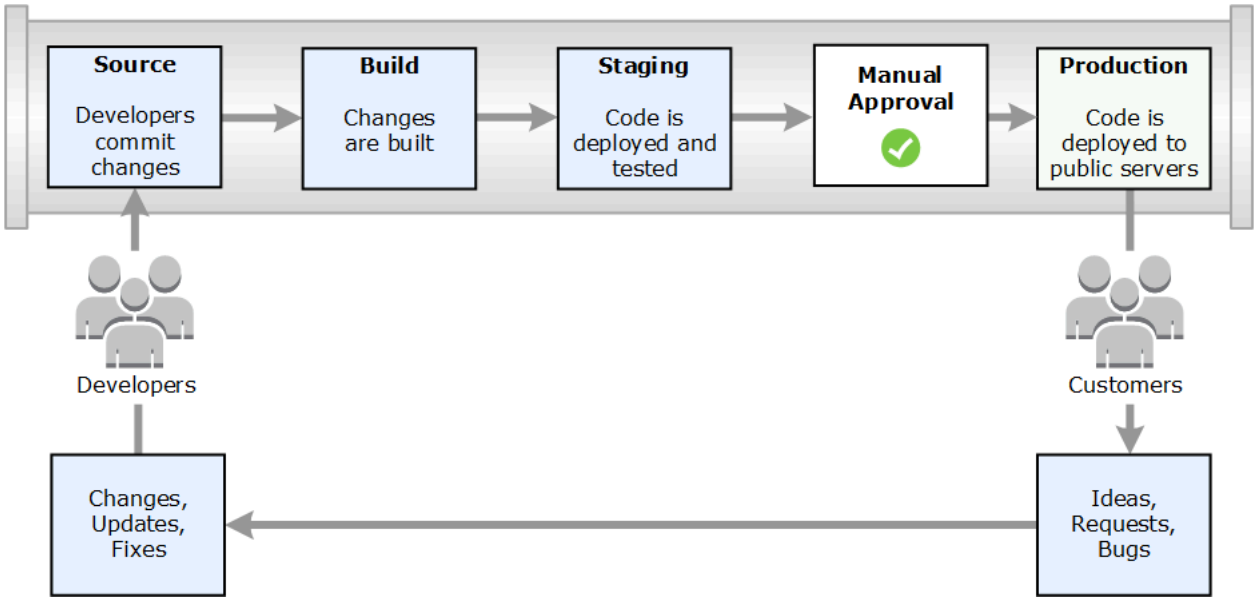
Sie können CodePipeline es verwenden, um Ihre Anwendungen automatisch in der Cloud zu erstellen, zu testen und bereitzustellen. Insbesondere können Sie Folgendes:

- Automatisieren Sie Ihre Release-Prozesse: Automatisiert Ihren Release-Prozess CodePipeline vollständig von Anfang bis Ende, angefangen von Ihrem Quell-Repository über Build, Test und Bereitstellung. Sie können die Verarbeitung von Änderungen in der Pipeline verhindern, indem Sie eine manuelle Genehmigungsaktion einfügen. Dies ist für alle Phasen außer der Quellphase möglich. Sie können Software-Änderungen zum gewünschten Zeitpunkt auf die gewünschte Weise auf den Systemen Ihrer Wahl und auf einer einzigen oder mehreren Instances veröffentlichen.
- Richten Sie einen konsistenten Release-Prozess ein: Definieren Sie für jede Codeänderung eine konsistente Reihe von Schritten. CodePipeline führt jede Phase Ihres Releases nach Ihren Kriterien aus.
- Sie können die Bereitstellung beschleunigen und gleichzeitig die Qualität verbessern: Sie können Ihren Veröffentlichungsprozess automatisieren, um Ihren Developern zu gestatten, Code inkrementell zu testen und zu veröffentlichen und die Veröffentlichung neuer Funktionen für Ihre Kunden zu beschleunigen.
- Sie können Ihre bevorzugten Tools verwenden: Sie können vorhandene Quellen, Builds und Bereitstellungs-Tools in Ihre Pipeline integrieren. Eine vollständige Liste der Tools AWS-Services und Tools von Drittanbietern, die derzeit von unterstützt werden CodePipeline, finden Sie unterProdukt- und Serviceintegrationen mit CodePipeline.

- Den Fortschritt auf einen Blick anzeigen: Sie können den Status Ihrer Pipelines in Echtzeit überprüfen, die Details aller Warnungen überprüfen, fehlgeschlagene Phasen oder Aktionen erneut versuchen, Details zu den Quellversionen anzeigen, die bei der letzten Pipeline-Ausführung in jeder Phase verwendet wurden, und jede Pipeline manuell erneut ausführen.
- Anzeigen von Pipeline-Verlaufsdaten: Sie können Informationen über Ausführungen einer Pipeline einschließlich Start- und Endzeiten, Ausführungsdauer und Ausführungs-IDs anzeigen.

Ein kurzer Blick auf CodePipeline

Das folgende Diagramm zeigt ein Beispiel für einen Freigabeprozess mit CodePipeline.



Wenn Entwickler in diesem Beispiel Änderungen an ein Quell-Repository übertragen, CodePipeline werden die Änderungen automatisch erkannt. Diese Änderungen werden erstellt. Wenn Tests konfiguriert werden, werden diese Tests ausgeführt. Nach Abschluss der Tests wird der erstellte Code für weitere Tests auf Staging-Servern bereitgestellt. CodePipeline Führt vom Staging-Server aus weitere Tests aus, z. B. Integrations- oder Auslastungstests. Nach erfolgreichem Abschluss dieser Tests und nach Genehmigung einer manuellen Genehmigungsaktion, die der Pipeline hinzugefügt wurde, wird der getestete und genehmigte Code auf Produktionsinstanzen CodePipeline bereitgestellt.

CodePipeline kann mithilfe von, oder Anwendungen auf EC2-Instances CodeDeploy bereitstellen. AWS Elastic Beanstalk AWS OpsWorks Stacks CodePipeline kann mithilfe von Amazon ECS auch containerbasierte Anwendungen für Services bereitstellen. Entwickler können die mitgelieferten

Integrationspunkte auch verwenden, CodePipeline um andere Tools oder Dienste zu integrieren, darunter Build-Services, Testanbieter oder andere Bereitstellungsziele oder -systeme.

Eine Pipeline kann so einfach oder so komplex sein, wie von Ihrem Veröffentlichungsprozess gefordert.

Wie fange ich an mit CodePipeline?

Um loszulegen mit CodePipeline:

1. Lesen Sie den CodePipeline Konzepte Abschnitt, um zu erfahren, wie das CodePipeline funktioniert.
2. Bereiten Sie sich auf die Verwendung vor, CodePipeline indem Sie die Schritte unter befolgenErste Schritte mit CodePipeline.
3. Experimentieren Sie damit, CodePipeline indem Sie die Schritte in den CodePipeline Anleitungen Tutorials befolgen.
4. Verwenden Sie es CodePipeline für Ihre neuen oder vorhandenen Projekte, indem Sie die Schritte unter befolgenErstellen Sie eine Pipeline in CodePipeline.

CodePipeline Konzepte

Die Modellierung und Konfiguration Ihres automatisierten Release-Prozesses ist einfacher, wenn Sie die in verwendeten Konzepte und Begriffe verstehen AWS CodePipeline. Im Folgenden finden Sie einige Konzepte, mit denen Sie sich bei der Verwendung vertraut machen sollten CodePipeline.

Ein Beispiel für eine DevOps Pipeline finden Sie unterDevOps Beispiel für eine Pipeline.

Die folgenden Begriffe werden in verwendet CodePipeline:

Themen

- Pipelines
- Pipeline-Ausführungen
- Operationen in Etappen
- Aktionsausführungen
- Arten der Ausführung
- Aktionstypen

- -Artefakte
- Quell-Revisionen
- Auslöser
- Variablen

Pipelines

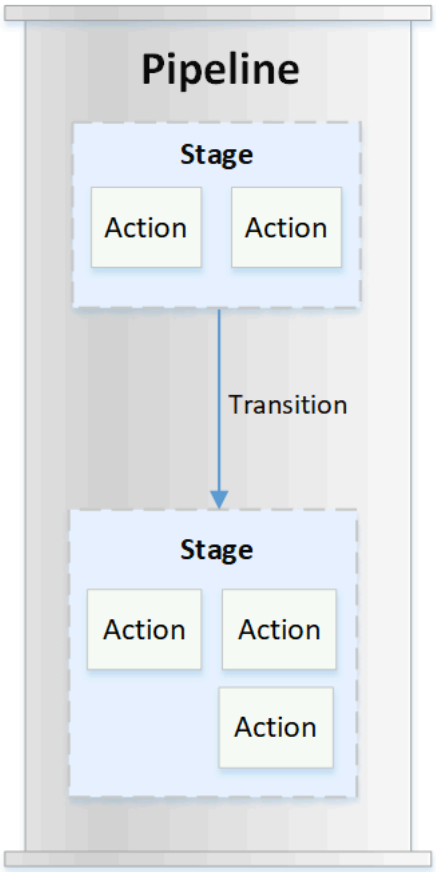
Eine Pipeline ist ein Workflow-Konstrukt, das den Veröffentlichungsprozess von Software-Änderungen beschreibt. Jede Pipeline besteht aus einer Reihe von Phasen.

Phasen

Eine Phase ist eine logische Einheit, mit der Sie eine Umgebung isolieren und die Anzahl der Änderungen einschränken können, die in dieser Umgebung gleichzeitig ausgeführt werden. Jede Phase enthält Aktionen, die für die Artefakte der Anwendung ausgeführt werden. Ihr Quellcode ist ein Beispiel für ein Artefakt. Bei einer Phase kann es sich um eine Build-Phase handeln, in der der Quellcode erstellt und getestet wird. Es kann sich auch um eine Bereitstellungsphase handeln, in der Code in Laufzeitumgebungen bereitgestellt wird. Jede Phase besteht aus einer Reihe serieller oder paralleler Aktionen.

Übergänge

Ein Übergang ist der Punkt, an dem eine Pipeline-Ausführung zur nächsten Phase in der Pipeline wechselt. Sie können den Eingangspunkt einer Phase deaktivieren, um zu verhindern, dass Ausführungen in diese Phase eintreten. Anschließend können Sie den Eingangspunkt aktivieren, damit Ausführungen fortgesetzt werden können. Wenn mehrere Ausführungen an einem deaktivierten Übergangspunkt eintreffen, wird nur die neueste Ausführung zur nächsten Phase fortgesetzt, wenn der Übergangspunkt aktiviert wird. Das bedeutet, dass neuere Ausführungen bereits wartende Ausführungen ersetzen, während der Übergangspunkt deaktiviert ist. Nach der Aktivierung des Übergangspunkts wird nur die ersetzende Ausführung fortgesetzt.



Aktionen

Eine Aktion ist ein Satz von Operationen, die auf Anwendungscode angewendet werden. Sie ist für die Ausführung an einem bestimmten Punkt der Pipeline konfiguriert. Dies können Quellaktionen aufgrund von Codeänderungen, Aktionen zum Bereitstellen der Anwendung auf Instances usw. sein. Eine Bereitstellungsphase kann beispielsweise eine Bereitstellungsaktion enthalten, die Code für einen Rechenservice wie Amazon EC2 oder bereitstellt. AWS Lambda

Gültige CodePipeline Aktionstypen sind`source`,`build`,`test`, `deployapproval`, und. `invoke` Eine Liste der Aktionsanbieter finden Sie unter Gültige Aktionstypen und Anbieter in CodePipeline .

Aktionen können seriell oder parallel ausgeführt werden. Informationen zu seriellen und parallel Aktionen in einer Phase finden Sie unter Anforderungen an die `runOrder` Aktionsstruktur.

Pipeline-Ausführungen

Eine Ausführung ist ein Satz von Änderungen, die von einer Pipeline freigegeben werden. Jede Pipeline-Ausführung ist eindeutig und besitzt eine eigene ID. Eine Ausführung entspricht einem Satz

von Änderungen, z. B. einem zusammengeführten Commit oder einer manuellen Freigabe des letzten Commits. Zwei Ausführungen können den gleichen Satz von Änderungen zu unterschiedlichen Zeitpunkten freigeben.

Während eine Pipeline mehrere Ausführungen gleichzeitig verarbeiten kann, verarbeitet eine Pipeline-Phase jeweils nur eine Ausführung zur selben Zeit. Hierzu wird die Phase gesperrt, während sie eine Ausführung verarbeitet. Zwei Pipeline-Ausführungen können nicht gleichzeitig dieselbe Phase belegen. Eine Ausführung, die darauf wartet, in die besetzte Phase überzugehen, wird als eingehende Ausführung bezeichnet. Eine eingehende Ausführung kann immer noch fehlschlagen, ersetzt oder manuell gestoppt werden. Weitere Hinweise zur Funktionsweise eingehender Ausführungen finden Sie unter. So funktionieren eingehende Ausführungen

Pipeline-Ausführungen durchlaufen die Pipeline-Phasen der Reihe nach. Gültige Statuswerte für Pipelines sind `InProgress`, `Stopping`, `Stopped`, `Succeeded`, `Superseded` und `Failed`.

Weitere Informationen finden Sie unter. PipelineExecution

Angehaltene Ausführungen

Die Pipeline-Ausführung kann manuell angehalten werden, sodass die laufende Pipeline-Ausführung nicht durch die Pipeline hindurch fortgesetzt wird. Wenn die Pipeline-Ausführung manuell angehalten wird, zeigt sie den Status `Stopping` an, bis sie vollständig gestoppt wird. Danach zeigt sie den Status `Stopped` an. Eine `Stopped`-Pipeline-Ausführung kann erneut ausgeführt werden.

Es gibt zwei Möglichkeiten, eine Pipeline-Ausführung anzuhalten:

- Anhalten und warten
- Anhalten und beenden

Informationen über Anwendungsfälle für das Anhalten einer Ausführung und Details zur Reihenfolge dieser Optionen finden Sie unter Pipeline-Ausführungen beenden.

Fehlgeschlagene Ausführungen

Wenn eine Ausführung fehlschlägt, wird sie angehalten und durchläuft die Pipeline nicht vollständig. Ihr Status ist `FAILED` und die Phase wird freigeschaltet. Eine neuere Ausführung kann in die entsperrte Phase eintreten und diese sperren. Sie können eine fehlgeschlagene Ausführung wiederholen, wenn diese nicht ersetzt wurde oder nicht wiederholbar ist. Sie können eine fehlgeschlagene Phase auf eine vorherige erfolgreiche Ausführung zurücksetzen.

Ausführungsmodi

Um die neuesten Änderungen über eine Pipeline bereitzustellen, holen neuere Ausführungen weniger aktuelle Ausführungen ein, die bereits über die Pipeline ausgeführt werden, und ersetzen diese. In diesem Fall wird die ältere Ausführung durch die neuere Ausführung ersetzt. Eine Ausführung kann durch eine neuere Ausführung an einem bestimmten Punkt ersetzt werden. Dies ist der Punkt zwischen den einzelnen Phasen. SUPERSEDED ist der Standard-Ausführungsmodus.

Wenn im SUPERSEDED-Modus eine Ausführung darauf wartet, in eine gesperrte Phase einzutreten, kann eine neuere Ausführung sie catch und ersetzen. Die neuere Ausführung wartet nun auf die Entsperrung der Phase und die ersetzte Ausführung wird mit dem Status `SUPERSEDED` beendet. Wenn eine Pipeline-Ausführung ersetzt wird, wird die Ausführung gestoppt und durchläuft die Pipeline nicht vollständig. Sie können die ersetzte Ausführung nicht wiederholen, nachdem sie in dieser Phase ersetzt wurde. Andere verfügbare Ausführungsmodi sind der PARALLEL- oder QUEUED-Modus.

Weitere Hinweise zu Ausführungsmodi und gesperrten Phasen finden Sie unterWie werden Ausführungen im Modus ABGELÖST verarbeitet.

Operationen in Etappen

Wenn eine Pipeline-Ausführung eine Phase durchläuft, ist die Phase dabei, alle darin enthaltenen Aktionen abzuschließen. Hinweise zur Funktionsweise von Phasenoperationen und Informationen zu gesperrten Phasen finden Sie unterWie werden Ausführungen im Modus ABGELÖST verarbeitet.

Gültige Status für Stufen sind`InProgress`,`Stopping`, `StoppedSucceeded`, und`Failed`. Sie können eine fehlgeschlagene Phase erneut versuchen, es sei denn, die fehlgeschlagene Phase kann nicht wiederholt werden. Weitere Informationen finden Sie unter. StageExecution Sie können eine Phase auf eine bestimmte vorherige erfolgreiche Ausführung zurücksetzen. Eine Phase kann so konfiguriert werden, dass bei einem Fehler ein automatisches Rollback ausgeführt wird, wie unter beschriebenStage Rollback konfigurieren. Weitere Informationen finden Sie unter RollbackStage.

Aktionsausführungen

Eine Aktionsausführung ist die Durchführung einer konfigurierten Aktion, die für bestimmteArtefakte ausgeführt wird. Dies können Eingabe-Artefakte, Ausgabe-Artefakte oder beides sein. Beispielsweise kann eine Build-Aktion Build-Befehle für ein Eingabe-Artefakt ausführen, z. B. das Anwendungsquellcode kompilieren. Die Details der Aktionsausführung umfassen die Aktionsausführungs-ID, den zugehörigen Pipeline-Ausführungsquellauslöser und die Eingabe- und Ausgabe-Artefakte der Aktion.

Gültige Status für Aktionen sind`InProgress`, `AbandonedSucceeded`, oder`Failed`. Weitere Informationen finden Sie unter ActionExecution.

Arten der Ausführung

Bei einer Pipeline- oder Phasenausführung kann es sich entweder um eine Standardausführung oder um eine Ausführung mit Rollback handeln.

Bei Standardtypen hat die Ausführung eine eindeutige ID und ist ein vollständiger Pipeline-Lauf. Ein Pipeline-Rollback hat eine Phase, für die ein Rollback ausgeführt werden muss, und eine erfolgreiche Ausführung für die Phase als Zielausführung, zu der ein Rollback durchgeführt werden soll. Die Ausführung der Zielpipeline wird verwendet, um Quellversionen und Variablen für die Phase abzurufen, die erneut ausgeführt werden soll.

Aktionstypen

Aktionstypen sind vorkonfigurierte Aktionen, die unter ausgewählt werden können. CodePipeline Der Aktionstyp wird durch seinen Besitzer, Anbieter, Version und Kategorie definiert. Der Aktionstyp stellt benutzerdefinierte Parameter bereit, die verwendet werden, um die Aktionsaufgaben in einer Pipeline abzuschließen.

Informationen zu den Produkten AWS-Services und Diensten von Drittanbietern, die Sie je nach Aktionstyp in Ihre Pipeline integrieren können, finden Sie unterIntegrationen mit CodePipeline Aktionstypen.

Informationen zu den Integrationsmodellen, die für Aktionstypen in unterstützt werden CodePipeline, finden Sie unterReferenz zum Integrationsmodell.

Informationen darüber, wie Drittanbieter Aktionstypen in einrichten und verwalten können CodePipeline, finden Sie unterMit Aktionstypen arbeiten.

-Artefakte

Der Begriff Artefakt bezieht sich auf die Sammlung von Daten, wie Anwendungsquellcode, erstellte Anwendungen, Abhängigkeiten, Definitionsdateien, Vorlagen usw., die von Pipeline-Aktionen bearbeitet werden. Artefakte werden von bestimmten Aktionen erzeugt und von anderen Aktionen verbraucht. Artefakten in einer Pipeline können die Gruppe von Dateien sein, die von einer Aktion bearbeitet werden (Eingabe-Artefakte) oder die aktualisierte Ausgabe einer abgeschlossenen Aktion (Ausgabe-Artefakte).

Aktionen leiten die Ausgabe an eine andere Aktion zur weiteren Verarbeitung mithilfe des Pipeline-Artefakt-Buckets weiter. CodePipeline kopiert Artefakte in den Artefaktspeicher, wo sie von der Aktion abgerufen werden. Weitere Informationen zu Artefakten finden Sie unter Eingabe- und Ausgabe-Artefakte.

Quell-Revisionen

Wenn Sie eine Quellcodeänderung ausführen, wird eine neue Version erstellt. Eine Quellrevision ist eine Quelländerungsversion, die eine Pipeline-Ausführung auslöst. Bei einer Ausführung werden Quellversionen verarbeitet. Für GitHub und CodeCommit Repositorien ist dies der Commit. Bei S3-Buckets oder -Aktionen ist dies die Objektversion.

Sie können eine Pipeline-Ausführung mit einer von Ihnen angegebenen Quellrevision starten, z. B. einem Commit. Die Ausführung verarbeitet die angegebene Revision und überschreibt die Version, die für die Ausführung verwendet worden wäre. Weitere Informationen finden Sie unter Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung.

Auslöser

Trigger sind Ereignisse, die Ihre Pipeline starten. Einige Auslöser, z. B. das manuelle Starten einer Pipeline, sind für alle Quellaktionsanbieter in einer Pipeline verfügbar. Bestimmte Trigger hängen vom Quellanbieter für eine Pipeline ab. CloudWatch Ereignisse müssen beispielsweise mit Ereignisressourcen von Amazon konfiguriert werden CloudWatch , denen der Pipeline-ARN als Ziel in der Ereignisregel hinzugefügt wurde. Amazon CloudWatch Events ist der empfohlene Auslöser für die automatische Änderungserkennung für Pipelines mit einer CodeCommit oder S3-Quellaktion. Webhooks sind ein Triggertyp, der für Repository-Ereignisse von Drittanbietern konfiguriert ist. WebHookV2 ist beispielsweise ein Triggertyp, der es ermöglicht, Git-Tags zu verwenden, um Pipelines mit externen Quellanbietern wie GitHub .com, GitHub Enterprise Server, GitLab .com, GitLab Self-managed oder Bitbucket Cloud zu starten. In der Pipeline-Konfiguration kannst du einen Filter für Trigger wie Push- oder Pull-Requests angeben. Du kannst Code-Push-Ereignisse nach Git-Tags, Branches oder Dateipfaden filtern. Du kannst Pull-Request-Ereignisse nach Ereignissen (geöffnet, aktualisiert, geschlossen), Branches oder Dateipfaden filtern.

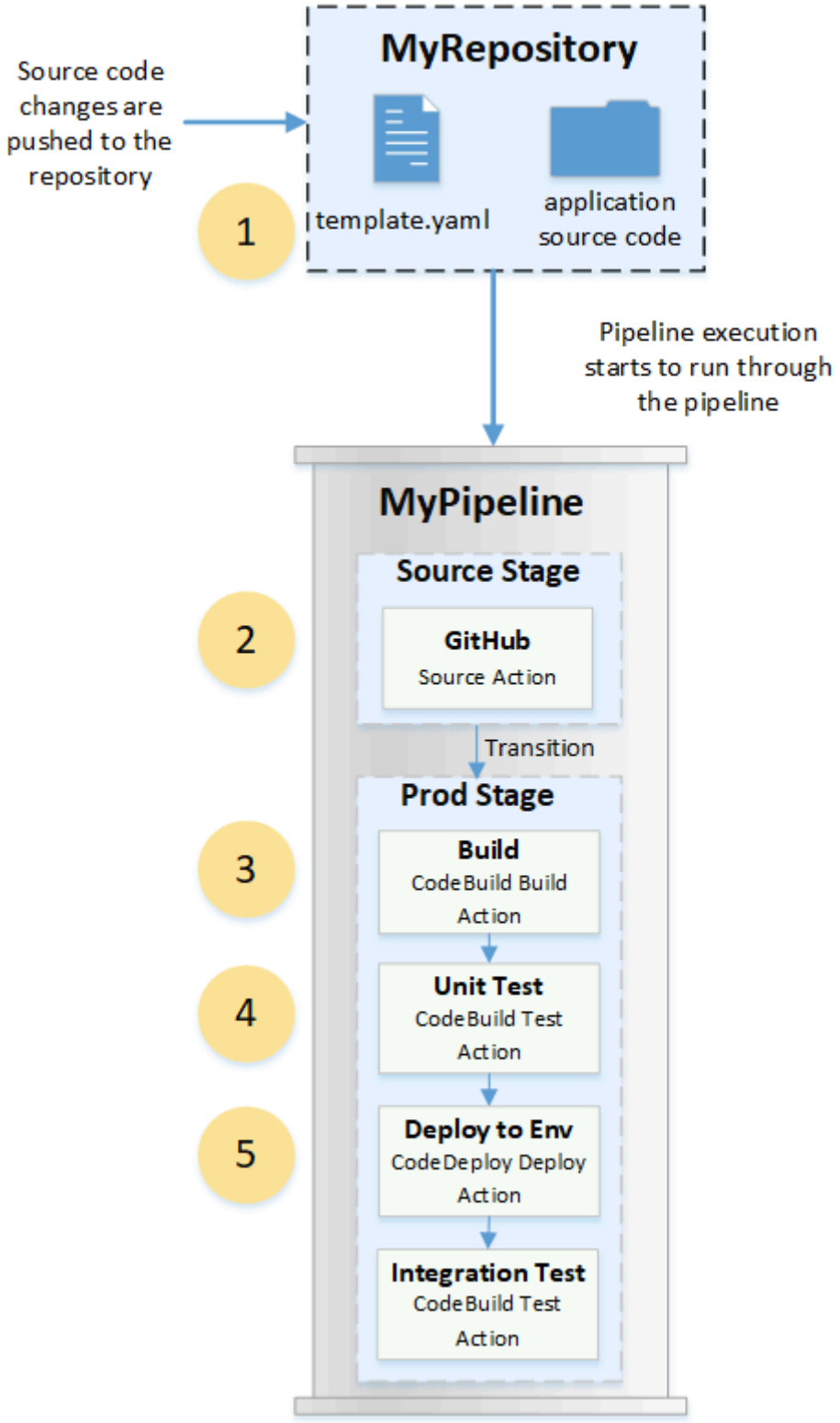
Weitere Informationen zu Auslösern finden Sie unter Starten Sie eine Pipeline in CodePipeline. Ein Tutorial, das Sie durch die Verwendung von Git-Tags als Trigger für Ihre Pipeline führt, finden Sie unterTutorial: Verwende Git-Tags, um deine Pipeline zu starten.

Variablen

Eine Variable ist ein Wert, der verwendet werden kann, um Aktionen in Ihrer Pipeline dynamisch zu konfigurieren. Variablen können entweder auf Pipeline-Ebene deklariert oder durch Aktionen in der Pipeline ausgegeben werden. Variablenwerte werden zum Zeitpunkt der Pipeline-Ausführung aufgelöst und können im Ausführungsverlauf eingesehen werden. Für Variablen, die auf Pipeline-Ebene deklariert wurden, können Sie entweder Standardwerte in der Pipeline-Konfiguration definieren oder sie für eine bestimmte Ausführung überschreiben. Bei Variablen, die von einer Aktion ausgegeben werden, ist der Wert verfügbar, nachdem eine Aktion erfolgreich abgeschlossen wurde. Weitere Informationen finden Sie unter Variablen.

DevOps Beispiel für eine Pipeline

Als Beispiel für eine DevOps Pipeline könnte eine zweistufige Pipeline eine Quellstufe namens Source und eine zweite Stufe namens Prod haben. In diesem Beispiel aktualisiert die Pipeline die Anwendung mit den neuesten Änderungen und stellt kontinuierlich das neueste Ergebnis bereit. Bevor die neueste Anwendung bereitgestellt wird, erstellt und testet die Pipeline die Webanwendung. In diesem Beispiel hat eine Gruppe von Entwicklern eine Infrastrukturvorlage und den Quellcode für eine Webanwendung in einem GitHub Repository namens eingerichtet. MyRepository



Ein Entwickler verschiebt beispielsweise eine Fehlerbehebung per Push zur Indexseite der Webanwendung. Folgendes tritt auf:

1. Der Quellcode der Anwendung wird in einem Repository verwaltet, das als GitHub Quellaktion in der Pipeline konfiguriert ist. Wenn Entwickler Commits per Push in das Repository übertragen, wird die übertragene Änderung CodePipeline erkannt und die Pipeline-Ausführung beginnt in der Quellphase.
2. Die GitHub Quellaktion wird erfolgreich abgeschlossen (das heißt, die neuesten Änderungen wurden heruntergeladen und in dem für diese Ausführung spezifischen Artefakt-Bucket gespeichert). Die von der GitHub Quellaktion erzeugten Ausgabeartefakte, bei denen es sich um die Anwendungsdateien aus dem Repository handelt, werden dann als Eingabeartefakte verwendet, an denen die Aktionen in der nächsten Phase arbeiten.
3. Die Pipeline-Ausführung wechselt von der Quellphase zur Produktionsphase. Die erste Aktion in der Produktionsphase führt ein Build-Projekt aus, das in der Pipeline erstellt CodeBuild und als Build-Aktion konfiguriert wurde. Die Build-Aufgabe ruft ein Build-Umgebungs-Image ab und erstellt die Webanwendung in einem virtuellen Container.
4. Die nächste Aktion in der Produktionsphase ist ein Komponententest-Projekt, das in der CodeBuild Pipeline als Testaktion erstellt und konfiguriert wurde.
5. Der komponentengetestete Code wird als Nächstes von einer Bereitstellungsaktion in der Produktionsphase bearbeitet, die die Anwendung zu einer Produktionsumgebung bereitstellt. Nachdem die Bereitstellungsaktion erfolgreich abgeschlossen wurde, ist die letzte Aktion in der Phase ein Integrationstestprojekt, das in der Pipeline erstellt CodeBuild und als Testaktion konfiguriert wurde. Die Testaktion ruft Shell-Skripte auf, die ein Testtool in der Webanwendung installieren und ausführen, z. B. einen Linkprüfer. Nach erfolgreichem Abschluss besteht die Ausgabe in einer erstellten Webanwendung und einer Reihe von Testergebnissen.

Entwickler können der Pipeline Aktionen hinzufügen, die die Anwendung bereitstellen oder weiter testen, nachdem sie bei jeder Änderung erstellt und getestet wurde.

Weitere Informationen finden Sie unter So funktionieren Pipeline-Ausführungen.

So funktionieren Pipeline-Ausführungen

Dieser Abschnitt bietet einen Überblick über die Art und Weise, wie eine Reihe von Änderungen CodePipeline verarbeitet werden. CodePipelineverfolgt jede Pipeline-Ausführung, die beginnt, wenn eine Pipeline manuell gestartet oder eine Änderung am Quellcode vorgenommen wird. CodePipeline verwendet die folgenden Ausführungsmodi, um die Art und Weise zu handhaben, wie jede Ausführung durch die Pipeline voranschreitet.

- Modus ABGELÖST: Eine neuere Ausführung kann eine ältere überholen. Dies ist die Standardeinstellung.
- WARTESCHLANGENMODUS: Ausführungen werden nacheinander in der Reihenfolge verarbeitet, in der sie sich in der Warteschlange befinden. Dies erfordert den Pipeline-Typ V2.
- PARALLEL-Modus: Im PARALLEL-Modus werden Ausführungen gleichzeitig und unabhängig voneinander ausgeführt. Ausführungen warten nicht darauf, dass andere Läufe abgeschlossen sind, bevor sie gestartet oder beendet werden. Dies erfordert den Pipeline-Typ V2.

So werden Pipeline-Ausführungen gestartet

Sie können eine Ausführung starten, wenn Sie Ihren Quellcode ändern, oder Sie können die Pipeline manuell starten. Sie können eine Ausführung auch über eine von Ihnen geplante Amazon CloudWatch Events-Regel auslösen. Wenn beispielsweise eine Quellcodeänderung per Push zu einem Repository übertragen wird, das als Quellaktion der Pipeline konfiguriert ist, erkennt die Pipeline die Änderung und startet eine Ausführung.

> **Note**
>
> Wenn eine Pipeline mehrere Quellaktionen enthält, werden alle erneut ausgeführt, auch wenn nur für eine Quellaktion eine Änderung entdeckt wird.

Wie Quellversionen in Pipeline-Ausführungen verarbeitet werden

Für jede Pipeline-Ausführung, die mit Änderungen am Quellcode (Quellrevisionen) beginnt, werden Quellrevisionen wie folgt bestimmt.

- Bei Pipelines mit einer CodeCommit Quelle wird der HEAD in dem Moment geklont, CodePipeline in dem der Commit übertragen wird. Beispielsweise wird ein Commit per Push übertragen, wodurch die Pipeline für Ausführung 1 gestartet wird. In dem Moment, in dem ein zweiter Commit übertragen wird, wird die Pipeline für Ausführung 2 gestartet.

  > **Note**
  >
  > Bei Pipelines im PARALLEL-Modus mit einer CodeCommit Quelle klont die Quellaktion unabhängig vom Commit, das die Pipeline-Ausführung ausgelöst hat, den HEAD immer zum Zeitpunkt des Starts. Weitere Informationen finden Sie unter CodeCommit oder S3-

Quellversionen im PARALLEL-Modus stimmen möglicherweise nicht mit dem Ereignis überein EventBridge .

- Für Pipelines mit einer S3-Quelle wird das EventBridge Ereignis für das S3-Bucket-Update verwendet. Das Ereignis wird beispielsweise generiert, wenn eine Datei im Quell-Bucket aktualisiert wird, wodurch die Pipeline für Ausführung 1 gestartet wird. In dem Moment, in dem das Ereignis für ein zweites Bucket-Update ausgelöst wird, wird damit die Pipeline für Ausführung 2 gestartet.

  > **Note**
  >
  > Bei Pipelines im PARALLEL-Modus mit einer S3-Quelle beginnt die Quellaktion unabhängig vom Image-Tag, das die Ausführung ausgelöst hat, immer mit dem neuesten Image-Tag. Weitere Informationen finden Sie unter CodeCommit oder S3-Quellversionen im PARALLEL-Modus stimmen möglicherweise nicht mit dem Ereignis überein EventBridge .

- Bei Pipelines mit einer Verbindungsquelle, z. B. zu Bitbucket, wird der HEAD in dem Moment geklont, CodePipeline in dem der Commit übertragen wird. Bei einer Pipeline im PARALLEL-Modus wird beispielsweise ein Commit per Push übertragen, wodurch die Pipeline für Ausführung 1 gestartet wird, und die zweite Pipeline-Ausführung verwendet den zweiten Commit.

Pipeline-Ausführungen beenden

Um die Konsole zum Beenden einer Pipeline-Ausführung zu verwenden, können Sie Stop execution (Ausführung beenden) auf der Seite mit der Pipeline-Visualisierung, auf der Seite mit dem Ausführungsverlauf oder auf der Seite mit dem detaillierten Verlauf auswählen. Um die CLI zum Beenden einer Pipeline-Ausführung zu verwenden, verwenden Sie den Befehl `stop-pipeline-execution`. Weitere Informationen finden Sie unter Stoppen Sie eine Pipeline-Ausführung in CodePipeline.

Es gibt zwei Möglichkeiten, eine Pipeline-Ausführung anzuhalten:

- Anhalten und warten: Alle laufenden Aktionsausführungen dürfen abgeschlossen werden, und nachfolgende Aktionen werden nicht gestartet. Die Pipeline-Ausführung wird nicht in den nachfolgenden Phasen fortgesetzt. Sie können diese Option nicht für eine Ausführung verwenden, die sich bereits in einem `Stopping`-Status befindet.

- Anhalten und beenden: Alle laufenden Aktionsausführungen werden angehalten und nicht abgeschlossen, und nachfolgende Aktionen werden nicht begonnen. Die Pipeline-Ausführung wird nicht in den nachfolgenden Phasen fortgesetzt. Sie können diese Option bei einer Ausführung verwenden, die sich bereits in einem `Stopping`-Status befindet.

> **Note**
>
> Diese Option kann zu fehlgeschlagenen oder nicht aufeinander folgenden Aufgaben führen.

Jede Option führt zu einer anderen Abfolge von Pipeline- und Aktionsausführungsphasen:

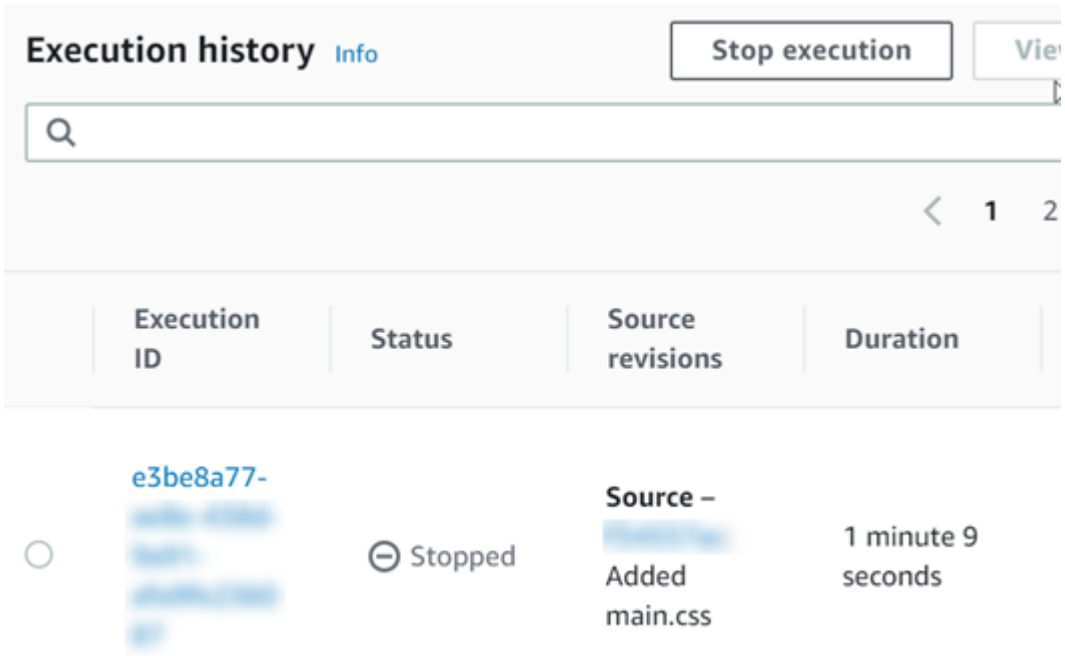
Option 1: Anhalten und warten

Wenn Sie sich für das Anhalten und Warten entscheiden, wird die ausgewählte Ausführung fortgesetzt, bis die laufenden Aktionen abgeschlossen sind. Beispielsweise wurde die folgende Pipeline-Ausführung angehalten, während die Build-Aktion lief.

1. In der Pipeline-Ansicht wird das Banner für Erfolgsmeldungen angezeigt, und die Build-Aktion wird fortgesetzt, bis sie abgeschlossen ist. Der Status der Pipeline-Ausführung ist Stopping (Wird angehalten).

   In der Verlaufsansicht ist der Status für laufende Aktionen, wie z. B. die Build-Aktion, In progress (In Bearbeitung), bis die Build-Aktion abgeschlossen ist. Während die Aktionen in Bearbeitung sind, ist der Status der Pipeline-Ausführung Stopping (Wird angehalten).
2. Die Ausführung wird angehalten, wenn der Anhaltevorgang abgeschlossen ist. Wenn die Build-Aktion erfolgreich abgeschlossen ist, hat sie den Status Succeeded (Erfolgreich abgeschlossen), und die Pipeline-Ausführung zeigt den Status Stopped (Angehalten). Nachfolgende Aktionen werden nicht gestartet. Die Schaltfläche Retry (Wiederholen) ist aktiviert.

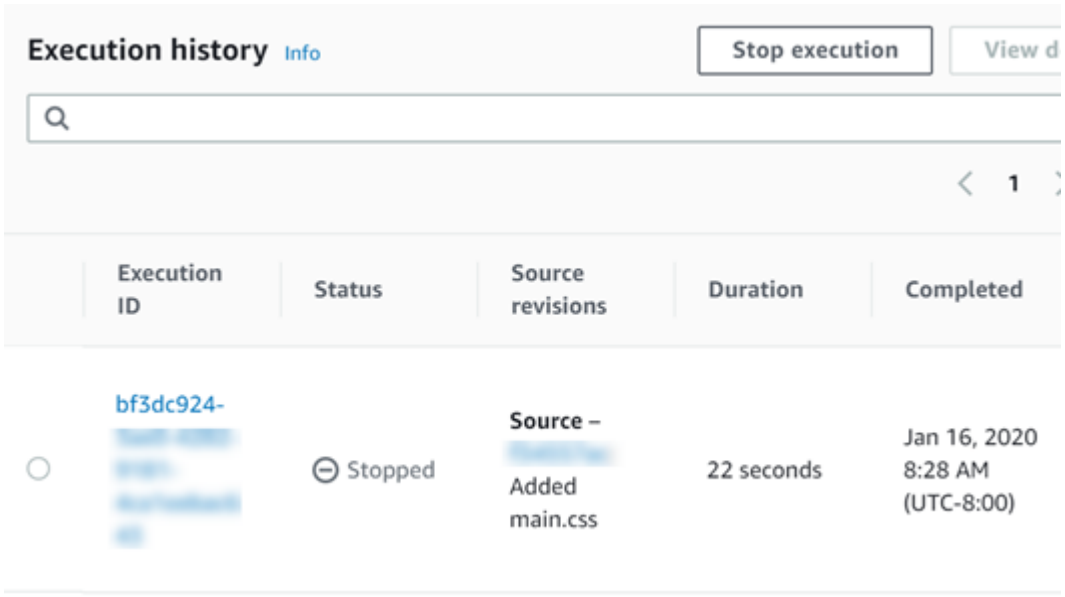
   In der Verlaufsansicht wird der Ausführungsstatus Stopped (Beendet) angezeigt, nachdem die laufende Aktion abgeschlossen ist.



Option 2: Anhalten und beenden

Wenn Sie sich zum Anhalten und Beenden entscheiden, wartet die ausgewählte Ausführung nicht auf den Abschluss der laufenden Aktionen. Die Aktionen werden beendet. Beispielsweise wurde die folgende Pipeline-Ausführung angehalten und beendet, während die Build-Aktion lief.

1. In der Pipeline-Ansicht wird die Erfolgsbanner-Meldung angezeigt, die Build-Aktion zeigt den Status In progress (In Bearbeitung) und die Pipeline-Ausführung den Status Stopping (Wird angehalten).
2. Nachdem die Pipeline-Ausführung beendet wurde, zeigt die Build-Aktion den Status Abandoned (Beendet) und die Pipeline-Ausführung den Status Stopped (Angehalten) an. Nachfolgende Aktionen werden nicht gestartet. Die Schaltfläche Retry (Wiederholen) ist aktiviert.
3. In der Verlaufsansicht ist der Ausführungsstatus Stopped (Beendet).



Nutzungsszenarien zum Beenden einer Pipeline-Ausführung

Wir empfehlen Ihnen, die Option „Anhalten und warten“ zu verwenden, um eine Pipeline-Ausführung zu beenden. Diese Option ist sicherer, da sie mögliche Fehlschläge oder out-of-sequence Aufgaben in Ihrer Pipeline vermeidet. Wenn eine Aktion abgebrochen wird CodePipeline, setzt der Aktionsanbieter alle Aufgaben im Zusammenhang mit der Aktion fort. Im Fall einer AWS CloudFormation Aktion wird die Bereitstellungsaktion in der Pipeline abgebrochen, aber das Stack-Update wird möglicherweise fortgesetzt und führt zu einem fehlgeschlagenen Update.

Ein Beispiel für aufgegebene Aktionen, die zu out-of-sequence Aufgaben führen können: Wenn Sie eine große Datei (1 GB) über eine S3-Bereitstellungsaktion bereitstellen und die Aktion beenden und abbrechen möchten, während die Bereitstellung bereits läuft, wird die Aktion in Amazon S3 abgebrochen CodePipeline, aber in Amazon S3 fortgesetzt. Amazon S3 erhält keine Anweisung, den Upload abzubrechen. Wenn Sie anschließend eine neue Pipeline-Ausführung mit einer sehr kleinen Datei starten, sind jetzt zwei Bereitstellungen im Gange. Da die Dateigröße der neuen Ausführung gering ist, wird die neue Bereitstellung abgeschlossen, während die alte Bereitstellung noch hochgeladen wird. Wenn die alte Bereitstellung abgeschlossen ist, wird die neue Datei durch die alte Datei überschrieben.

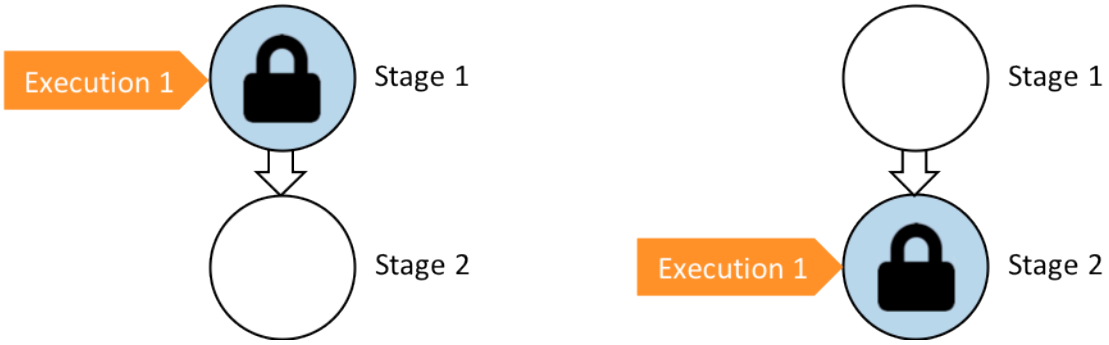
Sie können die Option zum Anhalten und Beenden verwenden, wenn Sie eine benutzerdefinierte Aktion haben. Sie können beispielsweise eine benutzerdefinierte Aktion abbrechen, deren Arbeit nicht abgeschlossen werden muss, bevor Sie eine neue Ausführung zur Behebung eines Fehlers starten.

Wie werden Ausführungen im Modus ABGELÖST verarbeitet

Der Standardmodus für die Verarbeitung von Ausführungen ist der Modus ABGELÖST. Eine Ausführung besteht aus einer Reihe von Änderungen, die von der Ausführung aufgenommen und verarbeitet werden. Pipelines können mehrere Ausführungen gleichzeitig verarbeiten. Jede Ausführung wird in der Pipeline getrennt ausgeführt. Die Pipeline verarbeitet die einzelnen Ausführung der Reihe nach und ersetzt möglicherweise frühere Ausführungen durch spätere Ausführungen. Die folgenden Regeln werden verwendet, um Ausführungen in einer Pipeline für den SUPERSEDED-Modus zu verarbeiten.

Regel 1: Phasen werden gesperrt, wenn eine Ausführung verarbeitet wird

Da jede Phase jeweils nur eine Ausführung verarbeiten kann, ist eine Phase während der Verarbeitung einer Ausführung gesperrt. Wenn die Ausführung eine Phase abgeschlossen hat, wechselt sie zur nächsten Phase in der Pipeline.

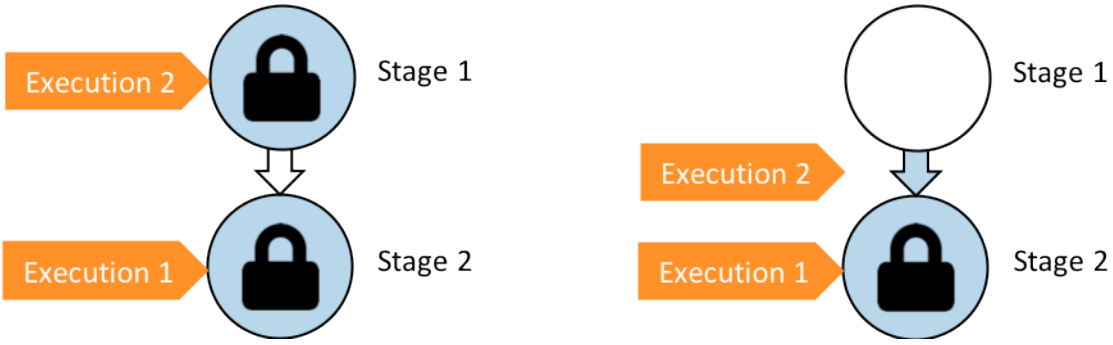


Vorher: Stage 1 is locked as Execution 1 enters. Nachher: Stage 2 is locked as Execution 1 enters.

Regel 2: Nachfolgende Ausführungen warten, bis die jeweilige Phase freigegeben wird

Wenn eine Phase gesperrt ist, warten nachfolgende Ausführungen vor der gesperrten Phase. Alle für eine Phase konfigurierten Aktionen müssen erfolgreich abgeschlossen sein, bevor die Phase als abgeschlossen gilt. Eine fehlgeschlagene Ausführung löst die Sperre der Phase auf. Wenn eine Ausführung beendet wird, wird die Ausführung in einer Phase nicht fortgesetzt und die Phase wird freigegeben.

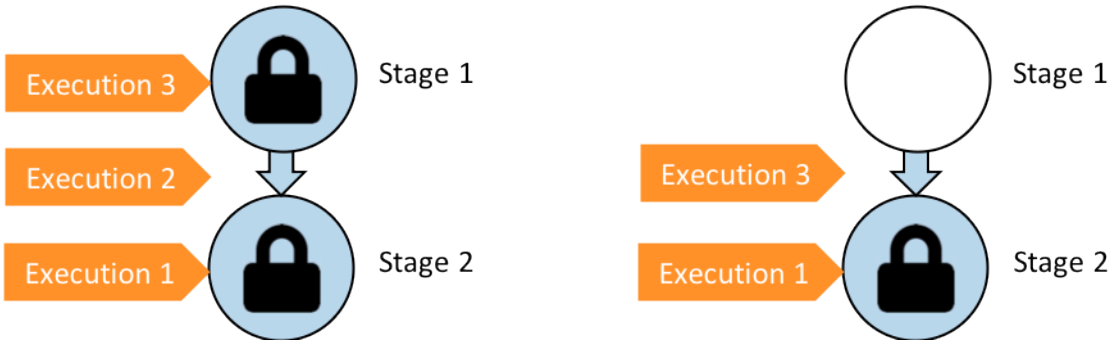
> **Note**
>
> Bevor Sie eine Ausführung beenden, empfehlen wir Ihnen, den Übergang vor der Phase zu deaktivieren. Auf diese Weise wird die Phase aufgrund der angehaltenen Ausführung entsperrt und akzeptiert keine nachfolgende Pipeline-Ausführung.



Vorher: Stage 2 is locked as Execution 1 enters. Nachher: Execution 2 exits Stage 1 and waits between stages.

Regel 3: Wartende Ausführungen werden durch neuere Ausführungen ersetzt

Ausführungen werden nur zwischen den Phasen ersetzt. Wenn eine Phase gesperrt ist, wartet eine nachfolgende Ausführung am Eingangspunkt der Phase, bis die Phase abgeschlossen ist. Eine neuere Ausführung holt eine wartende Ausführung ein und fährt zur nächsten Phase fort, sobald die Phase freigegeben wird. Die ersetzte Ausführung wird nicht fortgesetzt. In diesem Beispiel wurde Ausführung 2 durch Ausführung 3 ersetzt, während sie auf die Freigabe der gesperrten Phase wartet. Ausführung 3 tritt in die nächste Phase ein.



Vorher: Ausführung 2 wartet zwischen den Phasen, während Ausführung 3 in Phase 1 eintritt. Nachher: Ausführung 3 beendet Stufe 1. Ausführung 2 wird durch Ausführung 3 ersetzt.

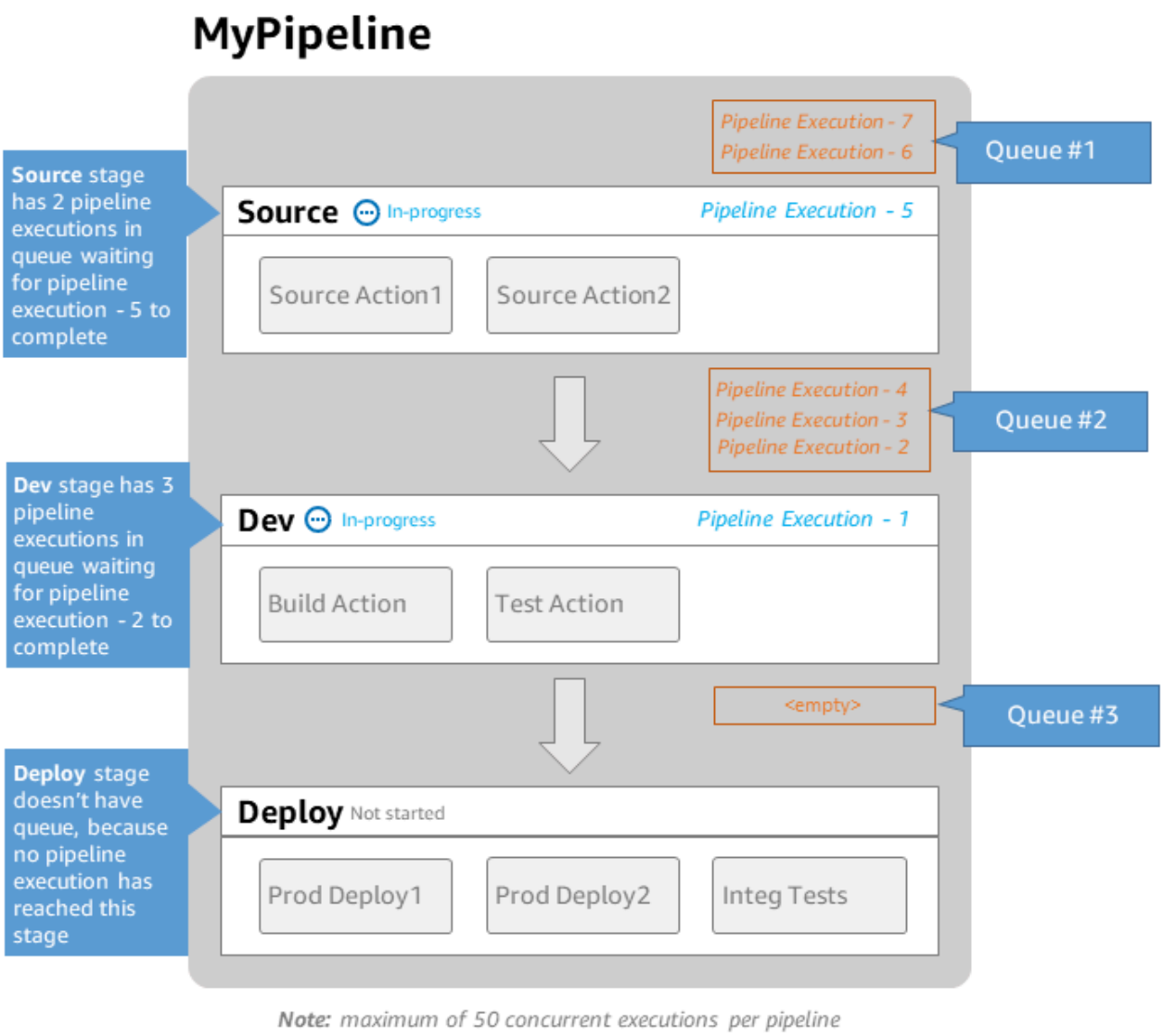
Weitere Hinweise zu Überlegungen beim Anzeigen und Umschalten zwischen den Ausführungsmodi finden Sie unterLegen Sie den Pipeline-Ausführungsmodus fest oder ändern Sie ihn. Weitere Informationen zu Kontingenten mit Ausführungsmodi finden Sie unterKontingente in AWS CodePipeline.

Wie werden Ausführungen im QUEUED-Modus verarbeitet

Bei Pipelines im QUEUED-Modus werden Phasen gesperrt, wenn eine Ausführung verarbeitet wird. Wartende Ausführungen überholen jedoch nicht Ausführungen, die bereits gestartet wurden.

Wartende Ausführungen sammeln sich an den Eintrittspunkten zu gesperrten Phasen in der Reihenfolge, in der sie die Phase erreichen, und bilden so eine Warteschlange wartender Ausführungen. Im QUEUED-Modus können Sie mehrere Warteschlangen in derselben Pipeline haben. Wenn eine Ausführung in der Warteschlange in eine Phase eintritt, ist die Phase gesperrt und es können keine weiteren Ausführungen mehr gestartet werden. Dieses Verhalten entspricht dem Modus SUPERSEDED. Wenn die Ausführung die Phase abgeschlossen hat, wird die Stufe entsperrt und ist bereit für die nächste Ausführung.

Das folgende Diagramm zeigt, wie Phasen in einer Pipeline im QUEUED-Modus die Ausführung verarbeiten. Während in der Quellphase beispielsweise Ausführung 5 verarbeitet wird, bilden die Ausführungen für 6 und 7 die Warteschlange #1 und warten am Startpunkt der Phase. Die nächste Ausführung in der Warteschlange wird verarbeitet, nachdem die Phase entsperrt wurde.



Weitere Informationen zu Überlegungen beim Anzeigen und Wechseln zwischen Ausführungsmodi finden Sie unter. Legen Sie den Pipeline-Ausführungsmodus fest oder ändern Sie ihn Weitere Informationen zu Kontingenten mit Ausführungsmodi finden Sie unterKontingente in AWS CodePipeline.

Wie werden Ausführungen im PARALLELEN Modus verarbeitet

Bei Pipelines im PARALLEL-Modus sind die Ausführungen unabhängig voneinander und warten nicht, bis andere Ausführungen abgeschlossen sind, bevor sie gestartet werden. Es gibt keine Warteschlangen. Verwenden Sie die Ausführungshistorie-Ansicht, um parallel Ausführungen in der Pipeline anzuzeigen.

Verwenden Sie den PARALLEL-Modus in Entwicklungsumgebungen, in denen jede Funktion über einen eigenen Funktionszweig verfügt und auf Zielen bereitgestellt wird, die nicht von anderen Benutzern gemeinsam genutzt werden.

Weitere Informationen zu Überlegungen beim Anzeigen und Wechseln zwischen Ausführungsmodi finden Sie unterLegen Sie den Pipeline-Ausführungsmodus fest oder ändern Sie ihn. Weitere Informationen zu Kontingenten mit Ausführungsmodi finden Sie unterKontingente in AWS CodePipeline.

Pipelinefluss verwalten

Der Ablauf von Pipeline-Ausführungen kann wie folgt gesteuert werden:

- Ein Übergang steuert den Fluss von Ausführungen in die Phase. Übergänge können aktiviert oder deaktiviert werden. Wenn ein Übergang deaktiviert ist, können Pipeline-Ausführungen nicht in die Phase übergehen. Die Pipeline-Ausführung, die darauf wartet, in eine Phase einzutreten, in der der Übergang deaktiviert ist, wird als eingehende Ausführung bezeichnet. Nachdem Sie den Übergang aktiviert haben, geht eine eingehende Ausführung in die Phase über und sperrt sie.

  Ähnlich wie bei Ausführungen, die auf die Freigabe einer gesperrten Phase warten, kann bei Deaktivierung eines Übergangs die Ausführung, die darauf wartet, in die Phase einzutreten, durch eine neue Ausführung ersetzt werden. Wenn ein deaktivierter Übergang erneut aktiviert wird, tritt die neueste Ausführung in die Phase ein. Dies gilt auch für Ausführungen, die ältere Ausführungen ersetzt haben, während der Übergang deaktiviert war.
- Eine Genehmigungsaktion, die verhindert, dass eine Pipeline zur nächsten Aktion übergeht, bis die entsprechende Genehmigung erteilt wurde (z. B. durch manuelle Genehmigung durch eine autorisierte Identität). Sie können eine Genehmigungsaktion beispielsweise verwenden, wenn Sie den Zeitpunkt steuern möchten, an dem eine Pipeline zur abschließenden Phase Production (Produktion) wechselt.

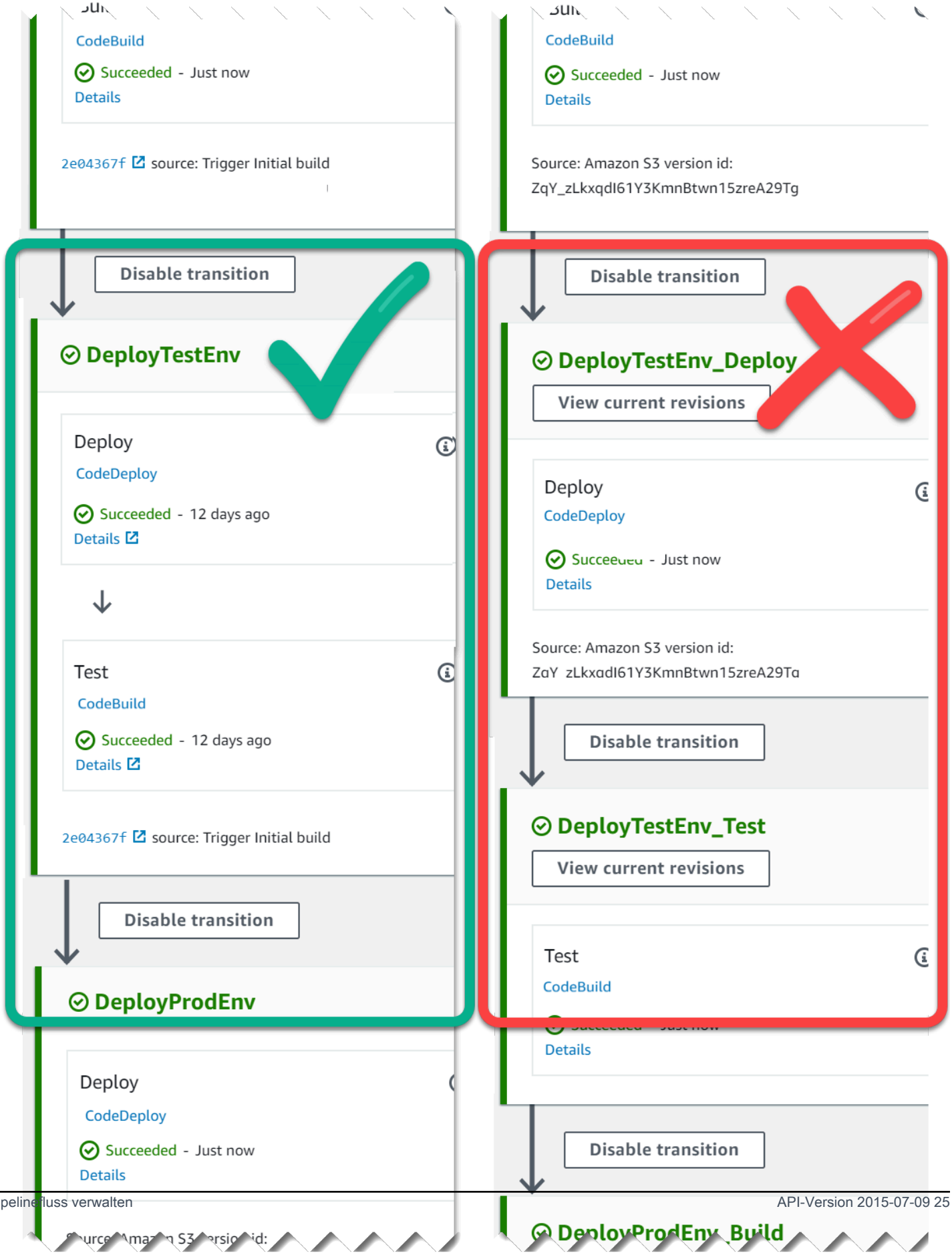
> **Note**
>
> Eine Phase mit einer Genehmigungsaktion ist gesperrt, bis die Genehmigungsaktion genehmigt oder abgelehnt wurde oder eine Zeitüberschreitung eintritt. Eine abgelaufene Genehmigungsaktion wird auf die gleiche Weise wie eine fehlgeschlagene Aktion verarbeitet.

- Ein Fehler bedeutet, dass eine Aktion in einer Phase nicht erfolgreich abgeschlossen wurde. Die Revision wird nicht zur nächsten Aktion in der Phase oder zur nächsten Phase in der Pipeline weitergeleitet. Folgendes kann auftreten:
  - Sie führen die Phase manuell erneut aus, die die fehlgeschlagenen Aktionen enthält. Hierdurch wird die Ausführung fortgesetzt (fehlgeschlagene Aktionen werden wiederholt und, wenn erfolgreich, in der Phase/Pipeline fortgesetzt).
  - Eine andere Ausführung tritt in die fehlgeschlagene Phase ein und ersetzt die fehlgeschlagene Ausführung. In diesem Fall kann die fehlgeschlagene Ausführung nicht wiederholt werden.

Empfohlene Pipeline-Struktur

Wenn Sie den Fluss einer Codeänderung durch die Pipeline festlegen, sollten Sie verwandte Aktionen innerhalb einer Phase gruppieren, sodass die Aktionen dieselbe Ausführung verarbeiten, wenn die Phase gesperrt wird. Sie können für jede Anwendungsumgebung AWS-Region, Availability Zone usw. eine Phase erstellen. Eine Pipeline mit zu vielen Phasen (die also zu detailliert definiert ist) kann zu viele gleichzeitige Änderungen ermöglichen. Eine Pipeline mit vielen Aktionen in einer großen Phase (die also zu grob definiert ist) kann zu viel Zeit benötigen, um eine Änderung freizugeben.

Beispiel: Eine Testaktion nach einer Bereitstellungsaktion in derselben Phase testet garantiert die Änderung, die bereitgestellt wurde. In diesem Beispiel wird eine Änderung in einer Testumgebung bereitgestellt und dann getestet. Dann wird die letzte Änderung aus der Testumgebung in eine Produktionsumgebung bereitgestellt. Im empfohlenen Beispiel handelt es sich bei Testumgebung und Produktionsumgebung um getrennte Phasen.


CodeBuild
Succeeded - Just now
Details
2e04367f source: Trigger Initial build
Disable transition
DeployTestEnv
Deploy
CodeDeploy
Succeeded - 12 days ago
Details
Test
CodeBuild
Succeeded - 12 days ago
Details
2e04367f source: Trigger Initial build
Disable transition
DeployProdEnv
Deploy
CodeDeploy
Succeeded - Just now
Details
CodeBuild
Succeeded - Just now
Details
Source: Amazon S3 version id:
ZqY_zLkxqdI61Y3KmnBtwn15zreA29Tg
Disable transition
DeployTestEnv_Deploy
View current revisions
Deploy
CodeDeploy
Succeeded - Just now
Details
Source: Amazon S3 version id:
ZqY_zLkxqdI61Y3KmnBtwn15zreA29Tg
Disable transition
DeployTestEnv_Test
View current revisions
Test
CodeBuild
Details
Disable transition
DeployProdEnv_Build

Links: Gruppierung von zuammengehörenden Test-, Bereitstellungs- und Genehmigungsaktionen (empfohlen). Rechts: Zusammengehörende Aktionen in getrennten Phasen (nicht empfohlen).

So funktionieren eingehende Ausführungen

Eine eingehende Ausführung ist eine Ausführung, die darauf wartet, dass eine Phase, ein Übergang oder eine Aktion verfügbar wird, die nicht verfügbar ist, bevor sie fortgeführt wird. Die nächste Phase, der Übergang oder die nächste Aktion ist möglicherweise aus folgenden Gründen nicht verfügbar:

- Eine weitere Exekution ist bereits in die nächste Phase eingetreten und wurde gesperrt.
- Der Übergang zum Eintritt in die nächste Phase ist deaktiviert.

Sie können einen Übergang deaktivieren, um eine eingehende Ausführung abzuhalten, wenn Sie kontrollieren möchten, ob eine aktuelle Ausführung Zeit hat, in nachfolgenden Phasen abgeschlossen zu werden, oder wenn Sie alle Aktionen an einem bestimmten Punkt beenden möchten. Um festzustellen, ob es sich um eine eingehende Ausführung handelt, können Sie sich die Pipeline in der Konsole oder die Ausgabe des Befehls ansehen. get-pipeline-state

Bei eingehenden Ausführungen sind die folgenden Überlegungen zu beachten:

- Sobald die Aktions-, Übergangs- oder Sperrphase verfügbar ist, geht die laufende eingehende Ausführung in die Phase über und durchläuft die Pipeline.
- Solange die eingehende Ausführung wartet, kann sie manuell gestoppt werden. Eine eingehende Ausführung kann den Status `InProgressStopped`, oder `Failed` haben.
- Wenn eine eingehende Ausführung gestoppt wurde oder fehlgeschlagen ist, kann sie nicht erneut versucht werden, da es keine fehlgeschlagenen Aktionen gibt, die wiederholt werden könnten. Wenn eine eingehende Ausführung gestoppt wurde und der Übergang aktiviert ist, wird die gestoppte eingehende Ausführung nicht in der Phase fortgesetzt.

Sie können eine eingehende Ausführung anzeigen oder beenden.

Eingabe- und Ausgabe-Artefakte

CodePipeline lässt sich in Entwicklungstools integrieren, um nach Codeänderungen zu suchen und anschließend in allen Phasen des Continuous Delivery-Prozesses für die Erstellung und Bereitstellung zu sorgen. Artefakte sind Dateien, die durch Aktionen in der Pipeline bearbeitet

werden, z. B. Dateien oder Ordner mit Anwendungscode, Indexseitendateien, Skripts usw. Das Amazon S3 S3-Artefakt für die Quellaktion ist beispielsweise ein Dateiname (oder Dateipfad), in dem die Quellcodedateien der Anwendung für die Pipeline-Quellaktion bereitgestellt werden, und die Dateien werden im Allgemeinen als ZIP-Datei bereitgestellt, z. B. der folgende Beispielartefaktname: SampleApp _Windows.zip. Das Ausgabeartefakt für die Quellaktion, die Quellcodedateien der Anwendung, sind das Ausgabeartefakt der Quellaktion und auch das Eingabeartefakt für die nächste Aktion, z. B. eine Build-Aktion. Ein weiteres Beispiel: Eine Build-Aktion könnte Build-Befehle ausführen, die den Anwendungsquellcode für ein Eingabeartefakt kompilieren, bei dem es sich um die Anwendungsquellcodedateien aus der Quellaktion handelt. Einzelheiten zu Artefaktparametern, z. B. für die Aktion, finden Sie auf der Referenzseite zur Aktionskonfiguration AWS CodeBuild für eine bestimmte Aktion. CodeBuild

Aktionen verwenden Eingabe- und Ausgabeartefakte, die in dem Amazon S3-Artefakt-Bucket gespeichert sind, den Sie bei der Erstellung der Pipeline ausgewählt haben. CodePipeline komprimiert und überträgt die Dateien für Eingabe- oder Ausgabeartefakte entsprechend dem Aktionstyp in der Phase.

> **Note**
>
> Der Artefakt-Bucket ist nicht derselbe Bucket wie der Bucket, der als Speicherort der Quelldatei für eine Pipeline verwendet wird, in der S3 als Quellaktion ausgewählt wurde.

Beispielsweise:

1. CodePipeline löst die Ausführung Ihrer Pipeline aus, wenn ein Commit in das Quell-Repository erfolgt, und stellt das Ausgabeartefakt (alle zu erstellenden Dateien) aus dem Source-Schritt bereit.
2. Das Ausgabe-Artefakt (alle Dateien, die erstellt werden sollen) aus dem vorherigen Schritt wird als Eingabe-Artefakt in die Phase Erstellen aufgenommen. Ein Ausgabe-Artefakt (die Build-Anwendung) aus der Phase Build kann eine aktualisierte Anwendung oder ein in einem Container erstelltes aktualisiertes Docker-Image sein.
3. Das Ausgabeartefakt aus dem vorherigen Schritt (die erstellte Anwendung) wird als Eingabeartefakt in die Bereitstellungsphase aufgenommen, z. B. für Staging- oder Produktionsumgebungen in der. AWS Cloud Sie können Anwendungen für die Bereitstellungsflotte bereitstellen, oder Sie können Container-basierte Anwendungen für in ECS-Clustern ausgeführte Aufgaben bereitstellen.

Wenn Sie eine Aktion erstellen oder bearbeiten, legen Sie das Eingabe- und Ausgabe-Artefakt oder die Eingabe- und Ausgabe-Artefakte für die Aktion fest. Beispiel: Für eine zweistufige Pipeline mit den Stufen „Source“ und „Deploy“ wählen Sie in „Aktion bearbeiten“ den Artefaktnamen der Quellaktion für das Eingabeartefakt für die Bereitstellungsaktion aus.

- Wenn Sie die Konsole verwenden, um Ihre erste Pipeline zu erstellen, CodePipeline erstellt in derselben einen Amazon S3 S3-Bucket AWS-Konto und AWS-Region speichert Artikel für alle Pipelines. Jedes Mal, wenn Sie die Konsole verwenden, um eine weitere Pipeline in dieser Region zu erstellen, CodePipeline wird im Bucket ein Ordner für diese Pipeline erstellt. Dieser Ordner wird zum Speichern von Artefakten für Ihre Pipeline verwendet, während der automatisierte Veröffentlichungsprozess ausgeführt wird. Dieser Bucket trägt den Namen codepipeline- *region* — *12345EXAMPLE*, wobei *Region die Region* ist, in der Sie die AWS Pipeline erstellt haben, und *12345EXAMPLE* eine 12-stellige Zufallszahl ist, die sicherstellt, dass der Bucket-Name eindeutig ist.

  > **Note**
  >
  > Wenn Sie in der Region, in der Sie die Pipeline erstellen, bereits einen Bucket haben, der mit codepipeline- region beginnt, CodePipeline verwendet diesen als Standard-Bucket. Außerdem folgt er der lexikografischen Reihenfolge. Beispielsweise wird codepipeline-region-abcexample vor codepipeline- region-defexample ausgewählt.

  CodePipeline kürzt Artefaktnamen, was dazu führen kann, dass einige Bucket-Namen ähnlich aussehen. Auch wenn der Artefaktname gekürzt zu sein scheint, wird er dem CodePipeline Artefakt-Bucket auf eine Weise zugeordnet, die von Artefakten mit gekürzten Namen nicht beeinträchtigt wird. Die Pipeline kann normal ausgeführt werden. Dies ist kein Problem mit dem Ordner oder den Artefakten. Für Pipelinenamen besteht eine Längenbegrenzung auf 100 Zeichen. Auch wenn der Name des Artefaktordners gekürzt angezeigt wird, ist er für Ihre Pipeline eindeutig.

  Wenn Sie eine Pipeline erstellen oder bearbeiten, müssen Sie einen Artefakt-Bucket in der Pipeline haben AWS-Konto und Sie müssen über einen Artefakt-Bucket pro Region verfügen AWS-Region, in der Sie eine Aktion ausführen möchten. Wenn Sie die Konsole verwenden, um eine Pipeline oder regionsübergreifende Aktionen zu erstellen, werden Standardartefact-Buckets CodePipeline in den Regionen konfiguriert, in denen Sie Aktionen haben.

  Wenn Sie die verwenden, AWS CLI um eine Pipeline zu erstellen, können Sie die Artefakte für diese Pipeline in einem beliebigen Amazon S3 S3-Bucket speichern, sofern sich dieser Bucket

im selben AWS-Konto und AWS-Region wie die Pipeline befindet. Sie können dies tun, wenn Sie befürchten, die für Ihr Konto zulässigen Amazon S3 S3-Buckets zu überschreiten. Wenn Sie die verwenden, AWS CLI um eine Pipeline zu erstellen oder zu bearbeiten, und Sie eine regionsübergreifende Aktion hinzufügen (eine Aktion mit einem AWS Anbieter in einer anderen Region als Ihrer Pipeline), müssen Sie für jede weitere Region, in der Sie eine Aktion ausführen möchten, einen Artefakt-Bucket bereitstellen.

- Jede Aktion hat einen Typ. Je nach Typ kann die Aktion über eine oder beide der folgenden Arten von Artefakten verfügen:
  - Ein Eingabeartefakt, bei dem es sich um das Artefakt handelt, das die Aktion während ihrer Ausführung nutzt oder bearbeitet.
  - Ein Ausgabeartefakt, bei dem es sich um das Ergebnis der Aktion handelt.

  Jedes Ausgabeartefakt in einer Pipeline muss einen eindeutigen Namen haben. Jedes Eingabeartefakt einer Aktion muss dem Ausgabeartefakt einer früheren Aktion in der Pipeline entsprechen. Dabei kann es sich um eine Aktion handeln, die unmittelbar vor der betreffenden Aktion in derselben Phase oder in einer früheren Phase ausgeführt wurde.

  Ein Artefakt kann von mehr als einer Aktion bearbeitet werden.

Pipeline-Typen

CodePipeline bietet die folgenden Pipeline-Typen, die sich in Eigenschaften und Preis unterscheiden, sodass Sie Ihre Pipeline-Funktionen und Kosten an die Anforderungen Ihrer Anwendungen anpassen können.

- Pipelines vom Typ V1 haben eine JSON-Struktur, die Standardparameter auf Pipeline-, Stufen- und Aktionsebene enthält.
- Pipelines vom Typ V2 haben dieselbe Struktur wie Pipelines vom Typ V1 und verfügen über zusätzliche Parameter für die Releasesicherheit und die Triggerkonfiguration.

Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

Auf der CodePipeline Referenz zur Pipeline-Struktur Seite finden Sie Einzelheiten zu den Parametern in den einzelnen Pipeline-Typen. Informationen darüber, welcher Pipeline-Typ Sie wählen sollten, finden Sie unterWelcher Pipeline-Typ ist der richtige für mich?.

Welcher Pipeline-Typ ist der richtige für mich?

Der Pipeline-Typ wird durch die Eigenschaften und Funktionen bestimmt, die von jeder Pipeline-Version unterstützt werden.

Im Folgenden finden Sie eine Zusammenfassung der Anwendungsfälle und Merkmale, die für jeden Pipeline-Typ verfügbar sind.

	Typ V1	Typ V2
Merkmale		
Anwendungsfälle	• Standardbereitstellungen	• Bereitstellungen mit Konfiguration durch Übergabe von Variablen auf Pipeline-Ebene zur Laufzeit • Bereitstellungen, bei denen Pipelines so konfiguriert sind, dass sie mit Git-Tags starten
Variablen auf Aktionsebene	Unterstützt	Unterstützt
PARALLELER Ausführungsmodus	Nicht unterstützt	Unterstützt
Variablen auf Pipeline-Ebene	Nicht unterstützt	Unterstützt
Ausführungsmodus „QUEUED“	Nicht unterstützt	Unterstützt
Rollback für Pipeline-Phasen	Nicht unterstützt	Unterstützt
Überschreibungen der Quellversion	Nicht unterstützt	Unterstützt
Triggert und filtert Git-Tags, Pull-Requests, Branches oder Dateipfade	Nicht unterstützt	Unterstützt

Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

Verwenden Sie das folgende Skript für eine Pipeline vom Typ V1, um die Kosten für die Umstellung der Pipeline auf eine Pipeline vom Typ V2 zu analysieren.

Um eine Kostenanalyse für Pipeline-Typen durchzuführen (Skript)

1. Öffnen Sie ein Terminal-Fenster. Führen Sie den folgenden Befehl aus, um ein neues Python-Skript namens PipelineCostAnalyzer.py zu erstellen.

```
vi PipelineCostAnalyzer.py
```

2. Kopieren Sie den folgenden Code und fügen Sie ihn in das PipelineCostAnalyzer.py-Skript ein.

```
import boto3
import sys
import math
from datetime import datetime, timedelta, timezone

if len(sys.argv) < 3:
    raise Exception("Please provide region name and pipeline name as arguments.
 Example usage: python PipelineCostAnalyzer.py us-east-1 MyPipeline")
session = boto3.Session(profile_name='default', region_name=sys.argv[1])
pipeline = sys.argv[2]
codepipeline = session.client('codepipeline')

def analyze_cost_in_v2(pipeline_name):
    if codepipeline.get_pipeline(name=pipeline)['pipeline']['pipelineType'] ==
 'V2':
        raise Exception("Provided pipeline is already of type V2.")
    total_action_executions = 0
    total_blling_action_executions = 0
    total_action_execution_minutes = 0
    cost = 0.0
    hasNextToken = True
    nextToken = ""

    while hasNextToken:
        if nextToken=="":
            response =
 codepipeline.list_action_executions(pipelineName=pipeline_name)
        else:
```

```
            response =
 codepipeline.list_action_executions(pipelineName=pipeline_name,
 nextToken=nextToken)
        if 'nextToken' in response:
            nextToken = response['nextToken']
        else:
            hasNextToken= False
        for action_execution in response['actionExecutionDetails']:
            start_time = action_execution['startTime']
            end_time = action_execution['lastUpdateTime']
            if (start_time < (datetime.now(timezone.utc) - timedelta(days=30))):
                hasNextToken= False
                continue
            total_action_executions += 1
            if (action_execution['status'] in ['Succeeded', 'Failed', 'Stopped']):
                action_owner = action_execution['input']['actionTypeId']['owner']
                action_category = action_execution['input']['actionTypeId']
['category']
                if (action_owner == 'Custom' or (action_owner == 'AWS' and
 action_category == 'Approval')):
                    continue

                total_blling_action_executions += 1
                action_execution_minutes = (end_time -
 start_time).total_seconds()/60
                action_execution_cost = math.ceil(action_execution_minutes) * 0.02
                total_action_execution_minutes += action_execution_minutes
                cost = round(cost + action_execution_cost, 2)

    print ("{:<40}".format('Activity in last 30 days:'))
    print ("| {:<40} | {:<10}".format('_____________________________________',
 '____________________'))
    print ("| {:<40} | {:<10}".format('Total action executions:',
 total_action_executions))
    print ("| {:<40} | {:<10}".format('Total billing action executions:',
 total_blling_action_executions))
    print ("| {:<40} | {:<10}".format('Total billing action execution minutes:',
 round(total_action_execution_minutes, 2)))
    print ("| {:<40} | {:<10}".format('Cost of moving to V2 in $:', cost - 1))

analyze_cost_in_v2(pipeline)
```

3. Führen Sie das Skript für die V1-Pipeline Ihrer Wahl in einer bestimmten AWS-Region Umgebung aus.

Führen Sie den folgenden Befehl aus, um das Python-Skript mit dem Namen PipelineCostAnalyzer.py auszuführen. In diesem Beispiel ist die Region us-west-2.

```
python3 PipelineCostAnalyzer.py us-west-2
```

4. In der folgenden Beispielausgabe des Skripts sehen wir die Liste der Aktionsausführungen, die Liste der Aktionsausführungen, die für die Abrechnung in Frage kamen, die Gesamtlaufzeit dieser Aktionsausführungen und schließlich die Kosten für die Aktualisierung der Pipeline auf den Typ V2.

```
Activity in last 30 days:
 | ____________________________________      | ___________________
 | Total action executions:                  | 9
 | Total billing action executions:          | 9
 | Total billing action execution minutes:   | 5.59
 | Cost of moving to V2 in $:                | -0.76
```

> **Note**
>
> In diesem Beispiel steht der negative Wert in der letzten Zeile für den Betrag, der durch die Umstellung auf Pipelines vom Typ V2 eingespart werden kann.

Erste Schritte mit CodePipeline

Wenn Sie noch keine Erfahrung damit haben CodePipeline, können Sie den Tutorials in diesem Handbuch folgen, nachdem Sie die Schritte in diesem Kapitel zur Einrichtung befolgt haben.

Die CodePipeline Konsole enthält hilfreiche Informationen in einem zusammenklappbaren Bereich, den Sie über das Informationssymbol oder einen beliebigen Informationslink auf der Seite öffnen können.

().

Sie können dieses Fenster jederzeit schließen.

Die CodePipeline Konsole bietet auch eine Möglichkeit, schnell nach Ihren Ressourcen wie Repositorys, Build-Projekten, Bereitstellungsanwendungen und Pipelines zu suchen. Wählen Sie Go to Ressource (Zur Ressource) oder drücken Sie die Taste / und geben Sie dann den Namen der Ressource ein. Alle Übereinstimmungen werden in der Liste angezeigt. Bei der Suche wird nicht zwischen Groß- und Kleinschreibung unterschieden. Sie sehen nur die Ressourcen, für die Sie die Berechtigung besitzen. Weitere Informationen finden Sie unter Anzeigen von Ressourcen in der Konsole.

Bevor Sie es AWS CodePipeline zum ersten Mal verwenden können, müssen Sie Ihren AWS-Konto und Ihren ersten Administratorbenutzer erstellen.

Themen

Schritt 1: Erstellen Sie einen AWS-Konto Benutzer mit Administratorrechten

Melde dich an für einen AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie einen Benutzer mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.

2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity Center Benutzerhandbuch.

Melden Sie sich als Benutzer mit Administratorzugriff an

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugriffsportal.

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen finden Sie im Benutzerhandbuch unter Einen Berechtigungssatz erstellen.AWS IAM Identity Center

2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Anweisungen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen.

Schritt 2: Wenden Sie eine verwaltete Richtlinie für den Administratorzugriff an auf CodePipeline

Sie müssen Berechtigungen für die Interaktion mit gewähren CodePipeline. Der schnellste Weg, dies zu tun, besteht darin, die `AWSCodePipeline_FullAccess` verwaltete Richtlinie auf den Administratorbenutzer anzuwenden.

> **Note**
>
> Die `AWSCodePipeline_FullAccess` Richtlinie umfasst Berechtigungen, die es dem Konsolenbenutzer ermöglichen, eine IAM-Rolle an eine andere Person zu CodePipeline übergeben. AWS-Services Damit kann der Service die Rolle annehmen und in Ihrem Auftrag Aktionen ausführen. Wenn Sie die Richtlinie an einen Benutzer, eine Rolle oder eine Gruppe anfügen, werden die `iam:PassRole`-Berechtigungen angewendet. Stellen Sie sicher, dass die Richtlinie nur auf vertrauenswürdige Benutzer angewendet wird. Wenn Benutzer mit diesen Berechtigungen die Konsole zum Erstellen oder Bearbeiten einer Pipeline verwenden, stehen die folgenden Optionen zur Verfügung:
>
> - Erstellen Sie eine CodePipeline Servicerolle oder wählen Sie eine bestehende aus und übergeben Sie die Rolle an CodePipeline
> - Sie könnten sich dafür entscheiden, eine CloudWatch Ereignisregel für die Erkennung von Änderungen zu erstellen und die Servicerolle CloudWatch Ereignisse an CloudWatch Ereignisse zu übergeben
>
> Weitere Informationen finden Sie unter Einem Benutzer Berechtigungen zur Übergabe einer Rolle an einen gewähren AWS-Service.

> **Note**
>
> Die `AWSCodePipeline_FullAccess` Richtlinie bietet Zugriff auf alle CodePipeline Aktionen und Ressourcen, auf die der IAM-Benutzer Zugriff hat, sowie auf alle möglichen Aktionen beim Erstellen von Phasen in einer Pipeline, z. B. beim Erstellen von Phasen CodeDeploy, die Elastic Beanstalk oder Amazon S3 beinhalten. Als bewährte Methode sollten Sie Personen nur die Berechtigungen erteilen, die sie für ihre Aufgaben benötigen. Weitere Informationen dazu, wie Sie IAM-Benutzer auf eine begrenzte Anzahl von CodePipeline Aktionen und Ressourcen beschränken können, finden Sie unter. Entfernen von Berechtigungen aus der CodePipeline-Servicerolle

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.

- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Schritt 3: Installieren Sie das AWS CLI

Um CodePipeline Befehle von der AWS CLI auf einem lokalen Entwicklungscomputer aufzurufen, müssen Sie die AWS CLI installieren. Dieser Schritt ist optional, wenn Sie zunächst nur die Schritte in diesem Handbuch für die CodePipeline Konsole verwenden möchten.

Um das zu installieren und zu konfigurieren AWS CLI

1. Laden Sie auf Ihrem lokalen Computer den herunter und installieren Sie ihn AWS CLI. Auf diese Weise können Sie CodePipeline von der Befehlszeile aus mit interagieren. Weitere Informationen finden Sie unter Einrichtung der AWS Befehlszeilenschnittstelle.

   > **Note**
   >
   > CodePipeline funktioniert nur mit den AWS CLI Versionen 1.7.38 und höher. Führen Sie den Befehl aus, um festzustellen AWS CLI , welche Version von Sie möglicherweise installiert haben. aws --version Um eine ältere Version von AWS CLI auf die neueste Version zu aktualisieren, folgen Sie den Anweisungen unter Deinstallieren von und dann den Anweisungen unter Installieren von. AWS CLI AWS Command Line Interface

2. Konfigurieren Sie das AWS CLI mit dem configure Befehl wie folgt:

   ```
   aws configure
   ```

Wenn Sie dazu aufgefordert werden, geben Sie den AWS Zugriffsschlüssel und den AWS geheimen Zugriffsschlüssel des IAM-Benutzers an, den Sie mit CodePipeline verwenden möchten. Wenn Sie den Standard-Regionennamen eingeben sollen, geben Sie den Namen Region an, in der Sie die Pipeline erstellen (z. B. `us-east-2`). Wenn Sie nach dem standardmäßigen Ausgabeformat gefragt werden, geben Sie `json` an. Beispielsweise:

```
AWS Access Key ID [None]: Type your target AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your target AWS secret access key here, and then
 press Enter
Default region name [None]: Type us-east-2 here, and then press Enter
Default output format [None]: Type json here, and then press Enter
```

> **Note**
>
> Weitere Informationen zu IAM, Zugriffsschlüsseln und geheimen Schlüsseln finden Sie unter Zugriffsschlüssel für IAM-Benutzer verwalten und Wie erhalte ich Anmeldeinformationen? .
> Weitere Informationen zu den Regionen und Endpunkten CodePipeline, für die verfügbar sind, finden Sie unter AWS CodePipeline Endpunkte und Kontingente.

Schritt 4: Öffnen Sie die Konsole für CodePipeline

- Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

Nächste Schritte

Sie haben die Voraussetzungen erfüllt. Sie können mit der Verwendung beginnen CodePipeline. Um mit der Arbeit zu beginnen CodePipeline, sehen Sie sich das anCodePipeline Anleitungen.

Produkt- und Serviceintegrationen mit CodePipeline

Standardmäßig AWS CodePipeline ist es in eine Reihe von Produkten AWS-Services und Dienstleistungen von Partnern integriert. Verwenden Sie die Informationen in den folgenden Abschnitten, um Sie bei der Konfiguration CodePipeline für die Integration mit den von Ihnen verwendeten Produkten und Diensten zu unterstützen.

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit diesem Service.

Themen

Integrationen mit CodePipeline Aktionstypen

Die Integrationsinformationen in diesem Thema sind nach CodePipeline Aktionstypen geordnet.

Themen

Quellaktions-Integrationen

Die folgenden Informationen sind nach CodePipeline Aktionstyp geordnet und können Ihnen bei der Konfiguration CodePipeline der Integration mit den folgenden Quellaktionsanbietern helfen.

Themen

- Verbindungen zu Bitbucket Cloud GitHub (Version 2), GitHub Enterprise Server, GitLab .com und Selbstverwaltung GitLab
- CodeCommit Quellaktionen
- GitHub (Version 1) Quellaktionen

Amazon ECR-Quellaktionen

Amazon ECR ist ein AWS Docker-Image-Repository-Service. Sie verwenden Docker-Push- und Pull-Befehle zum Hochladen von Docker-Images auf Ihr Repository. Eine Amazon ECR-Repository-URI und ein Bild werden in Amazon ECS-Aufgabendefinitionen verwendet, um Quell-Image-Informationen zu referenzieren.

Weitere Informationen:

- Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter Amazon ECR
- Erstellen Sie eine Pipeline in CodePipeline
- Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy

Amazon S3 S3-Quellaktionen

Amazon S3 ist Speicher für das Internet. Mit Amazon S3 können Sie jederzeit beliebige Mengen von Daten von überall aus im Internet speichern und aufrufen. Sie können so konfigurieren CodePipeline , dass ein versionierter Amazon S3 S3-Bucket als Quellaktion für Ihren Code verwendet wird.

> **Note**
>
> Amazon S3 kann auch als Bereitstellungsaktion in eine Pipeline aufgenommen werden.

Weitere Informationen:

- Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter Amazon S3 S3-Quellaktion
- Schritt 1: Erstellen eines S3-Buckets für Ihre Anwendung
- Erstellen einer Pipeline (CLI)

- CodePipeline verwendet Amazon EventBridge (früher Amazon CloudWatch Events), um Änderungen in Ihrem Amazon S3 S3-Quell-Bucket zu erkennen. Siehe Allgemeine Integrationen mit CodePipeline.

Verbindungen zu Bitbucket Cloud GitHub (Version 2), GitHub Enterprise Server, GitLab .com und Selbstverwaltung GitLab

Verbindungen (`CodeStarSourceConnection`Aktionen) werden verwendet, um auf deine Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com oder GitLab dein selbstverwaltetes Repository eines Drittanbieters zuzugreifen.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Afrika (Kapstadt), Naher Osten (Bahrain), Naher Osten (VAE), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv) oder AWS GovCloud (US-West) nicht verfügbar. Informationen zu anderen verfügbaren Aktionen finden Sie unter. Produkt- und Serviceintegrationen mit CodePipeline Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Bitbucket Cloud	Du kannst so konfigurieren CodePipeline , dass ein Bitbucket Cloud-Rep ository als Quelle für deinen Code verwendet wird. Du musst zuvor ein Bitbucket-Konto und mindestens ein Bitbucket Cloud-Repository erstellt haben. Du kannst eine Quellaktion für dein Bitbucket Cloud-Repository hinzufügen, indem du entweder eine Pipeline erstellst oder eine bestehende bearbeitest. **Note** Sie können Verbindungen mit einem Bitbucket-Cloud-Repository erstellen. Installierte Bitbucket-Anbietertypen wie Bitbucket Server werden nicht unterstützt.

	Sie können Ressourcen, die als Verbindungen bezeichnet werden, einrichten, damit Ihre Pipelines auf Code-Repositorys von Drittanbietern zugreifen können. Wenn du eine Verbindung herstellst, installierst du die AWS CodeStar App mit deinem Code-Repository eines Drittanbieters und ordnest sie dann deiner Verbindung zu. Verwende für Bitbucket Cloud die Bitbucket-Option in der Konsole oder die `CodestarSourceConnection` Aktion in der CLI. Siehe Bitbucket Cloud-Verbindungen. Sie können die Option Vollständiges Klonen für diese Aktion verwenden, um auf die Git-Metadaten des Repositorys zu verweisen, sodass nachgelagerte Aktionen Git-Befehle direkt ausführen können. Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden. Weitere Informationen: • Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter. CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen • Ein Tutorial „Erste Schritte“, in dem eine Pipeline mit einer Bitbucket Cloud-Quelle erstellt wird, findest du unter Erste Schritte mit Verbindungen.
GitHub oder GitHub Enterprise Cloud	Sie können so konfigurieren CodePipeline , dass ein GitHub Repository als Quelle für Ihren Code verwendet wird. Sie müssen zuvor ein GitHub Konto und mindestens ein GitHub Repository erstellt haben. Sie können eine Quellaktion für Ihr GitHub Repository hinzufügen, indem Sie entweder eine Pipeline erstellen oder eine bestehende bearbeiten. Sie können Ressourcen, die als Verbindungen bezeichnet werden, einrichten, damit Ihre Pipelines auf Code-Repositorys von Drittanbietern zugreifen können. Wenn Sie eine Verbindung herstellen, installieren Sie die AWS CodeStar App mit Ihrem Code-Repository eines Drittanbieters und verknüpfen sie dann mit Ihrer Verbindung.

	Verwenden Sie die Provider-Option GitHub (Version 2) in der Konsole oder die `CodestarSourceConnection` Aktion in der CLI. Siehe GitHub Verbindungen. Sie können die Option Vollständiges Klonen für diese Aktion verwenden, um auf die Git-Metadaten des Repositorys zu verweisen, sodass nachgelagerte Aktionen Git-Befehle direkt ausführen können. Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden. Weitere Informationen: • Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen • Ein Tutorial, das Ihnen zeigt, wie Sie eine Verbindung zu einem GitHub Repository herstellen und die Option Vollständiges Klonen verwenden, finden Sie unter. Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden • Die aktuelle GitHub Aktion ist die Quellaktion für GitHub Version 2. Die GitHub Aktion der Version 1 wird mit der OAuth-Token-Authentifizierung verwaltet. Wir empfehlen zwar nicht, die Aktion GitHub Version 1 zu verwenden, aber bestehende Pipelines mit der Aktion GitHub Version 1 funktionieren weiterhin ohne Auswirkungen. Sie können jetzt eine CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen Quellaktion in Ihrer Pipeline verwenden, die Ihre GitHub Quellaktion mit GitHub Apps verwaltet. Wenn Sie eine Pipeline haben, die die GitHub Aktion Version 1 verwendet, lesen Sie die Schritte zur Aktualisierung, um sie auf die Verwendung einer GitHub Aktion von Version 2 zu aktualisierenAktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2.
GitHub Unternehmensserver	Sie können so konfigurieren CodePipeline , dass ein GitHub Enterprise Server-Repository als Quelle für Ihren Code verwendet wird. Sie müssen zuvor ein GitHub Konto und mindestens ein GitHub Repository erstellt haben.

Sie können eine Quellaktion für Ihr GitHub Enterprise Server-Repository hinzufügen, indem Sie entweder eine Pipeline erstellen oder eine bestehende bearbeiten.

Sie können Ressourcen, die als Verbindungen bezeichnet werden, einrichte n, damit Ihre Pipelines auf Code-Repositorys von Drittanbietern zugreifen können. Wenn Sie eine Verbindung herstellen, installieren Sie die AWS CodeStar App mit Ihrem Code-Repository eines Drittanbieters und verknüpfen sie dann mit Ihrer Verbindung.

Verwenden Sie die GitHub Enterprise Server Provider-Option in der Konsole oder die `CodestarSourceConnection` Aktion in der CLI. Siehe GitHub Enterprise Server-Verbindungen.

> **Important**
>
> AWS CodeStar Connections unterstützt GitHub Enterprise Server Version 2.22.0 aufgrund eines bekannten Problems in der Version nicht. Um eine Verbindung zu erstellen, aktualisieren Sie auf Version 2.22.1 bzw. die neueste verfügbare Version.

Sie können die Option Vollständiges Klonen für diese Aktion verwenden, um auf die Git-Metadaten des Repositorys zu verweisen, sodass nachgelagerte Aktionen Git-Befehle direkt ausführen können. Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

Weitere Informationen:

- Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen
- Ein Tutorial, das Ihnen zeigt, wie Sie eine Verbindung zu einem GitHub Repository herstellen und die Option Vollständiges Klonen verwenden, finden Sie unter. Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden

GitLab.com	Sie können so konfigurieren CodePipeline , dass ein GitLab .com-Repo sitory als Quelle für Ihren Code verwendet wird. Sie müssen zuvor GitLab ein.com-Konto und mindestens ein GitLab .com-Repository erstellt haben. Sie können eine Quellaktion für Ihr GitLab .com-Repository hinzufügen, indem Sie entweder eine Pipeline erstellen oder eine bestehende bearbeiten. Verwenden Sie die GitLabProvider-Option in der Konsole oder die `CodestarSourceConnection` Aktion mit dem `GitLab` Anbieter in der CLI. Siehe GitLab.com-Verbindungen. Weitere Informationen: • Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen
GitLab selbst verwaltet	Sie können so konfigurieren CodePipeline , dass eine GitLab selbstverwaltete Installation als Quelle für Ihren Code verwendet wird. Sie müssen zuvor ein GitLab Konto erstellt haben und über ein Abonnement für Self-Managed GitLab (Enterprise Edition oder Community Edition) verfügen. Sie können eine Quellaktion für Ihr GitLab selbstverwaltetes Repository hinzufügen, indem Sie entweder eine Pipeline erstellen oder eine bestehende bearbeiten. Sie können Ressourcen, die als Verbindungen bezeichnet werden, einrichte n, damit Ihre Pipelines auf Code-Repositorys von Drittanbietern zugreifen können. Wenn Sie eine Verbindung herstellen, installieren Sie die AWS CodeStar App mit Ihrem Code-Repository eines Drittanbieters und verknüpfen sie dann mit Ihrer Verbindung. Verwenden Sie die Option für GitLab selbstverwalteten Anbieter in der Konsole oder die `CodestarSourceConnection` Aktion in der CLI. Siehe Verbindungen für GitLab selbstverwaltete. Sie können die Option Vollständiges Klonen für diese Aktion verwenden, um auf die Git-Metadaten des Repositorys zu verweisen, sodass nachgelagerte

Aktionen Git-Befehle direkt ausführen können. Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

Weitere Informationen:

- Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen
- Die Schritte zum Herstellen einer Verbindung mit diesem Anbietertyp finden Sie unter. Verbindungen für GitLab selbstverwaltete

CodeCommit Quellaktionen

CodeCommitist ein Dienst zur Versionskontrolle, mit dem Sie Assets (wie Dokumente, Quellcode und Binärdateien) privat in der Cloud speichern und verwalten können. Sie können so konfigurieren CodePipeline , dass ein Zweig in einem CodeCommit Repository als Quelle für Ihren Code verwendet wird. Erstellen Sie das Repository und ordnen Sie es einem Arbeitsverzeichnis auf Ihrem lokalen Computer zu. Anschließend können Sie eine Pipeline erstellen, die den Branch in einer Phase als Teil einer Quellaktion verwendet. Sie können eine Verbindung zum CodeCommit Repository herstellen, indem Sie entweder eine Pipeline erstellen oder eine bestehende bearbeiten.

Sie können die Option Vollständiges Klonen für diese Aktion verwenden, um auf die Git-Metadaten des Repositorys zu verweisen, sodass nachgelagerte Aktionen Git-Befehle direkt ausführen können. Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

Weitere Informationen:

- Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter. CodeCommit
- Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository)
- CodePipeline verwendet Amazon CloudWatch Events, um Änderungen in CodeCommit Repositorys zu erkennen, die als Quelle für eine Pipeline verwendet werden. Für jede Quellaktion gibt es eine entsprechende Regel. Diese Ereignisregel startet Ihre Pipeline, wenn eine Änderung im Repository auftritt. Siehe Allgemeine Integrationen mit CodePipeline.

GitHub (Version 1) Quellaktionen

Die Aktion der GitHub Version 1 wird mit OAuth Apps verwaltet. In verfügbaren Regionen können Sie auch eine CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen Quellaktion in Ihrer Pipeline verwenden, die Ihre GitHub Quellaktion mit GitHub Apps verwaltet. Wenn Sie eine Pipeline haben, die die Aktion GitHub Version 1 verwendet, lesen Sie die Schritte zur Aktualisierung auf die Verwendung einer Aktion von GitHub Version 2 inAktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2.

> **Note**
>
> Wir empfehlen zwar nicht, die Aktion GitHub Version 1 zu verwenden, bestehende Pipelines mit der Aktion GitHub Version 1 funktionieren jedoch weiterhin ohne Auswirkungen.

Weitere Informationen:

- Weitere Informationen zum OAuth-basierten GitHub Zugriff im Gegensatz zum App-basierten GitHub Zugriff finden Sie unter. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps
- Einen Anhang mit den Einzelheiten der GitHub Aktion für Version 1 finden Sie unter. Anhang A: Quellaktionen für GitHub Version 1

Erstellen von Aktionsintegrationen

Die folgenden Informationen sind nach CodePipeline Aktionstypen geordnet und können Ihnen bei der Konfiguration CodePipeline der Integration mit den folgenden Build-Aktionsanbietern helfen.

Themen

- CodeBuild Aktionen erstellen
- CloudBees Aktionen erstellen
- Jenkins-Build-Aktionen
- TeamCity Aktionen erstellen

CodeBuild Aktionen erstellen

CodeBuildist ein vollständig verwalteter Build-Service, der Ihren Quellcode kompiliert, Komponententests durchführt und Artefakte erzeugt, die sofort einsatzbereit sind.

Sie können es CodeBuild als Build-Aktion zur Build-Phase einer Pipeline hinzufügen. Weitere Informationen finden Sie in der Referenz zur CodePipeline Aktionskonfiguration fürAWS CodeBuild.

> **Note**
>
> CodeBuild kann auch als Testaktion mit oder ohne Build-Ausgabe in eine Pipeline aufgenommen werden.

Weitere Informationen:

- Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter. AWS CodeBuild
- Was ist? CodeBuild
- CodeBuild— Vollständig verwalteter Build-Service

CloudBees Aktionen erstellen

Sie können ihn so konfigurieren CodePipeline , dass Sie ihn CloudBeeszum Erstellen oder Testen Ihres Codes in einer oder mehreren Aktionen in einer Pipeline verwenden.

Weitere Informationen:

- re:Invent 2017: Cloud First mit AWS

Jenkins-Build-Aktionen

Sie können so konfigurieren CodePipeline , dass Sie Jenkins CI verwenden, um Ihren Code in einer oder mehreren Aktionen in einer Pipeline zu erstellen oder zu testen. Sie müssen zuvor ein Jenkins-Projekt erstellt und das CodePipeline Plugin für Jenkins für dieses Projekt installiert und konfiguriert haben. Sie können eine Verbindung zum Jenkins-Projekt herstellen, indem Sie entweder eine neue Pipeline erstellen oder eine vorhandene Pipeline bearbeiten.

Der Zugriff für Jenkins wird pro Projekt konfiguriert. Sie müssen das CodePipeline Plugin für Jenkins auf jeder Jenkins-Instanz installieren, mit der Sie es verwenden möchten. CodePipeline Sie müssen auch den CodePipeline Zugriff auf das Jenkins-Projekt konfigurieren. Schützen Sie Ihr Jenkins-Projekt, indem Sie es so konfigurieren, dass es ausschließlich HTTPS/SSL-Verbindungen akzeptiert. Wenn Ihr Jenkins-Projekt auf einer Amazon EC2 EC2-Instance installiert ist, sollten Sie erwägen, Ihre AWS Anmeldeinformationen anzugeben, indem Sie das AWS CLI auf jeder Instance installieren. Konfigurieren Sie dann auf diesen Instances ein AWS Profil mit den Anmeldeinformationen, die Sie für Verbindungen verwenden möchten. Dies ist eine Alternative zum Hinzufügen und Speichern über die Jenkins-Weboberfläche.

Weitere Informationen:

- Zugriff auf Jenkins
- Tutorial: Erstellen einer vierstufigen Pipeline

TeamCity Aktionen erstellen

Sie können CodePipeline es so konfigurieren, dass Sie TeamCityes verwenden, um Ihren Code in einer oder mehreren Aktionen in einer Pipeline zu erstellen und zu testen.

Weitere Informationen:

- TeamCity Plugin für CodePipeline

Testen von Aktionsintegrationen

Die folgenden Informationen sind nach CodePipeline Aktionstyp geordnet und können Ihnen bei der Konfiguration für CodePipeline die Integration mit den folgenden Testaktionsanbietern helfen.

Themen

- CodeBuild Testaktionen
- AWS Device Farm Aktionen testen
- Ghost Inspector-Testaktionen
- OpenText LoadRunner Cloud-Testaktionen

CodeBuild Testaktionen

CodeBuildist ein vollständig verwalteter Build-Service in der Cloud. CodeBuild kompiliert Ihren Quellcode, führt Komponententests durch und erzeugt Artefakte, die sofort einsatzbereit sind.

Sie können einer Pipeline als Testaktion etwas hinzufügen CodeBuild . Weitere Informationen finden Sie in der Referenz zur CodePipeline Aktionskonfiguration fürAWS CodeBuild.

> **Note**
>
> CodeBuild kann auch als Build-Aktion mit einem obligatorischen Build-Ausgabeartefakt in eine Pipeline aufgenommen werden.

Weitere Informationen:

- Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter. AWS CodeBuild
- Was ist? CodeBuild

AWS Device Farm Aktionen testen

AWS Device Farm ist ein Service für das Testen von Apps, den Sie verwenden können, um Android-, iOS- und Webanwendungen auf echten physischen Mobiltelefonen und Tablets zu testen und mit ihnen zu interagieren. Sie können so konfigurieren CodePipeline , AWS Device Farm dass Sie Ihren Code in einer oder mehreren Aktionen in einer Pipeline testen. AWS Device Farm ermöglicht es Ihnen, Ihre eigenen Tests hochzuladen oder integrierte, skriptfreie Kompatibilitätstests zu verwenden. Da Tests parallel durchgeführt werden, starten Tests auf mehreren Geräten innerhalb von wenigen Minuten. Ein Testbericht, der allgemeine Ergebnisse, Low-Level-Logs, pixel-to-pixel Screenshots und Leistungsdaten enthält, wird aktualisiert, sobald die Tests abgeschlossen sind. AWS Device Farm unterstützt das Testen von nativen und hybriden Android-, iOS- und Fire OS-Apps, einschließlich solcher, die mit Titanium PhoneGap, Xamarin, Unity und anderen Frameworks erstellt wurden. Es unterstützt den Remote-Zugriff auf Android-Apps, damit Sie direkt mit Testgeräten interagieren können.

Weitere Informationen:

- Konfigurationsparameter und ein Beispiel für ein JSON/YAML-Snippet finden Sie unter. AWS Device Farm

- Was ist? AWS Device Farm
- Verwendung AWS Device Farm in einer CodePipeline Testphase

Ghost Inspector-Testaktionen

Sie können so konfigurieren CodePipeline , dass Ghost Inspector verwendet wird, um Ihren Code in einer oder mehreren Aktionen in einer Pipeline zu testen.

Weitere Informationen:

- Ghost Inspector-Dokumentation für die Serviceintegration mit CodePipeline

OpenText LoadRunner Cloud-Testaktionen

Sie können so konfigurieren CodePipeline , dass OpenText LoadRunner Cloud in einer oder mehreren Aktionen in einer Pipeline verwendet wird.

Weitere Informationen:

- LoadRunner Cloud-Dokumentation für die Integration mit CodePipeline

Bereitstellungsaktions-Integrationen

Die folgenden Informationen sind nach CodePipeline Aktionstyp geordnet und können Ihnen bei der Konfiguration CodePipeline der Integration mit den folgenden Bereitstellungsaktionsanbietern helfen.

Themen

- Amazon S3 S3-Bereitstellungsaktionen
- AWS AppConfig Aktionen bereitstellen
- AWS CloudFormation Aktionen bereitstellen
- AWS CloudFormation StackSets Aktionen bereitstellen
- Amazon ECS-Bereitstellungsaktionen
- Elastic Beanstalk Beanstalk-Bereitstellungsaktionen
- AWS OpsWorks Aktionen bereitstellen
- Bereitstellungsaktionen für den Service Catalog

- Amazon Alexa Bereitstellungsaktionen
- CodeDeploy Aktionen bereitstellen
- XebiaLabs Aktionen bereitstellen

Amazon S3 S3-Bereitstellungsaktionen

Amazon S3 ist Speicher für das Internet. Mit Amazon S3 können Sie jederzeit beliebige Mengen von Daten von überall aus im Internet speichern und aufrufen. Sie können einer Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet, eine Aktion hinzufügen.

> **Note**
>
> Amazon S3 kann auch als Quellaktion in eine Pipeline aufgenommen werden.

Weitere Informationen:

- Erstellen Sie eine Pipeline in CodePipeline
- Tutorial: Erstellen Sie eine Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet

AWS AppConfig Aktionen bereitstellen

AWS AppConfig ist eine Fähigkeit, Anwendungskonfigurationen AWS Systems Manager zu erstellen, zu verwalten und schnell bereitzustellen. Sie können es AppConfig mit Anwendungen verwenden, die auf EC2-Instances, Containern AWS Lambda, mobilen Anwendungen oder IoT-Geräten gehostet werden.

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für AWS AppConfig
- Tutorial: Eine Pipeline erstellen, die AWS AppConfig als Bereitstellungsanbieter verwendet wird

AWS CloudFormation Aktionen bereitstellen

AWS CloudFormationbietet Entwicklern und Systemadministratoren eine einfache Möglichkeit, eine Sammlung verwandter Ressourcen zu erstellen und zu verwalten und diese AWS Ressourcen mithilfe von Vorlagen bereitzustellen und zu aktualisieren. Sie können die Beispielvorlagen des

Service verwenden oder eigene Beispielvorlagen erstellen. Vorlagen beschreiben die AWS Ressourcen und alle Abhängigkeiten oder Laufzeitparameter, die für die Ausführung Ihrer Anwendung erforderlich sind.

Das AWS Serverless Application Model (AWS SAM) wurde erweitert und bietet AWS CloudFormation nun eine vereinfachte Methode zur Definition und Bereitstellung serverloser Anwendungen. AWS SAM unterstützt Amazon API Gateway Gateway-APIs, AWS Lambda-Funktionen und Amazon DynamoDB-Tabellen. Sie können AWS SAM CodePipeline zusammen mit AWS CloudFormation und zur kontinuierlichen Bereitstellung Ihrer serverlosen Anwendungen verwenden.

Sie können einer Pipeline, die AWS CloudFormation als Bereitstellungsanbieter verwendet wird, eine Aktion hinzufügen. Wenn Sie sie AWS CloudFormation als Bereitstellungsanbieter verwenden, können Sie im Rahmen einer Pipeline-Ausführung Aktionen für AWS CloudFormation Stacks und Änderungssätze ausführen. AWS CloudFormation kann Stacks erstellen, aktualisieren, ersetzen und löschen und Sets ändern, wenn eine Pipeline ausgeführt wird. Daher können benutzerdefinierte Ressourcen während einer Pipeline-Ausführung gemäß den Spezifikationen, die Sie in AWS CloudFormation Vorlagen AWS und Parameterdefinitionen angeben, erstellt, bereitgestellt, aktualisiert oder beendet werden.

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für AWS CloudFormation
- Continuous Delivery mit CodePipeline — Erfahren Sie, wie Sie CodePipeline damit einen Continuous Delivery-Workflow für erstellen können AWS CloudFormation.
- Automatisieren der Bereitstellung von Lambda-basierten Anwendungen — Erfahren Sie, wie Sie das AWS serverlose Anwendungsmodell verwenden und einen Continuous-Delivery-Workflow AWS CloudFormation für Ihre Lambda-basierte Anwendung erstellen können.

AWS CloudFormation StackSets Aktionen bereitstellen

AWS CloudFormationbietet Ihnen die Möglichkeit, Ressourcen für mehrere Konten und AWS Regionen bereitzustellen.

Sie können CodePipeline with verwenden AWS CloudFormation , um Ihre Stack-Set-Definition zu aktualisieren und Updates für Ihre Instances bereitzustellen.

Sie können einer Pipeline die folgenden Aktionen hinzufügen, um sie AWS CloudFormation StackSets als Bereitstellungsanbieter zu verwenden.

- CloudFormationStackSet
- CloudFormationStackInstances

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für AWS CloudFormation StackSets
- Tutorial: Eine Pipeline mit AWS CloudFormation StackSets Bereitstellungsaktionen erstellen

Amazon ECS-Bereitstellungsaktionen

Amazon ECS ist ein hoch skalierbarer, leistungsstarker Container-Management-Service, mit dem Sie containerbasierte Anwendungen in der ausführen können. AWS Cloud Wenn Sie eine Pipeline erstellen, können Sie Amazon ECS als Bereitstellungsanbieter auswählen. Eine Änderung am Code in Ihrem Quellcodeverwaltungs-Repository veranlasst Ihre Pipeline, ein neues Docker-Image zu erstellen, es in Ihre Container-Registry zu übertragen und dann das aktualisierte Image in Amazon ECS bereitzustellen. Sie können auch die ECS-Anbieteraktion (Blau/Grün) verwenden CodePipeline , um Datenverkehr an Amazon ECS weiterzuleiten und bereitzustellen. CodeDeploy

Weitere Informationen:

- Was ist Amazon ECS?
- Tutorial: Kontinuierliche Bereitstellung mit CodePipeline
- Erstellen Sie eine Pipeline in CodePipeline
- Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy

Elastic Beanstalk Beanstalk-Bereitstellungsaktionen

Elastic Beanstalk ist ein Service für die Bereitstellung und Skalierung von Webanwendungen und Services, die mit Java, .NET, PHP, Node.js, Python, Ruby, Go und Docker auf vertrauten Servern wie Apache, Nginx, Passenger und IIS entwickelt wurden. Sie können so konfigurieren CodePipeline , dass Elastic Beanstalk für die Bereitstellung Ihres Codes verwendet wird. Sie können die Elastic Beanstalk Beanstalk-Anwendung und -Umgebung für eine Bereitstellungsaktion in einer Phase erstellen, entweder bevor Sie die Pipeline erstellen oder wenn Sie den Assistenten zum Erstellen einer Pipeline verwenden.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hyderabad), Asien-Pazifik (Melbourne), Naher Osten (VAE), Europa (Spanien) und Europa (Zürich) nicht verfügbar. Weitere verfügbare Aktionen finden Sie unterProdukt- und Serviceintegrationen mit CodePipeline.

Weitere Informationen:

- Erste Schritte mit Elastic Beanstalk
- Erstellen Sie eine Pipeline in CodePipeline

AWS OpsWorks Aktionen bereitstellen

AWS OpsWorks ist ein Konfigurationsverwaltungsdienst, der Sie bei der Konfiguration und dem Betrieb von Anwendungen aller Art und Größe mit Chef unterstützt. Mit AWS OpsWorks Stacks ihm können Sie die Architektur der Anwendung und die Spezifikation der einzelnen Komponenten definieren, einschließlich Paketinstallation, Softwarekonfiguration und Ressourcen wie Speicher. Sie können CodePipeline es so konfigurieren, AWS OpsWorks Stacks dass Sie Ihren Code in Verbindung mit benutzerdefinierten Chef-Kochbüchern und -Anwendungen in AWS OpsWorks bereitstellen.

- Benutzerdefinierte Chef-Kochbücher — AWS OpsWorks verwendet Chef Cookbooks, um Aufgaben wie die Installation und Konfiguration von Paketen und die Bereitstellung von Anwendungen zu erledigen.
- Anwendungen — Eine AWS OpsWorks Anwendung besteht aus Code, den Sie auf einem Anwendungsserver ausführen möchten. Der Anwendungscode wird in einem Repository gespeichert, z. B. in einem Amazon S3 S3-Bucket.

Bevor Sie die Pipeline erstellen, erstellen Sie den AWS OpsWorks Stack und die Ebene. Sie können die AWS OpsWorks Anwendung, die in einer Bereitstellungsaktion verwendet werden soll, in einer Phase erstellen, entweder bevor Sie die Pipeline erstellen oder wenn Sie den Assistenten zum Erstellen einer Pipeline verwenden.

CodePipeline Support für AWS OpsWorks ist derzeit nur in der Region USA Ost (Nord-Virginia) (us-east-1) verfügbar.

Weitere Informationen:

- Verwenden mit CodePipeline AWS OpsWorks Stacks
- Cookbooks und Rezepte
- AWS OpsWorks Apps

Bereitstellungsaktionen für den Service Catalog

Service Catalog ermöglicht es Unternehmen, Kataloge mit Produkten zu erstellen und zu verwalten, die für die Verwendung auf AWS zugelassen sind.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Naher Osten (VAE), Europa (Spanien), Europa (Zürich) oder Israel (Tel Aviv) nicht verfügbar. Hinweise zu anderen verfügbaren Aktionen finden Sie unter. Produkt- und Serviceintegrationen mit CodePipeline

Sie können so konfigurieren CodePipeline , dass Updates und Versionen Ihrer Produktvorlagen im Service Catalog bereitgestellt werden. Sie können das Service Catalog-Produkt zur Verwendung in einer Bereitstellungsaktion erstellen und dann den Assistenten „Pipeline erstellen“ verwenden, um die Pipeline zu erstellen.

Weitere Informationen:

- Tutorial: Erstellen Sie eine Pipeline, die in Service Catalog bereitgestellt wird
- Erstellen Sie eine Pipeline in CodePipeline

Amazon Alexa Bereitstellungsaktionen

Mit dem Amazon Alexa Skills Kit können Sie für Benutzer von Alexa-fähigen Geräten Cloud-basierte Qualifikationen entwickeln und verteilen.

> **Note**
>
> Diese Funktion ist in der Region Asien-Pazifik (Hongkong) oder Europa (Mailand) nicht verfügbar. Informationen zur Verwendung anderer Bereitstellungsaktionen, die in dieser Region verfügbar sind, finden Sie unterBereitstellungsaktions-Integrationen.

Sie können einer Pipeline eine Aktion hinzufügen, für die das Alexa Skills Kit als Bereitstellungsanbieter verwendet wird. Die Quelländerungen werden von Ihrer Pipeline erkannt und dann stellt die Pipeline Updates für Ihre Alexa-Qualifikation im Alexa-Service bereit.

Weitere Informationen:

- Tutorial: Erstellen einer Pipeline zum Bereitstellen eines Amazon Alexa-Skills

CodeDeploy Aktionen bereitstellen

CodeDeploykoordiniert Anwendungsbereitstellungen für Amazon EC2 EC2-Instances, lokale Instances oder beides. Sie können ihn so konfigurieren CodePipeline , dass er CodeDeploy zur Bereitstellung Ihres Codes verwendet wird. Sie können die CodeDeploy Anwendung, die Bereitstellung und die Bereitstellungsgruppe, die in einer Bereitstellungsaktion verwendet werden sollen, in einer Phase erstellen, entweder bevor Sie die Pipeline erstellen oder wenn Sie den Assistenten zum Erstellen einer Pipeline verwenden.

Weitere Informationen:

- Schritt 3: Erstellen Sie eine Anwendung in CodeDeploy
- Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository)

XebiaLabs Aktionen bereitstellen

Sie können so konfigurieren CodePipeline , XebiaLabsdass Ihr Code in einer oder mehreren Aktionen in einer Pipeline bereitgestellt wird.

Weitere Informationen:

- Verwenden Sie XL Deploy mit CodePipeline

Integration von Genehmigungsaktionen mit Amazon Simple Notification Service

Amazon SNS ist ein schneller, flexibler und vollständig verwalteter Push-Benachrichtigungsservice, mit dem Sie einzelne Nachrichten senden oder Nachrichten an eine große Anzahl von Empfängern auffächern können. Amazon SNS macht es einfach und kostengünstig, Push-Benachrichtigungen an Benutzer von Mobilgeräten, E-Mail-Empfänger oder sogar Nachrichten an andere verteilte Dienste zu senden.

Wenn Sie eine manuelle Genehmigungsanfrage in erstellen CodePipeline, können Sie optional zu einem Thema in Amazon SNS veröffentlichen, sodass alle IAM-Benutzer, die sie abonniert haben, darüber informiert werden, dass die Genehmigungsaktion zur Überprüfung bereit ist.

Weitere Informationen:

- Was ist Amazon SNS?
- Gewähren Sie Amazon SNS SNS-Berechtigungen für eine CodePipeline Servicerolle

Aufrufen von Aktionsintegrationen

Die folgenden Informationen sind nach CodePipeline Aktionstyp geordnet und können Ihnen bei der Konfiguration CodePipeline der Integration mit den folgenden Anbietern von Aufrufaktionen helfen.

Themen

- Lambda-Aktionen aufrufen
- Aktionen mit Snyk aufrufen
- Step Functions rufen Aktionen auf

Lambda-Aktionen aufrufen

Mit Lambda können Sie Code ausführen, ohne Server bereitzustellen oder zu verwalten. Sie können CodePipeline die Verwendung von Lambda-Funktionen konfigurieren, um Ihren Pipelines Flexibilität und Funktionalität zu verleihen. Sie können die Lambda-Funktion erstellen, um sie als Aktion in einer Phase hinzuzufügen, entweder bevor Sie die Pipeline erstellen oder wenn Sie den Assistenten Pipeline erstellen verwenden.

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für AWS Lambda
- Rufen Sie eine AWS Lambda Funktion in einer Pipeline auf in CodePipeline

Aktionen mit Snyk aufrufen

Sie können Snyk so konfigurieren CodePipeline , dass Sie Ihre Open-Source-Umgebungen schützen, indem Sie Sicherheitslücken erkennen und beheben und Abhängigkeiten in Ihrem Anwendungscode und Ihren Container-Images aktualisieren. Sie können die Snyk-Aktion auch verwenden, um die Kontrollen für Sicherheitstests in Ihrer Pipeline CodePipeline zu automatisieren.

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für Referenz zur Snyk-Aktionsstruktur
- Automatisieren Sie das Scannen von Sicherheitslücken AWS CodePipeline mit Snyk

Step Functions rufen Aktionen auf

Mit Step Functions können Sie Zustandsmaschinen erstellen und konfigurieren. Sie können so konfigurieren CodePipeline , dass Step Functions Aktionen aufrufen, um State-Machine-Ausführungen auszulösen.

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für AWS Step Functions
- Tutorial: Verwenden Sie eine AWS Step Functions Aufrufaktion in einer Pipeline

Allgemeine Integrationen mit CodePipeline

Die folgenden AWS-Service Integrationen basieren nicht auf CodePipeline Aktionstypen.

Amazon CloudWatch	Amazon CloudWatch überwacht Ihre AWS Ressourcen. Weitere Informationen: • Was ist Amazon CloudWatch?
Amazon EventBridge	Amazon EventBridge ist ein Webservice, der AWS-Services anhand von von Ihnen definierter Regeln Änderungen an Ihnen erkennt und

	eine Aktion in einer oder mehreren angegebenen Fällen auslöst, AWS-Services wenn eine Änderung eintritt. • Automatisches Starten einer Pipeline-Ausführung, wenn sich etwas ändert — Sie können CodePipeline sie als Ziel in in Amazon eingerichteten Regeln konfigurieren EventBridge. Dadurch werden Pipelines automatisch gestartet, wenn es Änderungen in einem anderen Service gibt. Weitere Informationen: • Was ist Amazon EventBridge? • Starten Sie eine Pipeline in CodePipeline. • CodeCommit Quellaktionen und EventBridge • Erhalten Sie Benachrichtigungen, wenn sich der Status einer Pipeline ändert — Sie können EventBridge Regeln einrichten, um Änderungen im Ausführungsstatus einer Pipeline, Phase oder Aktion zu erkennen und darauf zu reagieren. Weitere Informationen: • CodePipeline Ereignisse überwachen • Tutorial: Richten Sie eine CloudWatch Ereignisregel ein, um E-Mail-Benachrichtigungen für Änderungen des Pipeline-Status zu erhalten
AWS Cloud9	AWS Cloud9 ist eine Online-IDE, auf die Sie über Ihren Webbrowse r zugreifen. Die IDE bietet eine umfassende Codebearbeitung mit Unterstützung mehrerer Programmiersprachen und Runtime-Debugger sowie ein integriertes Terminal. Im Hintergrund hostet eine Amazon EC2 EC2-Instance eine AWS Cloud9 Entwicklungsumgebung. Weitere Informationen finden Sie im AWS Cloud9 -Benutzerhandbuch. Weitere Informationen: • Einrichten von AWS Cloud9

AWS CloudTrail	CloudTrailerfasst AWS API-Aufrufe und zugehörige Ereignisse, die von oder im Namen eines AWS Kontos getätigt wurden, und übermitte lt Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können so konfigurieren CloudTrail , dass API-Aufrufe von der CodePipeline Konsole, CodePipeline Befehle von der AWS CLI und von der CodePipeline API erfasst werden. Weitere Informationen: • CodePipeline API-Aufrufe protokollieren mit AWS CloudTrail
AWS CodeStar Benachrichtigungen	Sie können Benachrichtigungen einrichten, um Benutzer auf wichtige Änderungen aufmerksam zu machen, z. B. wenn eine Pipeline mit der Ausführung beginnt. Weitere Informationen finden Sie unter Erstellen einer Benachrichtigungsregel.

AWS Key Management Service	AWS KMS ist ein verwalteter Service, der das Erstellen und Kontrollieren der Schlüssel zum Verschlüsseln Ihrer Daten vereinfacht. Wird standardmäßig AWS KMS zum Verschlüsseln von Artefakten für Pipelines CodePipeline verwendet, die in Amazon S3 S3-Buckets gespeichert sind. Weitere Informationen: • Um eine Pipeline zu erstellen, die einen Quell-Bucket, einen Artefakt-Bucket und eine Servicerolle aus einem AWS Konto und CodeDeploy Ressourcen aus einem anderen AWS Konto verwendet, müssen Sie einen vom Kunden verwalteten KMS-Schlüssel erstellen, den Schlüssel zur Pipeline hinzufügen und Kontorichtlinien und Rollen einrichten, um den kontoübergreifenden Zugriff zu ermöglichen. Weitere Informationen finden Sie unter Erstellen Sie eine Pipeline CodePipeline , in der Ressourcen von einem anderen AWS Konto verwendet werden. • Um eine Pipeline von einem AWS Konto zu erstellen, das einen AWS CloudFormation Stack für ein anderes AWS Konto bereitstellt, müssen Sie einen vom Kunden verwalteten KMS-Schlüssel erstellen, den Schlüssel zur Pipeline hinzufügen und Kontorichtlinien und Rollen einrichten, um den Stack für ein anderes AWS Konto bereitzustellen. Weitere Informationen finden Sie unter Wie verwende ich, CodePipeline um einen AWS CloudFormation Stack in einem anderen Konto bereitzustellen? • Um die serverseitige Verschlüsselung für den S3-Artefakt-Bucket Ihrer Pipeline zu konfigurieren, können Sie den standardmäßigen AWS verwalteten KMS-Schlüssel verwenden oder einen vom Kunden verwalteten KMS-Schlüssel erstellen und die Bucket-Richtlinie für die Verwendung des Verschlüsselungsschlüssels einrichten. Weitere Informationen finden Sie unter Konfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte für CodePipeline. Für eine AWS KMS key können Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN verwenden.

> **Note**
>
> Aliase werden nur in dem Konto erkannt, das den KMS-Schlüssel erstellt hat. Für kontoübergreifende Aktionen können Sie zum Identifizieren des Schlüssels nur die Schlüssel-ID oder den Schlüssel-ARN vewenden. Bei kontoübergreifenden Aktionen wird die Rolle des anderen Kontos (AccountB) verwendet, sodass bei Angabe der Schlüssel-ID der Schlüssel des anderen Kontos (AccountB) verwendet wird.

Beispiele aus der Community

In den folgenden Abschnitten werden Links zu Blog-Posts, Artikel und von der Community bereitgestellte Beispiele vorgestellt.

> **Note**
>
> Diese Links werden nur zu Informationszwecken bereitgestellt und sollten weder als umfassende Liste noch als Bestätigung des Inhalts der Beispiele betrachtet werden. AWS ist nicht verantwortlich für den Inhalt oder die Richtigkeit externer Inhalte.

Themen

Integrationsbeispiele: Blog-Beiträge

- Den AWS CodePipeline Build-Status aus dem Git-Repository eines Drittanbieters verfolgen

  Erfahre, wie du Ressourcen einrichtest, die deinen Pipeline- und Build-Aktionsstatus in deinem Drittanbieter-Repository anzeigen, sodass der Entwickler den Status leicht verfolgen kann, ohne den Kontext wechseln zu müssen.

  Veröffentlicht im März 2021
- Komplettes CI/CD mit AWS CodeCommit, AWS CodeBuild, und AWS CodeDeployAWS CodePipeline

Erfahren Sie, wie Sie eine Pipeline einrichten CodeCommit, die die CodeDeploy Dienste,, und verwendet CodePipeline CodeBuild, um eine versionskontrollierte Java-Anwendung auf einer Reihe von Amazon EC2 EC2-Linux-Instances zu kompilieren, zu erstellen und zu installieren.

Veröffentlicht im September 2020

- Wie erfolgt die Bereitstellung von GitHub zu Amazon EC2 mit CodePipeline

  Erfahren Sie, wie Sie CodePipeline von Grund auf die Bereitstellung von Entwicklungs-, Test- und Produktionszweigen für separate Bereitstellungsgruppen einrichten. Erfahren Sie, wie Sie IAM-Rollen, den CodeDeploy Agenten und zusammen mit verwenden und CodeDeploy konfigurieren. CodePipeline

  Veröffentlicht im April 2020

- Testen und Erstellen von CI/CD-Pipelines für Step Functions AWS

  Erfahren Sie, wie Sie Ressourcen einrichten, die Ihre Step Functions Functions-Zustandsmaschine und Ihre Pipeline koordinieren.

  Veröffentlicht im März 2020

- Implementieren DevSecOps mit CodePipeline

  Erfahren Sie, wie Sie mithilfe einer CI/CD-Pipeline präventive und detektive Sicherheitskontrollen automatisieren können. CodePipeline In diesem Beitrag erfahren Sie, wie Sie mithilfe einer Pipeline eine einfache Sicherheitsgruppe erstellen und während der Quell-, Test- und Produktionsphase Sicherheitsprüfungen durchführen, um die Sicherheitslage Ihrer AWS Konten zu verbessern.

  Veröffentlichung März 2017

- Kontinuierliche Bereitstellung in Amazon ECS mithilfe von CodePipeline CodeBuild, Amazon ECR und AWS CloudFormation

  Erfahren Sie, wie Sie eine kontinuierliche Bereitstellungspipeline für Amazon Elastic Container Service (Amazon ECS) einrichten. Anwendungen werden als Docker-Container mit CodePipeline, CodeBuild, Amazon ECR und bereitgestellt. AWS CloudFormation

  - Laden Sie eine AWS CloudFormation Beispielvorlage und Anweisungen zur Erstellung Ihrer eigenen Continuous Deployment-Pipeline aus dem ECS-Reference Architecture: Continuous Deployment Repo unter herunter. GitHub

  Veröffentlicht im Januar 2017

- Kontinuierliche Bereitstellung für serverlose Anwendungen

  Erfahren Sie, wie Sie anhand einer Sammlung von AWS-Services eine kontinuierliche Bereitstellungspipeline für Ihre serverlosen Anwendungen erstellen können. Sie verwenden das Serverless Application Model (SAM), um die Anwendung und ihre Ressourcen zu definieren und die Anwendungsbereitstellung CodePipeline zu orchestrieren.

  - Zeigen Sie eine Beispielanwendung an, die in Go mit dem Gin-Framework und einem API Gateway-Proxy-Shin geschrieben wurde.

  Erschienen: Dezember 2016

- Skalierung von DevOps Bereitstellungen mit und Dynatrace CodePipeline

  Erfahren Sie, wie Sie mit den Überwachungslösungen von Dynatrace Pipelines skalieren, Testausführungen automatisch analysieren CodePipeline, bevor der Code festgeschrieben wird, und optimale Durchlaufzeiten einhalten können.

  Veröffentlichung November 2016

- Erstellen Sie eine Pipeline für in Using und AWS Elastic Beanstalk CodePipeline AWS CloudFormation CodeCommit

  Erfahren Sie in, wie Sie Continuous Delivery in einer CodePipeline Pipeline für eine Anwendung implementieren AWS Elastic Beanstalk. Alle AWS Ressourcen werden mithilfe einer AWS CloudFormation Vorlage automatisch bereitgestellt. Diese exemplarische Vorgehensweise beinhaltet auch CodeCommit und AWS Identity and Access Management (IAM).

  Veröffentlicht im Mai 2016

- Automatisieren und CodeCommit in CodePipeline AWS CloudFormation

  Wird verwendet AWS CloudFormation , um die Bereitstellung von AWS Ressourcen für eine Continuous-Delivery-Pipeline zu automatisieren, die CodeCommit, CodePipeline CodeDeploy, und AWS Identity and Access Management verwendet.

  Veröffentlicht im April 2016

- Erstellen Sie eine kontenübergreifende Pipeline in AWS CodePipeline

  Erfahren Sie, wie Sie mittels AWS CodePipeline die Bereitstellung eines kontoübergreifenden Zugriffs für Pipelines in AWS Identity and Access Management automatisieren. Enthält Beispiele in einer AWS CloudFormation Vorlage.

Veröffentlichung März 2016

- Informationen zu ASP.NET Core Teil 2: kontinuierliche Bereitstellung

  Erfahren Sie, wie Sie mit und ein vollständiges Continuous-Delivery-System für eine ASP.NET Core-Anwendung CodeDeploy erstellen. AWS CodePipeline

  Veröffentlichung März 2016

- Erstellen Sie mithilfe der Konsole eine Pipeline AWS CodePipeline

  Erfahren Sie in einer exemplarischen Vorgehensweise, wie Sie mit der AWS CodePipeline Konsole eine zweistufige Pipeline erstellen. AWS CodePipeline Tutorial: Erstellen einer vierstufigen Pipeline

  Veröffentlichung März 2016

- Pipelines AWS CodePipeline verspotten mit AWS Lambda

  Erfahren Sie, wie Sie eine Lambda-Funktion aufrufen, mit der Sie die Aktionen und Phasen eines CodePipeline Softwarebereitstellungsprozesses während des Entwurfs visualisieren können, bevor die Pipeline betriebsbereit ist. Beim Entwerfen Ihrer Pipeline-Struktur können Sie die Lambda-Funktion verwenden, um zu testen, ob Ihre Pipeline erfolgreich abgeschlossen wird.

  Veröffentlichung Februar 2016

- Ausführen von AWS Lambda Funktionen in Using CodePipeline AWS CloudFormation

  Erfahren Sie in der Benutzeranleitung, wie Sie einen AWS CloudFormation Stack erstellen, der alle in der Benutzeranleitung verwendeten AWS Ressourcen bereitstelltRufen Sie eine AWS Lambda Funktion in einer Pipeline auf in CodePipeline.

  Veröffentlichung Februar 2016

- Bereitstellung benutzerdefinierter CodePipeline Aktionen in AWS CloudFormation

  Informationen zur Bereitstellung benutzerdefinierter Aktionen finden Sie AWS CloudFormation unter CodePipeline.

  Veröffentlicht im Januar 2016

- Bereitstellung mit CodePipeline AWS CloudFormation

  Erfahren Sie, wie Sie im CodePipeline Einsatz AWS CloudFormation eine grundlegende Continuous-Delivery-Pipeline bereitstellen.

Erschienen: Dezember 2015

- Bereitstellung von CodePipeline bis AWS OpsWorks Mithilfe einer benutzerdefinierten Aktion und AWS Lambda

  Erfahren Sie, wie Sie Ihre Pipeline konfigurieren und welche AWS Lambda Funktion für die Bereitstellung AWS OpsWorks verwendet CodePipeline werden soll.

  Veröffentlichung Juli 2015

CodePipeline Anleitungen

Nachdem Sie die Schritte unter abgeschlossen habenErste Schritte mit CodePipeline, können Sie eines der AWS CodePipeline Tutorials in diesem Benutzerhandbuch ausprobieren:

Ich möchte den Assistenten verwenden, um eine Pipeline zu erstellen, mit der eine Beispielanwendung aus einem Amazon S3-Bucket für Amazon EC2 EC2-Instances bereitgestellt wird, auf denen Amazon Linux ausgeführt wird. CodeDeploy Nachdem ich meine Pipeline mit dem Assistenten erstellt haben, möchte ich eine dritte Stufe hinzufügen.	Siehe Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket).
Ich möchte eine zweistufige Pipeline erstellen , mit der eine Beispielanwendung aus einem CodeCommit Repository CodeDeploy auf einer Amazon EC2 EC2-Instance bereitgestellt wird, auf der Amazon Linux ausgeführt wird.	Siehe Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository).
Ich möchte der Pipeline mit drei Stufen, die ich im ersten Tutorial erstellt habe, eine Build-Stufe hinzufügen. Die neue Stufe verwendet Jenkins für die Erstellung meiner Anwendung.	Siehe Tutorial: Erstellen einer vierstufigen Pipeline.
Ich möchte eine CloudWatch Ereignisregel einrichten, die Benachrichtigungen sendet, wenn sich der Ausführungsstatus meiner Pipeline, Phase oder Aktion ändert.	Siehe Tutorial: Richten Sie eine CloudWatch Ereignisregel ein, um E-Mail-Benachrichtigungen für Änderungen des Pipeline-Status zu erhalten.
Ich möchte eine Pipeline mit einer GitHub Quelle erstellen, die eine Android-App mit und erstellt CodeBuild und testet AWS Device Farm.	Siehe Tutorial: Erstellen Sie eine Pipeline, die Ihre Android-App erstellt und testet mit AWS Device Farm.

Ich möchte eine Pipeline mit einer Amazon S3 S3-Quelle erstellen, mit der eine iOS-App getestet wird AWS Device Farm.	Siehe Tutorial: Erstellen Sie eine Pipeline, die Ihre iOS-App testet mit AWS Device Farm.
Ich möchte eine Pipeline erstellen, die meine Produktvorlage für Service Catalog bereitstellt.	Siehe Tutorial: Erstellen Sie eine Pipeline, die in Service Catalog bereitgestellt wird.
Ich möchte Beispielvorlagen verwenden, um mithilfe der AWS CloudFormation Konsole eine einfache Pipeline (mit einem Amazon S3 oder einer GitHub Quelle) zu erstellen. CodeCommit	Siehe Tutorial: Erstellen Sie eine Pipeline mit AWS CloudFormation.
Ich möchte eine zweistufige Pipeline erstellen, die Amazon ECS für die blaue/grüne Bereitstellung eines Images aus einem Amazon ECR-Repository in einem Amazon ECS-Cluster CodeDeploy und -Service verwendet.	Siehe Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy .
Ich möchte eine Pipeline erstellen, die meine serverlose Anwendung kontinuierlich zu AWS Serverless Application Repository veröffentlicht.	Siehe Tutorial: Erstellen Sie eine Pipeline, die Ihre serverlose Anwendung auf dem AWS Serverless Application Repository.

Die folgenden Tutorials in anderen Benutzerhandbüchern bieten Anleitungen zur Integration anderer Anleitungen AWS-Services in Ihre Pipelines:

- Erstellen Sie eine Pipeline, die CodeBuild im AWS CodeBuild Benutzerhandbuch verwendet wird
- Verwendung CodePipeline mit AWS OpsWorks Stacks im AWS OpsWorks Benutzerhandbuch
- Continuous Delivery mit CodePipeline AWS CloudFormation integriertem Benutzerhandbuch
- Erste Schritte mit Elastic Beanstalk im Entwicklerhandbuch AWS Elastic Beanstalk
- Richten Sie eine kontinuierliche Bereitstellungspipeline ein mit CodePipeline

Tutorial: Verwende Git-Tags, um deine Pipeline zu starten

In diesem Tutorial erstellen Sie eine Pipeline, die eine Verbindung zu Ihrem GitHub Repository herstellt, in dem die Quellaktion für den Triggertyp Git-Tags konfiguriert ist. Wenn bei einem Commit ein Git-Tag erstellt wird, startet deine Pipeline. Dieses Beispiel zeigt dir, wie du eine Pipeline erstellst,

die es ermöglicht, anhand der Syntax des Tag-Namens nach Tags zu filtern. Weitere Informationen zum Filtern mit Glob-Mustern finden Sie unterArbeiten mit Glob-Mustern in der Syntax.

In diesem Tutorial wird GitHub über den `CodeStarSourceConnection` Aktionstyp eine Verbindung hergestellt.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Afrika (Kapstadt), Naher Osten (Bahrain) und Europa (Zürich) nicht verfügbar. Hinweise zu anderen verfügbaren Aktionen finden Sie unterProdukt- und Serviceintegrationen mit CodePipeline. Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Themen

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Erstellen Sie mit Ihrem GitHub Konto ein GitHub Repository.
- Halten Sie Ihre GitHub Anmeldedaten bereit. Wenn Sie den verwenden AWS Management Console , um eine Verbindung herzustellen, werden Sie aufgefordert, sich mit Ihren GitHub Anmeldeinformationen anzumelden.

Schritt 1: Öffne CloudShell und klone dein Repository

Du kannst eine Befehlszeilenschnittstelle verwenden, um dein Repository zu klonen, Commits vorzunehmen und Tags hinzuzufügen. Dieses Tutorial startet eine CloudShell Instanz für die Befehlszeilenschnittstelle.

1. Melden Sie sich bei der an AWS Management Console.
2. Wählen Sie in der oberen Navigationsleiste das AWS Symbol aus. Die Hauptseite der AWS Management Console Displays.
3. Wählen Sie in der oberen Navigationsleiste das AWS CloudShell Symbol aus. CloudShell öffnet. Warten Sie, bis die CloudShell Umgebung erstellt ist.

   > **Note**
   >
   > Wenn Sie das CloudShell Symbol nicht sehen, vergewissern Sie sich, dass Sie sich in einer Region befinden, die von unterstützt wird CloudShell. In diesem Tutorial wird davon ausgegangen, dass Sie sich in der Region USA West (Oregon) befinden.

4. Navigieren GitHub Sie in zu Ihrem Repository. Wählen Sie Code und anschließend HTTPS aus. Kopieren Sie den Pfad. Die Adresse zum Klonen Ihres Git-Repositorys wird in die Zwischenablage kopiert.
5. Führen Sie den folgenden Befehl aus, um das Repository zu klonen.

   ```
   git clone https://github.com/<account>/MyGitHubRepo.git
   ```

6. Geben Sie Ihr GitHub Konto ein `Username` und `Password` wenn Sie dazu aufgefordert werden. Für den `Password` Eintrag müssen Sie ein vom Benutzer erstelltes Token anstelle Ihres Kontopassworts verwenden.

Schritt 2: Eine Pipeline zum Auslösen auf Git-Tags erstellen

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellphase mit einer Verbindung zu deinem GitHub Repository und einer Aktion.
- Eine Build-Phase mit einer AWS CodeBuild Build-Aktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich unter https://console.aws.amazon.com/codepipeline/ bei der CodePipeline Konsole an.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyGitHubTagsPipeline`** ein.
4. Behalten Sie unter Pipeline-Typ die Standardauswahl V2 bei. Pipeline-Typen unterscheiden sich in ihren Eigenschaften und im Preis. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle).

   > **Note**
   >
   > Wenn Sie stattdessen Ihre bestehende CodePipeline Servicerolle verwenden möchten, stellen Sie sicher, dass Sie die `codestar-connections:UseConnection` IAM-Berechtigung zu Ihrer Servicerollenrichtlinie hinzugefügt haben. Anweisungen für die CodePipeline Servicerolle finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline Servicerolle.

6. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie unter Artifact store (Artefaktspeicher) die Option Default location (Standardstandort) aus, um den Standard-Artefakt-Speicherort für die Pipeline in der entsprechenden Region zu verwenden, beispielsweise mit dem Amazon S3-Artefakt-Bucket als Standard.

   > **Note**
   >
   > Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket.

   Wählen Sie Next (Weiter).
7. Fügen Sie auf der Seite Step 2: Add source stage (Schritt 2: Quell-Stufe hinzufügen) eine Quellphase hinzu:

   a. Wählen Sie unter Quellanbieter die Option GitHub (Version 2) aus.

b. Wählen Sie unter Verbindung eine bestehende Verbindung aus, oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.

c. Wählen Sie unter Repository-Name den Namen Ihres GitHub Repositorys aus.

d. Wählen Sie unter Pipeline-Trigger die Option Git-Tags aus.

Geben Sie im Feld Include den Wert ein`release*`.

Wählen Sie unter Standardzweig den Branch aus, den Sie angeben möchten, wenn die Pipeline manuell oder mit einem Quellereignis gestartet wird, das kein Git-Tag ist. Wenn die Quelle der Änderung nicht der Auslöser ist oder wenn eine Pipeline-Ausführung manuell gestartet wurde, wird als Änderung der HEAD-Commit aus dem Standardbranch verwendet.

> **Important**
>
> Pipelines, die mit einem Triggertyp von Git-Tags beginnen, werden für WebhookV2-Ereignisse konfiguriert und verwenden nicht das Webhook-Ereignis (Änderungserkennung bei allen Push-Ereignissen), um die Pipeline zu starten.

Wählen Sie Weiter aus.

8. Fügen Sie unter Add build stage (Build-Phase hinzufügen) eine Build-Phase hinzu:

a. Wählen Sie unter Build provider (Build-Anbieter) die Option AWS CodeBuild aus. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region.

b. Wählen Sie Create project (Projekt erstellen) aus.

c. Geben Sie unter Project name (Projektname) einen Namen für dieses Build-Projekt ein.

d. Wählen Sie für Environment image (Umgebungs-Image) die Option Managed image (Verwaltetes Image) aus. Wählen Sie für Operating system (Betriebssystem) die Option Ubuntu aus.

e. Wählen Sie unter Runtime (Laufzeit) die Option Standard aus. Wählen Sie für Image (Abbild) die Option aws/codebuild/standard:5.0 aus.

f. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle) aus.

> **Note**
>
> Notieren Sie sich den Namen Ihrer Servicerolle. CodeBuild Sie benötigen den Rollennamen für den letzten Schritt in diesem Tutorial.

g. Wählen Sie unter BuildSpec bei Build specifications (Build-Spezifikationen) die Option Insert build commands (Build-Befehle einfügen) aus. Wählen Sie Zum Editor wechseln und fügen Sie Folgendes unter Build-Befehle ein.

```
version: 0.2
#env:
  #variables:
     # key: "value"
     # key: "value"
  #parameter-store:
     # key: "value"
     # key: "value"
  #git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
    #commands:
      # - command
      # - command
  #pre_build:
    #commands:
      # - command
      # - command
  build:
    commands:
      -
  #post_build:
    #commands:
      # - command
      # - command
artifacts:
```

```
  files:
     - '*'
    # - location
  name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

h. Wählen Sie Weiter zu CodePipeline. Dadurch kehren Sie zur CodePipeline Konsole zurück und erstellen ein CodeBuild Projekt, das Ihre Build-Befehle für die Konfiguration verwendet. Das Build-Projekt verwendet eine Servicerolle zur Verwaltung von AWS-Service Berechtigungen. Dieser Vorgang kann einige Minuten dauern.

i. Wählen Sie Weiter aus.

9. Wählen Sie auf der Seite Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen) die Option Skip deploy stage (Bereitstellungsstufe überspringen) aus und akzeptieren Sie anschließend die Warnmeldung, indem Sie erneut Skip (Überspringen) auswählen. Wählen Sie Weiter aus.

10. Wählen Sie unter Step 5: Review (Schritt 5: Überprüfen) die Option Create pipeline (Pipeline erstellen) aus.

Schritt 3: Markiere deine Commits zur Veröffentlichung

Nachdem du deine Pipeline erstellt und Git-Tags angegeben hast, kannst du Commits in deinem GitHub Repository taggen. In diesen Schritten kennzeichnest du einen Commit dem `release-1` Tag. Jeder Commit in einem Git-Repository muss ein eindeutiges Git-Tag haben. Wenn Sie den Commit auswählen und ihn taggen, können Sie auf diese Weise Änderungen aus verschiedenen Branches in Ihre Pipeline-Bereitstellung integrieren. Beachten Sie, dass der Tagname release nicht für das Konzept einer Veröffentlichung in gilt GitHub.

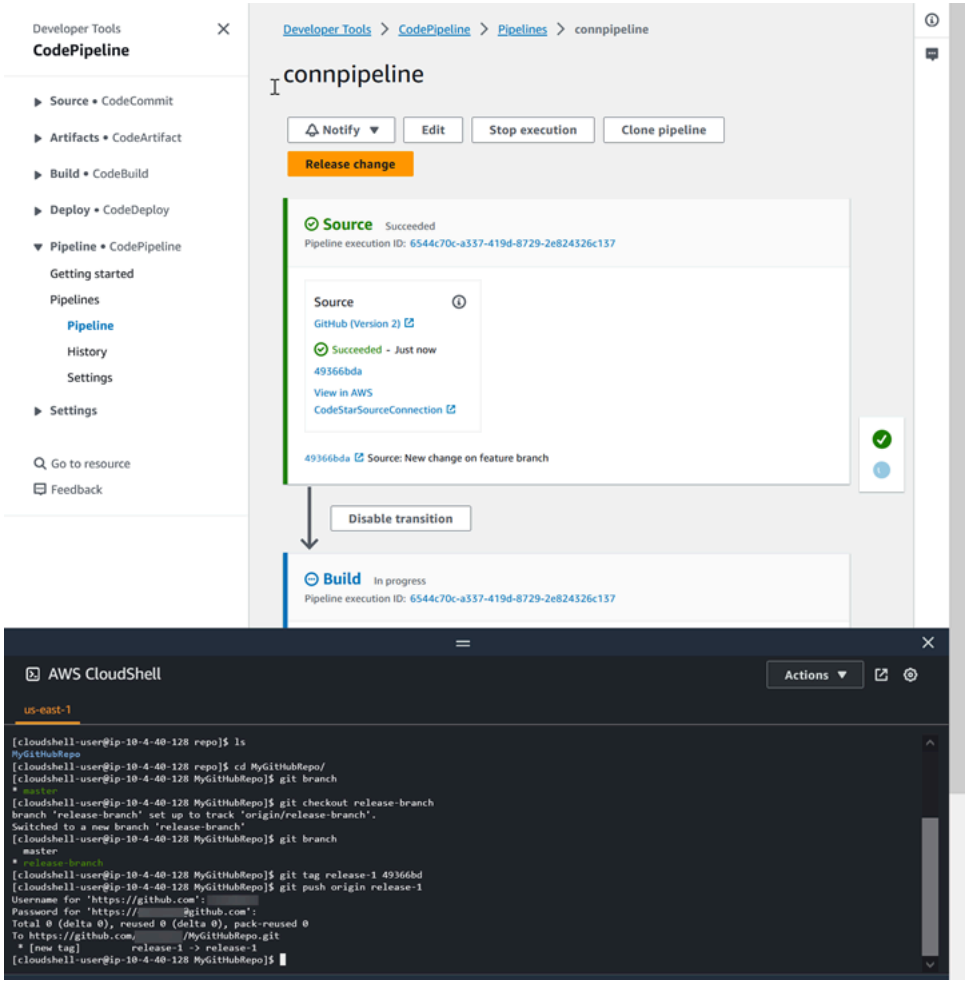
1. Verweisen Sie auf die kopierten Commit-IDs, die Sie taggen möchten. Um die Commits in den einzelnen Branches anzuzeigen, geben Sie im CloudShell Terminal den folgenden Befehl ein, um die Commit-IDs zu erfassen, die Sie taggen möchten:

```
git log
```

2. Gib im CloudShell Terminal den Befehl ein, um deinen Commit zu taggen und ihn an den Ursprung weiterzuleiten. Nachdem du deinen Commit markiert hast, verwendest du den Befehl git push, um den Tag an den Ursprung zu übertragen. Geben Sie im folgenden Beispiel den folgenden Befehl ein, um das `release-1` Tag für den zweiten Commit mit der ID zu verwenden49366bd. Dieses Tag wird durch den `release*` Pipeline-Tagfilter gefiltert und startet die Pipeline.

```
git tag release-1 49366bd
```

```
git push origin release-1
```



Schritt 4: Änderungen veröffentlichen und Logs einsehen

1. Nachdem die Pipeline erfolgreich ausgeführt wurde, wählen Sie in der erfolgreichen Buildphase die Option Protokoll anzeigen aus.

   Sehen Sie sich unter Protokolle die CodeBuild Build-Ausgabe an. Die Befehle geben den Wert der eingegebenen Variablen aus.
2. Sehen Sie sich auf der Seite Verlauf die Spalte Trigger an. Zeigen Sie den Triggertyp an GitTag : Release-1.

Tutorial: Filtere nach Branch-Namen für Pull-Requests, um deine Pipeline zu starten

In diesem Tutorial erstellen Sie eine Pipeline, die eine Verbindung zu Ihrem GitHub .com-Repository herstellt, wo die Quellaktion so konfiguriert ist, dass Ihre Pipeline mit einer Trigger-Konfiguration gestartet wird, die nach Pull-Requests filtert. Wenn ein bestimmtes Pull-Request-Ereignis für einen bestimmten Branch eintritt, wird Ihre Pipeline gestartet. Dieses Beispiel zeigt Ihnen, wie Sie eine Pipeline erstellen, die das Filtern nach Zweignamen ermöglicht. Weitere Informationen zum Arbeiten mit Triggern finden Sie unterTriggerfilterung in der Pipeline-JSON (CLI). Weitere Informationen zum Filtern mit Regex-Mustern im Glob-Format finden Sie unter. Arbeiten mit Glob-Mustern in der Syntax

In dieser Anleitung wird über den Aktionstyp eine Verbindung zu GitHub .com hergestellt. `CodeStarSourceConnection`

Themen

- Voraussetzungen
- Schritt 1: Erstellen Sie eine Pipeline, die bei einem Pull-Request für bestimmte Branches gestartet wird
- Schritt 2: Erstellen Sie eine Pull-Anfrage in GitHub .com und führen Sie sie zusammen, um Ihre Pipeline-Ausführungen zu starten

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Erstellen Sie ein GitHub .com-Repository mit Ihrem GitHub .com-Konto.

- Halten Sie Ihre GitHub Zugangsdaten bereit. Wenn Sie den verwenden AWS Management Console , um eine Verbindung herzustellen, werden Sie aufgefordert, sich mit Ihren GitHub Anmeldeinformationen anzumelden.

Schritt 1: Erstellen Sie eine Pipeline, die bei einem Pull-Request für bestimmte Branches gestartet wird

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellphase mit einer Verbindung zu Ihrem GitHub .com-Repository und einer Aktion.
- Eine Build-Phase mit einer AWS CodeBuild Build-Aktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich unter https://console.aws.amazon.com/codepipeline/ bei der CodePipeline Konsole an.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyFilterBranchesPipeline`** ein.
4. Behalten Sie unter Pipeline-Typ die Standardauswahl V2 bei. Pipeline-Typen unterscheiden sich in ihren Eigenschaften und im Preis. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle).

   > **Note**
   >
   > Wenn Sie stattdessen Ihre bestehende CodePipeline Servicerolle verwenden möchten, stellen Sie sicher, dass Sie die `codeconnections:UseConnection` IAM-Berechtigung zu Ihrer Servicerollenrichtlinie hinzugefügt haben. Anweisungen für die CodePipeline Servicerolle finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline Servicerolle.

6. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie unter Artifact store (Artefaktspeicher) die Option Default location (Standardstandort) aus, um den Standard-Artefakt-Speicherort für die Pipeline in der entsprechenden Region zu verwenden, beispielsweise mit dem Amazon S3-Artefakt-Bucket als Standard.

> **Note**
>
> Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket.

Wählen Sie Next (Weiter).

7. Fügen Sie auf der Seite Step 2: Add source stage (Schritt 2: Quell-Stufe hinzufügen) eine Quellphase hinzu:

   a. Wählen Sie unter Quellanbieter die Option GitHub (Version 2) aus.

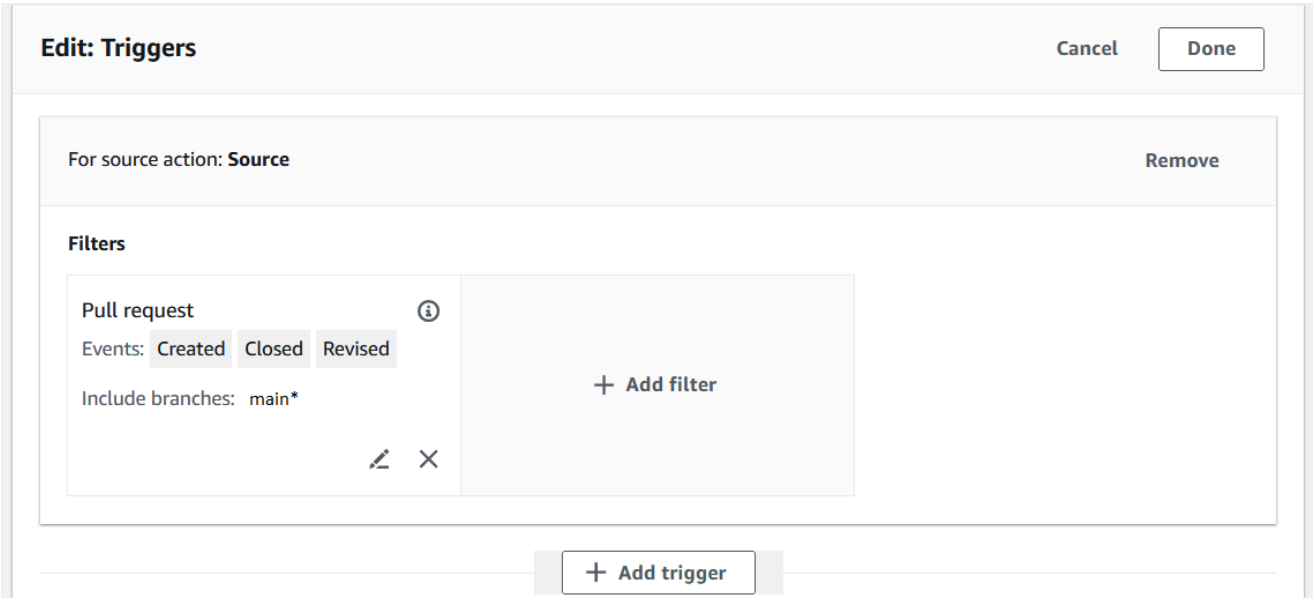
   b. Wählen Sie unter Verbindung eine bestehende Verbindung aus, oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.

   c. Wählen Sie unter Repository-Name den Namen Ihres GitHub .com-Repositorys aus.

   d. Wählen Sie unter Triggertyp die Option Filter angeben aus.

      Wählen Sie unter Ereignistyp die Option Pull-Anfrage aus. Wählen Sie unter Pull Request alle Ereignisse aus, sodass das Ereignis für erstellte, aktualisierte oder geschlossene Pull Requests eintritt.

      Geben Sie unter Branches in das Feld Include den Wert ein`main*`.



> **Important**
>
> Pipelines, die mit diesem Triggertyp beginnen, werden für WebhookV2-Ereignisse konfiguriert und verwenden nicht das Webhook-Ereignis (Änderungserkennung bei allen Push-Ereignissen), um die Pipeline zu starten.

Wählen Sie Weiter aus.

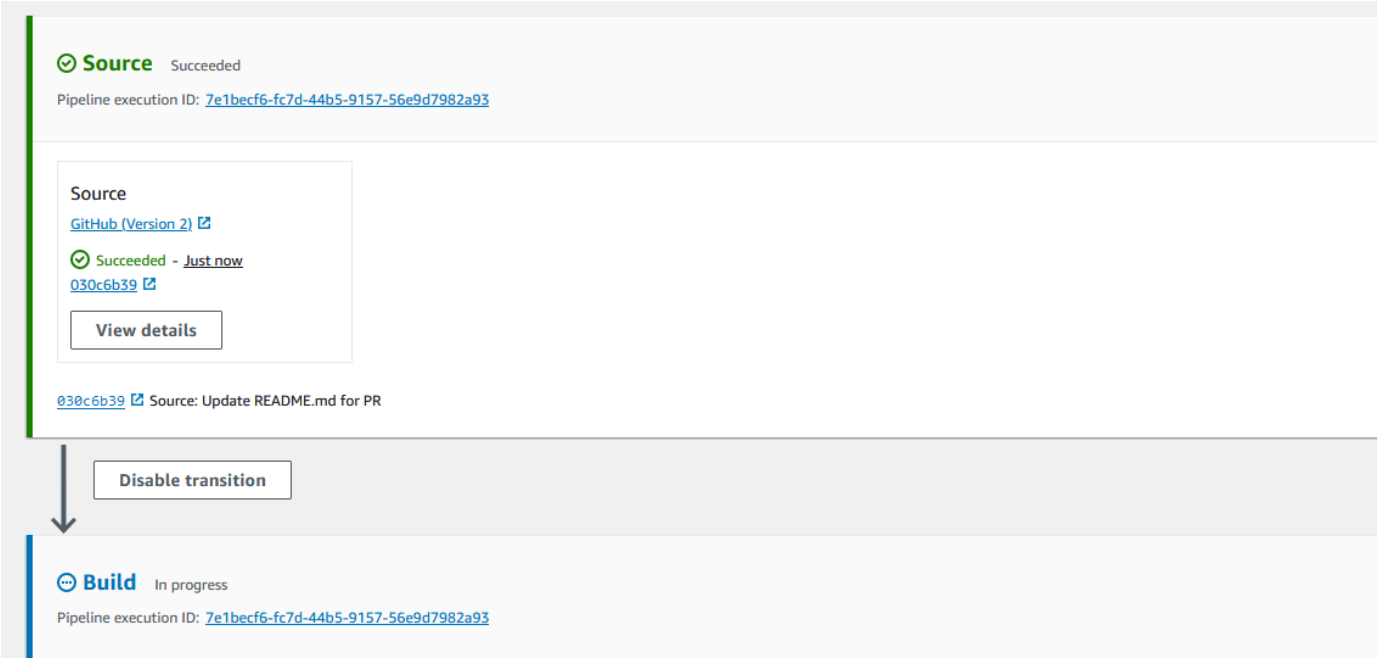
8. Wählen Sie unter Buildphase hinzufügen unter Build Provider die Option aus. AWS CodeBuild Belassen Sie unter Region als Standardeinstellung die Pipeline-Region. Wählen Sie das Build-Projekt aus, oder erstellen Sie es wie unter beschriebenTutorial: Verwende Git-Tags, um deine Pipeline zu starten. Diese Aktion wird in diesem Tutorial nur als zweite Phase verwendet, die zum Erstellen Ihrer Pipeline erforderlich ist.
9. Wählen Sie auf der Seite Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen) die Option Skip deploy stage (Bereitstellungsstufe überspringen) aus und akzeptieren Sie anschließend die Warnmeldung, indem Sie erneut Skip (Überspringen) auswählen. Wählen Sie Weiter aus.
10. Wählen Sie unter Step 5: Review (Schritt 5: Überprüfen) die Option Create pipeline (Pipeline erstellen) aus.

Schritt 2: Erstellen Sie eine Pull-Anfrage in GitHub .com und führen Sie sie zusammen, um Ihre Pipeline-Ausführungen zu starten

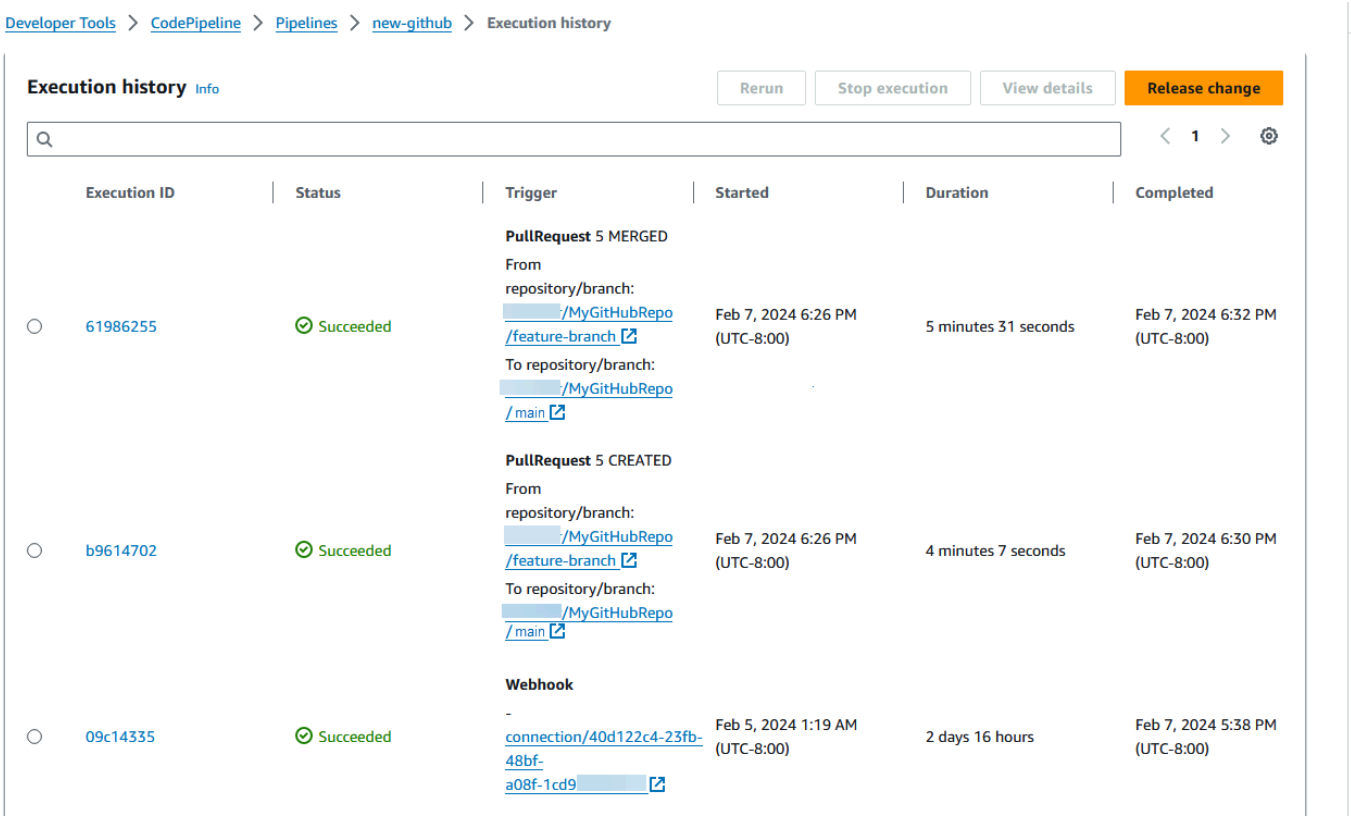
In diesem Abschnitt erstellen Sie einen Pull-Request und führen ihn zusammen. Dadurch wird Ihre Pipeline mit einer Ausführung für den geöffneten Pull-Request und einer Ausführung für den geschlossenen Pull-Request gestartet.

Um eine Pull-Anfrage zu erstellen und Ihre Pipeline zu starten

1. Erstellen Sie in GitHub .com eine Pull-Anfrage, indem Sie eine Änderung an der Datei README.md in einem Feature-Branch vornehmen und eine Pull-Anfrage an den Branch senden. `main` Bestätigen Sie die Änderung mit einer Nachricht wie. `Update README.md for PR`
2. Die Pipeline beginnt mit der Quellrevision, die die Quellnachricht für den Pull-Request als Update README.md for PR anzeigt.



3. Wählen Sie History (Verlauf) aus. Sehen Sie sich im Verlauf der Pipeline-Ausführung die Pull-Request-Statusereignisse CREATED und MERGED an, die die Pipeline-Ausführungen gestartet haben.



Tutorial: Variablen auf Pipeline-Ebene verwenden

In diesem Tutorial erstellen Sie eine Pipeline, in der Sie eine Variable auf Pipeline-Ebene hinzufügen und eine CodeBuild Build-Aktion ausführen, die Ihren Variablenwert ausgibt.

Themen

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Erstellen Sie ein CodeCommit Repository.
- Fügen Sie dem Repository eine .txt-Datei hinzu.

Schritt 1: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellphase mit einer Verbindung zu Ihrem CodeCommit Repository.
- Eine Build-Phase mit einer AWS CodeBuild Build-Aktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich unter https://console.aws.amazon.com/codepipeline/ bei der CodePipeline Konsole an.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyVariablesPipeline`** ein.
4. Behalten Sie unter Pipeline-Typ die Standardauswahl V2 bei. Pipeline-Typen unterscheiden sich in ihren Eigenschaften und im Preis. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle).

> **Note**
>
> Wenn Sie stattdessen Ihre bestehende CodePipeline Servicerolle verwenden möchten, stellen Sie sicher, dass Sie die `codeconnections:UseConnection` IAM-Berechtigung zu Ihrer Servicerollenrichtlinie hinzugefügt haben. Anweisungen für die CodePipeline Servicerolle finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline Servicerolle.

6. Wählen Sie unter Variablen die Option Variable hinzufügen aus. Geben Sie unter Name `timeout` ein. Geben Sie als Standard 1000 ein. Geben Sie im Feld Beschreibung die folgende Beschreibung ein:**`Timeout`**.

   Dadurch wird eine Variable erstellt, in der Sie den Wert deklarieren können, wenn die Pipeline-Ausführung beginnt. Variablennamen müssen übereinstimmen `[A-Za-z0-9@\-_]+` und können alles außer einer leeren Zeichenfolge sein.

7. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie unter Artifact store (Artefaktspeicher) die Option Default location (Standardstandort) aus, um den Standard-Artefakt-Speicherort für die Pipeline in der entsprechenden Region zu verwenden, beispielsweise mit dem Amazon S3-Artefakt-Bucket als Standard.

   > **Note**
   >
   > Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket.

   Wählen Sie Next (Weiter).

8. Fügen Sie auf der Seite Step 2: Add source stage (Schritt 2: Quell-Stufe hinzufügen) eine Quellphase hinzu:

   a. Wählen Sie unter Source provider (Quell-Anbieter) die Option AWS CodeCommit.

   b. Wählen Sie unter Repository-Name und Branch-Name Ihr Repository und Ihren Branch aus.

   Wählen Sie Weiter aus.

9. Fügen Sie unter Add build stage (Build-Phase hinzufügen) eine Build-Phase hinzu:

a. Wählen Sie unter Build provider (Build-Anbieter) die Option AWS CodeBuild aus. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region.

b. Wählen Sie Create project (Projekt erstellen) aus.

c. Geben Sie unter Project name (Projektname) einen Namen für dieses Build-Projekt ein.

d. Wählen Sie für Environment image (Umgebungs-Image) die Option Managed image (Verwaltetes Image) aus. Wählen Sie für Operating system (Betriebssystem) die Option Ubuntu aus.

e. Wählen Sie unter Runtime (Laufzeit) die Option Standard aus. Wählen Sie für Image (Abbild) die Option aws/codebuild/standard:5.0 aus.

f. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle) aus.

> **Note**
>
> Notieren Sie sich den Namen Ihrer CodeBuild Servicerolle. Sie benötigen den Rollennamen für den letzten Schritt in diesem Tutorial.

g. Wählen Sie unter BuildSpec bei Build specifications (Build-Spezifikationen) die Option Insert build commands (Build-Befehle einfügen) aus. Wählen Sie Zum Editor wechseln und fügen Sie Folgendes unter Build-Befehle ein. In der Buildspec `$CUSTOM_VAR1` wird die Kundenvariable verwendet, um die Pipeline-Variable im Build-Log auszugeben. Im folgenden Schritt erstellen Sie die `$CUSTOM_VAR1` Ausgabevariable als Umgebungsvariable.

```
version: 0.2
#env:
  #variables:
     # key: "value"
     # key: "value"
  #parameter-store:
     # key: "value"
     # key: "value"
  #git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
```

```
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
    #commands:
      # - command
      # - command
  #pre_build:
    #commands:
      # - command
      # - command
  build:
    commands:
      - echo $CUSTOM_VAR1
  #post_build:
    #commands:
      # - command
      # - command
artifacts:
  files:
     - '*'
    # - location
  name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

h. Wählen Sie Weiter zu CodePipeline. Dadurch kehren Sie zur CodePipeline Konsole zurück und erstellen ein CodeBuild Projekt, das Ihre Build-Befehle für die Konfiguration verwendet. Das Build-Projekt verwendet eine Servicerolle zur Verwaltung von AWS-Service Berechtigungen. Dieser Vorgang kann einige Minuten dauern.

i. Wählen Sie unter Umgebungsvariablen — optional die Option Umgebungsvariable hinzufügen aus, um eine Umgebungsvariable als Eingabevariable für die Build-Aktion zu erstellen, die durch die Variable auf Pipeline-Ebene aufgelöst wird. Dadurch wird die in der Buildspec angegebene Variable als erstellt. `$CUSTOM_VAR1` Geben Sie unter Name `CUSTOM_VAR1` ein. Geben Sie unter Value (Wert) `#{variables.timeout}` ein. Wählen Sie unter Typ die Option. `Plaintext`

Der `#{variables.timeout}` Wert für die Umgebungsvariable basiert auf dem Variablennamespace auf Pipelineebene `variables` und der Variablen auf Pipelineebene, die in Schritt 5 für die Pipeline `timeout` erstellt wurde.

j. Wählen Sie Weiter aus.

10. Wählen Sie auf der Seite Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen) die Option Skip deploy stage (Bereitstellungsstufe überspringen) aus und akzeptieren Sie anschließend die Warnmeldung, indem Sie erneut Skip (Überspringen) auswählen. Wählen Sie Weiter aus.
11. Wählen Sie unter Step 5: Review (Schritt 5: Überprüfen) die Option Create pipeline (Pipeline erstellen) aus.

Schritt 2: Änderungen veröffentlichen und Protokolle anzeigen

1. Nachdem die Pipeline erfolgreich ausgeführt wurde, wählen Sie in der erfolgreichen Buildphase die Option Details anzeigen aus.

   Wählen Sie auf der Detailseite die Registerkarte Protokolle aus. Sehen Sie sich die CodeBuild Build-Ausgabe an. Die Befehle geben den Wert der eingegebenen Variablen aus.
2. Wählen Sie in der linken Navigationsleiste Historie aus.

   Wählen Sie die letzte Ausführung und dann die Registerkarte Variablen aus. Zeigen Sie den aufgelösten Wert für die Pipeline-Variable an.

Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket)

Der einfachste Weg, eine Pipeline zu erstellen, besteht darin, den Assistenten zum Erstellen von Pipelines in der AWS CodePipeline Konsole zu verwenden.

In diesem Tutorial erstellen Sie eine zweistufige Pipeline, die einen versionierten S3-Bucket verwendet, und CodeDeploy zur Veröffentlichung einer Beispielanwendung.

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, können Sie Ihre Quelldatei (en) in eine einzige ZIP-Datei komprimieren und die ZIP-Datei in Ihren Quell-Bucket hochladen. Sie

können auch eine einzelne Datei ungezippt hochladen, aber nachgelagerte Aktionen, die eine ZIP-Datei erwarten, schlagen dann fehl.

Nach dem Erstellen dieser einfachen Pipeline fügen Sie eine weitere Stufe hinzu und deaktivieren und aktivieren anschließend den Übergang zwischen Stufen.

Important

Viele der Aktionen, die Sie in diesem Verfahren zu Ihrer Pipeline hinzufügen, beinhalten AWS Ressourcen, die Sie erstellen müssen, bevor Sie die Pipeline erstellen. AWS Ressourcen für Ihre Quellaktionen müssen immer in derselben AWS Region erstellt werden, in der Sie Ihre Pipeline erstellen. Wenn Sie Ihre Pipeline beispielsweise in der Region USA Ost (Ohio) erstellen, muss sich Ihr CodeCommit Repository in der Region USA Ost (Ohio) befinden. Sie können beim Erstellen Ihrer Pipeline regionsübergreifende Aktionen hinzufügen. AWS Ressourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

Stellen Sie vor Beginn sicher, dass die unter Erste Schritte mit CodePipeline beschriebenen Voraussetzungen erfüllt sind.

Themen

Schritt 1: Erstellen eines S3-Buckets für Ihre Anwendung

Sie können Ihre Quelldateien oder Anwendungen an jedem versionierten Speicherort speichern. In diesem Tutorial erstellen Sie einen S3-Bucket für die Beispielanwendungsdateien und aktivieren die Versionierung für diesen Bucket. Nach der Aktivierung des Versioning kopieren Sie die Beispielanwendungen zu diesem Bucket.

So erstellen Sie einen S3-Bucket

1. Melden Sie sich bei der Konsole an unter AWS Management Console. Öffnen Sie die S3-Konsole.
2. Wählen Sie Bucket erstellen aus.
3. Geben Sie in Bucket Name (Bucket-Name) einen Namen für Ihren Bucket ein (z. B. **`awscodepipeline-demobucket-example-date`**).

   > **Note**
   >
   > Da alle Bucket-Namen in Amazon S3 eindeutig sein müssen, verwenden Sie einen eigenen Namen, nicht den im Beispiel gezeigten Namen. Sie können den Beispielnamen abändern, indem Sie ihm das Datum hinzufügen. Notieren Sie sich diesen Namen, da Sie ihn für den Rest dieser Anleitung benötigen.

   Wählen Sie unter Region die Region aus, in der Sie Ihre Pipeline erstellen möchten, z. B. USA West (Oregon), und wählen Sie dann Create Bucket aus.
4. Nachdem der Bucket erstellt wurde, wird ein Erfolgsbanner angezeigt. Wählen Sie Go to bucket details (Zu Bucket-Details wechseln) aus.
5. Wählen Sie auf der Registerkarte Properties (Eigenschaften) die Option Versioning aus. Wählen Sie Enable versioning (Versioning aktivieren) und dann Save (Speichern)aus.

   Wenn die Versionierung aktiviert ist, speichert Amazon S3 jede Version jedes Objekts im Bucket.
6. Behalten Sie auf der Registerkarte Permissions (Berechtigungen) die Standardwerte bei. Weitere Informationen über Bucket- und Objekt-Berechtigungen in S3 finden Sie unter Specifying Permissions in a Policy (Berechtigungen in Richtlinien festlegen).
7. Laden Sie als Nächstes ein Beispiel herunter und speichern Sie es in einem Ordner oder Verzeichnis auf Ihrem lokalen Computer.

a. Wählen Sie eine der folgenden Optionen aus. Wählen Sie `SampleApp_Windows.zip` aus, wenn Sie die Schritte in diesem Tutorial für Windows Server-Instances ausführen möchten.

   - Wenn Sie die Bereitstellung auf Amazon Linux-Instances mithilfe durchführen möchten CodeDeploy, laden Sie die Beispielanwendung hier herunter: SampleApp_Linux.zip.
   - Wenn Sie die Bereitstellung auf Windows Server-Instances mithilfe durchführen möchten CodeDeploy, laden Sie die Beispielanwendung hier herunter: SampleApp_Windows.zip.

   Die Beispielanwendung enthält die folgenden Dateien für die Bereitstellung mit CodeDeploy:

   - `appspec.yml`— Die Anwendungsspezifikationsdatei (AppSpecDatei) ist eine Datei im YAML-Format, die CodeDeploy zur Verwaltung einer Bereitstellung verwendet wird. Weitere Informationen zu der AppSpec Datei finden Sie unter CodeDeploy AppSpec Dateireferenz im AWS CodeDeploy Benutzerhandbuch.
   - `index.html`— Die Indexdatei enthält die Homepage für die bereitgestellte Beispielanwendung.
   - `LICENSE.txt`— Die Lizenzdatei enthält Lizenzinformationen für die Beispielanwendung.
   - Dateien für Skripts — Die Beispielanwendung verwendet Skripts, um Textdateien an einen Speicherort auf Ihrer Instance zu schreiben. Für jedes der verschiedenen Ereignisse im CodeDeploy Bereitstellungslebenszyklus wird wie folgt eine Datei geschrieben:
     - `scripts`Ordner (nur Linux-Beispiel) — Der Ordner enthält die folgenden Shell-Skripts zum Installieren von Abhängigkeiten und zum Starten und Beenden der Beispielanwendung für die automatisierte Bereitstellung: `install_dependencies`, `start_server` und `stop_server`.
     - (Nur Windows-Beispiel) `before-install.bat` — Dies ist ein Batch-Skript für das `BeforeInstall` Deployment-Lifecycle-Ereignis, das ausgeführt wird, um alte Dateien zu entfernen, die während früherer Bereitstellungen dieses Beispiels geschrieben wurden, und einen Speicherort auf Ihrer Instanz zu erstellen, in den die neuen Dateien geschrieben werden.

b. Laden Sie die komprimierte ZIP-Datei herunter. Entpacken Sie die Datei nicht.

8. Laden Sie in der Amazon S3 S3-Konsole für Ihren Bucket die Datei hoch:

   a. Klicken Sie auf Hochladen.

   b. Ziehen Sie die Datei oder wählen Sie Add Files (Dateien hinzufügen) aus und suchen Sie die Datei.

c. Klicken Sie auf Hochladen.

Schritt 2: Amazon EC2 EC2-Windows-Instances erstellen und den CodeDeploy Agenten installieren

> **Note**
>
> Dieses Tutorial enthält Beispielschritte für die Erstellung von Amazon EC2 EC2-Windows-Instances. Beispielschritte zum Erstellen von Amazon EC2 EC2-Linux-Instances finden Sie unterSchritt 3: Erstellen Sie eine Amazon EC2 EC2-Linux-Instance und installieren Sie den Agenten CodeDeploy . Wenn Sie zur Eingabe der Anzahl der zu erstellenden Instances aufgefordert werden, geben Sie 2 an.

In diesem Schritt erstellen Sie die Windows Server Amazon EC2 EC2-Instances, auf denen Sie eine Beispielanwendung bereitstellen werden. Im Rahmen dieses Prozesses erstellen Sie eine Instance-Rolle mit Richtlinien, die die Installation und Verwaltung des CodeDeploy Agenten auf den Instances ermöglichen. Der CodeDeploy Agent ist ein Softwarepaket, das die Verwendung einer Instanz in CodeDeploy Bereitstellungen ermöglicht. Sie fügen auch Richtlinien hinzu, die es der Instanz ermöglichen, Dateien abzurufen, die der CodeDeploy Agent zur Bereitstellung Ihrer Anwendung verwendet, und die Verwaltung der Instanz durch SSM zu ermöglichen.

So erstellen Sie eine Instance-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/).
2. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen AWS-Servicedie Option aus. Wählen Sie unter Choose a use case (Anwendungsfall auswählen) die Option EC2 und dann Next: Permissions (Weiter: Berechtigungen) aus.
5. Suchen Sie nach der genannten Richtlinie und wählen Sie sie aus **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Suchen Sie nach der genannten Richtlinie und wählen Sie sie aus **`AmazonSSMManagedInstanceCore`**. Wählen Sie Next: Tags (Weiter: Tags) aus.

7. Klicken Sie auf Weiter: Prüfen. Geben Sie einen Namen für die Rolle ein (z. B. **`EC2InstanceRole`**).

   > **Note**
   >
   > Notieren Sie sich Ihren Rollennamen für den nächsten Schritt. Sie wählen diese Rolle, wenn Sie Ihre Instance erstellen.

   Wählen Sie Rolle erstellen aus.

Starten von Instances

1. Öffnen Sie die Amazon EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie in der Seitennavigation Instances und dann oben auf der Seite Launch Instances aus.
3. Geben Sie unter Name und Tags im Feld Name den Wert ein**`MyCodePipelineDemo`**. Dadurch wird den Instanzen ein Tag-Schlüssel von **`Name`** und ein Tag-Wert von **`MyCodePipelineDemo`** zugewiesen. Später erstellen Sie eine CodeDeploy Anwendung, die die Beispielanwendung auf den Instanzen bereitstellt. CodeDeploywählt die bereitzustellenden Instanzen auf der Grundlage der Tags aus.
4. Wählen Sie unter Anwendungs- und Betriebssystem-Images (Amazon Machine Image) die Option Windows aus. (Dieses AMI wird als Microsoft Windows Server 2019 Base beschrieben und trägt die Bezeichnung „Kostenloses Nutzungskontingent“. Es ist unter Quick Start zu finden.)
5. Wählen Sie unter Instance-Typ den `t2.micro` Typ aus, für den das kostenlose Kontingent in Frage kommt, als Hardwarekonfiguration für Ihre Instance.
6. Wählen Sie unter key pair (Anmeldung) ein Schlüsselpaar aus oder erstellen Sie eines.

   Sie können auch Proceed without a key pair wählen.

   > **Note**
   >
   > Für die Zwecke dieses Tutorials können Sie ohne Schlüsselpaar fortfahren. Um SSH zu verwenden, wenn eine Verbindung zu Ihren Instances herstellen, erstellen Sie ein Schlüsselpaar oder verwenden Sie ein vorhandenes.

7. Gehen Sie unter Netzwerkeinstellungen wie folgt vor.

Vergewissern Sie sich, dass unter Öffentliche IP automatisch zuweisen der Status Enable lautet.

- Klicken Sie neben Assign a security group (Sicherheitsgruppe zuweisen) auf Create a new security group (Neue Sicherheitsgruppe zuweisen).
- Wählen Sie in der Zeile für SSH unter Quelltyp die Option Meine IP aus.
- Wählen Sie Sicherheitsgruppe hinzufügen, wählen Sie HTTP und dann unter Quelltyp die Option Meine IP aus.

8. Erweitern Sie Advanced Details (Erweiterte Details). Wählen Sie im IAM-Instanzprofil die IAM-Rolle aus, die Sie im vorherigen Verfahren erstellt haben (z. B.**`EC2InstanceRole`**).
9. Geben Sie unter Zusammenfassung unter Anzahl der Instanzen den Wert.. 2
10. Wählen Sie Launch Instance (Instance starten) aus.
11. Wählen Sie View all Instances (Alle Instances anzeigen) aus, um die Bestätigungsseite zu schließen und zur Konsole zurückzukehren.
12. Sie können den Status des Starts auf der Seite Instances anzeigen. Wenn Sie eine Instance starten, lautet ihr anfänglicher Status `pending`. Nachdem die Instance gestartet ist, lautet ihr Status `running`. Sie erhält dann einen öffentlichen DNS-Namen. (Wenn die Spalte Public DNS nicht angezeigt wird, wählen Sie das Show/Hide-Symbol aus und wählen Sie dann Public DNS aus.)
13. Es kann ein paar Minuten dauern, bis die Instance zur Verbindung bereitsteht. Stellen Sie sicher, dass Ihre Instance ihre Statusprüfungen bestanden hat. Sie können diese Informationen in der Spalte Statusprüfungen abrufen.

Schritt 3: Erstellen Sie eine Anwendung in CodeDeploy

CodeDeployIn ist eine Anwendung eine Kennung in Form eines Namens für den Code, den Sie bereitstellen möchten. CodeDeploy verwendet diesen Namen, um sicherzustellen, dass während einer Bereitstellung auf die richtige Kombination aus Version, Bereitstellungskonfiguration und Bereitstellungsgruppe verwiesen wird. Sie wählen den Namen der CodeDeploy Anwendung, die Sie in diesem Schritt erstellen, wenn Sie später in diesem Tutorial Ihre Pipeline erstellen.

Sie erstellen zunächst eine Servicerolle CodeDeploy zur Verwendung. Wenn Sie bereits eine Servicerolle erstellt haben, müssen Sie keine weitere erstellen.

Um eine CodeDeploy Servicerolle zu erstellen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/).
2. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Vertrauenswürdige Entität auswählen die Option AWS-Service. Wählen Sie unter Use case (Anwendungsfall) CodeDeploy aus. Wählen Sie CodeDeployaus den aufgelisteten Optionen aus. Wählen Sie Weiter aus. Die `AWSCodeDeployRole`-verwaltete Richtlinie ist der Rolle bereits zugewiesen.
5. Wählen Sie Weiter aus.
6. Geben Sie einen Namen für die Rolle ein (beispielsweise **`CodeDeployRole`**) und wählen Sie Next (Weiter) aus.

Um eine Anwendung zu erstellen in CodeDeploy

1. Öffnen Sie die CodeDeploy Konsole unter https://console.aws.amazon.com/codedeploy.
2. Wenn die Seite „Anwendungen“ nicht angezeigt wird, wählen Sie im AWS CodeDeploy Menü „Anwendungen“.
3. Wählen Sie Create application aus.
4. Geben Sie unter Application name (Anwendungsname) `MyDemoApplication` ein.
5. Wählen Sie unter Compute Platform (Plattform für die Datenverarbeitung) die Option EC2/On-premises (EC2/Lokal) aus.
6. Wählen Sie Create application aus.

Um eine Bereitstellungsgruppe zu erstellen in CodeDeploy

1. Wählen Sie auf der Seite mit Ihrer Anwendung Create deployment group (Bereitstellungsgruppe erstellen).
2. Geben Sie unter Deployment group name (Name der Bereitstellungsgruppe) **`MyDemoDeploymentGroup`** ein.
3. Wählen Sie unter Servicerolle die Servicerolle aus, die Sie zuvor erstellt haben. Sie müssen eine Servicerolle verwenden, die mindestens den unter Eine Servicerolle erstellen für CodeDeploy beschriebenen Vertrauensstellungen und Berechtigungen vertraut AWS CodeDeploy .

Informationen zum Erhalt des Servicerollen-ARN finden Sie unter Abrufen des Servicerollen-ARN (Konsole).

4. Wählen Sie unter Deployment type (Bereitstellungstyp) die Option In-place (Direkt).
5. Wählen Sie unter Environment configuration (Umgebungskonfiguration) die Option Amazon EC2-Instances aus. Wählen Sie im Feld Schlüssel die Option Name und geben Sie im Feld Wert die Eingabe ein**`MyCodePipelineDemo`**.

   > **Important**
   >
   > Sie müssen hier denselben Wert für den Schlüssel Name auswählen, den Sie Ihrer EC2-Instance beim Erstellen zugewiesen haben. Wenn Sie die Instances nicht mit **`MyCodePipelineDemo`** markiert haben, sollten Sie Ihren Wert verwenden.

6. Wählen Sie unter Agentenkonfiguration mit AWS Systems Manager die Option Jetzt und planen Sie Updates. Dadurch wird der Agent auf der Instanz installiert. Die Windows-Instanz ist bereits mit dem SSM-Agenten konfiguriert und wird nun mit dem CodeDeploy Agenten aktualisiert.
7. Wählen Sie unter Bereitstellungseinstellungen die Option`CodeDeployDefault.OneAtaTime`.
8. Stellen Sie sicher, dass unter Load Balancer das Feld Load Balancing aktivieren nicht ausgewählt ist. Sie müssen keinen Load Balancer einrichten oder eine Zielgruppe für dieses Beispiel auswählen. Nachdem Sie das Kontrollkästchen deaktiviert haben, werden die Load Balancer-Optionen nicht angezeigt.
9. Behalten Sie im Abschnitt Advanced (Fortgeschritten) die Standardwerte bei.
10. Wählen Si Create deployment group (Bereitstellungsgruppe erstellen).

Schritt 4: Erstellen Sie Ihre erste Pipeline in CodePipeline

In diesem Teil des Tutorials erstellen Sie die Pipeline. Das Beispiel wird automatisch durch die Pipeline geführt.

Um einen CodePipeline automatisierten Release-Prozess zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) aus, oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).

3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyFirstPipeline`** ein.

   > **Note**
   >
   > Wenn Sie einen anderen Namen für Ihre Pipeline verwenden, stellen Sie sicher, dass Sie im Rest dieses Tutorials diesen Namen anstelle von **`MyFirstPipeline`** verwenden. Der Name einer erstellten Pipeline kann nicht nachträglich geändert werden. Pipeline-Namen unterliegen einigen Einschränkungen. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline.

4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:
   - Wählen Sie Neue Servicerolle aus, CodePipeline um die Erstellung einer neuen Servicerolle in IAM zu ermöglichen.
   - Wählen Sie Existing service role (Vorhandene Servicerolle), um eine Servicerolle zu verwenden, die in IAM bereits erstellt wurde. Wählen Sie unter Name der Rolle in der Liste Ihre Servicerolle aus.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie in Step 2: Add source stage (Schritt 2: Quellstufe hinzufügen) unter Source provider (Quellanbieter) die Option Amazon S3. Geben Sie unter Bucket den Namen des S3-Buckets ein, den Sie in Schritt 1: Erstellen eines S3-Buckets für Ihre Anwendung erstellt haben. Geben Sie im S3-Objektschlüssel den Objektschlüssel mit oder ohne Dateipfad ein, und denken Sie daran, die Dateierweiterung einzuschließen. Geben Sie beispielsweise für `SampleApp_Windows.zip` den Namen der Beispieldatei ein, wie in diesem Beispiel gezeigt:

   ```
   SampleApp_Windows.zip
   ```

   Klicken Sie auf Nächster Schritt.

   Lassen Sie die Standardwerte in Change detection options unverändert. Auf diese Weise können CodePipeline Sie Amazon CloudWatch Events verwenden, um Änderungen in Ihrem Quell-Bucket zu erkennen.

Wählen Sie Weiter aus.

8. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken. Wählen Sie Weiter aus.
9. Wählen Sie in Schritt 4: Bereitstellungsphase hinzufügen unter Bereitstellungsanbieter die Option CodeDeploy . Das Feld Region entspricht AWS-Region standardmäßig dem Feld Ihrer Pipeline. Geben Sie unter Application name (Anwendungsname) `MyDemoApplication` ein, oder klicken Sie auf die Schaltfläche Refresh (Aktualisieren), und wählen Sie dann den Namen der Anwendung aus der Liste aus. Geben Sie unter Deployment group (Bereitstellungsgruppe) **`MyDemoDeploymentGroup`** ein oder wählen Sie den Eintrag aus der Liste. Klicken Sie dann auf Next (Weiter).

   > **Note**
   >
   > Der Name „Deploy“ (Bereitstellen) ist der Name, der der Phase in Step 4: Add deploy stage (Schritt 4: Bereitstellungsphase hinzufügen) standardmäßig gegeben wird, so wie auch der ersten Phase der Pipeline der Name „Source“ (Quelle) gegeben wird.

10. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.
11. Die Pipeline startet die Ausführung. Sie können sich Fortschritts-, Erfolgs- und Fehlschlagsmeldungen anzeigen lassen, während das CodePipeline Beispiel eine Webseite für jede Amazon EC2 EC2-Instance in der CodeDeploy Bereitstellung bereitstellt.

Herzlichen Glückwunsch! Sie haben gerade eine einfache Pipeline in erstellt. CodePipeline Die Pipeline hat zwei Phasen:

- Eine Quellstufe mit dem Namen Source (Quelle), die Änderungen in der Versioning-Beispielanwendung erkennt, die im S3-Bucket gespeichert ist, und diese Änderungen in die Pipeline zieht.
- Eine Bereitstellungsphase, in der diese Änderungen auf EC2-Instances bereitgestellt werden. CodeDeploy

Anschließend überprüfen Sie die Ergebnisse.

So verifizieren Sie, ob Ihre Pipeline erfolgreich ausgeführt wurde

1. Zeigen Sie den anfänglichen Fortschritt der Pipeline an. Der Status jeder Phase ändert sich von Zurzeit keine Ausführungen zu Laufend. Danach ändert er sich entweder zu Erfolgreich oder Fehlgeschlagen. Die Pipeline sollte die erste Ausführung innerhalb weniger Minuten abschließen.
2. Nachdem Succeeded (Erfolgreich) für den Aktionsstatus angezeigt wird, wählen Sie im Statusbereich für die Phase Deploy (Bereitstellen) die Option Details aus. Dadurch wird die CodeDeploy Konsole geöffnet.
3. Wählen Sie auf der Registerkarte Deployment group (Bereitstellungsgruppe) unter Deployment lifecycle events (Bereitstellungslebenszyklusereignisse) eine Instance-ID aus. Daraufhin wird die EC2-Konsole geöffnet.
4. Kopieren Sie auf der Registerkarte Description in Public DNS die Adresse und fügen Sie diese in die Adressleiste Ihres Webbrowsers ein. Zeigen Sie die Indexseite für die Beispielanwendung an, die Sie in Ihren S3-Bucket hochgeladen haben.

   Die Webseite für die Beispielanwendung, die Sie in Ihren S3-Bucket hochgeladen haben, wird angezeigt.

Weitere Informationen über Phasen, Aktionen und die Funktionsweise von Pipelines finden Sie unter CodePipeline Konzepte .

Schritt 5: Hinzufügen einer weiteren Phase zur Pipeline

Fügen Sie nun eine weitere Phase in der Pipeline hinzu, um die Bereitstellung von Staging-Servern auf Produktionsserver mithilfe CodeDeploy von. Zunächst erstellen Sie eine weitere Bereitstellungsgruppe CodePipelineDemoApplication in CodeDeploy. Anschließend fügen Sie eine Phase hinzu, die eine Aktion enthält, die diese Bereitstellungsgruppe verwendet. Um eine weitere Phase hinzuzufügen, verwenden Sie die CodePipeline Konsole oder die, AWS CLI um die Struktur der Pipeline in einer JSON-Datei abzurufen und manuell zu bearbeiten, und führen dann den update-pipeline Befehl aus, um die Pipeline mit Ihren Änderungen zu aktualisieren.

Themen

- Erstellen Sie eine zweite Bereitstellungsgruppe in CodeDeploy
- Hinzufügen der Bereitstellungsgruppe als weitere Phase in Ihrer Pipeline

Erstellen Sie eine zweite Bereitstellungsgruppe in CodeDeploy

> **Note**
>
> In diesem Teil des Tutorials erstellen Sie eine zweite Bereitstellungsgruppe, die Bereitstellung erfolgt jedoch auf denselben Amazon EC2 EC2-Instances wie zuvor. Dies erfolgt lediglich zu Demonstrationszwecken. Es ist bewusst so konzipiert, dass es Ihnen nicht zeigt, wie Fehler in angezeigt werden. CodePipeline

Um eine zweite Bereitstellungsgruppe in zu erstellen CodeDeploy

1. Öffnen Sie die CodeDeploy Konsole unter https://console.aws.amazon.com/codedeploy.
2. Wählen Sie die Liste Applications (Anwendungen) aus, und wählen Sie in der Liste `MyDemoApplication` aus.
3. Wählen Sie die Registerkarte Deployment groups (Bereitstellungsgruppen) und klicken Sie dann auf Create deployment group (Bereitstellungsgruppe erstellen).
4. Geben Sie auf der Seite Create deployment group (Bereitstellungsgruppe erstellen) unter Deployment group name (Name der Bereitstellungsgruppe) einen Namen für die zweite Bereitstellungsgruppe ein (z. B. **`CodePipelineProductionFleet`**).
5. Wählen Sie unter Service Role dieselbe CodeDeploy Servicerolle aus, die Sie für die erste Bereitstellung verwendet haben (nicht die CodePipeline Servicerolle).
6. Wählen Sie unter Deployment type (Bereitstellungstyp) die Option In-place (Direkt).
7. Wählen Sie unter Environment configuration (Umgebungskonfiguration) die Option Amazon EC2-Instances aus. Wählen Sie Name im Feld Key (Schlüssel) aus und wählen Sie im Feld Value (Wert) aus der Liste die Option `MyCodePipelineDemo` aus. Lassen Sie die Standardkonfiguration für Deployment sttings (Bereitstellungseinstellungen) unverändert.
8. Wählen Sie in Deployment configuration (Bereitstellungskonfiguration) die Option `CodeDeployDefault.OneAtaTime` aus.
9. Deaktivieren Sie unter Load Balancer die Option Enable load balancing (Load Balancing aktivieren).
10. Wählen Si Create deployment group (Bereitstellungsgruppe erstellen).

Hinzufügen der Bereitstellungsgruppe als weitere Phase in Ihrer Pipeline

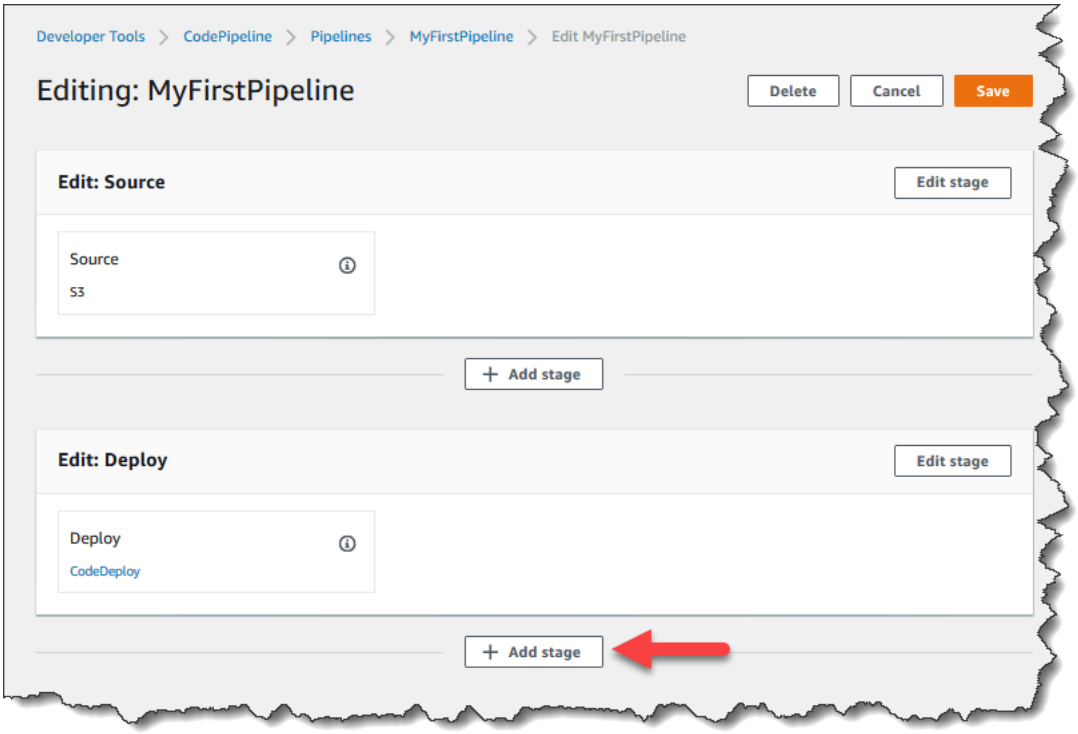
Da Sie nun eine weitere Bereitstellungsgruppe erstellt haben, können Sie eine Phase hinzufügen, die diese Bereitstellungsgruppe verwendet, um eine Bereitstellung für die gleichen EC2-Instances wie zuvor auszuführen. Sie können die CodePipeline Konsole oder die verwenden AWS CLI , um diese Phase hinzuzufügen.

Themen

Erstellen einer dritten Phase (Konsole)

Sie können die CodePipeline Konsole verwenden, um eine neue Phase hinzuzufügen, die die neue Bereitstellungsgruppe verwendet. Da diese Bereitstellungsgruppe Bereitstellungen für die gleichen EC2-Instances wie zuvor ausführt, schlägt die Bereitstellungsaktion in dieser Phase fehl.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie unter Name den Namen der Pipeline aus, die Sie erstellt haben, MyFirstPipeline.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie auf der Seite Edit (Bearbeiten) die Option + Add stage (Phase hinzufügen) aus, um eine Phase unmittelbar hinter der Phase "Deploy" (Bereitstellung) hinzuzufügen.



5. Geben Sie unter Add stage (Stufe hinzufügen) in das Feld Stage name (Stufenname) **`Production`** ein. Klicken Sie auf Add stage (Phase hinzufügen).
6. Wählen Sie in der neuen Phase + Add action group (Aktionsgruppe hinzufügen).
7. Geben Sie unter Edit action (Aktion bearbeiten) in das Feld Action name (Aktionsname) **`Deploy-Second-Deployment`** ein. Wählen Sie unter Aktionsanbieter unter Bereitstellen die Option aus CodeDeploy.
8. Wählen Sie im CodeDeploy Abschnitt Anwendungsname `MyDemoApplication` aus der Dropdownliste aus, wie Sie es beim Erstellen der Pipeline getan haben. Wählen Sie in Deployment group (Bereitstellungsgruppe) die Bereitstellungsgruppe aus, die Sie gerade erstellt haben, **`CodePipelineProductionFleet`**. Wählen Sie unter Input artifacts (Eingabe-Artefakte) das Eingabe-Artefakt aus der Quellaktion aus. Wählen Sie Speichern.
9. Klicken Sie auf der Seite Edit (Bearbeiten) auf Save (Speichern). Klicken Sie unter Save pipeline changes (Pipeline-Änderungen speichern) auf Save (Speichern).
10. Obwohl die neue Stufe der Pipeline hinzugefügt wurde, wird der Status No executions yet (Zurzeit keine Ausführungen) angezeigt. Dies liegt daran, dass noch keine Änderungen eine weitere Ausführung der Pipeline ausgelöst haben. Sie müssen die letzte Revision manuell erneut ausführen, um die Ausführung der Pipeline nach der Bearbeitung zu testen. Wählen Sie auf der Seite mit den Pipeline-Details die Option Versionsänderung aus und klicken Sie dann auf

Freigabe, wenn Sie dazu aufgefordert werden. Dadurch wird die letzte Revision gestartet, die in jedem in einer Quellaktion der Pipeline angegebenen Quellspeicherort vorhanden ist.

AWS CLI Um die Pipeline erneut auszuführen, führen Sie alternativ den Befehl von einem Terminal auf Ihrem lokalen Linux-, macOS- oder Unix-Computer oder von einer Befehlszeile auf Ihrem lokalen Windows-Computer aus und geben Sie den Namen der Pipeline an. start-pipeline-execution Dadurch wird die Anwendung in Ihrem Quell-Bucket über die Pipeline ein zweites Mal ausgeführt.

```
aws codepipeline start-pipeline-execution --name MyFirstPipeline
```

Dieser Befehl gibt ein `pipelineExecutionId`-Objekt zurück.

11. Kehren Sie zur CodePipeline Konsole zurück und wählen Sie in der Liste der Pipelines, ob Sie die Ansichtsseite öffnen MyFirstPipelinemöchten.

    Die Pipeline zeigt drei Phasen und den Status des ausgeführten Artefakts in den drei Phasen an. Es kann bis zu fünf Minuten dauern, bis die Pipeline alle Phasen durchlaufen hat. Für die ersten zwei Stufen wird wie zuvor die erfolgreiche Bereitstellung angezeigt. Die Stufe Production (Produktion) zeigt jedoch an, dass die Aktion Deploy-Second-Deployment fehlgeschlagen ist.

12. Klicken Sie in der Deploy-Second-Deployment--Aktion auf Details. Sie werden auf die Seite für die CodeDeploy Bereitstellung weitergeleitet. In diesem Fall wird der Fehler durch die Bereitstellung der ersten Instance-Gruppe in allen EC2-Instances verursacht. Somit verbleiben keine Instances für die zweite Bereitstellungsgruppe.

    > **Note**
    >
    > Dieser Fehler ist beabsichtigt. Er soll demonstrieren, was passiert, wenn es einen Fehler in einer Pipeline-Phase gibt.

Erstellen einer dritten Phase (CLI)

Das Hinzufügen einer Phase AWS CLI zu Ihrer Pipeline ist zwar komplexer als das Verwenden der Konsole, bietet jedoch einen besseren Einblick in die Struktur der Pipeline.

So erstellen Sie eine dritte Phase für Ihre Pipeline

1. Öffnen Sie eine Terminalsitzung auf Ihrem lokalen Linux-, macOS- oder Unix-Computer oder eine Befehlszeile auf Ihrem lokalen Windows-Computer und führen Sie den get-pipeline Befehl aus, um die Struktur der Pipeline anzuzeigen, die Sie gerade erstellt haben. Für **MyFirstPipeline** geben Sie beispielsweise den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name "MyFirstPipeline"
   ```

   Dieser Befehl gibt die Struktur von zurück MyFirstPipeline. Der erste Teil der Ausgabe sollte wie folgt aussehen:

   ```
   {
       "pipeline": {
           "roleArn": "arn:aws:iam::80398EXAMPLE:role/AWS-CodePipeline-Service",
           "stages": [
       ...
   ```

   Der letzte Teil der Ausgabe enthält die Pipelinemetadaten und sollte in etwa wie folgt aussehen:

   ```
       ...
           ],
           "artifactStore": {
               "type": "S3"
               "location": "codepipeline-us-east-2-250656481468",
           },
           "name": "MyFirstPipeline",
           "version": 4
       },
       "metadata": {
           "pipelineArn": "arn:aws:codepipeline:us-
   east-2:80398EXAMPLE:MyFirstPipeline",
           "updated": 1501626591.112,
           "created": 1501626591.112
       }
   }
   ```

2. Kopieren Sie diese Struktur in einen Texteditor und speichern Sie die Datei als **pipeline.json**. Speichern Sie diese Datei der Einfachheit halber in dem Verzeichnis, in dem Sie die aws codepipeline-Befehle ausgeführt haben.

> **Note**
>
> Sie können den JSON-Code mit dem Befehl get-pipeline folgendermaßen direkt in eine Datei umleiten:
>
> ```
> aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
> ```

3. Kopieren Sie den Phasenabschnitt Deploy (Bereitstellen) und fügen Sie ihn hinter den ersten beiden Phasen ein. Da es sich um eine Bereitstellungsphase handelt (genau wie die Phase Staging), können Sie diese als Vorlage für die dritte Phase nutzen.
4. Ändern Sie den Namen der Phase und die Details der Bereitstellungsgruppe.

   Das folgende Beispiel zeigt den JSON-Code, den Sie der Datei pipeline.json nach der Bereitstellungsphase hinzufügen. Ändern Sie die Werte der hervorgehobenen Elemente. Denken Sie daran, die Phasendefinitionen Deploy (Bereitstellen) und Production (Produktion) durch ein Komma zu trennen.

```
,
{
    "name": "Production",
     "actions": [
        {
         "inputArtifacts": [
             {
              "name": "MyApp"
             }
          ],
          "name": "Deploy-Second-Deployment",
          "actionTypeId": {
              "category": "Deploy",
              "owner": "AWS",
              "version": "1",
              "provider": "CodeDeploy"
              },
          "outputArtifacts": [],
          "configuration": {
              "ApplicationName": "CodePipelineDemoApplication",
              "DeploymentGroupName": "CodePipelineProductionFleet"
               },
          "runOrder": 1
```

```
        }
    ]
}
```

5. Wenn Sie mit einer Pipeline-Struktur arbeiten, die Sie mit dem Befehl get-pipeline abgerufen haben, müssen Sie die `metadata`-Zeilen aus der JSON-Datei entfernen. Andernfalls kann der update-pipeline-Befehl sie nicht nutzen. Entfernen Sie die `"metadata": { }`-Zeilen und die Felder `"created"`, `"pipelineARN"` und `"updated"`.

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

   ```
   "metadata": {
     "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
     "created": "date",
     "updated": "date"
     }
   ```

   Speichern Sie die Datei.

6. Führen Sie den Befehl update-pipeline aus. Geben Sie die Pipeline-JSON-Datei dabei folgendermaßen an:

   ```
   aws codepipeline update-pipeline --cli-input-json file://pipeline.json
   ```

   Dieser Befehl gibt die gesamte Struktur der aktualisierten Pipeline zurück.

   > **Important**
   >
   > Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

7. Führen Sie den Befehl start-pipeline-execution aus und geben Sie dabei den Namen der Pipeline an. Dadurch wird die Anwendung in Ihrem Quell-Bucket über die Pipeline ein zweites Mal ausgeführt.

   ```
   aws codepipeline start-pipeline-execution --name MyFirstPipeline
   ```

   Dieser Befehl gibt ein `pipelineExecutionId`-Objekt zurück.

8. Öffnen Sie die CodePipeline Konsole und wählen Sie MyFirstPipelineaus der Liste der Pipelines aus.

   Die Pipeline zeigt drei Phasen und den Status des ausgeführten Artefakts in den drei Phasen an. Es kann bis zu fünf Minuten dauern, bis die Pipeline alle Phasen durchlaufen hat. Die ersten zwei Phasen werden, wie zuvor, erfolgreich bereitgestellt. Die Phase Production zeigt an, dass die Aktion Deploy-Second-Deployment- fehlgeschlagen ist.

9. Klicken Sie in der Aktion Deploy-Second-Deployment- auf Details, um Details zum Fehler anzuzeigen. Sie werden auf die Detailseite für die CodeDeploy Bereitstellung weitergeleitet. In diesem Fall wird der Fehler durch die Bereitstellung der ersten Instance-Gruppe in allen EC2-Instances verursacht. Somit verbleiben keine Instances für die zweite Bereitstellungsgruppe.

> **Note**
>
> Dieser Fehler ist beabsichtigt. Er soll demonstrieren, was passiert, wenn es einen Fehler in einer Pipeline-Phase gibt.

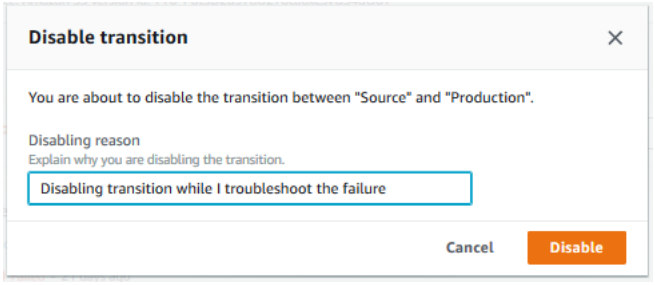
(Optional) Schritt 6: Deaktivieren und aktivieren Sie Übergänge zwischen den Phasen in CodePipeline

Sie können den Übergang zwischen Phasen in einer Pipeline aktivieren oder deaktivieren. Durch die Deaktivierung des Übergangs zwischen Phasen können Sie die Übergänge zwischen Phasen manuell steuern. Beispielsweise möchten Sie die ersten beiden Phasen einer Pipeline ausführen, den Übergang zur dritten Phase jedoch deaktivieren, bis Sie für die Bereitstellung für die Produktion bereit sind oder während Sie ein Problem oder einen Ausfall für die betreffende Phase beheben.

Um Übergänge zwischen Stufen in einer CodePipeline Pipeline zu deaktivieren und zu aktivieren

1. Öffnen Sie die CodePipeline Konsole und wählen Sie MyFirstPipelineaus der Liste der Pipelines aus.
2. Klicken Sie auf der Detailseite für die Pipeline auf die Schaltfläche Disable transition (Übergang deaktivieren) zwischen der zweiten Phase, Deploy (Bereitstellen)), und der dritten Phase, Production (Produktion), die Sie im vorigen Abschnitt hinzugefügt haben.
3. Geben Sie unter Disable transition (Übergang deaktivieren) einen Grund für das Deaktivieren des Übergangs zwischen den Stufen ein und klicken Sie dann auf Disable (Deaktivieren).

Der Pfeil zwischen den Phasen zeigt ein Symbol an, die Farbe ändert sich und die Schaltfläche Enable transition (Übergang aktivieren) wird angezeigt.



4. Laden Sie Ihr Beispiel erneut in den S3-Bucket hoch. Da der Bucket versioniert ist, startet diese Änderung die Pipeline.
5. Kehren Sie zur Detailseite für Ihre Pipeline zurück und prüfen Sie den Status der Phasen. Die Pipelineanzeige ändert sich und zeigt den Fortschritt und den Erfolg für die ersten zwei Phasen an. Für die dritte Phase werden keine Änderungen durchgeführt. Dieser Vorgang kann einige Minuten dauern.
6. Aktivieren Sie den Übergang, indem Sie die Schaltfläche Enable transition (Übergang aktivieren) zwischen den beiden Phasen auswählen. Wählen Sie im Dialogfeld Enable transition die Option Enable aus. Die Phase wird in ein paar Minuten ausgeführt und versucht, das bereits für die ersten beiden Phasen der Pipeline ausgeführte Artefakt zu verarbeiten.

> **Note**
>
> Wenn Sie möchten, dass diese dritte Phase erfolgreich ist, bearbeiten Sie die CodePipelineProductionFleet Bereitstellungsgruppe, bevor Sie den Übergang aktivieren, und geben Sie einen anderen Satz von EC2-Instances an, auf denen die Anwendung bereitgestellt wird. Weitere Informationen dazu finden Sie unter Ändern der Einstellungen von Bereitstellungsgruppen. Wenn Sie mehrere EC2-Instances erstellen, können für Sie zusätzliche Kosten entstehen.

Schritt 7: Bereinigen von Ressourcen

Sie können einige der Ressourcen, die Sie in diesem Tutorial erstellt haben, für Tutorial: Erstellen einer vierstufigen Pipeline verwenden. Sie können beispielsweise die CodeDeploy Anwendung und die Bereitstellung wiederverwenden. Sie können eine Build-Aktion mit einem Anbieter wie CodeBuild, einem vollständig verwalteten Build-Service in der Cloud, konfigurieren. Sie können eine Build-Aktion

auch so konfigurieren, dass ein Anbieter mit einem Build-Server oder -System verwendet wird, z. B. Jenkins.

Nach Abschluss dieses und jedes anderen Tutorials sollten Sie jedoch die Pipeline und die von ihr verwendeten Ressourcen löschen, damit Ihnen keine Kosten für die weitere Nutzung der Ressourcen entstehen. Löschen Sie zuerst die Pipeline, dann die CodeDeploy Anwendung und die zugehörigen Amazon EC2 EC2-Instances und schließlich den S3-Bucket.

So bereinigen Sie die in diesem Tutorial verwendeten Ressourcen

1. Um Ihre CodePipeline Ressourcen zu bereinigen, folgen Sie den Anweisungen unter Eine Pipeline löschen in AWS CodePipeline.
2. Um Ihre CodeDeploy Ressourcen zu bereinigen, folgen Sie den Anweisungen unter So bereinigen Sie Ressourcen (Konsole).
3. Folgen Sie den Anweisungen unter Löschen oder Leeren von Buckets, um den S3-Bucket zu löschen. Wenn Sie nicht noch weitere Pipelines erstellen möchten, löschen Sie den S3-Bucket, den Sie für die Speicherung Ihrer Pipeline-Artefakte erstellt haben. Unter CodePipeline Konzepte finden Sie weitere Informationen zu diesem Bucket.

Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository)

In diesem Tutorial verwenden Sie, CodePipeline um Code, der in einem CodeCommit Repository verwaltet wird, auf einer einzelnen Amazon EC2 EC2-Instance bereitzustellen. Ihre Pipeline wird ausgelöst, wenn Sie eine Änderung an das CodeCommit Repository übertragen. Die Pipeline stellt Ihre Änderungen an einer Amazon EC2 EC2-Instance bereit, die CodeDeploy als Bereitstellungsservice verwendet wird.

Die Pipeline hat zwei Phasen:

- Eine Quellstufe (Source) für Ihre CodeCommit Quellaktion.
- Eine Bereitstellungsphase (Deploy) für Ihre CodeDeploy Bereitstellungsaktion.

Der einfachste Weg, damit zu beginnen, AWS CodePipeline ist die Verwendung des Assistenten zum Erstellen einer Pipeline in der CodePipeline Konsole.

> **Note**
>
> Bevor du anfängst, stelle sicher, dass du deinen Git-Client so eingerichtet hast, dass er damit arbeiten kann CodeCommit. Anweisungen dazu findest du unter Einrichtung für CodeCommit.

Schritt 1: Erstellen Sie ein CodeCommit Repository

Zunächst erstellen Sie ein Repository in CodeCommit. Ihre Pipeline erhält Quellcode aus diesem Repository, wenn sie ausgeführt wird. Sie erstellen auch ein lokales Repository, in dem Sie Code verwalten und aktualisieren, bevor Sie ihn in das CodeCommit Repository übertragen.

Um ein CodeCommit Repository zu erstellen

1. Öffnen Sie die CodeCommit Konsole unter https://console.aws.amazon.com/codecommit/.
2. Wählen Sie in der Regionsauswahl den Ort aus, AWS-Region an dem Sie das Repository und die Pipeline erstellen möchten. Weitere Informationen finden Sie unter AWS-Regionen und Endpunkte.
3. Wählen Sie auf der Seite Repositorys die Option Repository auswählen.
4. Geben Sie auf der Seite Create repository (Repository erstellen) unter Repository name (Repository-Name) einen Namen für Ihr Repository ein (z. B. **`MyDemoRepo`**).
5. Wählen Sie Erstellen.

> **Note**
>
> Die verbleibenden Schritte in diesem Tutorial werden **`MyDemoRepo`** für den Namen Ihres CodeCommit Repositorys verwendet. Wenn Sie einen anderen Namen auswählen, müssen Sie diesen im gesamten Tutorial verwenden.

So richten Sie ein lokales Repository ein

In diesem Schritt richten Sie ein lokales Repository ein, um eine Verbindung zu Ihrem CodeCommit Remote-Repository herzustellen.

> **Note**
>
> Sie müssen kein lokales Repository einrichten. Sie können die Konsole auch verwenden, um Dateien hochzuladen, wie unter beschriebenSchritt 2: Fügen Sie Ihrem CodeCommit Repository Beispielcode hinzu.

1. Wenn Ihr neues Repository in der Konsole geöffnet ist, wählen Sie Clone URL (URL klonen) oben rechts auf der Seite und dann Clone SSH (SSH klonen) aus. Die Adresse zum Klonen Ihres Git-Repositorys wird in die Zwischenablage kopiert.
2. Navigieren Sie in Ihrem Terminal oder in der Befehlszeile zu einem lokalen Verzeichnis, in dem Ihr lokales Repository gespeichert werden soll. In diesem Tutorial verwenden wir `/tmp`.
3. Führen Sie den folgenden Befehl aus, um das Repository zu klonen. Ersetzen Sie dabei die SSH-Adresse durch die Adresse, die Sie im vorherigen Schritt kopiert haben. Mit diesem Befehl wird ein Verzeichnis namens `MyDemoRepo` erstellt. Kopieren Sie eine Beispielanwendung in dieses Verzeichnis.

```
git clone ssh://git-codecommit.us-west-2.amazonaws.com/v1/repos/MyDemoRepo
```

Schritt 2: Fügen Sie Ihrem CodeCommit Repository Beispielcode hinzu

In diesem Schritt laden Sie Code für eine Beispielanwendung herunter, die für eine CodeDeploy exemplarische Vorgehensweise erstellt wurde, und fügen ihn Ihrem CodeCommit Repository hinzu.

1. Laden Sie als Nächstes ein Beispiel herunter und speichern Sie es in einem Ordner oder Verzeichnis auf Ihrem lokalen Computer.

   a. Wählen Sie eine der folgenden Optionen aus. Wählen Sie aus, `SampleApp_Linux.zip` ob Sie die Schritte in diesem Tutorial für Linux-Instances befolgen möchten.

   - Wenn Sie die Bereitstellung auf Amazon Linux-Instances mithilfe durchführen möchten CodeDeploy, laden Sie die Beispielanwendung hier herunter: SampleApp_Linux.zip.
   - Wenn Sie die Bereitstellung auf Windows Server-Instances mithilfe durchführen möchten CodeDeploy, laden Sie die Beispielanwendung hier herunter: SampleApp_Windows.zip.

   Die Beispielanwendung enthält die folgenden Dateien für die Bereitstellung mit CodeDeploy:

- `appspec.yml`— Die Anwendungsspezifikationsdatei (AppSpecDatei) ist eine Datei im YAML-Format, die CodeDeploy zur Verwaltung einer Bereitstellung verwendet wird. Weitere Informationen zu der AppSpec Datei finden Sie unter CodeDeploy AppSpec Dateireferenz im AWS CodeDeploy Benutzerhandbuch.
- `index.html`— Die Indexdatei enthält die Homepage für die bereitgestellte Beispielanwendung.
- `LICENSE.txt`— Die Lizenzdatei enthält Lizenzinformationen für die Beispielanwendung.
- Dateien für Skripts — Die Beispielanwendung verwendet Skripts, um Textdateien an einen Speicherort auf Ihrer Instance zu schreiben. Für jedes der verschiedenen Ereignisse im CodeDeploy Bereitstellungszyklus wird wie folgt eine Datei geschrieben:
  - `scripts`Ordner (nur Linux-Beispiel) — Der Ordner enthält die folgenden Shell-Skripts zum Installieren von Abhängigkeiten und zum Starten und Beenden der Beispielanwendung für die automatisierte Bereitstellung: `install_dependencies`, `start_server` und `stop_server`.
  - (Nur Windows-Beispiel) `before-install.bat` — Dies ist ein Batch-Skript für das `BeforeInstall` Deployment-Lifecycle-Ereignis, das ausgeführt wird, um alte Dateien zu entfernen, die während früherer Bereitstellungen dieses Beispiels geschrieben wurden, und einen Speicherort auf Ihrer Instanz zu erstellen, in den die neuen Dateien geschrieben werden.

b. Laden Sie die komprimierte ZIP-Datei herunter.

2. Entpacken Sie die Dateien aus SampleApp_Linux.zip in das lokale Verzeichnis, das Sie zuvor erstellt haben (z. B. `/tmp/MyDemoRepo` oder `c:\temp\MyDemoRepo`).

   Stellen Sie sicher, dass Sie die Dateien direkt in Ihrem lokalen Repository platzieren. Beziehen Sie keinen `SampleApp_Linux` Ordner ein. Auf Ihrem lokalen Linux-, macOS- oder Unix-Computer sollte Ihre Verzeichnis- und Dateihierarchie beispielsweise so aussehen:

```
/tmp
   #-- MyDemoRepo
       #-- appspec.yml
       #-- index.html
       #-- LICENSE.txt
       #-- scripts
           #-- install_dependencies
           #-- start_server
           #-- stop_server
```

3. Verwenden Sie eine der folgenden Methoden, um Dateien in Ihr Repository hochzuladen.

   a. So verwenden Sie die CodeCommit Konsole zum Hochladen Ihrer Dateien:

      i. Öffnen Sie die CodeCommit Konsole und wählen Sie Ihr Repository aus der Repository-Liste aus.

      ii. Wählen Sie Add file (Datei hinzufügen) und dann Upload file (Datei hochladen) aus.

      iii. Wählen Sie Choose file (Datei auswählen) und navigieren Sie dann zu Ihrer Datei. Um eine Datei unter einem Ordner hinzuzufügen, wählen Sie Datei erstellen und geben Sie dann den Ordnernamen mit dem Dateinamen ein, z. B. `scripts/install_dependencies` Fügen Sie den Dateiinhalt in die neue Datei ein.

      Übernehmen Sie die Änderung, indem Sie Ihren Benutzernamen und Ihre E-Mail-Adresse eingeben.

      Wählen Sie Commit changes (Änderungen übernehmen) aus.

      iv. Wiederholen Sie diesen Schritt für jede Datei.

      Der Inhalt Ihres Repositorys sollte wie folgt aussehen:

```
#-- appspec.yml
#-- index.html
#-- LICENSE.txt
#-- scripts
    #-- install_dependencies
    #-- start_server
    #-- stop_server
```

   b. So verwendest du Git-Befehle, um deine Dateien hochzuladen:

      i. Ändern von Verzeichnisses in Ihrem lokalen Repository:

```
(For Linux, macOS, or Unix) cd /tmp/MyDemoRepo
(For Windows) cd c:\temp\MyDemoRepo
```

      ii. Führen Sie den folgenden Befehl aus, um alle Dateien auf einmal zu übertragen:

```
git add -A
```

iii. Führen Sie den folgenden Befehl aus, um einen Commit für Dateien mit einer Commit-Nachricht durchzuführen:

```
git commit -m "Add sample application files"
```

iv. Führen Sie den folgenden Befehl aus, um die Dateien von Ihrem lokalen Repository in Ihr CodeCommit Repository zu übertragen:

```
git push
```

4. Die Dateien, die Sie heruntergeladen und zu Ihrem lokalen Repo hinzugefügt haben, wurden nun dem `main` Zweig in Ihrem CodeCommit `MyDemoRepo` Repository hinzugefügt und können nun in eine Pipeline aufgenommen werden.

Schritt 3: Erstellen Sie eine Amazon EC2 EC2-Linux-Instance und installieren Sie den Agenten CodeDeploy

In diesem Schritt erstellen Sie die Amazon EC2 EC2-Instance, in der Sie eine Beispielanwendung bereitstellen. Erstellen Sie im Rahmen dieses Prozesses eine Instance-Rolle, die die Installation und Verwaltung des CodeDeploy Agenten auf der Instance ermöglicht. Der CodeDeploy Agent ist ein Softwarepaket, das die Verwendung einer Instanz in CodeDeploy Bereitstellungen ermöglicht. Sie fügen auch Richtlinien hinzu, die es der Instanz ermöglichen, Dateien abzurufen, die der CodeDeploy Agent zur Bereitstellung Ihrer Anwendung verwendet, und die Verwaltung der Instanz durch SSM zu ermöglichen.

So erstellen Sie eine Instance-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/).
2. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen AWS-Servicedie Option aus. Wählen Sie unter Anwendungsfall auswählen die Option EC2 aus. Wählen Sie in Select your use case (Anwendungsfall auswählen) die Option EC2 aus. Wählen Sie Weiter: Berechtigungen aus.
5. Suchen Sie nach der genannten **`AmazonEC2RoleforAWSCodeDeploy`**Richtlinie und wählen Sie sie aus.

6. Suchen Sie nach der genannten Richtlinie und wählen Sie sie aus **`AmazonSSMManagedInstanceCore`**. Wählen Sie Next: Tags (Weiter: Tags) aus.
7. Klicken Sie auf Weiter: Prüfen. Geben Sie einen Namen für die Rolle ein (z. B. **`EC2InstanceRole`**).

   > **Note**
   >
   > Notieren Sie sich Ihren Rollennamen für den nächsten Schritt. Sie wählen diese Rolle, wenn Sie Ihre Instance erstellen.

   Wählen Sie Rolle erstellen aus.

So starten Sie eine Instance

1. Öffnen Sie die Amazon EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie in der Seitennavigation Instances und dann oben auf der Seite Launch Instances aus.
3. Geben Sie unter Name **`MyCodePipelineDemo`** ein. Dadurch wird der Instance ein Tag-Schlüssel von **`Name`** und ein Tag-Wert von **`MyCodePipelineDemo`** zugewiesen. Später erstellen Sie eine CodeDeploy Anwendung, die die Beispielanwendung auf dieser Instanz bereitstellt. CodeDeploywählt die bereitzustellenden Instanzen auf der Grundlage der Tags aus.
4. Suchen Sie unter Anwendungs- und Betriebssystem-Images (Amazon Machine Image) die Amazon Linux AMI-Option mit dem AWS Logo und stellen Sie sicher, dass sie ausgewählt ist. (Dieses AMI wird als Amazon Linux 2 AMI (HVM) bezeichnet und trägt die Bezeichnung „Für das kostenlose Kontingent in Frage“.)
5. Wählen Sie unter Instance-Typ den `t2.micro` Typ aus, für den das kostenlose Kontingent in Frage kommt, als Hardwarekonfiguration für Ihre Instance.
6. Wählen Sie unter key pair (Anmeldung) ein Schlüsselpaar aus oder erstellen Sie eines.

   Sie können auch Proceed without a key pair wählen.

> **Note**
>
> Für die Zwecke dieses Tutorials können Sie ohne Schlüsselpaar fortfahren. Um SSH zu verwenden, wenn eine Verbindung zu Ihren Instances herstellen, erstellen Sie ein Schlüsselpaar oder verwenden Sie ein vorhandenes.

7. Gehen Sie unter Netzwerkeinstellungen wie folgt vor.

   Vergewissern Sie sich, dass unter Öffentliche IP automatisch zuweisen der Status Enable lautet.

   - Klicken Sie neben Assign a security group (Sicherheitsgruppe zuweisen) auf Create a new security group (Neue Sicherheitsgruppe zuweisen).
   - Wählen Sie in der Zeile für SSH unter Quelltyp die Option Meine IP aus.
   - Wählen Sie Sicherheitsgruppe hinzufügen, wählen Sie HTTP und dann unter Quelltyp die Option Meine IP aus.
8. Erweitern Sie Advanced Details (Erweiterte Details). Wählen Sie im IAM-Instanzprofil die IAM-Rolle aus, die Sie im vorherigen Verfahren erstellt haben (z. B.**`EC2InstanceRole`**).
9. Geben Sie unter Zusammenfassung unter Anzahl der Instanzen den Wert.. 1
10. Wählen Sie Launch Instance (Instance starten) aus.
11. Sie können den Status des Starts auf der Seite Instances anzeigen. Wenn Sie eine Instance starten, lautet ihr anfänglicher Status `pending`. Nachdem die Instance gestartet ist, lautet ihr Status `running`. Sie erhält dann einen öffentlichen DNS-Namen. (Wenn die Spalte Public DNS nicht angezeigt wird, wählen Sie das Show/Hide-Symbol aus und wählen Sie dann Public DNS aus.)

Schritt 4: Erstellen Sie eine Anwendung in CodeDeploy

In CodeDeploy ist eine Anwendung eine Ressource, die die Softwareanwendung enthält, die Sie bereitstellen möchten. Später verwenden Sie diese Anwendung, CodePipeline um die Bereitstellung der Beispielanwendung auf Ihrer Amazon EC2 EC2-Instance zu automatisieren.

Zunächst erstellen Sie eine Rolle, die die Durchführung von CodeDeploy Bereitstellungen ermöglicht. Anschließend erstellen Sie eine CodeDeploy -Anwendung.

Um eine CodeDeploy Servicerolle zu erstellen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/).
2. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Vertrauenswürdige Entität auswählen die Option AWS-Service. Wählen Sie unter Use case (Anwendungsfall) CodeDeploy aus. Wählen Sie CodeDeployaus den aufgelisteten Optionen. Wählen Sie Weiter aus. Die `AWSCodeDeployRole`-verwaltete Richtlinie ist der Rolle bereits zugewiesen.
5. Wählen Sie Weiter aus.
6. Geben Sie einen Namen für die Rolle ein (beispielsweise **`CodeDeployRole`**) und wählen Sie Next (Weiter) aus.

Um eine Anwendung zu erstellen in CodeDeploy

1. Öffnen Sie die CodeDeploy Konsole unter https://console.aws.amazon.com/codedeploy.
2. Wenn die Seite „Anwendungen“ nicht angezeigt wird, wählen Sie im Menü „Anwendungen“.
3. Wählen Sie Create application aus.
4. Geben Sie unter Application name (Anwendungsname) **`MyDemoApplication`** ein.
5. Wählen Sie unter Compute Platform (Plattform für die Datenverarbeitung) die Option EC2/On-premises (EC2/Lokal) aus.
6. Wählen Sie Create application aus.

Um eine Bereitstellungsgruppe zu erstellen in CodeDeploy

Eine Bereitstellungsgruppe ist eine Ressource, die Bereitstellungseinstellungen definiert, z. B. für welche Instances und wie schnell sie bereitgestellt werden sollen.

1. Wählen Sie auf der Seite mit Ihrer Anwendung Create deployment group (Bereitstellungsgruppe erstellen).
2. Geben Sie unter Deployment group name (Name der Bereitstellungsgruppe) **`MyDemoDeploymentGroup`** ein.
3. Wählen Sie unter Servicerolle den ARN der Servicerolle aus, die Sie zuvor erstellt haben (z. B.**`arn:aws:iam::`*`account_ID`*`:role/CodeDeployRole`**).
4. Wählen Sie unter Deployment type (Bereitstellungstyp) die Option In-place (Direkt).

5. Wählen Sie unter Environment configuration (Umgebungskonfiguration) die Option Amazon EC2-Instances aus. Geben Sie im Feld Schlüssel den Wert ein**`Name`**. Geben Sie im Feld Wert den Namen ein, mit dem Sie die Instanz markiert haben (z. B.**`MyCodePipelineDemo`**).
6. Wählen Sie unter Agentenkonfiguration mit AWS Systems Manager die Option Jetzt und planen Sie Updates. Dadurch wird der Agent auf der Instanz installiert. Die Linux-Instanz ist bereits mit dem SSM-Agenten konfiguriert und wird nun mit dem CodeDeploy Agenten aktualisiert.
7. Wählen Sie in Deployment configuration (Bereitstellungskonfiguration) die Option `CodeDeployDefault.OneAtaTime` aus.
8. Stellen Sie sicher, dass unter Load Balancer die Option Load Balancing aktivieren nicht ausgewählt ist. Sie müssen keinen Load Balancer einrichten oder eine Zielgruppe für dieses Beispiel auswählen.
9. Wählen Si Create deployment group (Bereitstellungsgruppe erstellen).

Schritt 5: Erstellen Sie Ihre erste Pipeline in CodePipeline

Sie sind jetzt bereit, um Ihre erste Pipeline zu erstellen und auszuführen. In diesem Schritt erstellen Sie eine Pipeline, die automatisch ausgeführt wird, wenn Code in Ihr CodeCommit Repository übertragen wird.

Um eine CodePipeline Pipeline zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie Create pipeline (Pipeline erstellen) aus.
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyFirstPipeline`** ein.
4. Wählen Sie unter Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.

7. Wählen Sie in Schritt 2: Quellstufe hinzufügen im Feld Quellanbieter die Option CodeCommit. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, in dem Sie es erstellt habenSchritt 1: Erstellen Sie ein CodeCommit Repository. Wählen Sie unter Branch name (Name der Verzweigung) die Option `main` und dann Next step (Nächster Schritt) aus.

   Nachdem Sie den Repository-Namen und den Branch ausgewählt haben, wird in einer Meldung die Amazon CloudWatch Events-Regel angezeigt, die für diese Pipeline erstellt werden soll.

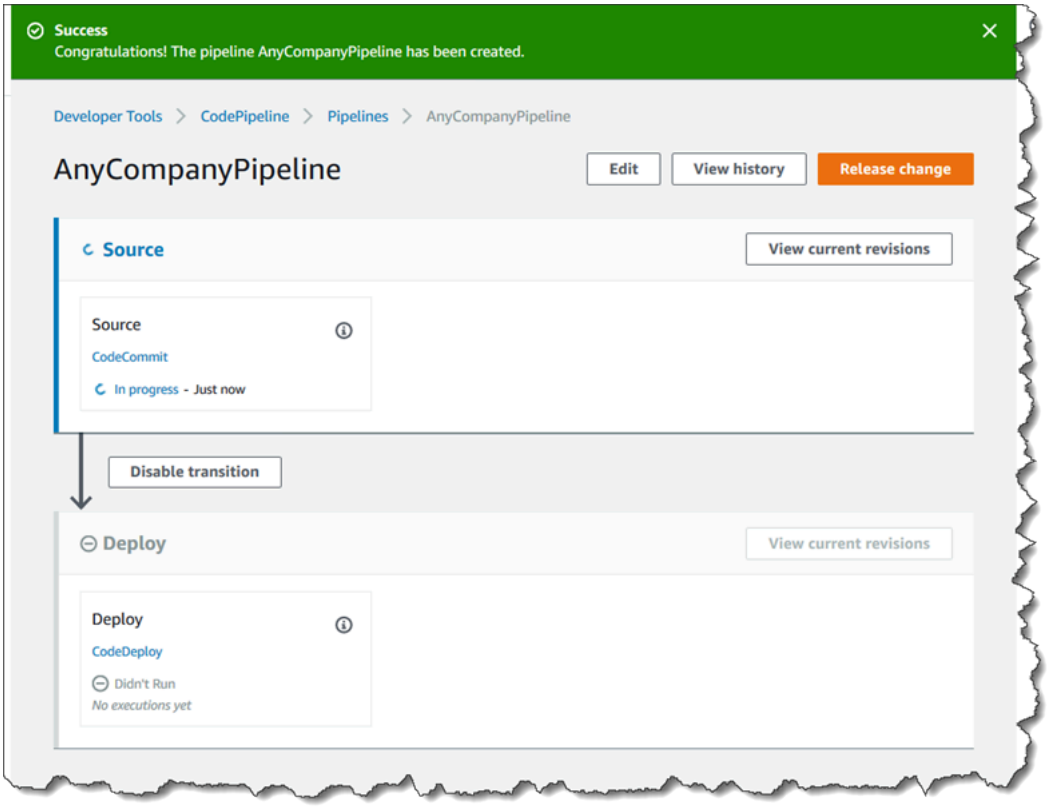
   Lassen Sie die Standardwerte in Change detection options unverändert. Auf diese Weise können CodePipeline Sie Amazon CloudWatch Events verwenden, um Änderungen in Ihrem Quell-Repository zu erkennen.

   Wählen Sie Weiter aus.

8. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken. Wählen Sie Weiter aus.

   > **Note**
   >
   > In diesem Tutorial stellen Sie Code bereit, der keinen Build-Service erfordert, sodass Sie diesen Schritt überspringen können. Wenn Ihr Quellcode jedoch erstellt werden muss, bevor er für Instances bereitgestellt wird, können Sie ihn CodeBuildin diesem Schritt konfigurieren.

9. Wählen Sie in Schritt 4: Bereitstellungsphase hinzufügen unter Bereitstellungsanbieter die Option aus CodeDeploy. Wählen Sie unter Application name (Anwendungsname) die Option **`MyDemoApplication`** aus. Wählen Sie unter Deployment group (Bereitstellungsgruppe) die Option **`MyDemoDeploymentGroup`** , und dann Next step (Nächster Schritt) aus.
10. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.
11. Die Pipeline wird ausgeführt, nachdem sie erstellt wurde. Es lädt den Code aus Ihrem CodeCommit Repository herunter und erstellt eine CodeDeploy Bereitstellung für Ihre EC2-Instance. Sie können sich Fortschritts-, Erfolgs- und Fehlschlagsmeldungen anzeigen lassen, während das CodePipeline Beispiel die Webseite für die Amazon EC2 EC2-Instance in der CodeDeploy Bereitstellung bereitstellt.



Herzlichen Glückwunsch! Sie haben gerade eine einfache Pipeline in erstellt CodePipeline.

Im nächsten Schritt überprüfen Sie die Ergebnisse.

So verifizieren Sie, ob Ihre Pipeline erfolgreich ausgeführt wurde

1. Zeigen Sie den anfänglichen Fortschritt der Pipeline an. Der Status jeder Phase ändert sich von Zurzeit keine Ausführungen zu Laufend. Danach ändert er sich entweder zu Erfolgreich oder Fehlgeschlagen. Die Pipeline sollte die erste Ausführung innerhalb weniger Minuten abschließen.
2. Nachdem Succeeded für den Pipeline-Status angezeigt wird, wählen Sie im Statusbereich für die Bereitstellungsphase die Option CodeDeploy. Dadurch wird die CodeDeploy Konsole geöffnet. Wenn Succeeded (Erfolgreich) nicht angezeigt wird, finden Sie weitere Informationen unter Problembehebung CodePipeline.
3. Wählen Sie in der Registerkarte Deployments (Bereitstellungen) die Bereitstellungs-ID aus. Wählen Sie auf der Seite für die Bereitstellung unter Deployment lifecycle events (Bereitstellungslebenszyklusereignisse) die Instance-ID aus. Daraufhin wird die EC2-Konsole geöffnet.

4. Kopieren Sie in der Registerkarte Description (Beschreibung) in Public DNS (Öffentliches DNS) die Adresse (z. B. `ec2-192-0-2-1.us-west-2.compute.amazonaws.com`) und fügen Sie diese in die Adressleiste Ihres Webbrowsers ein.

   Die Webseite für die Beispielanwendung, die Sie heruntergeladen und in Ihr CodeCommit Repository übertragen haben, wird angezeigt.

Weitere Informationen über Phasen, Aktionen und die Funktionsweise von Pipelines finden Sie unter CodePipeline Konzepte .

Schritt 6: Ändern Sie den Code in Ihrem CodeCommit Repository

Ihre Pipeline ist so konfiguriert, dass sie immer dann ausgeführt wird, wenn Codeänderungen an Ihrem CodeCommit Repository vorgenommen werden. In diesem Schritt nehmen Sie Änderungen an der HTML-Datei vor, die Teil der CodeDeploy Beispielanwendung im CodeCommit Repository ist. Wenn Sie diese Änderungen übertragen, wird Ihre Pipeline erneut ausgeführt. Die Änderungen, die Sie vornehmen, sind an der Webadresse sichtbar, auf die Sie zuvor zugegriffen haben.

1. Ändern von Verzeichnisses in Ihrem lokalen Repository:

```
(For Linux, macOS, or Unix) cd /tmp/MyDemoRepo
(For Windows) cd c:\temp\MyDemoRepo
```

2. Verwenden Sie einen Texteditor zum Ändern der `index.html`-Datei:

```
(For Linux or Unix)gedit index.html
(For OS X)open -e index.html
(For Windows)notepad index.html
```

3. Ändern Sie den Inhalt der `index.html`-Datei auf die gewünschte Hintergrundfarbe und ändern Sie einigen Text auf der Webseite. Speichern Sie dann die Datei.

```
<!DOCTYPE html>
<html>
<head>
  <title>Updated Sample Deployment</title>
  <style>
    body {
      color: #000000;
      background-color: #CCFFCC;
```

```
      font-family: Arial, sans-serif;
      font-size:14px;
    }

    h1 {
      font-size: 250%;
      font-weight: normal;
      margin-bottom: 0;
    }

    h2 {
      font-size: 175%;
      font-weight: normal;
      margin-bottom: 0;
    }
  </style>
</head>
<body>
  <div align="center"><h1>Updated Sample Deployment</h1></div>
  <div align="center"><h2>This application was updated using CodePipeline,
 CodeCommit, and CodeDeploy.</h2></div>
  <div align="center">
    <p>Learn more:</p>
    <p><a href="https://docs.aws.amazon.com/codepipeline/latest/
userguide/">CodePipeline User Guide</a></p>
    <p><a href="https://docs.aws.amazon.com/codecommit/latest/
userguide/">CodeCommit User Guide</a></p>
    <p><a href="https://docs.aws.amazon.com/codedeploy/latest/
userguide/">CodeDeploy User Guide</a></p>
  </div>
</body>
</html>
```

4. Übernehmen Sie Ihre Änderungen und übertragen Sie sie in Ihr CodeCommit Repository, indem Sie nacheinander die folgenden Befehle ausführen:

```
git commit -am "Updated sample application files"
```

```
git push
```

So verifizieren Sie, ob Ihre Pipeline erfolgreich ausgeführt wurde

1. Zeigen Sie den anfänglichen Fortschritt der Pipeline an. Der Status jeder Phase ändert sich von Zurzeit keine Ausführungen zu Laufend. Danach ändert er sich entweder zu Erfolgreich oder Fehlgeschlagen. Die Ausführung der Pipeline sollte innerhalb weniger Minuten abgeschlossen sein.
2. Nachdem Succeeded (Erfolgreich) für den Aktionsstatus angezeigt wird, aktualisieren Sie die Demo-Seite, auf die Sie zuvor in Ihrem Browser zugegriffen haben.

   Die aktualisierte Webseite wird angezeigt.

Schritt 7: Bereinigen von Ressourcen

Sie können einige der Ressourcen, die Sie in diesem Tutorial erstellt haben, für andere Tutorials in diesem Handbuch verwenden. Sie können beispielsweise die CodeDeploy Anwendung und die Bereitstellung wiederverwenden. Nach Abschluss dieses und jedes anderen Tutorials sollten Sie jedoch die Pipeline und die von ihr verwendeten Ressourcen löschen, damit Ihnen keine Kosten für die weitere Nutzung der Ressourcen entstehen. Löschen Sie zuerst die Pipeline, dann die CodeDeploy Anwendung und die zugehörige Amazon EC2 EC2-Instance und schließlich das CodeCommit Repository.

So bereinigen Sie die in diesem Tutorial verwendeten Ressourcen

1. Um Ihre CodePipeline Ressourcen zu bereinigen, folgen Sie den Anweisungen unter Eine Pipeline löschen in AWS CodePipeline.
2. Um Ihre CodeDeploy Ressourcen zu bereinigen, folgen Sie den Anweisungen unter Ressourcen zum Aufräumen der Bereitstellung.
3. Folgen Sie den Anweisungen unter Löschen eines CodeCommit Repositorys, um das CodeCommitRepository zu löschen.

Schritt 8: Weitere Informationen

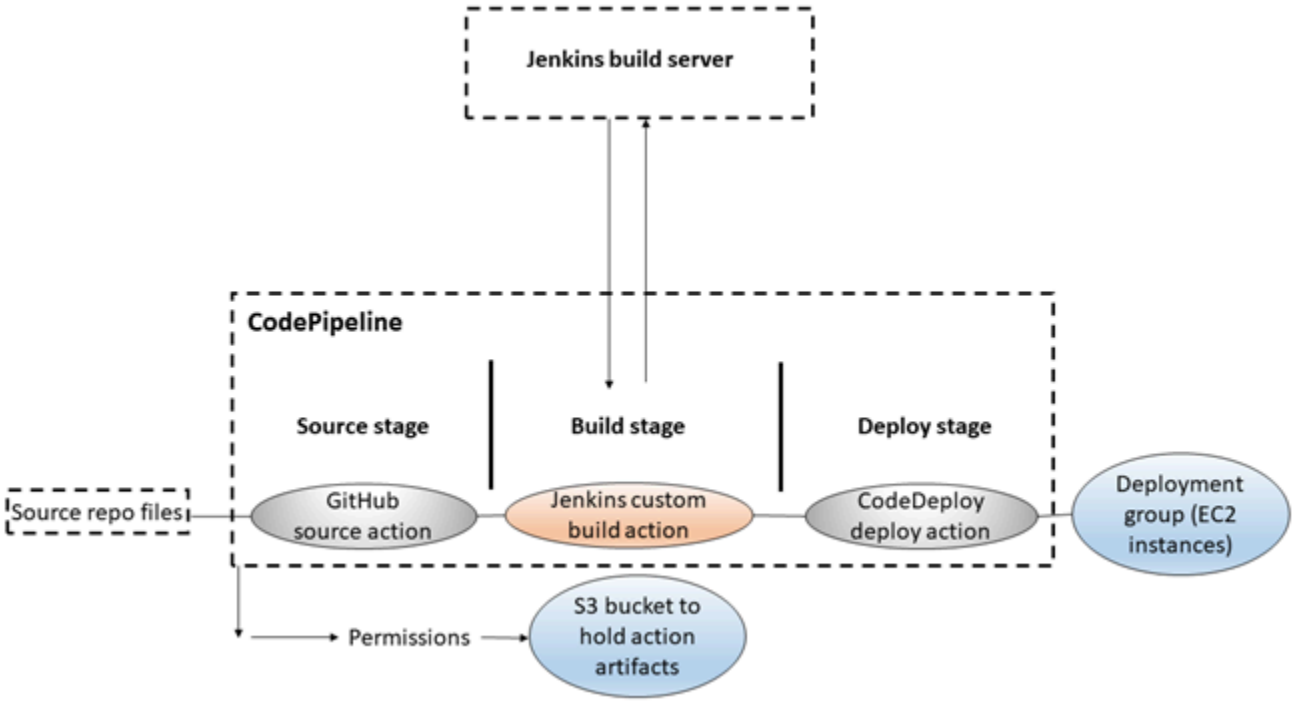
Erfahre mehr darüber, wie das CodePipeline funktioniert:

- Weitere Informationen über Phasen, Aktionen und die Funktionsweise von Pipelines finden Sie unter CodePipeline Konzepte .

- Informationen zu den Aktionen, die Sie mit ausführen können CodePipeline, finden Sie unterIntegrationen mit CodePipeline Aktionstypen.
- Testen Sie dieses erweiterte Tutorial: Tutorial: Erstellen einer vierstufigen Pipeline. Es wird eine mehrstufige Pipeline erstellt, die einen Schritt enthält, mit dem Code vor der Bereitstellung erstellt wird.

Tutorial: Erstellen einer vierstufigen Pipeline

Nachdem Sie Ihre erste Pipeline in Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) oder Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository) erstellt haben, können Sie mit dem Erstellen komplexerer Pipelines anfangen. Dieses Tutorial führt Sie durch die Erstellung einer vierstufigen Pipeline, die ein GitHub Repository für Ihre Quelle, einen Jenkins-Build-Server zum Erstellen des Projekts und eine CodeDeploy Anwendung zur Bereitstellung des erstellten Codes auf einem Staging-Server verwendet. Das folgende Diagramm zeigt die erste dreistufige Pipeline.



Nach Erstellung der Pipeline bearbeiten Sie sie, um eine Stufe mit einer Testaktion zum Testen des Codes hinzuzufügen. Hierfür verwenden Sie ebenfalls Jenkins.

Vor dem Erstellen dieser Pipeline müssen Sie die erforderlichen Ressourcen konfigurieren. Wenn Sie beispielsweise ein GitHub Repository für Ihren Quellcode verwenden möchten, müssen Sie das Repository erstellen, bevor Sie es einer Pipeline hinzufügen können. Im Zuge der

Konfiguration führt Sie dieses Tutorial durch die Einrichtung von Jenkins bei einer EC2-Instance für Demonstrationszwecke.

> **Important**
>
> Viele der Aktionen, die Sie Ihrer Pipeline in diesem Verfahren hinzufügen, beinhalten AWS Ressourcen, die Sie erstellen müssen, bevor Sie die Pipeline erstellen. AWS Ressourcen für Ihre Quellaktionen müssen immer in derselben AWS Region erstellt werden, in der Sie Ihre Pipeline erstellen. Wenn Sie Ihre Pipeline beispielsweise in der Region USA Ost (Ohio) erstellen, muss sich Ihr CodeCommit Repository in der Region USA Ost (Ohio) befinden. Sie können beim Erstellen Ihrer Pipeline regionsübergreifende Aktionen hinzufügen. AWS Ressourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

Bevor Sie mit diesem Tutorial beginnen, sollten bereits die allgemeinen Voraussetzungen in Erste Schritte mit CodePipeline erfüllt haben.

Themen

Schritt 1: Erfüllen der Voraussetzungen

Für die Integration mit Jenkins AWS CodePipeline müssen Sie das CodePipeline Plugin für Jenkins auf einer beliebigen Jenkins-Instanz installieren, mit der Sie es verwenden möchten. CodePipeline Sie sollten auch einen dedizierten IAM-Benutzer oder eine spezielle IAM-Rolle konfigurieren, die für Berechtigungen zwischen Ihrem Jenkins-Projekt und verwendet werden soll. CodePipeline Der einfachste Weg, Jenkins zu integrieren, CodePipeline besteht darin, Jenkins auf einer EC2-Instance zu installieren, die eine IAM-Instanzrolle verwendet, die Sie für die Jenkins-Integration erstellen. Damit Links in der Pipeline für Jenkins-Aktionen erfolgreich eine Verbindung erstellen, müssen Sie Proxy- und Firewall-Einstellungen auf dem Server oder der EC2-Instance entsprechend konfigurieren, um eingehende Verbindungen beim Port zu erlauben, der von Ihrem Jenkins-Projekt

verwendet wird. Sie müssen Jenkins für die Authentifizierung von Benutzern und die Durchsetzung der Zugriffskontrolle konfiguriert haben, um Verbindungen auf diesen Ports (z. B. 443 und 8443, wenn Sie Jenkins entsprechend gesichert haben, ausschließlich HTTPS-Verbindungen zu verwenden, oder 80 und 8080, wenn Sie HTTP-Verbindungen zulassen) zu erlauben. Weitere Informationen finden Sie unter Sichern von Jenkins.

> **Note**
>
> In diesem Tutorial werden ein Code-Beispiel verwendet und Build-Schritte konfiguriert, die das Beispiel von Haml in HTML konvertieren. Sie können den Open-Source-Beispielcode aus dem Repository herunterladen, indem Sie die Schritte unter befolgen. GitHub Kopieren oder klonen Sie das Beispiel in ein Repository GitHub Sie benötigen das gesamte Beispiel in Ihrem GitHub Repository, nicht nur die ZIP-Datei.
> In diesem Tutorial wird außerdem von Folgendem ausgegangen:
>
> - Sie sind mit der Installation und Administration von Jenkins und der Erstellung von Jenkins-Projekten vertraut.
> - Sie haben Rake und das Haml gem für Ruby auf dem Computer oder der Instance installiert, auf dem/der Ihr Jenkins-Projekt gehostet wird.
> - Sie haben die erforderlichen Systemumgebungsvariablen festgelegt, damit Rake-Befehle vom Terminal oder der Befehlszeile ausgeführt werden können (z. B. auf Windows-Systemen durch Modifizierung der PATH-Variable, um das Verzeichnis einzuschließen, in dem Sie Rake installiert haben).

Themen

Kopieren oder klonen Sie das Beispiel in ein Repository GitHub

Um das Beispiel zu klonen und in ein GitHub Repository zu pushen

1. Laden Sie den Beispielcode aus dem GitHub Repository herunter oder klonen Sie die Repositorys auf Ihren lokalen Computer. Es gibt zwei Beispielpakete:

- Wenn Sie Ihr Beispiel auf Amazon Linux-, RHEL- oder Ubuntu-Server-Instances bereitstellen möchten, wählen Sie codepipeline-jenkins-aws-codedeploy_linux.zip.
- Wenn Sie Ihr Beispiel auf Windows Server-Instances bereitstellen möchten, wählen Sie CodePipeline-Jenkins - .zip. AWSCodeDeploy_Windows

2. Klicken Sie im Repository auf Fork, um das Beispiel-Repository in Ihrem Github-Konto zu klonen. Weitere Informationen finden Sie in der Dokumentation. GitHub

Erstellen Sie eine IAM-Rolle, die für die Jenkins-Integration verwendet werden soll

Als bewährte Methode sollten Sie erwägen, eine EC2-Instance zu starten, um Ihren Jenkins-Server zu hosten und der Instance mithilfe einer IAM-Rolle die erforderlichen Berechtigungen für die Interaktion zu gewähren. CodePipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Klicken Sie im Navigationsbereich der IAM-Konsole auf Rollen und wählen Sie dann Rolle erstellen aus.
3. Wählen Sie unter Select type of trusted entity (Typ der vertrauenswürdigen Entität auswählen) die Option AWS-Service Service aus. Wählen Sie in Choose the service that will use this role (Service auswählen, der diese Rolle verwenden wird) die Option EC2 aus. Wählen Sie in Select your use case (Anwendungsfall auswählen) die Option EC2 aus.
4. Wählen Sie Weiter: Berechtigungen aus. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien anfügen) die verwaltete `AWSCodePipelineCustomActionAccess`-Richtlinie und anschließend Next: Tags (Weiter: Tags) aus. Wählen Sie Weiter: Prüfen aus.
5. Geben Sie auf der Überprüfungsseite im Feld Rollenname den Namen der Rolle ein, die speziell für die Jenkins-Integration erstellt werden soll (z. B. ***`JenkinsAccess`***), und wählen Sie dann Rolle erstellen aus.

Wenn Sie die EC2-Instanz erstellen, auf der Sie Jenkins installieren möchten, stellen Sie sicher, dass Sie in Schritt 3: Instanzdetails konfigurieren die Instanzrolle auswählen (z. B.). *`JenkinsAccess`*

Weitere Informationen zu Instance-Rollen und Amazon EC2 finden Sie unter IAM-Rollen für Amazon EC2, Verwenden von IAM-Rollen zur Erteilung von Berechtigungen für Anwendungen, die auf Amazon EC2 EC2-Instances ausgeführt werden, und Erstellen einer Rolle zum Delegieren von Berechtigungen an eine. AWS-Service

Installieren und konfigurieren Sie Jenkins und das Plugin für Jenkins CodePipeline

Um Jenkins und das Plugin für Jenkins zu installieren CodePipeline

1. Erstellen Sie eine EC2-Instanz, in der Sie Jenkins installieren werden, und stellen Sie in Schritt 3: Instanzdetails konfigurieren sicher, dass Sie die Instanzrolle auswählen, die Sie erstellt haben (z. B.). *JenkinsAccess* Weitere Informationen zum Erstellen von EC2-Instances finden Sie unter Starten einer Amazon EC2 EC2-Instance im Amazon EC2 EC2-Benutzerhandbuch.

   > **Note**
   >
   > Wenn Sie bereits über Jenkins-Ressourcen verfügen, die Sie verwenden möchten, können Sie dies tun. Sie müssen jedoch einen speziellen IAM-Benutzer erstellen, die `AWSCodePipelineCustomActionAccess` verwaltete Richtlinie auf diesen Benutzer anwenden und dann die Zugangsdaten für diesen Benutzer auf Ihrer Jenkins-Ressource konfigurieren und verwenden. Wenn Sie die Jenkins-Benutzeroberfläche zur Bereitstellung der Anmeldeinformationen verwenden möchten, konfigurieren Sie Jenkins so, dass nur HTTPS zugelassen ist. Weitere Informationen finden Sie unter Problembehebung CodePipeline.

2. Installieren Sie Jenkins in der EC2-Instance. Weitere Informationen finden Sie in der Jenkins-Dokumentation unter Installing Jenkins und Starting and accessing Jenkins sowie unter details of integration with Jenkins in Produkt- und Serviceintegrationen mit CodePipeline.
3. Starten Sie Jenkins. Wählen Sie auf der Startseite Manage Jenkins aus.
4. Wählen Sie auf der Seite Manage Jenkins Manage Plugins aus.
5. Wählen Sie die Registerkarte Available (Verfügbar) aus. Geben Sie im Suchfeld Filter „**AWS CodePipeline**“ ein. Wählen Sie CodePipeline Plugin für Jenkins aus der Liste aus und klicken Sie auf Jetzt herunterladen und nach dem Neustart installieren.
6. Klicken Sie auf der Seite Installing Plugins/Upgrades auf Restart Jenkins when installation is complete and no jobs are running.
7. Wählen Sie Back to Dashboard aus.
8. Klicken Sie auf der Hauptseite auf New Item.
9. Geben Sie im Feld Elementname einen Namen für das Jenkins-Projekt ein (z. B.). *MyDemoProject* Wählen Sie Freestyle project und OK aus.

> **Note**
>
> Stellen Sie sicher, dass der Name für Ihr Projekt die Anforderungen für erfüllt. CodePipeline Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline.

10. Wählen Sie auf der Konfigurationsseite des Projekts das Kontrollkästchen Execute concurrent builds if necessary aus. Wählen Sie in Source Code Management (Quellcodeverwaltung) AWS CodePipeline. Wenn Sie Jenkins auf einer EC2-Instance installiert und die AWS CLI mit dem Profil für den IAM-Benutzer konfiguriert haben, den Sie für die Integration zwischen CodePipeline und Jenkins erstellt haben, lassen Sie alle anderen Felder leer.
11. Wählen Sie Erweitert und geben Sie unter Anbieter einen Namen für den Anbieter der Aktion ein, wie er angezeigt werden soll (z. B.). CodePipeline *`MyJenkinsProviderName`* Stellen Sie sicher, dass der Name eindeutig und leicht zu merken ist. Sie werden ihn verwenden, wenn Sie zu einem späteren Zeitpunkt in diesem Tutorial eine Build- und -Test-Aktion zu Ihrer Pipeline hinzufügen.

    > **Note**
    >
    > Dieser Aktionsname muss die Benennungsanforderungen für Aktionen in erfüllen CodePipeline. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline.

12. Deaktivieren Sie in Build Triggers alle Kontrollkästchen und wählen Sie Poll SCM aus. Geben Sie in Schedule (Zeitplan) fünf durch Leerzeichen getrennte Sternchen ein:

    ```
    * * * * *
    ```

    Dieser wird CodePipeline jede Minute abgefragt.
13. Wählen Sie in Build Add build step aus. Wählen Sie Shell ausführen (Amazon Linux, RHEL oder Ubuntu Server) Batch-Befehl ausführen (Windows Server) und geben Sie dann Folgendes ein:

    ```
    rake
    ```

> **Note**
>
> Stellen Sie sicher, dass Ihre Umgebung mit den Variablen und Einstellungen konfiguriert ist, die für die Ausführung von Rake erforderlich sind. Andernfalls schlägt der Build fehl.

14. Wählen Sie „Post-Build-Aktion hinzufügen“ und anschließend „AWS CodePipeline Publisher“. Wählen Sie Add aus. Lassen Sie den Speicherort in Build Output Locations leer. Dies ist die Standardkonfiguration. Sie erstellt am Ende des Build-Prozesses eine komprimierte Datei.
15. Wählen Sie Save, um Ihr Jenkins-Projekt zu speichern.

Schritt 2: Erstellen Sie eine Pipeline in CodePipeline

In diesem Teil des Tutorial erstellen Sie die Pipeline mit dem Assistenten Create Pipeline (Pipeline erstellen).

Um einen CodePipeline automatisierten Release-Prozess zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Falls erforderlich, verwenden Sie die Regionsauswahl, um die Region auf diejenige zu ändern, in der sich Ihre Pipeline-Ressourcen befinden. Wenn Sie beispielsweise Ressourcen für das vorherige Tutorial in erstellt haben`us-east-2`, stellen Sie sicher, dass die Regionsauswahl auf USA Ost (Ohio) eingestellt ist.

   Weitere Informationen zu den Regionen und Endpunkten, für die verfügbar sind CodePipeline, finden Sie unter AWS CodePipeline Endpunkte und Kontingente.
3. Wählen sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) aus, oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
4. Geben Sie auf der Seite Step 1: Choose pipeline settings (Schritt 1: Pipeline-Einstellungen auswählen) unter Pipeline name (Pipeline-Name) den Namen für Ihre Pipeline ein.
5. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
6. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.

7. Belassen Sie die Advanced settings (Erweiterte Einstellungen) auf den Standardwerten und wählen Sie Next (Weiter) aus.
8. Wählen Sie auf der Seite Schritt 2: Quellstufe hinzufügen im Feld Quellanbieter die Option aus GitHub.
9. Wählen Sie unter Verbindung eine bestehende Verbindung aus, oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.
10. Wählen Sie in Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Add Jenkins (Jenkins hinzufügen). Geben Sie im Feld Anbietername den Namen der Aktion ein, die Sie im CodePipeline Plugin für Jenkins angegeben haben (z. B. *`MyJenkinsProviderName`*). Dieser Name muss genau mit dem Namen im CodePipeline Plugin für Jenkins übereinstimmen. Geben Sie unter Server URL (Server-URL) die URL der EC2-Instance ein, in der Jenkins installiert ist. Geben Sie unter Projektname den Namen des Projekts ein, das Sie in Jenkins erstellt haben, z. B. ***`MyDemoProject`***, und wählen Sie dann Weiter.
11. Verwenden Sie in Schritt 4: Bereitstellungsphase hinzufügen die CodeDeploy Anwendung und die Bereitstellungsgruppe, in der Sie sie erstellt haben, erneut. Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option CodeDeploy aus. Geben Sie unter Application name (Anwendungsname) „**`CodePipelineDemoApplication`**“ ein, oder klicken Sie auf die Schaltfläche „Aktualisieren“, und wählen Sie dann den Namen der Anwendung aus der Liste aus. Geben Sie unter Deployment group (Bereitstellungsgruppe) **`CodePipelineDemoFleet`** ein oder wählen Sie den Eintrag aus der Liste. Klicken Sie dann auf Next (Weiter).

    > **Note**
    >
    > Sie können Ihre eigenen CodeDeploy Ressourcen verwenden oder neue erstellen, es können jedoch zusätzliche Kosten anfallen.

12. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.
13. Die Pipeline wird automatisch gestartet und führt das Beispiel über die Pipeline aus. Sie können sich Fortschritts-, Erfolgs- und Fehlschlagsmeldungen anzeigen lassen, während die Pipeline das Haml-Beispiel als HTML erstellt und es auf einer Webseite für jede Amazon EC2 EC2-Instance in der Bereitstellung bereitstellt. CodeDeploy

Schritt 3: Hinzufügen einer weiteren Phase zur Pipeline

Nun werden Sie eine Testphase (und dieser eine Testaktion) hinzufügen, die den Jenkins-Test aus dem Beispiel verwendet, um zu bestimmen, ob die Webseite über einen Inhalt verfügt. Dieser Test wird nur zu Demonstrationszwecken verwendet.

> **Note**
>
> Wenn Sie keine weitere Stufe Ihrer Pipeline hinzufügen wollen, könnten Sie der Staging-Stufe der Pipeline vor oder nach der Bereitstellungsaktion eine Testaktion hinzufügen.

Hinzufügen einer Testphase zur Pipeline

Themen

Abfragen der IP-Adresse einer Instance

So prüfen Sie die IP-Adresse einer Instance, in der Sie Ihren Code bereitgestellt haben

1. Nachdem Succeeded für den Pipelinestatus angezeigt wird, wählen Sie im Statusbereich für die Stufe "Staging" die Option Details aus.
2. Klicken Sie im Abschnitt Deployment Details in Instance ID die Instance-ID einer der erfolgreich bereitgestellten Instances an.
3. Kopieren Sie die IP-Adresse der Instance (z. B. `192.168.0.4`). Sie verwenden diese IP-Adresse in Ihrem Jenkins-Test.

Erstellen eines Jenkins-Projekts zum Testen der Bereitstellung

So erstellen Sie das Jenkins-Projekt

1. Öffnen Sie in der Instance, in der Sie Jenkins installiert haben, Jenkins und wählen Sie auf der Hauptseite New Item aus.

2. Geben Sie im Feld Artikelname einen Namen für das Jenkins-Projekt ein (z. B.). *MyTestProject* Wählen Sie Freestyle project und OK aus.

> **Note**
>
> Stellen Sie sicher, dass der Name für Ihr Projekt den CodePipeline Anforderungen entspricht. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline.

3. Wählen Sie auf der Konfigurationsseite des Projekts das Kontrollkästchen Execute concurrent builds if necessary aus. Wählen Sie in Source Code Management (Quellcodeverwaltung) AWS CodePipeline. Wenn Sie Jenkins auf einer EC2-Instance installiert und das AWS CLI mit dem Profil für den IAM-Benutzer konfiguriert haben, den Sie für die Integration zwischen CodePipeline und Jenkins erstellt haben, lassen Sie alle anderen Felder leer.

> **Important**
>
> Wenn Sie ein Jenkins-Projekt konfigurieren und es nicht auf einer Amazon EC2 EC2-Instance installiert ist, oder wenn es auf einer EC2-Instance installiert ist, auf der ein Windows-Betriebssystem ausgeführt wird, füllen Sie die Felder entsprechend Ihren Proxy-Host- und Port-Einstellungen aus und geben Sie die Anmeldeinformationen des IAM-Benutzers oder der IAM-Rolle ein, die Sie für die Integration zwischen Jenkins und konfiguriert haben. CodePipeline

4. Wählen Sie Advanced aus und wählen Sie in Category Test aus.

5. Geben Sie im Feld Provider denselben Namen ein, den Sie für das Build-Projekt verwendet haben (z. B.). *MyJenkinsProviderName* Sie verwenden diesen Namen zu einem späteren Zeitpunkt in diesem Tutorial, wenn Sie die Test-Aktion zu Ihrer Pipeline hinzufügen.

> **Note**
>
> Dieser Name muss die CodePipeline Benennungsanforderungen für Aktionen erfüllen. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline.

6. Deaktivieren Sie in Build Triggers alle Kontrollkästchen und wählen Sie Poll SCM aus. Geben Sie in Schedule (Zeitplan) fünf durch Leerzeichen getrennte Sternchen ein:

```
* * * * *
```

Dieser wird CodePipeline jede Minute abgefragt.

7. Wählen Sie in Build Add build step aus. Wenn Sie die Bereitstellung auf Amazon Linux-, RHEL- oder Ubuntu-Server-Instances durchführen, wählen Sie Shell ausführen. Geben Sie dann Folgendes ein, wobei die IP-Adresse die Adresse der EC2-Instance ist, die Sie zuvor kopiert haben:

   ```
   TEST_IP_ADDRESS=192.168.0.4 rake test
   ```

   Wenn Sie die Bereitstellung auf Windows Server-Instances durchführen, wählen Sie Batch-Befehl ausführen und geben Sie dann Folgendes ein, wobei die IP-Adresse die Adresse der EC2-Instance ist, die Sie zuvor kopiert haben:

   ```
   set TEST_IP_ADDRESS=192.168.0.4 rake test
   ```

   > **Note**
   >
   > Der Test geht davon aus, dass der Standard-Port 80 verwendet wird. Wenn Sie einen anderen Port festlegen, fügen Sie wie folgt eine Test-Port-Anweisung ein:
   >
   > ```
   > TEST_IP_ADDRESS=192.168.0.4 TEST_PORT=8000 rake test
   > ```

8. Wählen Sie „Post-Build-Aktion hinzufügen“ und anschließend „AWS CodePipeline Publisher“. Wählen Sie nicht Add aus.
9. Wählen Sie Save, um Ihr Jenkins-Projekt zu speichern.

Erstellen einer vierten Phase

So fügen Sie eine Stufe mit der Jenkins-Testaktion zu der Pipeline hinzu

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie unter Name den Namen der Pipeline aus, die Sie erstellt haben, MySecondPipeline.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie auf der Seite Edit (Bearbeiten) + Stage (+ Stufe) aus, um eine Stufe direkt nach der Build-Stufe hinzuzufügen.

5. Geben Sie im Namensfeld für die neue Phase einen Namen für die Phase ein (z. B. **`Testing`**) und wählen Sie dann Add action group (Aktionsgruppe hinzufügen) aus.
6. Geben Sie im Feld Aktionsname den Wert *`MyJenkinsTest-Action`* ein. Wählen Sie unter Testanbieter den Anbieternamen aus, den Sie in Jenkins angegeben haben (z. B.). *`MyJenkinsProviderName`* Geben Sie im Feld Projektname den Namen des Projekts ein, das Sie in Jenkins erstellt haben (z. B.). *`MyTestProject`* Wählen Sie unter Eingabeartefakte das Artefakt aus dem Jenkins-Build aus, dessen Standardname lautet ***`BuildArtifact`***, und wählen Sie dann Fertig aus.

   > **Note**
   >
   > Da die Jenkins-Testaktion auf der Anwendung ausgeführt wird, die im Jenkins-Build-Schritt erstellt wurde, verwenden Sie das Build-Artefakt als das Eingabeartefakt für die Testaktion.

   Weitere Informationen über die Eingabe- und Ausgabeartefakte und die Struktur von Pipelines finden Sie in CodePipeline Referenz zur Pipeline-Struktur.
7. Wählen Sie auf der Seite Edit Save pipeline changes aus. Klicken Sie im Dialogfeld Save pipeline changes auf Save and continue.
8. Obwohl die neue Stufe der Pipeline hinzugefügt wurde, wird der Status No executions yet für die Stufe angezeigt. Dies liegt daran, dass noch keine Änderungen eine weitere Ausführung der Pipeline ausgelöst haben. Um das Beispiel durch die überarbeitete Pipeline laufen zu lassen, wählen Sie auf der Seite mit den Pipeline-Details die Option Versionsänderung aus.

   Die Pipelineansicht zeigt die Stufen und Aktionen in Ihrer Pipeline sowie den Status der Revision bei der Ausführung über die vier Stufen an. Die Zeit, die die Pipeline zur Ausführung über alle Stufen benötigt, hängt von der Größe der Artefakte sowie von der Komplexität Ihrer Build- und Testaktionen und von anderen Faktoren ab.

Schritt 4: Bereinigen von Ressourcen

Nach Abschluss dieses Tutorials sollten Sie die Pipeline und die von ihr verwendeten Ressourcen löschen, damit Ihnen keine Kosten für die weitere Nutzung der Ressourcen entstehen. Wenn Sie nicht weiter verwenden möchten CodePipeline, löschen Sie die Pipeline, dann die CodeDeploy Anwendung und die zugehörigen Amazon EC2 EC2-Instances und schließlich den Amazon S3-

Bucket, der zum Speichern von Artefakten verwendet wurde. Sie sollten auch überlegen, ob Sie andere Ressourcen, wie das GitHub Repository, löschen sollten, wenn Sie nicht beabsichtigen, sie weiter zu verwenden.

So bereinigen Sie die in diesem Tutorial verwendeten Ressourcen

1. Öffnen Sie eine Terminalsitzung auf Ihrem lokalen Linux-, macOS- oder Unix-Computer oder eine Befehlszeile auf Ihrem lokalen Windows-Computer und führen Sie den delete-pipeline Befehl zum Löschen der von Ihnen erstellten Pipeline aus. Für **`MySecondPipeline`** geben Sie beispielsweise den folgenden Befehl ein:

   ```
   aws codepipeline delete-pipeline --name "MySecondPipeline"
   ```

   Mit diesem Befehl wird kein Inhalt zurückgegeben.
2. Folgen Sie den Anweisungen unter Aufräumen, um Ihre CodeDeploy Ressourcen zu bereinigen.
3. Löschen Sie zum Bereinigen Ihrer Instance-Ressourcen die EC2-Instance, in der Sie Jenkins installiert haben. Weitere Informationen finden Sie unter Bereinigen Ihrer Instance.
4. Wenn Sie nicht beabsichtigen, weitere Pipelines zu erstellen oder CodePipeline erneut zu verwenden, löschen Sie den Amazon S3 S3-Bucket, der zum Speichern von Artefakten für Ihre Pipeline verwendet wurde. Folgen Sie den Anweisungen in Löschen eines Buckets, um den Bucket zu löschen.
5. Wenn Sie die anderen Ressourcen für diese Pipelines nicht erneut verwenden möchten, können Sie diese mithilfe der Anweisungen für die entsprechende Ressource löschen. Wenn Sie beispielsweise das GitHub Repository löschen möchten, folgen Sie den Anweisungen unter Löschen eines Repositorys auf der GitHub Website.

Tutorial: Richten Sie eine CloudWatch Ereignisregel ein, um E-Mail-Benachrichtigungen für Änderungen des Pipeline-Status zu erhalten

Nachdem Sie eine Pipeline eingerichtet haben AWS CodePipeline, können Sie eine CloudWatch Ereignisregel einrichten, um Benachrichtigungen zu senden, wenn sich der Ausführungsstatus Ihrer Pipelines oder die Phasen oder Aktionen in Ihren Pipelines ändern. Weitere Informationen zur Verwendung von CloudWatch Ereignissen zum Einrichten von Benachrichtigungen für Änderungen des Pipeline-Status finden Sie unter. CodePipeline Ereignisse überwachen

In diesem Tutorial konfigurieren Sie eine Benachrichtigung zum Senden einer E-Mail, wenn der Status einer Pipeline sich zu FAILED ändert. In diesem Tutorial wird beim Erstellen der CloudWatch Ereignisregel eine Methode zur Eingabe eines Transformators verwendet. Diese wandelt die Details des Nachrichtenschemas um, um die Nachricht in Form von lesbarem Text bereitzustellen.

> **Note**
>
> Achten Sie beim Erstellen der Ressourcen für dieses Tutorial, wie z. B. die Amazon SNS SNS-Benachrichtigung und die CloudWatch Ereignisregel, darauf, dass die Ressourcen in derselben AWS Region wie Ihre Pipeline erstellt wurden.

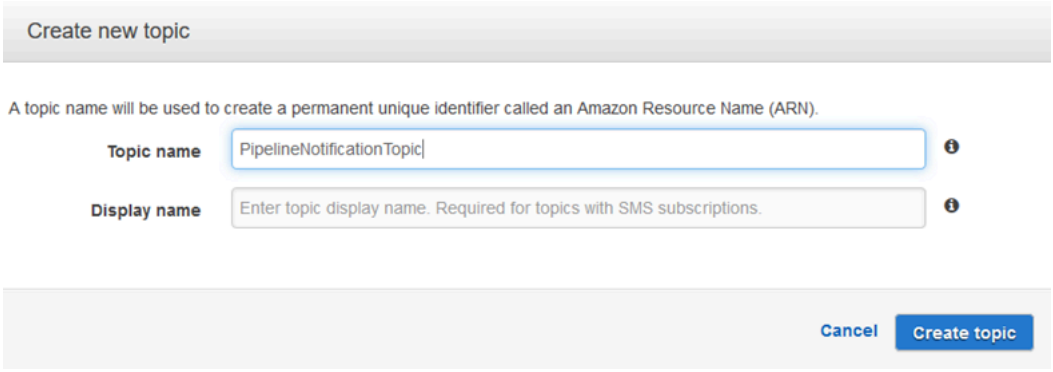
Themen

Schritt 1: E-Mail-Benachrichtigung mit Amazon SNS einrichten

Amazon SNS koordiniert die Verwendung von Themen für die Zustellung von Nachrichten an abonnierte Endpunkte oder Kunden. Verwenden Sie Amazon SNS, um ein Benachrichtigungsthema zu erstellen, und abonnieren Sie das Thema dann mit Ihrer E-Mail-Adresse. Das Amazon SNS SNS-Thema wird Ihrer CloudWatch Events-Regel als Ziel hinzugefügt. Weitere Informationen finden Sie im Amazon Simple Notification Service-Entwicklerhandbuch.

Erstellen oder identifizieren Sie ein Thema in Amazon SNS. CodePipeline verwendet CloudWatch Events, um Benachrichtigungen zu diesem Thema über Amazon SNS zu senden. Erstellen Sie ein Thema wie folgt:

1. Öffnen Sie die Amazon SNS SNS-Konsole unter https://console.aws.amazon.com/sns.
2. Wählen Sie Thema erstellen aus.
3. Geben Sie in das Dialogfeld Create new topic (Neues Thema erstellen) für Topic name (Themennamen) einen Namen für das Thema ein (z. B. **`PipelineNotificationTopic`**).



4. Wählen Sie Thema erstellen aus.

   Weitere Informationen finden Sie unter Thema erstellen im Amazon SNS SNS-Entwicklerhandbuch.

Abonnieren Sie das Thema für einen oder mehrere Empfänger, um E-Mail-Benachrichtigungen zu empfangen. So abonnieren Sie ein Thema für einen Empfänger:

1. Aktivieren Sie in der Amazon SNS SNS-Konsole in der Themenliste das Kontrollkästchen neben Ihrem neuen Thema. Wählen Sie Actions, Subscribe to topic aus.
2. Vergewissern Sie sich, dass im Dialogfeld Create subscription unter Topic ARN ein ARN angezeigt wird.
3. Wählen Sie unter Protocol (Protokoll) die Option Email (E-Mail) aus.
4. Geben Sie für Endpoint die vollständige E-Mail-Adresse des Empfängers ein.
5. Klicken Sie auf Create subscription (Abonnement erstellen).
6. Amazon SNS sendet eine Bestätigungs-E-Mail für das Abonnement an den Empfänger. Um E-Mail-Benachrichtigungen zu empfangen, muss der Empfänger den Link Confirm subscription in dieser E-Mail aufrufen. Nachdem der Empfänger auf den Link geklickt hat, zeigt Amazon SNS bei erfolgreichem Abonnement eine Bestätigungsnachricht im Webbrowser des Empfängers an.

   Weitere Informationen finden Sie unter Thema abonnieren im Amazon SNS Developer Guide.

Schritt 2: Erstellen einer Regel und Hinzufügen des SNS-Themas als Ziel

Erstellen Sie eine Regel für die Benachrichtigung über CloudWatch Ereignisse mit CodePipeline der Ereignisquelle.

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.

2. Wählen Sie im Navigationsbereich die Option Events.
3. Wählen Sie Regel erstellen aus. Wählen Sie unter Event source (Ereignisquelle) die Option AWS CodePipeline aus. Wählen Sie für Event Type die Option Pipeline Execution State Change aus.
4. Wählen Sie Specific state(s) (Spezifische(r) Status) aus und dann die Option **`FAILED`**.
5. Wählen Sie Edit aus, um den JSON-Editor für den Bereich Event Pattern Preview zu öffnen. Fügen Sie den **`pipeline`**-Parameter mit dem Namen Ihrer Pipeline wie im folgenden Beispiel für eine Pipeline mit dem Namen "myPipeline" gezeigt hinzu.

   Sie können das Ereignismuster hier kopieren und in die Konsole einfügen:

   ```
   {
     "source": [
       "aws.codepipeline"
     ],
     "detail-type": [
       "CodePipeline Pipeline Execution State Change"
     ],
     "detail": {
       "state": [
         "FAILED"
       ],
       "pipeline": [
         "myPipeline"
       ]
     }
   }
   ```
6. Wählen Sie für Targets die Option Add target aus.
7. Wählen Sie in der Liste der Ziele SNS topic aus. Geben Sie für Topic das erstellte Thema an.
8. Erweitern Sie Configure input und wählen Sie dann Input Transformer aus.
9. Geben Sie im Feld Input Path die folgenden Schlüssel-Wert-Paare ein.

   ```
   { "pipeline" : "$.detail.pipeline" }
   ```

   Geben Sie im Feld Input Template Folgendes ein:

   ```
   "The Pipeline <pipeline> has failed."
   ```
10. Wählen Sie Details konfigurieren.

11. Geben Sie auf der Seite Configure rule details einen Namen und optional eine Beschreibung ein. Lassen Sie für State das Kontrollkästchen Enabled aktiviert.
12. Wählen Sie Regel erstellen aus.
13. Bestätigen Sie, CodePipeline dass jetzt Build-Benachrichtigungen gesendet werden. Überprüfen Sie beispielsweise, ob sich die Build-Benachrichtigungs-E-Mails jetzt in Ihrem Posteingang befinden.
14. Um das Verhalten einer Regel zu ändern, wählen Sie in der CloudWatch Konsole die Regel aus und klicken Sie dann auf Aktionen, Bearbeiten. Bearbeiten Sie die Regel, wählen Sie Configure details und anschließend Update rule aus.

    Um eine Regel nicht mehr zum Senden von Build-Benachrichtigungen zu verwenden, wählen Sie in der CloudWatch Konsole die Regel aus und klicken Sie dann auf Aktionen, Deaktivieren.

    Um eine Regel zu löschen, wählen Sie in der CloudWatch Konsole die Regel und dann Aktionen, Löschen aus.

Schritt 3: Bereinigen von Ressourcen

Nach Abschluss dieses Tutorials sollten Sie die Pipeline und die von ihr verwendeten Ressourcen löschen, damit Ihnen keine Kosten für die weitere Nutzung der Ressourcen entstehen.

Informationen dazu, wie Sie die SNS-Benachrichtigung bereinigen und die Amazon CloudWatch Events-Regel löschen können, finden Sie unter Clean Up (Abmeldung von einem Amazon SNS SNS-Thema) und Referenz `DeleteRule` in der Amazon CloudWatch Events API-Referenz.

Tutorial: Erstellen Sie eine Pipeline, die Ihre Android-App erstellt und testet mit AWS Device Farm

Sie können AWS CodePipeline damit einen kontinuierlichen Integrationsablauf konfigurieren, in dem Ihre App jedes Mal erstellt und getestet wird, wenn ein Commit übertragen wird. Dieses Tutorial zeigt, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre Android-App mit Quellcode in einem GitHub Repository zu erstellen und zu testen. Die Pipeline erkennt das Eintreffen eines neuen GitHub Commits und verwendet es dann CodeBuild, um die App und die Device Farm zu erstellen, um sie zu testen.

> **Important**
>
> Viele der Aktionen, die Sie Ihrer Pipeline in diesem Verfahren hinzufügen, beinhalten AWS Ressourcen, die Sie erstellen müssen, bevor Sie die Pipeline erstellen. AWS Ressourcen für Ihre Quellaktionen müssen immer in derselben AWS Region erstellt werden, in der Sie Ihre Pipeline erstellen. Wenn Sie Ihre Pipeline beispielsweise in der Region USA Ost (Ohio) erstellen, muss sich Ihr CodeCommit Repository in der Region USA Ost (Ohio) befinden. Sie können beim Erstellen Ihrer Pipeline regionsübergreifende Aktionen hinzufügen. AWS Ressourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

Sie können dies mit Ihrer vorhandenen Android-App und Ihren Testdefinitionen ausprobieren, oder Sie können die von Device Farm bereitgestellten Beispiel-Apps und Testdefinitionen verwenden.

> **Note**

Bevor Sie beginnen

1. Melden Sie sich bei der AWS Device Farm Konsole an und wählen Sie Neues Projekt erstellen aus.
2. Wählen Sie Ihr Projekt. Kopieren Sie im Browser die URL Ihres neuen Projekts. Die URL enthält die Projekt-ID.
3. Kopieren Sie diese Produkt-ID und bewahren Sie sie auf. Sie verwenden es, wenn Sie Ihre Pipeline in erstellen CodePipeline.

   Hier ist eine Beispiel-URL für ein Projekt. Um die Projekt-ID zu extrahieren, kopieren Sie den Wert nach `projects/`. In diesem Beispiel lautet die Produkt-ID `eec4905f-98f8-40aa-9afc-4c1cfexample`.

   ```
   https://<region-URL>/devicefarm/home?region=us-west-2#/projects/
   eec4905f-98f8-40aa-9afc-4c1cfexample/runs
   ```

Für CodePipeline die Verwendung Ihrer Device Farm Farm-Tests konfigurieren

1. Fügen Sie eine Datei mit dem Namen `buildspec.yml` im Stammverzeichnis Ihres App-Codes hinzu, übertragen Sie sie und übertragen Sie sie in Ihr Repository. CodeBuild verwendet diese Datei, um Befehle auszuführen und auf Artefakte zuzugreifen, die für die Erstellung Ihrer App erforderlich sind.

```
version: 0.2

phases:
  build:
    commands:
      - chmod +x ./gradlew
      - ./gradlew assembleDebug
artifacts:
  files:
     - './android/app/build/outputs/**/*.apk'
  discard-paths: yes
```

2. (Optional) Wenn Sie Calabash or Appium zum Testen Ihrer App verwenden, fügen Sie die Testdefinitionsdatei zu Ihrem Repository hinzu. In einem späteren Schritt können Sie Device Farm so konfigurieren, dass die Definitionen zur Ausführung Ihrer Testsuite verwendet werden.

   Wenn Sie die integrierten Tests von Device Farm verwenden, können Sie diesen Schritt überspringen.

3. Um Ihre Pipeline zu erstellen und eine Quellphase hinzuzufügen, gehen Sie wie folgt vor:

   a. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   b. Wählen Sie Create pipeline (Pipeline erstellen) aus. Geben Sie auf der Seite Step 1: Choose pipeline settings (Schritt 1: Pipeline-Einstellungen auswählen) unter Pipeline name (Pipeline-Name) den Namen für Ihre Pipeline ein.

   c. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.

d. Wählen Sie unter Service role weiterhin New service role aus und lassen Sie Role name unverändert. Sie können auch eine vorhandene Service-Rolle verwenden, sofern eine vorhanden ist.

> **Note**
>
> Wenn Sie eine CodePipeline Servicerolle verwenden, die vor Juli 2018 erstellt wurde, müssen Sie Berechtigungen für Device Farm hinzufügen. Öffnen Sie dazu die IAM-Konsole, suchen Sie nach der Rolle und fügen Sie dann der Rollenrichtlinie die folgenden Berechtigungen hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.
>
> ```
> {
>      "Effect": "Allow",
>      "Action": [
>         "devicefarm:ListProjects",
>         "devicefarm:ListDevicePools",
>         "devicefarm:GetRun",
>         "devicefarm:GetUpload",
>         "devicefarm:CreateUpload",
>         "devicefarm:ScheduleRun"
>      ],
>      "Resource": "*"
> }
> ```

e. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.

f. Wählen Sie auf der Seite Schritt 2: Quellstufe hinzufügen unter Quellanbieter die Option GitHub.

g. Wählen Sie unter Verbindung eine bestehende Verbindung aus, oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.

h. Wählen Sie unter Repository das Quell-Repository aus.

i. Wählen Sie unter Branch (Verzweigung) die Verzweigung aus, Sie verwenden möchten.

j. Behalten Sie die übrigen Standardeinstellungen für die Quellaktion bei. Wählen Sie Weiter aus.

4. Fügen Sie unter Add build stage (Build-Phase hinzufügen) eine Build-Phase hinzu:

a. Wählen Sie unter Build provider (Build-Anbieter) die Option AWS CodeBuild aus. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region.

b. Wählen Sie Create project (Projekt erstellen) aus.

c. Geben Sie unter Project name (Projektname) einen Namen für dieses Build-Projekt ein.

d. Wählen Sie für Environment image (Umgebungs-Image) die Option Managed image (Verwaltetes Image) aus. Wählen Sie für Operating system (Betriebssystem) die Option Ubuntu aus.

e. Wählen Sie unter Runtime (Laufzeit) die Option Standard aus. Wählen Sie für Image (Abbild) die Option aws/codebuild/standard:5.0 aus.

   CodeBuild verwendet dieses Betriebssystem-Image, auf dem Android Studio installiert ist, um Ihre App zu erstellen.

f. Wählen Sie unter Servicerolle Ihre bestehende CodeBuild Servicerolle aus oder erstellen Sie eine neue.

g. Wählen Sie unter Build specifications (Build-Spezifikationen) die Option Use a buildspec file (Eine buildspec-Datei verwenden).

h. Wählen Sie Weiter zu CodePipeline. Dadurch kehren Sie zur CodePipeline Konsole zurück und erstellen ein CodeBuild Projekt, das das `buildspec.yml` in Ihrem Repository für die Konfiguration verwendet. Das Build-Projekt verwendet eine Servicerolle zur Verwaltung von AWS-Service Berechtigungen. Dieser Vorgang kann einige Minuten dauern.

i. Wählen Sie Weiter aus.

5. Wählen Sie auf der Seite Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen) die Option Skip deploy stage (Bereitstellungsstufe überspringen) aus und akzeptieren Sie anschließend die Warnmeldung, indem Sie erneut Skip (Überspringen) auswählen. Wählen Sie Weiter aus.

6. Wählen Sie unter Step 5: Review (Schritt 5: Überprüfen) die Option Create pipeline (Pipeline erstellen) aus. Sie sollten ein Diagramm sehen, das Ihre Pipeline-Quell- und -Build-Stufen anzeigt.

7. Fügen Sie Ihrer Pipeline eine Device Farm Farm-Testaktion hinzu:

   a. Wählen Sie rechts oben Edit (Bearbeiten) aus.

   b. Wählen Sie unten im Diagramm + Add stage (+ Stufe hinzufügen) aus. Geben Sie unter Stage name (Name der Phase), einen Namen ein, z. B. **`Test`**.

   c. Wählen Sie + Add action group (Aktionsgruppe hinzufügen).

d. Geben Sie unter Action name (Aktionsname) einen Namen ein.

e. Wählen Sie unter Aktionsanbieter die Option AWS Device Farm aus. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region.

f. Wählen Sie unter Input artifacts (Eingabeartefakte) das Eingabeartefakt aus, das mit dem Ausgabeartefakt der Phase übereinstimmt, die sich vor der Testphase befindet, wie etwa `BuildArtifact`.

   In der AWS CodePipeline Konsole finden Sie den Namen des Ausgabeartefakts für jede Phase, indem Sie den Mauszeiger über das Informationssymbol im Pipeline-Diagramm bewegen. Wenn Ihre Pipeline Ihre App direkt von der Quellphase aus testet, wählen Sie. SourceArtifact Wenn die Pipeline eine Build-Phase enthält, wählen Sie BuildArtifact.

g. Geben Sie Ihre Device Farm Farm-Projekt-ID ein. ProjectId Führen Sie die Schritte zu Beginn dieses Tutorials aus, um Ihre Projekt-ID abzurufen.

h. Geben Sie DevicePoolArnunter den ARN für den Gerätepool ein. Geben Sie mit der AWS CLI den folgenden Befehl ein, um die verfügbaren Gerätepool-ARNs für das Projekt abzurufen, einschließlich des ARN für Top-Geräte:

   ```
   aws devicefarm list-device-pools --arn arn:aws:devicefarm:us-
   west-2:account_ID:project:project_ID
   ```

i. Geben Sie Android ein. AppType

   Im Folgenden finden Sie eine Liste gültiger Werte für AppType:

   - iOS
   - Android
   - Web

j. Geben Sie unter App den Pfad des kompilierten Anwendungspakets ein. Der Pfad ist relativ zum Stamm des Eingabeartefakts für die Testphase. In der Regel sieht dieser Pfad in etwa wie `app-release.apk` aus.

k. Geben Sie unter Ihren Testtyp und anschließend im Feld Test den Pfad der Testspezifikationsdatei ein. TestType Der Pfad ist relativ zum Stamm des Eingabeartefakts für Ihren Test.

   Die folgende Liste enthält gültige Werte für TestType:

   - APPIUM_JAVA_JUNIT

- APPIUM_JAVA_TESTNG
- APPIUM_KNOTEN
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- EINGEBAUTER FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> Knoten für benutzerdefinierte Umgebungen werden nicht unterstützt.

l. Geben Sie in den übrigen Feldern die Konfiguration ein, die für Ihren Test- und Anwendungstyp geeignet ist.

m. (Optional) Geben Sie unter Advanced (Erweitert) Konfigurationsinformationen für Ihren Testlauf ein.

n. Wählen Sie Speichern.

o. Wählen Sie in der Phase, die Sie bearbeiten, Fertig. Wählen Sie im AWS CodePipeline - Fenster Save (Speichern) und dann in der Warnmeldung Save (Speichern).

p. Um Ihre Änderungen zu übertragen und einen Pipeline-Build zu starten, wählen Sie Release change (Änderung freigeben) und dann Release (Freigeben).

Tutorial: Erstellen Sie eine Pipeline, die Ihre iOS-App testet mit AWS Device Farm

Sie können AWS CodePipeline damit ganz einfach einen kontinuierlichen Integrationsablauf konfigurieren, in dem Ihre App bei jeder Änderung des Quell-Buckets getestet wird. In diesem Tutorial wird gezeigt, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre erstellte iOS-App aus einem S3-Bucket zu testen. Die Pipeline erkennt das Eintreffen einer gespeicherten Änderung über Amazon CloudWatch Events und verwendet dann Device Farm, um die erstellte Anwendung zu testen.

> **Important**
>
> Viele der Aktionen, die Sie Ihrer Pipeline in diesem Verfahren hinzufügen, beinhalten AWS Ressourcen, die Sie erstellen müssen, bevor Sie die Pipeline erstellen. AWS Ressourcen für Ihre Quellaktionen müssen immer in derselben AWS Region erstellt werden, in der Sie Ihre Pipeline erstellen. Wenn Sie Ihre Pipeline beispielsweise in der Region USA Ost (Ohio) erstellen, muss sich Ihr CodeCommit Repository in der Region USA Ost (Ohio) befinden. Sie können beim Erstellen Ihrer Pipeline regionsübergreifende Aktionen hinzufügen. AWS Ressourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

Sie können dies mit Ihrer vorhandenen iOS-App oder der iOS-Beispiel-App ausprobieren.

> **Note**

Bevor Sie beginnen

1. Melden Sie sich bei der AWS Device Farm Konsole an und wählen Sie Neues Projekt erstellen aus.
2. Wählen Sie Ihr Projekt. Kopieren Sie im Browser die URL Ihres neuen Projekts. Die URL enthält die Projekt-ID.
3. Kopieren Sie diese Produkt-ID und bewahren Sie sie auf. Sie verwenden es, wenn Sie Ihre Pipeline in erstellen CodePipeline.

Hier ist eine Beispiel-URL für ein Projekt. Um die Projekt-ID zu extrahieren, kopieren Sie den Wert nach `projects/`. In diesem Beispiel lautet die Produkt-ID `eec4905f-98f8-40aa-9afc-4c1cfexample`.

```
https://<region-URL>/devicefarm/home?region=us-west-2#/projects/
eec4905f-98f8-40aa-9afc-4c1cfexample/runs
```

Für CodePipeline die Verwendung Ihrer Device Farm Farm-Tests konfigurieren (Beispiel Amazon S3)

1. Erstellen oder verwenden Sie einen S3-Bucket mit aktiviertem Versioning. Sie können den Anweisungen unter Schritt 1: Erstellen eines S3-Buckets für Ihre Anwendung folgen, um einen S3-Bucket zu erstellen.
2. Wählen Sie in der Amazon S3 S3-Konsole für Ihren Bucket Upload und folgen Sie den Anweisungen, um Ihre ZIP-Datei hochzuladen.

   Ihre Beispielanwendung muss in komprimierter Form als ZIP-Datei vorliegen.
3. Um Ihre Pipeline zu erstellen und eine Quellphase hinzuzufügen, gehen Sie wie folgt vor:

   a. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   b. Wählen Sie Create pipeline (Pipeline erstellen) aus. Geben Sie auf der Seite Step 1: Choose pipeline settings (Schritt 1: Pipeline-Einstellungen auswählen) unter Pipeline name (Pipeline-Name) den Namen für Ihre Pipeline ein.

   c. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.

   d. Wählen Sie unter Service role weiterhin New service role aus und lassen Sie Role name unverändert. Sie können auch eine vorhandene Service-Rolle verwenden, sofern eine vorhanden ist.

> **Note**
>
> Wenn Sie eine CodePipeline Servicerolle verwenden, die vor Juli 2018 erstellt wurde, müssen Sie Berechtigungen für Device Farm hinzufügen. Öffnen Sie dazu die IAM-Konsole, suchen Sie nach der Rolle und fügen Sie dann der Rollenrichtlinie die folgenden Berechtigungen hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.
>
> ```
> {
>      "Effect": "Allow",
>      "Action": [
>         "devicefarm:ListProjects",
>         "devicefarm:ListDevicePools",
>         "devicefarm:GetRun",
>         "devicefarm:GetUpload",
>         "devicefarm:CreateUpload",
>         "devicefarm:ScheduleRun"
>      ],
>      "Resource": "*"
> }
> ```

e. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.

f. Auf der Seite Step 2: Add source stage (Schritt 2: Hinzufügen der Quell-Stufe) wählen Sie für Source provider (Quell-Anbieter) die Option Amazon S3.

g. Geben Sie im Amazon S3 S3-Speicherort den Bucket (z. `my-storage-bucket` B.) und den Objektschlüssel (z. B. `s3-ios-test-1.zip` für Ihre ZIP-Datei) ein.

h. Wählen Sie Weiter aus.

4. Erstellen Sie unter Build eine Platzhalter-Build-Stufe für Ihre Pipeline. So können Sie die Pipeline im Assistenten erstellen. Wenn Sie den Assistenten zum Erstellen Ihrer zweistufigen Pipeline verwendet haben, brauchen Sie diese Platzhalter-Build-Stufe nicht mehr. Nachdem die Pipeline abgeschlossen ist, wird die zweite Stufe gelöscht und die neue Teststufe in Schritt 5 hinzugefügt.

   a. Wählen Sie unter Build provider (Build-Anbieter) auf Add Jenkins (Jenkins hinzufügen). Diese Build-Auswahl ist nur ein Platzhalter. Sie wird nicht verwendet.

b. Geben Sie unter Provider name (Anbietername) einen Namen ein. Der Name ist ein Platzhalter. Sie wird nicht verwendet.

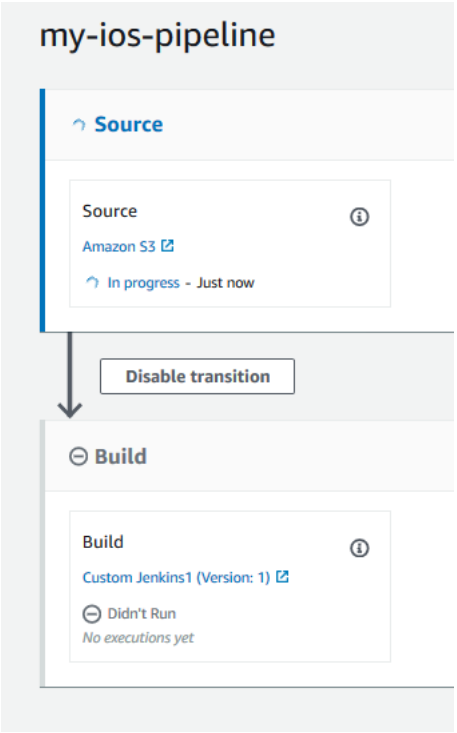
c. Geben Sie unter Server URL Text ein. Der Text ist ein Platzhalter. Sie wird nicht verwendet.

d. Geben Sie unter Project name (Projektname) einen Namen ein. Der Name ist ein Platzhalter. Sie wird nicht verwendet.

e. Wählen Sie Weiter aus.

f. Wählen Sie auf der Seite Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen) die Option Skip deploy stage (Bereitstellungsstufe überspringen) aus und akzeptieren Sie anschließend die Warnmeldung, indem Sie erneut Skip (Überspringen) auswählen.

g. Wählen Sie unter Step 5: Review (Schritt 5: Überprüfen) die Option Create pipeline (Pipeline erstellen) aus. Sie sollten ein Diagramm sehen, das Ihre Pipeline-Quell- und -Build-Stufen anzeigt.



5. Fügen Sie Ihrer Pipeline wie folgt eine Device Farm Farm-Testaktion hinzu:

a. Wählen Sie rechts oben Edit (Bearbeiten) aus.

b. Wählen Sie Edit stage (Phase bearbeiten). Wählen Sie Löschen aus. Dadurch wird die Platzhalter-Stufe gelöscht, die Sie nun für die Pipeline-Erstellung nicht mehr benötigen.

c. Wählen Sie unten im Diagramm + Add stage (+ Stufe hinzufügen) aus.

d. Geben Sie unter „Stage Name (Name der Phase)“ einen Namen für die Phase ein, etwa Test, und wählen Sie dann Add stage (Phase hinzufügen).

e. Wählen Sie + Add action group (Aktionsgruppe hinzufügen).

f. Geben Sie im Feld Aktionsname einen Namen ein, z. DeviceFarmTest B.

g. Wählen Sie unter Aktionsanbieter die Option AWS Device Farm aus. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region.

h. Wählen Sie unter Input artifacts (Eingabeartefakte) das Eingabeartefakt aus, das mit dem Ausgabeartefakt der Phase übereinstimmt, die sich vor der Testphase befindet, wie etwa `SourceArtifact`.

   In der AWS CodePipeline Konsole finden Sie den Namen des Ausgabeartefakts für jede Phase, indem Sie den Mauszeiger über das Informationssymbol im Pipeline-Diagramm bewegen. Wenn Ihre Pipeline Ihre App direkt von der Quellphase aus testet, wählen Sie. SourceArtifact Wenn die Pipeline eine Build-Phase enthält, wählen Sie BuildArtifact.

i. Wählen Sie ProjectIdunter Ihre Device Farm Farm-Projekt-ID aus. Führen Sie die Schritte zu Beginn dieses Tutorials aus, um Ihre Projekt-ID abzurufen.

j. Geben Sie DevicePoolArnunter den ARN für den Gerätepool ein. Geben Sie mit der AWS CLI den folgenden Befehl ein, um die verfügbaren Gerätepool-ARNs für das Projekt abzurufen, einschließlich des ARN für Top-Geräte:

```
aws devicefarm list-device-pools --arn arn:aws:devicefarm:us-
west-2:account_ID:project:project_ID
```

k. Geben AppTypeSie iOS ein.

   Im Folgenden finden Sie eine Liste gültiger Werte für AppType:

   - iOS
   - Android
   - Web

l. Geben Sie unter App den Pfad des kompilierten Anwendungspakets ein. Der Pfad ist relativ zum Stamm des Eingabeartefakts für die Testphase. In der Regel sieht dieser Pfad in etwa wie `ios-test.ipa` aus.

m. Geben Sie unter Ihren Testtyp und anschließend im Feld Test den Pfad der Testspezifikationsdatei ein. TestType Der Pfad ist relativ zum Stamm des Eingabeartefakts für Ihren Test.

Wenn Sie einen der integrierten Device Farm Farm-Tests verwenden, geben Sie den in Ihrem Device Farm Farm-Projekt konfigurierten Testtyp ein, z. B. BUILTIN_FUZZ. Geben Sie unter eine Zeit in FuzzEventCountMillisekunden ein, z. B. 6000. Geben Sie FuzzEventThrottleunter eine Zeit in Millisekunden ein, z. B. 50.

Wenn Sie keinen der integrierten Device Farm Farm-Tests verwenden, geben Sie Ihren Testtyp und dann im Feld Test den Pfad der Testdefinitionsdatei ein. Der Pfad ist relativ zum Stamm des Eingabeartefakts für Ihren Test.

Die folgende Liste enthält gültige Werte für TestType:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_KNOTEN
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- EINGEBAUTETER_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> Knoten für benutzerdefinierte Umgebungen werden nicht unterstützt.

n. Geben Sie in den übrigen Feldern die Konfiguration ein, die für Ihren Test- und Anwendungstyp geeignet ist.

o. (Optional) Geben Sie unter Advanced (Erweitert) Konfigurationsinformationen für Ihren Testlauf ein.

p. Wählen Sie Speichern.

q. Wählen Sie in der Phase, die Sie bearbeiten, Fertig. Wählen Sie in dem AWS CodePipeline Bereich Speichern aus, und wählen Sie dann in der Warnmeldung Speichern aus.

r. Um Ihre Änderungen zu übertragen und eine Pipelineausführung zu starten, wählen Sie Release change (Änderung freigeben) und dann Release (Freigeben).

Tutorial: Erstellen Sie eine Pipeline, die in Service Catalog bereitgestellt wird

Service Catalog ermöglicht es Ihnen, Produkte auf der Grundlage von AWS CloudFormation Vorlagen zu erstellen und bereitzustellen. In diesem Tutorial erfahren Sie, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre Produktvorlage in Service Catalog bereitzustellen und die Änderungen zu übertragen, die Sie in Ihrem Quell-Repository vorgenommen haben (bereits in GitHub, CodeCommit, oder Amazon S3 erstellt).

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, müssen Sie alle Quelldateien, die als eine einzige ZIP-Datei verpackt sind, in Ihren Bucket hochladen. Andernfalls schlägt die Quellaktion fehl.

Zuerst erstellen Sie ein Produkt im Service Catalog und dann eine Pipeline in AWS CodePipeline. Dieses Tutorial bietet zwei Optionen für die Einrichtung der Bereitstellungskonfiguration:

- Erstellen Sie ein Produkt im Service Catalog und laden Sie eine Vorlagendatei in Ihr Quell-Repository hoch. Geben Sie die Produktversion und die Bereitstellungskonfiguration in der CodePipeline Konsole an (ohne separate Konfigurationsdatei). Siehe Option 1: Bereitstellung in Service Catalog ohne Konfigurationsdatei.

  > **Note**
  >
  > Die Vorlagendatei kann im Format YAML oder JSON erstellt werden.

- Erstellen Sie ein Produkt im Service Catalog und laden Sie eine Vorlagendatei in Ihr Quell-Repository hoch. Stellen Sie die Produktversion und Bereitstellungskonfiguration in einer separaten

Konfigurationsdatei bereit. Siehe Option 2: Bereitstellung in Service Catalog mithilfe einer Konfigurationsdatei.

Option 1: Bereitstellung in Service Catalog ohne Konfigurationsdatei

In diesem Beispiel laden Sie die AWS CloudFormation Beispielvorlagendatei für einen S3-Bucket hoch und erstellen dann Ihr Produkt im Service Catalog. Als Nächstes erstellen Sie Ihre Pipeline und geben die Bereitstellungskonfiguration in der CodePipeline Konsole an.

Schritt 1: Hochladen der Beispielvorlagendatei in das Quellrepository

1. Öffnen Sie einen Texteditor. Erstellen Sie eine Beispielvorlage, indem Sie den folgenden Code in die Datei einfügen. Speichern Sie die Datei als `S3_template.json`.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "CloudFormation Sample Template S3_Bucket: Sample template showing
 how to create a privately accessible S3 bucket. **WARNING** This template creates
 an S3 bucket. You will be billed for the resources used if you create a stack from
 this template.",
  "Resources": {
    "S3Bucket": {
      "Type": "AWS::S3::Bucket",
      "Properties": {}
    }
  },
  "Outputs": {
    "BucketName": {
      "Value": {
        "Ref": "S3Bucket"
      },
      "Description": "Name of Amazon S3 bucket to hold website content"
    }
  }
}
```

   Diese Vorlage ermöglicht AWS CloudFormation die Erstellung eines S3-Buckets, der von Service Catalog verwendet werden kann.

2. Laden Sie die Datei `S3_template.json` in Ihr AWS CodeCommit -Repository hoch.

Schritt 2: Erstellen Sie ein Produkt im Service Catalog

1. Melden Sie sich als IT-Administrator bei der Service Catalog-Konsole an, wechseln Sie zur Produktseite und wählen Sie dann Neues Produkt hochladen aus.
2. Führen Sie auf der Seite Upload new product (Neues Produkt hochladen) die folgenden Schritte aus:

   a. Geben Sie unter Product name (Produktname) den gewünschten Namen für Ihr neues Produkt ein.

   b. Geben Sie unter Description (Beschreibung) die Beschreibung des Produktkatalogs an. Diese Beschreibung wird in der Produktliste angezeigt, um den Benutzern die Auswahl des richtigen Produkts zu erleichtern.

   c. Geben Sie unter Provided by (Bereitgestellt von) den Namen Ihrer IT-Abteilung oder des Administrators ein.

   d. Wählen Sie Weiter aus.
3. (Optional) Geben Sie unter Enter support details (Supportdetails eingeben) die Kontaktinformationen für den Produktsupport ein und klicken Sie auf Next (Weiter).
4. Führen Sie unter Version details (Versionsdetails) die folgenden Schritte aus:

   a. Wählen Sie Upload a template file (Vorlagendatei hochladen). Suchen Sie die Datei `S3_template.json` und laden Sie sie hoch.

   b. Geben Sie unter Version title (Versionstitel) den Namen der Produktversion ein (z. B. **`devops S3 v2`**).

   c. Geben Sie unter Description (Beschreibung) die Details ein, die diese Version von anderen Versionen unterscheidet.

   d. Wählen Sie Weiter aus.
5. Überprüfen Sie auf der Seite Review (Überprüfung) die Richtigkeit der Informationen und klicken Sie dann auf Create (Erstellen).
6. Kopieren Sie die URL Ihres neuen Produkts im Browser auf der Seite Products (Produkte). Diese enthält die Produkt-ID. Kopieren Sie diese Produkt-ID und bewahren Sie sie auf. Sie verwenden es, wenn Sie Ihre Pipeline in erstellen CodePipeline.

   Hier sehen Sie die URL für ein Produkt mit dem Namen `my-product`. Um die Produkt-ID zu extrahieren, kopieren Sie den Wert zwischen den Gleichheitszeichen (=) und dem kaufmännischen Und (&). In diesem Beispiel lautet die Produkt-ID `prod-example123456`.

```
https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
productCreated=prod-example123456&createdProductTitle=my-product
```

> **Note**
>
> Kopieren Sie die URL für Ihr Produkt, bevor Sie die Seite verlassen. Nachdem Sie diese Seite verlassen haben, müssen Sie die CLI verwenden, um Ihre Produkt-ID abzurufen.

Nach einigen Sekunden wird das Produkt auf der Seite Products (Produkte) angezeigt. Möglicherweise müssen Sie Ihren Browser aktualisieren, damit das Produkt in der Liste angezeigt wird.

Schritt 3: Erstellen Ihrer Pipeline

1. Führen Sie die folgenden Schritte aus, um Ihre Pipeline zu benennen und Parameter für diese auszuwählen:

   a. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   b. Wählen Sie Getting started (Erste Schritte). Wählen Sie Create pipeline (Pipeline erstellen) und geben Sie dann einen Namen für Ihre Pipeline ein.

   c. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.

   d. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.

   e. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.

2. Führen Sie die folgenden Schritte aus, um eine Quellphase hinzuzufügen:

   a. Wählen Sie unter Source provider (Quell-Anbieter) die Option AWS CodeCommit.

b. Geben Sie unter Repository name (Repository-Name) und Branch name (Name der Verzweigung) das Repository und die Verzweigung ein, die für Ihre Quellaktion verwendet werden sollen.

c. Wählen Sie Weiter aus.

3. Wählen Sie unter Add build stage (Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken.

4. Führen Sie unter Add deploy stage (Bereitstellungsphase hinzufügen) die folgenden Schritte aus:

a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option AWS Service Catalog aus.

b. Wählen Sie für die Bereitstellungskonfiguration Enter deployment configuration (Bereitstellungskonfiguration eingeben).

c. Fügen Sie unter Produkt-ID die Produkt-ID ein, die Sie aus der Service Catalog-Konsole kopiert haben.

d. Geben Sie unter Template file path (Pfad zur Vorlagendatei) den relativen Pfad ein, unter dem die Vorlagendatei gespeichert ist.

e. Wählen Sie unter Produkttyp die AWS CloudFormation Vorlage aus.

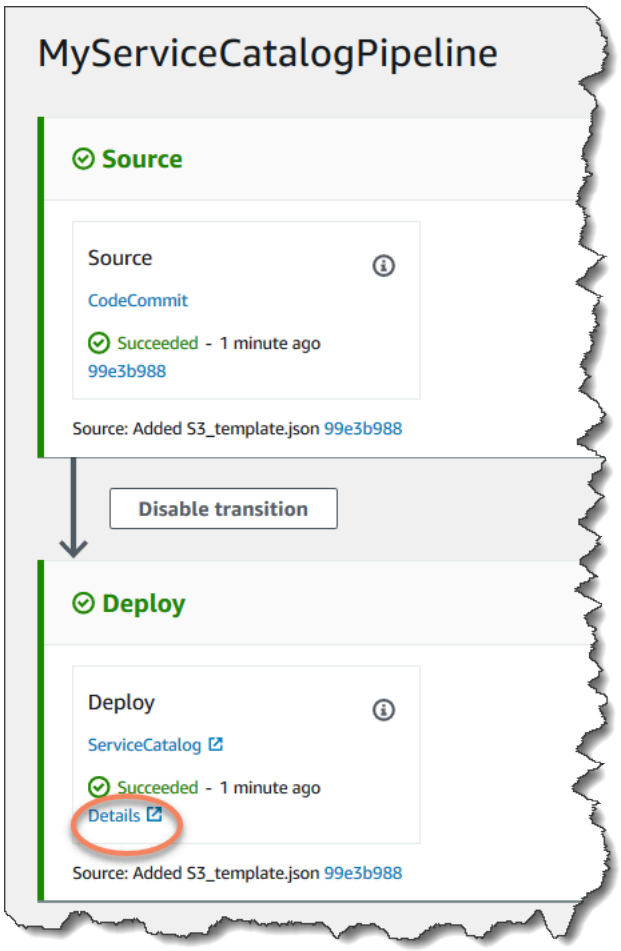
f. Geben Sie im Feld Produktversionsname den Namen der Produktversion ein, die Sie im Service Catalog angegeben haben. Wenn die Vorlagenänderung in einer neuen Produktversion bereitgestellt werden soll, geben Sie einen Namen für die Produktversion ein, der noch nicht für eine vorherige Produktversion im selben Produkt verwendet wurde.

g. Wählen Sie für Input artifact (Eingabeartefakt) das Quell-Eingabeartefakt.

h. Wählen Sie Weiter aus.

5. Prüfen Sie Ihre Pipeline-Einstellungen unter Review (Überprüfen) und wählen Sie dann Create (Erstellen).

6. Wenn Ihre Pipeline in der Bereitstellungsphase erfolgreich ausgeführt wird, wählen Sie Details. Dadurch wird Ihr Produkt im Service Catalog geöffnet.



7. Wählen Sie den Namen Ihrer Version in den Produktinformationen, um die Produktvorlage zu öffnen. Zeigen Sie die Vorlagenbereitstellung an.

Schritt 4: Führen Sie eine Änderung durch und verifizieren Sie Ihr Produkt im Service Catalog

1. Sehen Sie sich Ihre Pipeline in der CodePipeline Konsole an und wählen Sie in Ihrer Quellphase Details aus. Ihr AWS CodeCommit Quell-Repository wird in der Konsole geöffnet. Wählen Sie Edit (Bearbeiten) und nehmen Sie eine Änderung an der Datei vor (z. B. an der Beschreibung).

   ```
   "Description": "Name of Amazon S3 bucket to hold and version website content"
   ```

2. Führen Sie einen Commit für die Änderung durch und übertragen Sie diese. Ihre Pipeline startet, nach der Übertragung der Änderung. Wenn die Ausführung der Pipeline abgeschlossen ist, wählen Sie in der Bereitstellungsphase Details aus, um Ihr Produkt im Service Catalog zu öffnen.

3. Wählen Sie den neuen Namen Ihrer Version in den Produktinformationen, um die Produktvorlage zu öffnen. Zeigen Sie die Änderung an der bereitgestellten Vorlage an.

Option 2: Bereitstellung in Service Catalog mithilfe einer Konfigurationsdatei

In diesem Beispiel laden Sie die AWS CloudFormation Beispielvorlagendatei für einen S3-Bucket hoch und erstellen dann Ihr Produkt im Service Catalog. Sie können auch eine separate Konfigurationsdatei hochladen, in der Ihre Bereitstellungskonfiguration angegeben ist. Als Nächstes erstellen Sie Ihre Pipeline und geben den Speicherort Ihrer Konfigurationsdatei an.

Schritt 1: Hochladen der Beispielvorlagendatei in das Quellrepository

1. Öffnen Sie einen Texteditor. Erstellen Sie eine Beispielvorlage, indem Sie den folgenden Code in die Datei einfügen. Speichern Sie die Datei als `S3_template.json`.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "CloudFormation Sample Template S3_Bucket: Sample template showing
 how to create a privately accessible S3 bucket. **WARNING** This template creates
 an S3 bucket. You will be billed for the resources used if you create a stack from
 this template.",
  "Resources": {
    "S3Bucket": {
      "Type": "AWS::S3::Bucket",
      "Properties": {}
    }
  },
  "Outputs": {
    "BucketName": {
      "Value": {
        "Ref": "S3Bucket"
      },
      "Description": "Name of Amazon S3 bucket to hold website content"
    }
  }
}
```

   Diese Vorlage ermöglicht AWS CloudFormation die Erstellung eines S3-Buckets, der von Service Catalog verwendet werden kann.

2. Laden Sie die Datei `S3_template.json` in Ihr AWS CodeCommit -Repository hoch.

Schritt 2: Erstellen einer Konfigurationsdatei für die Produktbereitstellung

1. Öffnen Sie einen Texteditor. Erstellen Sie die Konfigurationsdatei für Ihr Produkt. Die Konfigurationsdatei wird verwendet, um Ihre Service Catalog-Bereitstellungsparameter/Einstellungen zu definieren. Sie verwenden diese Datei für die Erstellung Ihrer Pipeline.

   In diesem Beispiel wird für `ProductVersionName` "devops S3 v2" und für `ProductVersionDescription` die Option `MyProductVersionDescription` verwendet. Wenn die Vorlagenänderung in einer neuen Produktversion bereitgestellt werden soll, geben Sie einfach einen Namen für die Produktversion ein, der noch nicht für eine vorherige Produktversion im selben Produkt verwendet wurde.

   Speichern Sie die Datei als `sample_config.json`.

   ```
   {
       "SchemaVersion": "1.0",
       "ProductVersionName": "devops S3 v2",
       "ProductVersionDescription": "MyProductVersionDescription",
       "ProductType": "CLOUD_FORMATION_TEMPLATE",
       "Properties": {
           "TemplateFilePath": "/S3_template.json"
       }
   }
   ```

   Diese Datei erstellt bei jeder Ausführung der Pipeline die Informationen zur Produktversion.

2. Laden Sie die Datei `sample_config.json` in Ihr AWS CodeCommit -Repository hoch. Stellen Sie sicher, dass Sie diese Datei in Ihr Quellrepository hochladen.

Schritt 3: Erstellen Sie ein Produkt im Service Catalog

1. Melden Sie sich als IT-Administrator bei der Service Catalog-Konsole an, wechseln Sie zur Produktseite und wählen Sie dann Neues Produkt hochladen aus.
2. Führen Sie auf der Seite Upload new product (Neues Produkt hochladen) die folgenden Schritte aus:

   a. Geben Sie unter Product name (Produktname) den gewünschten Namen für Ihr neues Produkt ein.

b. Geben Sie unter Description (Beschreibung) die Beschreibung des Produktkatalogs an. Diese Beschreibung wird in der Produktliste angezeigt, um den Benutzern die Auswahl des richtigen Produkts zu erleichtern.

c. Geben Sie unter Provided by (Bereitgestellt von) den Namen Ihrer IT-Abteilung oder des Administrators ein.

d. Wählen Sie Weiter aus.

3. (Optional) Geben Sie unter Enter support details (Supportdetails eingeben) die Kontaktinformationen zum Produktsupport ein und klicken Sie dann auf Next (Weiter).

4. Führen Sie unter Version details (Versionsdetails) die folgenden Schritte aus:

a. Wählen Sie Upload a template file (Vorlagendatei hochladen). Suchen Sie die Datei `S3_template.json` und laden Sie sie hoch.

b. Geben Sie unter Version title (Versionstitel) den Namen der Produktversion ein (z. B. "devops S3 v2").

c. Geben Sie unter Description (Beschreibung) die Details ein, die diese Version von anderen Versionen unterscheidet.

d. Wählen Sie Weiter aus.

5. Überprüfen Sie die Informationen auf der Seite Review (Überprüfen) auf Richtigkeit und klicken Sie dann auf Confirm and upload (Bestätigen und hochladen).

6. Kopieren Sie die URL Ihres neuen Produkts im Browser auf der Seite Products (Produkte). Diese enthält die Produkt-ID. Kopieren Sie diese Produkt-ID und bewahren Sie sie auf. Sie verwenden, wenn Sie Ihre Pipeline in erstellen CodePipeline.

   Hier sehen Sie die URL für ein Produkt mit dem Namen `my-product`. Um die Produkt-ID zu extrahieren, kopieren Sie den Wert zwischen den Gleichheitszeichen (=) und dem kaufmännischen Und (&). In diesem Beispiel lautet die Produkt-ID `prod-example123456`.

```
https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
productCreated=prod-example123456&createdProductTitle=my-product
```

> **Note**
>
> Kopieren Sie die URL für Ihr Produkt, bevor Sie die Seite verlassen. Nachdem Sie diese Seite verlassen haben, müssen Sie die CLI verwenden, um Ihre Produkt-ID abzurufen.

Nach einigen Sekunden wird das Produkt auf der Seite Products (Produkte) angezeigt. Möglicherweise müssen Sie Ihren Browser aktualisieren, damit das Produkt in der Liste angezeigt wird.

Schritt 4: Erstellen Ihrer Pipeline

1. Führen Sie die folgenden Schritte aus, um Ihre Pipeline zu benennen und Parameter für diese auszuwählen:

   a. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   b. Wählen Sie Getting started (Erste Schritte). Wählen Sie Create pipeline (Pipeline erstellen) und geben Sie dann einen Namen für Ihre Pipeline ein.

   c. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.

   d. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.

2. Führen Sie die folgenden Schritte aus, um eine Quellphase hinzuzufügen:

   a. Wählen Sie unter Source provider (Quell-Anbieter) die Option AWS CodeCommit.

   b. Geben Sie unter Repository name (Repository-Name) und Branch name (Name der Verzweigung) das Repository und die Verzweigung ein, die für Ihre Quellaktion verwendet werden sollen.

   c. Wählen Sie Weiter aus.

3. Wählen Sie unter Add build stage (Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken.

4. Führen Sie unter Add deploy stage (Bereitstellungsphase hinzufügen) die folgenden Schritte aus:

   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option AWS Service Catalog aus.

   b. Wählen Sie Use configuration file (Konfigurationsdatei verwenden).

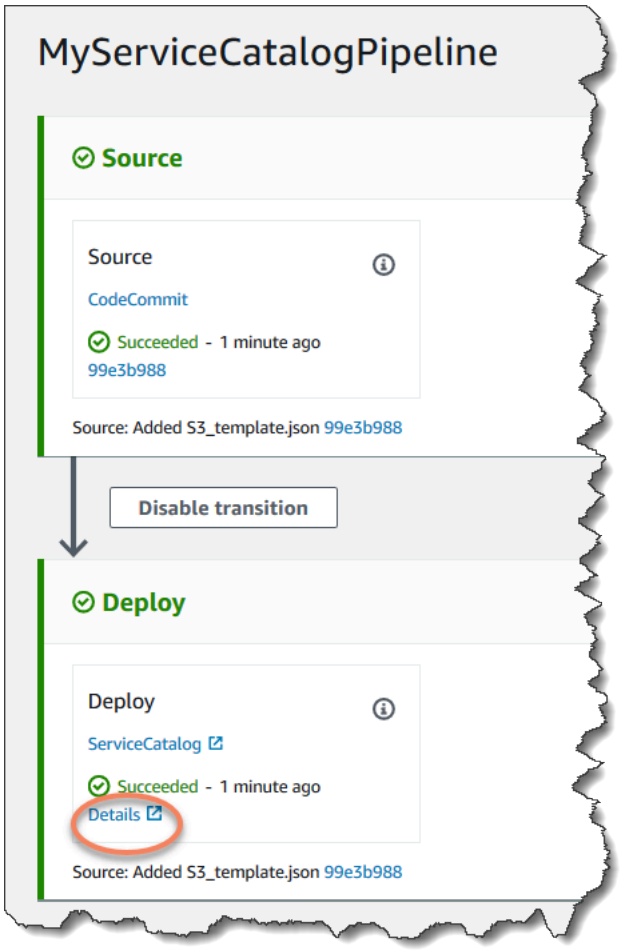
   c. Fügen Sie unter Produkt-ID die Produkt-ID ein, die Sie aus der Service Catalog-Konsole kopiert haben.

d. Geben Sie unter Configuration file path (Pfad zur Konfigurationsdatei) den Dateipfad der Konfigurationsdatei in Ihrem Repository ein.

e. Wählen Sie Weiter aus.

5. Prüfen Sie Ihre Pipeline-Einstellungen unter Review (Überprüfen) und wählen Sie dann Create (Erstellen).

6. Nachdem Ihre Pipeline erfolgreich ausgeführt wurde, wählen Sie in der Bereitstellungsphase Details aus, um Ihr Produkt im Service Catalog zu öffnen.



7. Wählen Sie den Namen Ihrer Version in den Produktinformationen, um die Produktvorlage zu öffnen. Zeigen Sie die Vorlagenbereitstellung an.

Schritt 5: Übertragen einer Änderung und Überprüfen Ihres Produkts im Service Catalog

1. Sehen Sie sich Ihre Pipeline in der CodePipeline Konsole an und wählen Sie in der Quellphase Details aus. Ihr AWS CodeCommit Quell-Repository wird in der Konsole geöffnet. Wählen

Sie Edit (Bearbeiten) und nehmen Sie dann eine Änderung an der Datei vor (z. B. an der Beschreibung).

```
"Description": "Name of Amazon S3 bucket to hold and version website content"
```

2. Führen Sie einen Commit für die Änderung durch und übertragen Sie diese. Ihre Pipeline startet, nach der Übertragung der Änderung. Wenn die Ausführung der Pipeline abgeschlossen ist, wählen Sie in der Bereitstellungsphase Details aus, um Ihr Produkt im Service Catalog zu öffnen.
3. Wählen Sie den neuen Namen Ihrer Version in den Produktinformationen, um die Produktvorlage zu öffnen. Zeigen Sie die Änderung an der bereitgestellten Vorlage an.

Tutorial: Erstellen Sie eine Pipeline mit AWS CloudFormation

Die Beispiele enthalten Beispielvorlagen, mit denen Sie eine Pipeline erstellen können, die Ihre Anwendung bei jeder Änderung des Quellcodes auf Ihren Instanzen bereitstellt. AWS CloudFormation Die Beispielvorlage erstellt eine Pipeline, die in AWS CodePipeline angezeigt werden kann. Die Pipeline erkennt den Eingang einer gespeicherten Änderung über Amazon CloudWatch Events.

Themen

- Beispiel 1: Erstellen Sie eine AWS CodeCommit Pipeline mit AWS CloudFormation
- Beispiel 2: Erstellen Sie eine Amazon S3 S3-Pipeline mit AWS CloudFormation

Beispiel 1: Erstellen Sie eine AWS CodeCommit Pipeline mit AWS CloudFormation

In dieser exemplarischen Vorgehensweise erfahren Sie, wie Sie mit der AWS CloudFormation Konsole eine Infrastruktur erstellen, die eine Pipeline umfasst, die mit einem CodeCommit Quell-Repository verbunden ist. In diesem Tutorial verwenden Sie die bereitgestellte Beispielvorlagendatei, um Ihren Ressourcenstapel zu erstellen, der Ihren Artefaktspeicher, Ihre Pipeline und Ressourcen zur Änderungserkennung, wie z. B. Ihre Amazon CloudWatch Events-Regel, umfasst. Nachdem Sie Ihren Ressourcen-Stack erstellt haben AWS CloudFormation, können Sie Ihre Pipeline in der Konsole einsehen. AWS CodePipeline Die Pipeline ist eine zweistufige Pipeline mit einer CodeCommit Quellphase und einer CodeDeploy Bereitstellungsphase.

Voraussetzungen:

Sie müssen die folgenden Ressourcen für die Verwendung mit der AWS CloudFormation Beispielvorlage erstellt haben:

- Sie müssen ein Quellrepository erstellt haben. Sie können das AWS CodeCommit Repository verwenden, in dem Sie es erstellt habenTutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository).
- Sie müssen eine CodeDeploy Anwendung und eine Bereitstellungsgruppe erstellt haben. Sie können die CodeDeploy Ressourcen verwenden, in denen Sie sie erstellt habenTutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository).
- Wählen Sie einen dieser Links, um die AWS CloudFormation Beispielvorlagendatei für die Erstellung einer Pipeline herunterzuladen: YAML | JSON

  Entpacken Sie die Datei und speichern Sie sie auf Ihrem lokalen Computer.
- Laden Sie die Beispielanwendungsdatei SampleApp_Linux.zip herunter.

Erstellen Sie Ihre Pipeline in AWS CloudFormation

1. Entpacken Sie die Dateien aus SampleApp_Linux.zip und laden Sie die Dateien in Ihr AWS CodeCommit Repository hoch. Sie müssen die entpackten Dateien in das Stammverzeichnis Ihres Repositorys hochladen. Sie können den Anweisungen unter Schritt 2: Fügen Sie Ihrem CodeCommit Repository Beispielcode hinzu folgen, um die Dateien in Ihr Repository zu übertragen.
2. Öffnen Sie die AWS CloudFormation Konsole und wählen Sie Create Stack. Wählen Sie Mit neuen Ressourcen (Standard).
3. Wählen Sie unter Vorlage angeben die Option Vorlage hochladen aus. Wählen Sie Datei auswählen und wählen Sie dann die Vorlagendatei von Ihrem lokalen Computer aus. Wählen Sie Weiter aus.
4. Geben Sie in das Feld Stack Name (Stack-Name) einen Namen für Ihre Pipeline ein. Die durch die Beispielvorlage angegebenen Parameter werden angezeigt. Legen Sie die folgenden Parameter fest:

   a. Geben Sie unter den Namen Ihrer CodeDeploy Anwendung ein. ApplicationName
   b. Geben Sie BetaFleetunter den Namen Ihrer CodeDeploy Bereitstellungsgruppe ein.
   c. Geben Sie BranchNameunter den Repository-Zweig ein, den Sie verwenden möchten.
   d. Geben Sie RepositoryNameunter den Namen Ihres CodeCommit Quell-Repositorys ein.

5. Wählen Sie Weiter aus. Übernehmen Sie die Standardeinstellungen auf der folgenden Seite und wählen Sie Next (Weiter) aus.
6. Wählen Sie unter Funktionen die Option Ich bestätige, dass AWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden, und wählen Sie dann Stack erstellen aus.
7. Nachdem die Erstellung Ihres Stacks abgeschlossen wurde, zeigen Sie die Ereignisliste an, um zu überprüfen, ob Fehler aufgetreten sind.

   Fehlersuche

   Der IAM-Benutzer, in dem die Pipeline erstellt wird, benötigt AWS CloudFormation möglicherweise zusätzliche Berechtigungen, um Ressourcen für die Pipeline zu erstellen. Die folgenden Berechtigungen sind in der Richtlinie erforderlich, um die erforderlichen Amazon CloudWatch Events-Ressourcen für die CodeCommit Pipeline erstellen AWS CloudFormation zu können:

   ```
   {
        "Effect": "Allow",
        "Action": [
           "events:PutRule",
           "events:PutEvents",
           "events:PutTargets",
           "events:DeleteRule",
           "events:RemoveTargets",
           "events:DescribeRule"
        ],
        "Resource": "resource_ARN"
   }
   ```

8. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.

   > **Note**
   >
   > Um die erstellte Pipeline einzusehen, suchen Sie auf der Registerkarte Ressourcen nach der Spalte Logische ID für Ihren Stack in AWS CloudFormation. Notieren Sie sich den Namen in der Spalte Physikalische ID für die Pipeline. In CodePipeline können Sie die

Pipeline mit derselben physischen ID (Pipeline-Namen) in der Region anzeigen, in der Sie Ihren Stack erstellt haben.

9. Führen Sie in Ihrem Quellrepository einen Commit für eine Änderung durch und übertragen Sie diese. Ihre Ressourcen für die Änderungserkennung übernehmen die Änderung und Ihre Pipeline startet.

Beispiel 2: Erstellen Sie eine Amazon S3 S3-Pipeline mit AWS CloudFormation

Diese exemplarische Vorgehensweise zeigt Ihnen, wie Sie mit der AWS CloudFormation Konsole eine Infrastruktur erstellen, die eine Pipeline umfasst, die mit einem Amazon S3 S3-Quell-Bucket verbunden ist. In diesem Tutorial verwenden Sie die bereitgestellte Beispielvorlagendatei, um Ihren Ressourcenstapel zu erstellen, der Ihren Quell-Bucket, Ihren Artefaktspeicher, Ihre Pipeline und Ressourcen zur Änderungserkennung umfasst, z. B. Ihre Amazon Events-Regel und Ihren Amazon CloudWatch Events-Trail. CloudTrail Nachdem Sie Ihren Ressourcen-Stack erstellt haben AWS CloudFormation, können Sie Ihre Pipeline in der Konsole einsehen. AWS CodePipeline Die Pipeline ist eine zweistufige Pipeline mit einer Amazon S3 S3-Quellphase und einer CodeDeploy Bereitstellungsphase.

Voraussetzungen:

Sie benötigen die folgenden Ressourcen, um sie mit der AWS CloudFormation Beispielvorlage verwenden zu können:

- Sie müssen die Amazon EC2 EC2-Instances erstellt haben, in denen Sie den CodeDeploy Agenten auf den Instances installiert haben. Sie müssen eine CodeDeploy Anwendung und eine Bereitstellungsgruppe erstellt haben. Verwenden Sie Amazon EC2 und die CodeDeploy Ressourcen, in Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository) denen Sie sie erstellt haben.
- Wählen Sie die folgenden Links, um die AWS CloudFormation Beispielvorlagendateien für die Erstellung einer Pipeline mit einer Amazon S3 S3-Quelle herunterzuladen:
  - Laden Sie die Beispielvorlage für Ihre Pipeline herunter: YAML | JSON
  - Laden Sie die Beispielvorlage für Ihren CloudTrail Bucket und Trail herunter: YAML | JSON
  - Entpacken Sie die Dateien und speichern Sie sie auf Ihrem lokalen Computer.
- Laden Sie die Beispielanwendung von SampleApp_Linux.zip herunter.

Speichern Sie die ZIP-Datei auf Ihrem lokalen Computer. Laden Sie die ZIP-Datei hoch, nachdem der Stack erstellt wurde.

Erstellen Sie Ihre Pipeline in AWS CloudFormation

1. Öffnen Sie die AWS CloudFormation Konsole und wählen Sie Create Stack aus. Wählen Sie Mit neuen Ressourcen (Standard).
2. Wählen Sie unter Vorlage auswählen die Option Vorlage hochladen aus. Wählen Sie Datei auswählen und wählen Sie dann die Vorlagendatei von Ihrem lokalen Computer aus. Wählen Sie Weiter aus.
3. Geben Sie in das Feld Stack Name (Stack-Name) einen Namen für Ihre Pipeline ein. Die durch die Beispielvorlage angegebenen Parameter werden angezeigt. Legen Sie die folgenden Parameter fest:

   a. Geben Sie unter den Namen Ihrer CodeDeploy Anwendung ein. ApplicationName Sie können den `DemoApplication`-Standardnamen ersetzen.
   b. Geben Sie BetaFleetunter den Namen Ihrer CodeDeploy Bereitstellungsgruppe ein. Sie können den `DemoFleet`-Standardnamen ersetzen.
   c. Geben SourceObjectKeySie ein`SampleApp_Linux.zip`. Laden Sie diese Datei in Ihren Bucket hoch, nachdem die Vorlage den Bucket und die Pipeline erstellt hat.
4. Wählen Sie Weiter aus. Übernehmen Sie die Standardeinstellungen auf der folgenden Seite und wählen Sie Next (Weiter) aus.
5. Wählen Sie unter Funktionen die Option Ich bestätige, dass AWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden, und wählen Sie dann Stack erstellen aus.
6. Nachdem die Erstellung Ihres Stacks abgeschlossen wurde, zeigen Sie die Ereignisliste an, um zu überprüfen, ob Fehler aufgetreten sind.

   Fehlersuche

   Der IAM-Benutzer, in dem die Pipeline erstellt wird, benötigt AWS CloudFormation möglicherweise zusätzliche Berechtigungen, um Ressourcen für die Pipeline zu erstellen. Die folgenden Berechtigungen sind in der Richtlinie erforderlich, um die erforderlichen Amazon CloudWatch Events-Ressourcen für die Amazon S3 S3-Pipeline erstellen zu können AWS CloudFormation :

   ```
   {
   ```

```
      "Effect": "Allow",
      "Action": [
         "events:PutRule",
         "events:PutEvents",
         "events:PutTargets",
         "events:DeleteRule",
         "events:RemoveTargets",
         "events:DescribeRule"
      ],
      "Resource": "resource_ARN"
}
```

7. Sehen Sie sich AWS CloudFormation auf der Registerkarte Ressourcen für Ihren Stack die Ressourcen an, die für Ihren Stack erstellt wurden.

> **Note**
>
> Um die Pipeline zu sehen, die erstellt wurde, suchen Sie in der Spalte Logische ID auf dem Tab Ressourcen für Ihren Stack AWS CloudFormation. Notieren Sie sich den Namen in der Spalte Physikalische ID für die Pipeline. In CodePipeline können Sie die Pipeline mit derselben physischen ID (Pipeline-Namen) in der Region anzeigen, in der Sie Ihren Stack erstellt haben.

Wählen Sie den S3-Bucket mit einer `sourcebucket`-Beschriftung im Namen aus, z. B. `s3-cfn-codepipeline-sourcebucket-y04EXAMPLE.`. Wählen Sie nicht den Pipeline-Artefakt-Bucket aus.

Der Quell-Bucket ist leer, da die Ressource von AWS CloudFormation neu erstellt wird. Öffnen Sie die Amazon S3 S3-Konsole und suchen Sie Ihren `sourcebucket` Bucket. Wählen Sie Upload (Hochladen) aus, und folgen Sie den Anweisungen, um Ihre `SampleApp_Linux.zip`-ZIP-Datei hochzuladen.

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, müssen Sie alle Quelldateien, die als eine einzige ZIP-Datei verpackt sind, in Ihren Bucket hochladen. Andernfalls schlägt die Quellaktion fehl.

8. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.

9. Führen Sie die Schritte im folgenden Verfahren aus, um Ihre AWS CloudTrail -Ressourcen zu erstellen.

Erstellen Sie Ihre AWS CloudTrail Ressourcen in AWS CloudFormation

1. Öffnen Sie die AWS CloudFormation Konsole und wählen Sie Create Stack aus.
2. Wählen Sie unter Choose a template (Auswahl einer Vorlage) die Option Upload a template to Amazon S3 (Vorlage auf Amazon S3 hochladen). Wählen Sie Durchsuchen und wählen Sie dann die Vorlagendatei für die AWS CloudTrail Ressourcen auf Ihrem lokalen Computer aus. Wählen Sie Weiter aus.
3. Geben Sie unter Stack name (Stack-Name) einen Namen für Ihren Ressourcenstapel ein. Die durch die Beispielvorlage angegebenen Parameter werden angezeigt. Legen Sie die folgenden Parameter fest:

   - SourceObjectKeyAkzeptieren Sie unter die Standardeinstellung für die ZIP-Datei der Beispielanwendung.

4. Wählen Sie Weiter aus. Übernehmen Sie die Standardeinstellungen auf der folgenden Seite und wählen Sie Next (Weiter) aus.
5. Wählen Sie unter Funktionen die Option Ich bestätige, dass AWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden, und wählen Sie dann Erstellen aus.
6. Nachdem die Erstellung Ihres Stacks abgeschlossen wurde, zeigen Sie die Ereignisliste an, um zu überprüfen, ob Fehler aufgetreten sind.

   Die folgenden Berechtigungen sind in der Richtlinie erforderlich, um die erforderlichen CloudTrail Ressourcen für die Amazon S3 S3-Pipeline erstellen AWS CloudFormation zu können:

```
{
     "Effect": "Allow",
     "Action": [
        "cloudtrail:CreateTrail",
        "cloudtrail:DeleteTrail",
        "cloudtrail:StartLogging",
```

```
            "cloudtrail:StopLogging",
            "cloudtrail:PutEventSelectors"
        ],
        "Resource": "resource_ARN"
}
```

7. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.

8. Führen Sie in Ihrem Quell-Bucket einen Commit für eine Änderung durch und übertragen Sie diese. Ihre Ressourcen für die Änderungserkennung übernehmen die Änderung und Ihre Pipeline startet.

Tutorial: Eine Pipeline erstellen, die Variablen aus AWS CloudFormation Bereitstellungsaktionen verwendet

In diesem Tutorial verwenden Sie die AWS CodePipeline Konsole, um eine Pipeline mit einer Bereitstellungsaktion zu erstellen. Wenn die Pipeline ausgeführt wird, erstellt die Vorlage einen Stack und auch eine `outputs`-Datei. Die von der Stack-Vorlage generierten Ausgaben sind die Variablen, die durch die AWS CloudFormation Aktion in generiert wurden CodePipeline.

In der Aktion, in der Sie den Stack aus der Vorlage erstellen, legen Sie einen Variablen-Namespace fest. Die Variablen, die von der `outputs`-Datei erzeugt werden, können dann durch nachfolgende Aktionen verwendet werden. In diesem Beispiel erstellen Sie einen Änderungssatz auf der Grundlage der durch die AWS CloudFormation Aktion erzeugten `StackName` Variablen. Nach einer manuellen Genehmigung führen Sie den Änderungssatz aus und erstellen dann eine Stack-Löschaktion, die den Stack basierend auf der `StackName`-Variablen löscht.

Themen

- Voraussetzungen: Erstellen Sie eine AWS CloudFormation Servicerolle und ein CodeCommit Repository
- Schritt 1: Laden Sie die Beispielvorlage herunter, bearbeiten Sie sie und laden Sie sie hoch AWS CloudFormation
- Schritt 2: Erstellen der Pipeline

- Schritt 3: Fügen Sie eine AWS CloudFormation Bereitstellungsaktion hinzu, um den Änderungssatz zu erstellen
- Schritt 4: Hinzufügen einer manuellen Genehmigungsaktion
- Schritt 5: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Änderungssatz auszuführen
- Schritt 6: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Stack zu löschen

Voraussetzungen: Erstellen Sie eine AWS CloudFormation Servicerolle und ein CodeCommit Repository

Sie müssen bereits über Folgendes verfügen:

- Ein CodeCommit Repository. Sie können das AWS CodeCommit Repository verwenden, in dem Sie es erstellt habenTutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository).
- In diesem Beispiel wird ein Amazon DocumentDB DocumentDB-Stack aus einer Vorlage erstellt. Sie müssen AWS Identity and Access Management (IAM) verwenden, um eine AWS CloudFormation Servicerolle mit den folgenden Berechtigungen für Amazon DocumentDB zu erstellen.

```
"rds:DescribeDBClusters",
"rds:CreateDBCluster",
"rds:DeleteDBCluster",
"rds:CreateDBInstance"
```

Schritt 1: Laden Sie die Beispielvorlage herunter, bearbeiten Sie sie und laden Sie sie hoch AWS CloudFormation

Laden Sie die AWS CloudFormation Beispielvorlagendatei herunter und laden Sie sie in Ihr CodeCommit Repository hoch.

1. Navigieren Sie zur Beispielvorlagenseite für Ihre Region. Die Seite für us-west-2 befindet sich beispielsweise unter https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/sample-templates-services-us-west-2.html. Laden Sie unter Amazon DocumentDB die Vorlage für einen Amazon DocumentDB-Cluster herunter. Der Dateiname lautet `documentdb_full_stack.yaml`.

2. Entpacken Sie die `documentdb_full_stack.yaml`-Datei, und öffnen Sie sie in einem Texteditor. Nehmen Sie die folgenden Änderungen vor.

   a. Fügen Sie in diesem Beispiel den folgenden `Purpose:`-Parameter zu Ihrem `Parameters`-Abschnitt in der Vorlage hinzu.

   ```
   Purpose:
     Type: String
     Default: testing
     AllowedValues:
       - testing
       - production
     Description: The purpose of this instance.
   ```

   b. Fügen Sie in diesem Beispiel die folgende `StackName`-Ausgabe zu Ihrem `Outputs:`-Abschnitt in der Vorlage hinzu.

   ```
   StackName:
     Value: !Ref AWS::StackName
   ```

3. Laden Sie die Vorlagendatei in Ihr AWS CodeCommit Repository hoch. Sie müssen die entpackte und bearbeitete Vorlagendatei in das Stammverzeichnis Ihres Repositorys hochladen.

   So verwenden Sie die CodeCommit Konsole, um Ihre Dateien hochzuladen:

   a. Öffnen Sie die CodeCommit Konsole und wählen Sie Ihr Repository aus der Repository-Liste aus.

   b. Wählen Sie Add file (Datei hinzufügen) und dann Upload file (Datei hochladen) aus.

   c. Wählen Sie Choose file (Datei auswählen) und navigieren Sie dann zu Ihrer Datei. Übernehmen Sie die Änderung, indem Sie Ihren Benutzernamen und Ihre E-Mail-Adresse eingeben. Wählen Sie Commit changes (Änderungen übernehmen) aus.

   Ihre Datei sollte auf der Stammebene in Ihrem Repository wie folgt aussehen:

   ```
   documentdb_full_stack.yaml
   ```

Schritt 2: Erstellen der Pipeline

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellstufe mit einer CodeCommit Aktion, bei der das Quellartefakt Ihre Vorlagendatei ist.
- Eine Bereitstellungsphase mit einer AWS CloudFormation Bereitstellungsaktion.

Jeder Aktion in der Quell- und Bereitstellungsstufe, die vom Assistenten erstellt wurde, wird jeweils ein Variablen-Namespace, `SourceVariables`, und `DeployVariables` zugewiesen. Da den Aktionen ein Namespace zugewiesen ist, stehen die in diesem Beispiel konfigurierten Variablen für nachgelagerte Aktionen zur Verfügung. Weitere Informationen finden Sie unter Variablen.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyCFNDeployPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:
   - Wählen Sie Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
   - Wählen Sie Existing service role (Vorhandene Servicerolle). Wählen Sie unter Name der Rolle in der Liste Ihre Servicerolle aus.
6. Unter Artifact store (Artefaktspeicher):
   a. Wählen Sie Standardstandort, um den Standard-Artefaktspeicher, z. B. den Amazon S3 S3-Artefakt-Bucket, der als Standard festgelegt wurde, für Ihre Pipeline in der Region zu verwenden, die Sie für Ihre Pipeline ausgewählt haben.
   b. Wählen Sie Benutzerdefinierter Standort, wenn Sie bereits einen Artefaktspeicher, z. B. einen Amazon S3 S3-Artefakt-Bucket, in derselben Region wie Ihre Pipeline haben.

> **Note**
>
> Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket. Wenn Sie eine Pipeline erstellen oder bearbeiten, benötigen Sie einen Artefakt-Bucket in der Pipeline-Region und einen Artefakt-Bucket pro AWS Region, in der Sie eine Aktion ausführen.
> Weitere Informationen finden Sie unter Eingabe- und Ausgabe-Artefakte und CodePipeline Referenz zur Pipeline-Struktur.

Wählen Sie Weiter aus.

7. Unter Step 2: Add source stage (Schritt 2: Quell-Stufe hinzufügen):

   a. Wählen Sie unter Source provider (Quell-Anbieter) die Option AWS CodeCommit.

   b. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, in dem Sie es erstellt haben. Schritt 1: Erstellen Sie ein CodeCommit Repository

   c. Wählen Sie in Branch name den Namen des Branch aus, der das neueste Code-Update enthält.

   Nachdem Sie den Repository-Namen und den Branch ausgewählt haben, wird die Amazon CloudWatch Events-Regel angezeigt, die für diese Pipeline erstellt werden soll.

   Wählen Sie Weiter aus.

8. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken.

   Wählen Sie Weiter aus.

9. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

   a. Wählen Sie unter Action name (Aktionsname) die Option Deploy (Bereitstellen). Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option CloudFormation aus.

   b. Wählen Sie unter Action mode (Aktionsmodus) Create or update a stack (Einen Stack erstellen oder aktualisieren) aus.

c. Geben Sie unter Stack name (Stack-Name) einen Namen für den Stapel ein. Dies ist der Name des Stacks, den die Vorlage erstellt.

d. Geben Sie unter Output file name (Ausgabedateiname) einen Namen für die Ausgabedatei ein, z. B. **`outputs`**. Dies ist der Name der Datei, die durch die Aktion erstellt wird, nachdem der Stack erstellt wurde.

e. Erweitern Sie Advanced (Erweitert). Geben Sie unter Parameter overrides (Parameterüberschreibungen) Ihre Vorlagenüberschreibungen als Schlüssel-Wert-Paare ein. Diese Vorlage erfordert beispielsweise die folgenden Überschreibungen.

```
{
"DBClusterName": "MyDBCluster",
"DBInstanceName": "MyDBInstance",
"MasterUser": "UserName",
"MasterPassword": "Password",
"DBInstanceClass": "db.r4.large",
"Purpose": "testing"}
```

Wenn Sie keine Überschreibungen eingeben, erstellt die Vorlage einen Stack mit Standardwerten.

f. Wählen Sie Weiter aus.

g. Wählen Sie Create pipeline (Pipeline erstellen) aus. Lassen Sie die Ausführung Ihrer Pipeline zu. Ihre zweistufige Pipeline ist damit vollständig und bereit für die zusätzlichen Stufen , die hinzugefügt werden sollen.

Schritt 3: Fügen Sie eine AWS CloudFormation Bereitstellungsaktion hinzu, um den Änderungssatz zu erstellen

Erstellen Sie eine nächste Aktion in Ihrer Pipeline, mit der Sie AWS CloudFormation den Änderungssatz vor der manuellen Genehmigungsaktion erstellen können.

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.

2. Wählen Sie, ob Sie die Pipeline bearbeiten möchten, oder fahren Sie mit der Anzeige der Pipeline im Modus Edit (Bearbeiten) fort.

3. Wählen Sie, ob Sie die Bereitstellungsphase bearbeiten möchten.
4. Fügen Sie eine Bereitstellungsaktion hinzu, die einen Änderungssatz für den Stack erstellt, der in der vorherigen Aktion erstellt wurde. Sie fügen diese Aktion nach der vorhandenen Aktion in der Phase hinzu.

   a. Geben Sie unter Action name (Aktionsname) Change_Set ein. Wählen Sie unter Aktionsanbieter die Option aus AWS CloudFormation .

   b. Wählen Sie unter Eingabeartefakt die Option aus SourceArtifact.

   c. Wählen Sie unter Action mode (Aktionsmodus) die Option Create or replace a change set (Einen Änderungssatz erstellen oder ersetzen) aus.

   d. Geben Sie unter Stack name (Stack-Name) die Variablensyntax ein, wie hier gezeigt. Dies ist der Name des Stacks, für den der Änderungssatz erstellt wird, wobei der Standard-Namespace `DeployVariables` der Aktion zugewiesen wird.

   ```
   #{DeployVariables.StackName}
   ```

   e. Geben Sie unter Change set name (Name des Änderungssatzes) den Namen des Änderungssatzes ein.

   ```
   my-changeset
   ```

   f. Ändern Sie unter Parameter overrides (Parameterüberschreibungen) den `Purpose`-Parameter von `testing` zu `production`.

   ```
   {
   "DBClusterName": "MyDBCluster",
   "DBInstanceName": "MyDBInstance",
   "MasterUser": "UserName",
   "MasterPassword": "Password",
   "DBInstanceClass": "db.r4.large",
   "Purpose": "production"}
   ```

   g. Wählen Sie Done (Fertig), um die Aktion zu speichern.

Schritt 4: Hinzufügen einer manuellen Genehmigungsaktion

Erstellen Sie eine manuelle Genehmigungsaktion in Ihrer Pipeline.

1. Wählen Sie, ob Sie die Pipeline bearbeiten möchten, oder fahren Sie mit der Anzeige der Pipeline im Modus Edit (Bearbeiten) fort.
2. Wählen Sie, ob Sie die Bereitstellungsphase bearbeiten möchten.
3. Fügen Sie nach der Bereitstellungsaktion, die den Änderungssatz erstellt, eine manuelle Genehmigungsaktion hinzu. Mit dieser Aktion können Sie den erstellten Ressourcen-Änderungssatz überprüfen, AWS CloudFormation bevor die Pipeline den Änderungssatz ausführt.

Schritt 5: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Änderungssatz auszuführen

Erstellen Sie eine nächste Aktion in Ihrer Pipeline, mit der Sie den Änderungssatz nach der manuellen Genehmigungsaktion ausführen können AWS CloudFormation .

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
2. Wählen Sie, ob Sie die Pipeline bearbeiten möchten, oder fahren Sie mit der Anzeige der Pipeline im Modus Edit (Bearbeiten) fort.
3. Wählen Sie, ob Sie die Bereitstellungsphase bearbeiten möchten.
4. Fügen Sie eine Bereitstellungsaktion hinzu, mit der der Änderungssatz ausgeführt wird, der in der vorherigen manuellen Aktion genehmigt wurde:

   a. Geben Sie unter Action name (Aktionsname) Execute_Change_Set ein. Wählen Sie unter Aktionsanbieter die Option aus AWS CloudFormation.

   b. Wählen Sie unter Eingabeartefakt die Option aus SourceArtifact.

   c. Wählen Sie unter Action mode (Aktionsmodus) die Option Execute a change set (Einen Änderungssatz ausführen).

   d. Geben Sie unter Stack name (Stack-Name) die Variablensyntax ein, wie hier gezeigt. Dies ist der Name des Stacks, für den der Änderungssatz erstellt wird.

   ```
   #{DeployVariables.StackName}
   ```

e. Geben Sie unter Change set name (Name des Änderungssatzes) den Namen des Änderungssatzes ein, den Sie in der vorherigen Aktion erstellt haben.

```
my-changeset
```

f. Wählen Sie Done (Fertig), um die Aktion zu speichern.

g. Fahren Sie mit der Pipeline-Ausführung fort.

Schritt 6: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Stack zu löschen

Erstellen Sie eine letzte Aktion in Ihrer Pipeline, mit der Sie AWS CloudFormation den Stack-Namen aus der Variablen in der Ausgabedatei abrufen und den Stack löschen können.

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
2. Wählen Sie diese Option, um die Pipeline zu bearbeiten.
3. Wählen Sie, ob Sie die Bereitstellungsphase bearbeiten möchten.
4. Fügen Sie eine Bereitstellungsaktion hinzu, die den Stack löscht:

   a. Wählen Sie im Feld Aktionsname die Option DeleteStack. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option CloudFormation aus.

   b. Wählen Sie unter Action mode (Aktionsmodus) die Option Delete a Stack (Stack löschen) aus.

   c. Geben Sie unter Stack name (Stack-Name) die Variablensyntax ein, wie hier gezeigt. Dies ist der Name des Stacks, den die Aktion löschen wird.

   d. Wählen Sie Done (Fertig), um die Aktion zu speichern.

   e. Wählen Sie Save (Speichern), um die Pipeline zu speichern.

   Die Pipeline wird ausgeführt, wenn sie gespeichert wird.

Tutorial: Amazon ECS-Standardbereitstellung mit CodePipeline

Dieses Tutorial hilft Ihnen dabei, eine vollständige, end-to-end kontinuierliche Bereitstellungspipeline (CD) mit Amazon ECS mit zu erstellen CodePipeline.

> **Note**
>
> Dieses Tutorial bezieht sich auf die Amazon ECS-Standardbereitstellungsaktion für CodePipeline. Ein Tutorial, das die Aktion Amazon ECS to CodeDeploy Blue/Green Deployment in verwendet CodePipeline, finden Sie unterTutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy .

Voraussetzungen

Es müssen bestimmte Ressourcen zur Verfügung stehen, damit Sie mithilfe dieses Tutorials eine CD-Pipeline erstellen können. Zum Einstieg benötigen Sie Folgendes:

> **Note**
>
> All diese Ressourcen sollten in derselben AWS Region erstellt werden.

- Ein Quellcodeverwaltungs-Repository (das in diesem Tutorial verwendet wird CodeCommit) mit Ihrem Dockerfile und Ihrer Anwendungsquelle. Weitere Informationen finden Sie unter Erstellen eines CodeCommit Repositorys im AWS CodeCommit Benutzerhandbuch.
- Ein Docker-Image-Repository (dieses Tutorial verwendet Amazon ECR), das ein Image enthält, das Sie aus Ihrer Docker-Datei und Anwendungsquelle erstellt haben. Weitere Informationen finden Sie unter Creating a Repository and Push an Image im Amazon Elastic Container Registry User Guide.
- Eine Amazon ECS-Aufgabendefinition, die auf das in Ihrem Image-Repository gehostete Docker-Image verweist. Weitere Informationen finden Sie unter Erstellen einer Aufgabendefinition im Amazon Elastic Container Service Developer Guide.

> **Important**
>
> Die Amazon ECS-Standardbereitstellungsaktion für CodePipeline erstellt eine eigene Revision der Aufgabendefinition, die auf der vom Amazon ECS-Service verwendeten

Version basiert. Wenn Sie neue Revisionen für die Aufgabendefinition erstellen, ohne den Amazon ECS-Service zu aktualisieren, ignoriert die Bereitstellungsaktion diese Revisionen.

Im Folgenden finden Sie ein Beispiel für eine Aufgabendefinition, die für dieses Tutorial verwendet wurde. Der Wert, den Sie für Ihre Build-Spezifikationsdatei verwenden `name` und der im nächsten Schritt verwendet `family` wird.

```
{
  "ipcMode": null,
  "executionRoleArn": "role_ARN",
  "containerDefinitions": [
    {
      "dnsSearchDomains": null,
      "environmentFiles": null,
      "logConfiguration": {
        "logDriver": "awslogs",
        "secretOptions": null,
        "options": {
          "awslogs-group": "/ecs/hello-world",
          "awslogs-region": "us-west-2",
          "awslogs-stream-prefix": "ecs"
        }
      },
      "entryPoint": null,
      "portMappings": [
        {
          "hostPort": 80,
          "protocol": "tcp",
          "containerPort": 80
        }
      ],
      "command": null,
      "linuxParameters": null,
      "cpu": 0,
      "environment": [],
      "resourceRequirements": null,
      "ulimits": null,
      "dnsServers": null,
      "mountPoints": [],
      "workingDirectory": null,
      "secrets": null,
```

```
      "dockerSecurityOptions": null,
      "memory": null,
      "memoryReservation": 128,
      "volumesFrom": [],
      "stopTimeout": null,
      "image": "image_name",
      "startTimeout": null,
      "firelensConfiguration": null,
      "dependsOn": null,
      "disableNetworking": null,
      "interactive": null,
      "healthCheck": null,
      "essential": true,
      "links": null,
      "hostname": null,
      "extraHosts": null,
      "pseudoTerminal": null,
      "user": null,
      "readonlyRootFilesystem": null,
      "dockerLabels": null,
      "systemControls": null,
      "privileged": null,
      "name": "hello-world"
    }
  ],
  "placementConstraints": [],
  "memory": "2048",
  "taskRoleArn": null,
  "compatibilities": [
    "EC2",
    "FARGATE"
  ],
  "taskDefinitionArn": "ARN",
  "family": "hello-world",
  "requiresAttributes": [],
  "pidMode": null,
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "networkMode": "awsvpc",
  "cpu": "1024",
  "revision": 1,
  "status": "ACTIVE",
  "inferenceAccelerators": null,
```

```
  "proxyConfiguration": null,
  "volumes": []
}
```

- Ein Amazon ECS-Cluster, auf dem ein Service ausgeführt wird, der Ihre zuvor erwähnte Aufgabendefinition verwendet. Weitere Informationen finden Sie unter Creating a Cluster and Creating a Service im Amazon Elastic Container Service Developer Guide.

Nachdem Sie diese Voraussetzungen erfüllt haben, können Sie mit dem Tutorial fortfahren und Ihre CD-Pipeline erstellen.

Schritt 1: Hinzufügen einer Build-Spezifikationsdatei zu Ihrem Quell-Repository

In diesem Tutorial CodeBuild können Sie Ihr Docker-Image erstellen und das Image an Amazon ECR übertragen. Fügen Sie Ihrem Quellcode-Repository eine `buildspec.yml` Datei hinzu, um zu erklären, CodeBuild wie das geht. Die nachfolgende Beispiel-Buildspezifikation bewirkt Folgendes:

- Phase vor dem Build:
  - Melden Sie sich bei Amazon ECR an.
  - Stellen Sie die Repository-URI auf Ihr ECR-Image ein und fügen Sie ein Image-Tag mit den ersten sieben Zeichen der Git-Commit-ID der Quelle hinzu.
- Build-Phase:
  - Erstellen Sie das Docker-Image und taggen Sie das Image sowohl als `latest` als auch mit der Git-Commit-ID.
- Phase nach dem Build:
  - Führen Sie die Push-Übertragung des Image zu Ihrem ECR-Repository mit beiden Tags durch.
  - Schreiben Sie eine Datei namens `imagedefinitions.json` Build-Root, die den Container-Namen Ihres Amazon ECS-Service sowie das Image und das Tag enthält. Die Bereitstellungsphase Ihrer CD-Pipeline erstellt anhand dieser Informationen eine neue Revision Ihrer Service-Aufgabendefinition und aktualisiert dann den Service für die Nutzung der neuen Aufgabendefinition. Die Datei `imagedefinitions.json` wird für den ECS-Auftragsworker benötigt.

Fügen Sie diesen Beispieltext ein, um Ihre `buildspec.yml` Datei zu erstellen, und ersetzen Sie die Werte für Ihr Bild und Ihre Aufgabendefinition. In diesem Text wird die Beispielkonto-ID 111122223333 verwendet.

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
      - REPOSITORY_URI=012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"hello-world","imageUri":"%s"}]' $REPOSITORY_URI:$IMAGE_TAG >
 imagedefinitions.json
artifacts:
    files: imagedefinitions.json
```

Die Build-Spezifikation wurde für die Beispielaufgabendefinition geschriebenVoraussetzungen, die in bereitgestellt wurde und vom Amazon ECS-Service für dieses Tutorial verwendet wird. Der `REPOSITORY_URI`-Wert entspricht dem `image`-Repository (ohne Image-Tag), und der *`hello-world`*-Wert am Ende der Datei entspricht dem Container-Namen in der Aufgabendefinition des Service.

So fügen Sie eine **`buildspec.yml`**-Datei zu Ihrem Quell-Repository hinzu

1. Öffnen Sie einen Text-Editor und kopieren Sie die Build-Spezifikation in eine neue Datei.
2. Ersetzen Sie den `REPOSITORY_URI` Wert (*`012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world`*) durch Ihre Amazon ECR-Repository-URI (ohne Image-Tag) für Ihr Docker-Image. Ersetzen Sie *`hello-world`* durch den Container-Namen in der Aufgabendefinition Ihres Service, die auf das Docker-Image verweist.
3. Führen Sie einen Commit und eine Push-Übertragung Ihrer `buildspec.yml`-Datei zu Ihrem Quell-Repository durch.

   a. Fügen Sie die Datei hinzu.

   ```
   git add .
   ```

   b. Führen Sie einen Commit der Änderung durch.

   ```
   git commit -m "Adding build specification."
   ```

   c. Führen Sie eine Push-Übertragung für das Commit durch.

   ```
   git push
   ```

Schritt 2: Erstellen einer durchgängigen Bereitstellungs-Pipeline

Verwenden Sie den CodePipeline Assistenten, um Ihre Pipeline-Stufen zu erstellen und Ihr Quell-Repository mit Ihrem ECS-Service zu verbinden.

So erstellen Sie Ihre Pipeline

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Klicken Sie auf der Seite Welcome auf Create pipeline.

   Wenn Sie dies zum ersten Mal verwenden CodePipeline, wird anstelle von Willkommen eine Einführungsseite angezeigt. Wählen Sie Get Started Now.
3. Geben Sie auf der Seite Schritt 1: Name in das Feld Pipeline-Name den Namen für Ihre Pipeline ein. Bei diesem Tutorial lautet der Pipeline-Name hello-world.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und

im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen. Wählen Sie Weiter.

5. Wählen Sie auf der Seite Schritt 2: Quellstufe hinzufügen für Quellanbieter die Option AWS CodeCommit.

   a. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, das als Quellpfad für Ihre Pipeline verwendet werden soll.

   b. Wählen Sie für Branch name (Zweigname) den zu verwendenden Zweig und dann Next (Weiter) aus.

6. Wählen Sie auf der Seite Schritt 3: Build-Phase hinzufügen die Option AWS CodeBuildBuild Provider und dann Create project aus.

   a. Wählen Sie für Project name einen eindeutigen Namen für Ihr Build-Projekt aus. Bei diesem Tutorial lautet der Projektname hello-world.

   b. Wählen Sie für Environment image (Umgebungs-Abbild) die Option Managed image (Verwaltetes Abbild) aus.

   c. Wählen Sie für Operating system (Betriebssystem) die Option Amazon Linux 2 aus.

   d. Wählen Sie unter Runtime (Laufzeit) die Option Standard aus.

   e. Wählen Sie für Image die Option **`aws/codebuild/amazonlinux2-x86_64-standard:3.0`**.

   f. Verwenden Sie für Image version (Abbildversion) und Environment type (Umgebungstyp) die Standardwerte.

   g. Wählen Sie Enable this flag if you want to build Docker images or want your builds to get elevated privileges (Dieses Flag aktivieren, wenn Docker-Abbilder erstellt oder die Builds erweiterte Berechtigungen erhalten sollen) aus.

   h. Wählen Sie CloudWatch Protokolle ab. Möglicherweise müssen Sie Erweitert erweitern.

   i. Wählen Sie Weiter zu CodePipeline.

   j. Wählen Sie Weiter aus.

   > **Note**
   >
   > Der Assistent erstellt eine CodeBuild Servicerolle mit dem Namen codebuild-***`build-project-name`***-service-role für Ihr Build-Projekt. Notieren Sie sich diesen Rollennamen, wenn Sie ihm später Amazon ECR-Berechtigungen hinzufügen.

7. Wählen Sie auf der Seite Schritt 4: Bereitstellungsphase hinzufügen als Bereitstellungsanbieter Amazon ECS aus.

   a. Wählen Sie als Clustername den Amazon ECS-Cluster aus, in dem Ihr Service ausgeführt wird. Für dieses Tutorial lautet der Cluster default.

   b. Wählen Sie für Service name (Service-Name) den zu aktualisierenden Service und dann Next (Weiter) aus. Bei diesem Tutorial lautet der Servicename hello-world.

8. Überprüfen Sie auf der Seite Step 5: Review (Schritt 5: Überprüfen) die Pipeline-Konfiguration und wählen Sie Create pipeline (Pipeline erstellen), um die Pipeline zu erstellen.

> **Note**
>
> Nachdem die Pipeline erstellt wurde, versucht sie, die verschiedenen Pipeline-Phasen zu durchlaufen. Die vom Assistenten erstellte CodeBuild Standardrolle ist jedoch nicht berechtigt, alle in der `buildspec.yml` Datei enthaltenen Befehle auszuführen, sodass die Erstellungsphase fehlschlägt. Der nächste Abschnitt fügt die Berechtigungen für die Build-Phase hinzu.

Schritt 3: Amazon ECR-Berechtigungen zur CodeBuild Rolle hinzufügen

Der CodePipeline Assistent hat eine IAM-Rolle mit dem Namen CodeBuild codebuild- -service-role für das Build-Projekt erstellt. *`build-project-name`* Für dieses Tutorial lautet der Name -role. codebuild-hello-world-service Da die `buildspec.yml` Datei Amazon ECR-API-Operationen aufruft, muss die Rolle über eine Richtlinie verfügen, die Berechtigungen für diese Amazon ECR-Aufrufe zulässt. Das folgende Verfahren hilft Ihnen, die richtigen Berechtigungen an die Rolle anzufügen.

So fügen Sie Amazon ECR-Berechtigungen zur Rolle hinzu CodeBuild

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im linken Navigationsbereich Roles aus.
3. Geben Sie im Suchfeld codebuild- ein und wählen Sie die Rolle aus, die CodePipeline vom Assistenten erstellt wurde. Für dieses Tutorial lautet der Rollenname codebuild-hello-world-service-role.
4. Wählen Sie auf der Seite Summary (Zusammenfassung) die Option Attach policy (Richtlinie anfügen) aus.

5. Markieren Sie das Feld links neben der ContainerRegistryPowerUserAmazonEC2-Richtlinie und wählen Sie Richtlinie anhängen aus.

Schritt 4: Testen Ihrer Pipeline

Ihre Pipeline sollte über alles verfügen, was für die Ausführung einer end-to-end systemeigenen AWS kontinuierlichen Bereitstellung erforderlich ist. Testen Sie nun ihre Funktionalität, indem Sie eine Code-Änderung mithilfe von Push zu Ihrem Quell-Repository übertragen.

So testen Sie Ihre Pipeline

1. Nehmen Sie eine Code-Änderung an Ihrem konfigurierten Quell-Repository vor und führen Sie den Commit-Vorgang und die Push-Übertragung der Änderung durch.
2. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
3. Wählen Sie Ihre Pipeline in der Liste aus.
4. Sehen Sie sich an, wie die Pipeline ihre Phasen durchläuft. Ihre Pipeline sollte abgeschlossen sein und Ihr Amazon ECS-Service führt das Docker-Image aus, das aus Ihrer Codeänderung erstellt wurde.

Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy

In diesem Tutorial konfigurieren Sie eine Pipeline AWS CodePipeline , in der Containeranwendungen mithilfe einer blauen/grünen Bereitstellung bereitgestellt werden, die Docker-Images unterstützt. In einer Blau/Grün-Bereitstellung können Sie die neue Version Ihrer Anwendung zusammen mit der alten Version starten und die neue Version testen, bevor Sie Datenverkehr umleiten. Sie können auch den Bereitstellungsprozess überwachen und bei Problemen schnell ein Rollback durchführen.

> **Note**
>
> Dieses Tutorial bezieht sich auf die Aktion Amazon ECS to CodeDeploy Blue/Green Deployment für CodePipeline. Ein Tutorial, das die Amazon ECS-Standardbereitstellungsaktion in verwendet CodePipeline, finden Sie unterTutorial: Amazon ECS-Standardbereitstellung mit CodePipeline.

Die abgeschlossene Pipeline erkennt Änderungen an Ihrem Image, das in einem Image-Repository wie Amazon ECR gespeichert ist und verwendet wird, CodeDeploy um den Datenverkehr an einen Amazon ECS-Cluster und einen Load Balancer weiterzuleiten und bereitzustellen. CodeDeploy verwendet einen Listener, um den Datenverkehr an den Port des aktualisierten Containers umzuleiten, der in der Datei angegeben ist. AppSpec Informationen darüber, wie der Load Balancer, der Produktions-Listener, die Zielgruppen und Ihre Amazon ECS-Anwendung in einer blauen/grünen Bereitstellung verwendet werden, finden Sie unter Tutorial: Bereitstellen eines Amazon ECS-Service.

Die Pipeline ist auch so konfiguriert, dass sie einen Quellspeicherort verwendet CodeCommit, z. B. wo Ihre Amazon ECS-Aufgabendefinition gespeichert ist. In diesem Tutorial konfigurieren Sie jede dieser AWS Ressourcen und erstellen dann Ihre Pipeline mit Phasen, die Aktionen für jede Ressource enthalten.

Ihre Continuous Delivery-Pipeline erstellt und stellt automatisch Container-Images bereit, wenn der Quellcode geändert oder ein neues Basis-Image in Amazon ECR hochgeladen wird.

In diesem Ablauf werden die folgenden Artefakte verwendet:

- Eine Docker-Image-Datei, die den Container-Namen und die Repository-URI Ihres Amazon ECR-Image-Repositorys angibt.
- Eine Amazon ECS-Aufgabendefinition, die Ihren Docker-Image-Namen, Container-Namen, Amazon ECS-Servicenamen und Load Balancer-Konfiguration auflistet.
- Eine CodeDeploy AppSpec Datei, die den Namen der Amazon ECS-Aufgabendefinitionsdatei, den Namen des Containers der aktualisierten Anwendung und den Container-Port angibt, an den der CodeDeploy Produktionsverkehr umgeleitet wird. Sie kann auch optionale Netzwerkkonfigurations- und Lambda-Funktionen angeben, die Sie für Lebenszyklusereignis-Hooks während der Bereitstellung ausführen.

> **Note**
>
> Wenn Sie eine Änderung an Ihrem Amazon ECR-Image-Repository festschreiben, erstellt die Pipeline-Quellaktion eine `imageDetail.json` Datei für diesen Commit. Informationen zur `imageDetail.json`-Datei finden Sie unter ImageDetail.json-Datei für Amazon ECS-Bereitstellungsaktionen in Blau/Grün.

Wenn Sie Ihre Pipeline erstellen oder bearbeiten und Quellartefakte für Ihre Bereitstellungsphase aktualisieren oder angeben, müssen Sie auf die Quellartefakte mit dem neuesten Namen und der

Version verweisen, die Sie verwenden möchten. Nach dem Einrichten Ihrer Pipeline müssen Sie beim Ändern Ihres Images oder der Aufgabendefinition möglicherweise Ihre Quellartefaktdateien in Ihren Repositorys aktualisieren und dann die Bereitstellungsstufe in Ihrer Pipeline bearbeiten.

Themen

Voraussetzungen

Sie müssen die folgenden Ressourcen erstellt haben:

- Ein CodeCommit Repository. Sie können das AWS CodeCommit Repository verwenden, in dem Sie es erstellt habenTutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository).
- Starten Sie eine Amazon EC2 EC2-Linux-Instance und installieren Sie Docker, um ein Image zu erstellen, wie in diesem Tutorial gezeigt. Wenn Sie bereits ein Image haben, das Sie verwenden möchten, können Sie diesen Schritt überspringen.

Schritt 1: Image erstellen und in ein Amazon ECR-Repository übertragen

In diesem Abschnitt verwenden Sie Docker, um ein Image zu erstellen, und verwenden dann das, AWS CLI um ein Amazon ECR-Repository zu erstellen und das Image in das Repository zu übertragen.

> **Note**
>
> Wenn Sie bereits ein Image haben, das Sie verwenden möchten, können Sie diesen Schritt überspringen.

So erstellen Sie ein Image

1. Melden Sie sich bei Ihrer Linux-Instance an, in der Sie Docker installiert haben.

   Rufen Sie ein Image für `nginx` ab. Dieser Befehl stellt das `nginx:latest` Bild bereit:

   ```
   docker pull nginx
   ```

2. Führen Sie docker images. Das Image sollte in der Liste angezeigt werden.

   ```
   docker images
   ```

Um ein Amazon ECR-Repository zu erstellen und Ihr Image zu pushen

1. Erstellen Sie ein Amazon ECR-Repository, um Ihr -Image zu speichern. Notieren Sie sich den `repositoryUri` in der Ausgabe.

   ```
   aws ecr create-repository --repository-name nginx
   ```

   Ausgabe:

   ```
   {
       "repository": {
           "registryId": "aws_account_id",
           "repositoryName": "nginx",
           "repositoryArn": "arn:aws:ecr:us-east-1:aws_account_id:repository/nginx",
           "createdAt": 1505337806.0,
           "repositoryUri": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx"
       }
   }
   ```

2. Versehen Sie das Image mit dem `repositoryUri`-Wert aus dem vorherigen Schritt als Tag.

```
docker tag nginx:latest aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
```

3. Führen Sie den aws ecr get-login-password Befehl aus, wie in diesem Beispiel für die `us-west-2` Region und die Konto-ID 111122223333 gezeigt.

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com/nginx
```

4. Pushen Sie das Bild mit dem `repositoryUri` aus dem vorherigen Schritt an Amazon ECR.

```
docker push 111122223333.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
```

Schritt 2: Erstellen Sie Aufgabendefinitions- und AppSpec Quelldateien und übertragen Sie sie in ein Repository CodeCommit

In diesem Abschnitt erstellen Sie eine JSON-Datei mit Aufgabendefinitionen und registrieren sie bei Amazon ECS. Anschließend erstellen Sie eine AppSpec Datei für Ihren Git-Client CodeDeploy und verwenden diesen, um die Dateien in Ihr CodeCommit Repository zu übertragen.

So erstellen Sie eine Aufgabendefinition für Ihr Image

1. Erstellen Sie eine Datei mit dem Namen `taskdef.json` und dem folgenden Inhalt. Geben Sie für `image` Ihren Image-Namen, z. B. nginx, ein. Dieser Wert wird aktualisiert, wenn Ihre Pipeline ausgeführt wird.

> **Note**
>
> Stellen Sie sicher, dass die in der Aufgabendefinition angegebene Ausführungsrolle den `AmazonECSTaskExecutionRolePolicy` enthält. Weitere Informationen finden Sie unter Amazon ECS Task Execution IAM Role im Amazon ECS Developer Guide.

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "name": "sample-website",
```

```
            "image": "nginx",
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
    "family": "ecs-demo"
}
```

2. Registrieren Sie Ihre Aufgabendefinition mit der Datei `taskdef.json`.

```
aws ecs register-task-definition --cli-input-json file://taskdef.json
```

3. Nachdem die Aufgabendefinition registriert ist, bearbeiten Sie Ihre Datei, um den Image-Namen zu entfernen und den <IMAGE1_NAME>-Platzhaltertext in das Image-Feld einzufügen.

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "name": "sample-website",
            "image": "<IMAGE1_NAME>",
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
```

```
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
    "family": "ecs-demo"
}
```

Um eine Datei zu erstellen AppSpec

- Die AppSpec Datei wird für CodeDeploy Bereitstellungen verwendet. Die Datei, die optionale Felder enthält, verwendet das folgende Format:

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "task-definition-ARN"
        LoadBalancerInfo:
          ContainerName: "container-name"
          ContainerPort: container-port-number
# Optional properties
        PlatformVersion: "LATEST"
        NetworkConfiguration:
            AwsvpcConfiguration:
              Subnets: ["subnet-name-1", "subnet-name-2"]
              SecurityGroups: ["security-group"]
              AssignPublicIp: "ENABLED"
Hooks:
- BeforeInstall: "BeforeInstallHookFunctionName"
- AfterInstall: "AfterInstallHookFunctionName"
- AfterAllowTestTraffic: "AfterAllowTestTrafficHookFunctionName"
- BeforeAllowTraffic: "BeforeAllowTrafficHookFunctionName"
- AfterAllowTraffic: "AfterAllowTrafficHookFunctionName"
```

  Weitere Informationen zu der AppSpec Datei, einschließlich Beispielen, finden Sie unter CodeDeploy AppSpec Dateireferenz.

Erstellen Sie eine Datei mit dem Namen `appspec.yaml` und dem folgenden Inhalt. Ändern Sie für `TaskDefinition` den <TASK_DEFINITION>-Platzhaltertext nicht. Dieser Wert wird aktualisiert, wenn Ihre Pipeline ausgeführt wird.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: <TASK_DEFINITION>
        LoadBalancerInfo:
          ContainerName: "sample-website"
          ContainerPort: 80
```

Um Dateien in Ihr CodeCommit Repository zu übertragen

1. Push oder lade die Dateien in dein CodeCommit Repository hoch. Diese Dateien sind das Quellartefakt, das vom Assistenten zum Erstellen einer Pipeline für Ihre Bereitstellungsaktion in CodePipeline erstellt wurde. Ihre Dateien sollten in Ihrem lokalen Verzeichnis wie folgt aussehen:

```
/tmp
  |my-demo-repo
    |-- appspec.yaml
    |-- taskdef.json
```

2. Wählen Sie die Methode aus, mit der Sie Ihre Dateien hochladen möchten:

   a. So verwenden Sie Ihre Git-Befehlszeile aus einem geklonten Repository auf Ihrem lokalen Computer:

      i. Wechseln Sie zu Ihrem lokalen Repository:

```
(For Linux, macOS, or Unix) cd /tmp/my-demo-repo
(For Windows) cd c:\temp\my-demo-repo
```

      ii. Führen Sie den folgenden Befehl aus, um alle Dateien auf einmal zu übertragen:

```
git add -A
```

iii. Führen Sie den folgenden Befehl aus, um einen Commit für Dateien mit einer Commit-Nachricht durchzuführen:

```
git commit -m "Added task definition files"
```

iv. Führen Sie den folgenden Befehl aus, um die Dateien von Ihrem lokalen Repository in Ihr CodeCommit Repository zu übertragen:

```
git push
```

b. So verwenden Sie die CodeCommit Konsole, um Ihre Dateien hochzuladen:

i. Öffnen Sie die CodeCommit Konsole und wählen Sie Ihr Repository aus der Repository-Liste aus.

ii. Wählen Sie Add file (Datei hinzufügen) und dann Upload file (Datei hochladen) aus.

iii. Wählen Sie Choose file (Datei auswählen) aus und navigieren Sie anschließend zu Ihrer Datei. Übernehmen Sie die Änderung, indem Sie Ihren Benutzernamen und Ihre E-Mail-Adresse eingeben. Wählen Sie Commit changes (Änderungen übernehmen) aus.

iv. Wiederholen Sie diesen Schritt für jede Datei, die Sie hochladen möchten.

Schritt 3: Erstellen Ihres Application Load Balancer und der Zielgruppen

In diesem Abschnitt erstellen Sie einen Amazon EC2 Application Load Balancer. Sie verwenden die Subnetznamen und Zielgruppenwerte, die Sie mit Ihrem Load Balancer erstellen, später, wenn Sie Ihren Amazon ECS-Service erstellen. Sie können einen Application Load Balancer oder einen Network Load Balancer erstellen. Der Load Balancer muss eine VPC mit zwei öffentlichen Subnetzen in verschiedenen Availability Zones verwenden. In diesen Schritten bestätigen Sie Ihre Standard-VPC, erstellen einen Load Balancer und legen dann zwei Zielgruppen für Ihren Load Balancer fest. Weitere Informationen finden Sie unter Zielgruppen für Ihre Network Load Balancer.

So überprüfen Sie Ihre Standard-VPC und öffentliche Subnetze

1. Melden Sie sich bei der Amazon VPC-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/vpc/.
2. Überprüfen Sie die zu verwendende Standard-VPC. Wählen Sie im Navigationsbereich Your VPCs (Ihre VPCs) aus. Beachten Sie, für welche VPC Yes (Ja) in der Spalte Default VPC

(Standard-VPC) angezeigt wird. Dies ist die Standard-VPC. Sie enthält Standardsubnetze, die Sie auswählen können.

3. Wählen Sie Subnets (Subnetze) aus. Wählen Sie zwei Subnetze aus, für die Yes (Ja) in der Spalte Default subnet (Standardsubnetz) angezeigt wird.

> **Note**
>
> Notieren Sie sich die Subnetz-IDs. Sie benötigen den Wert später in diesem Tutorial.

4. Wählen Sie die Subnetze und anschließend die Registerkarte Description (Beschreibung) aus. Stellen Sie sicher, dass sich die Subnetze, die Sie verwenden möchten, in verschiedenen Availability Zones befinden.
5. Wählen Sie die Subnetze aus und klicken Sie dann auf die Registerkarte Route Table (Routing-Tabelle). Um sicherzustellen, dass jedes Subnetz, das Sie verwenden möchten, ein öffentliches Subnetz ist, vergewissern Sie sich, dass eine Gateway-Zeile in der Routing-Tabelle enthalten ist.

So erstellen Sie einen Amazon EC2 Application Load Balancer

1. Melden Sie sich bei der Amazon EC2 EC2-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/ec2/.
2. Klicken Sie im Navigationsbereich auf Load Balancers.
3. Klicken Sie auf Load Balancer erstellen.
4. Wählen Sie Application Load Balancer aus und klicken Sie anschließend auf Create (Erstellen).
5. Geben Sie unter Name den Namen Ihres Load Balancer ein.
6. Wählen Sie für Scheme (Schema) die Option internet-facing (mit dem Internet verbunden) aus.
7. Wählen Sie für IP address type (IP-Adresstyp) die Option ipv4 aus.
8. Konfigurieren Sie zwei Listener-Ports für Ihren Load Balancer:
   a. Wählen Sie unter Load Balancer Protocol die Option HTTP aus. Geben Sie unter Load Balancer Port (Load Balancer-Port) „**`80`**“ ein.
   b. Wählen Sie Add listener (Listener hinzufügen) aus.
   c. Wählen Sie unter Load Balancer Protocol für den zweiten Listener HTTP aus. Geben Sie unter Load Balancer Port (Load Balancer-Port) „**`8080`**“ ein.
9. Wählen Sie unter Availability Zones in VPC die Standard-VPC aus. Wählen Sie als Nächstes die zwei Standardsubnetze aus, die Sie verwenden möchten.

10. Klicken Sie auf Next: Configure Security Settings (Weiter: Sicherheitseinstellungen konfigurieren).
11. Wählen Sie Next: Configure Security Groups (Weiter: Sicherheitsgruppen konfigurieren) aus.
12. Wählen Sie Select an existing security group (Eine vorhandene Sicherheitsgruppe wählen) aus und notieren Sie sich die Sicherheitsgruppen-ID.
13. Wählen Sie Weiter: Routing konfigurieren aus.
14. Wählen Sie in Target group (Zielgruppe) die Option New target group (Neue Zielgruppe) aus und konfigurieren Sie Ihre erste Zielgruppe:
    a. Geben Sie im Feld Name einen Namen für die Zielgruppe ein (z. B. **`target-group-1`**).
    b. Wählen Sie für Target type (Zieltyp) die Option IP aus.
    c. Wählen Sie für Protocol die Option HTTP aus. Geben Sie im Feld Port den Wert **`80`** ein.
    d. Klicken Sie auf Weiter: Ziele registrieren.
15. Wählen Sie Next: Review (Weiter: Prüfen) und anschließend Create role (Rolle erstellen) aus.

So erstellen Sie eine zweite Zielgruppe für Ihren Load Balancer.

1. Nachdem Ihr Load Balancer bereitgestellt wurde, öffnen Sie die Amazon EC2 EC2-Konsole. Wählen Sie im Navigationsbereich Target Groups aus.
2. Wählen Sie Zielgruppe erstellen aus.
3. Geben Sie im Feld Name einen Namen für die Zielgruppe ein (z. B. **`target-group-2`**).
4. Wählen Sie für Target type (Zieltyp) die Option IP aus.
5. Wählen Sie für Protocol die Option HTTP aus. Geben Sie im Feld Port den Wert **`8080`** ein.
6. Wählen Sie unter VPC die Standard-VPC aus.
7. Wählen Sie Erstellen.

> **Note**
>
> Sie müssen zwei Zielgruppen für Ihren Load Balancer erstellt haben, damit die Bereitstellung ausgeführt werden kann. Sie müssen sich nur den ARN Ihrer ersten Zielgruppe notieren. Dieser ARN wird in der `create-service`-JSON-Datei im nächsten Schritt verwendet.

So aktualisieren Sie Ihren Load Balancer, um Ihre zweite Zielgruppe einzubeziehen

1. Öffnen Sie die Amazon EC2-Konsole. Klicken Sie im Navigationsbereich auf Load Balancers.
2. Wählen Sie Ihren Load Balancer aus und klicken Sie dann auf die Registerkarte Listeners (Listener). Wählen Sie den Listener mit Port 8080 und klicken Sie dann auf Edit (Bearbeiten).
3. Klicken Sie auf das Stiftsymbol neben Forward to (Weiterleiten an). Wählen Sie die zweite Zielgruppe aus und klicken Sie dann auf das Häkchen. Klicken Sie auf Update (Aktualisieren), um die Aktualisierungen zu speichern.

Schritt 4: Erstellen Sie Ihren Amazon ECS-Cluster und -Service

In diesem Abschnitt erstellen Sie einen Amazon ECS-Cluster und -Service, der den Datenverkehr während der Bereitstellung CodeDeploy weiterleitet (zu einem Amazon ECS-Cluster statt zu EC2-Instances). Um Ihren Amazon ECS-Service zu erstellen, müssen Sie die Subnetznamen, Sicherheitsgruppen und Zielgruppenwerte verwenden, die Sie mit Ihrem Load Balancer erstellt haben, um Ihren Service zu erstellen.

> **Note**
>
> Wenn Sie diese Schritte verwenden, um Ihren Amazon ECS-Cluster zu erstellen, verwenden Sie die Cluster-Vorlage nur für Netzwerke, die AWS Fargate-Container bereitstellt. AWS Fargate ist eine Technologie, die Ihre Container-Instance-Infrastruktur für Sie verwaltet. Sie müssen Amazon EC2-Instances für Ihren Amazon ECS-Cluster nicht auswählen oder manuell erstellen.

Erstellen eines Amazon ECS-Clusters

1. Öffnen Sie die klassische Amazon-ECS-Konsole unter https://console.aws.amazon.com/ecs/.
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Wählen Sie Cluster erstellen.
4. Wählen Sie die Clustervorlage Nur Networking aus, die AWS Fargate verwendet, und klicken Sie dann auf Nächster Schritt.
5. Geben Sie auf der Seite Configure cluster (Cluster konfigurieren) einen Namen für den Cluster ein. Sie können ein optionales Tag für Ihre Ressource hinzufügen. Wählen Sie Erstellen.

So erstellen Sie einen Amazon ECS-Service

Verwenden Sie den AWS CLI , um Ihren Service in Amazon ECS zu erstellen.

1. Erstellen Sie eine JSON-Datei und geben Sie ihr den Namen `create-service.json`. Fügen Sie Folgendes in die JSON-Datei ein.

   Wenn Sie für dieses `taskDefinition` Feld eine Aufgabendefinition in Amazon ECS registrieren, geben Sie ihr eine Familie. Dies ähnelt einem Namen für mehrere Versionen der Aufgabendefinition und wird mit einer Versionsnummer versehen. Verwenden Sie in diesem Beispiel für Ihre Datei „`ecs-demo:1`“ als Familie und Versionsnummer. Verwenden Sie die Subnetz-Namen, die Sicherheitsgruppe und den Zielgruppenwert, die bzw. den Sie mit Ihrem Load Balancer in Schritt 3: Erstellen Ihres Application Load Balancer und der Zielgruppen erstellt haben.

   > **Note**
   >
   > Sie müssen Ihren Zielgruppen-ARN in dieser Datei angeben. Öffnen Sie die Amazon EC2 EC2-Konsole und wählen Sie im Navigationsbereich unter LOAD BALANCING die Option Target Groups aus. Wählen Sie Ihre erste Zielgruppe aus. Kopieren Sie Ihren ARN aus der Registerkarte Description (Beschreibung).

```
{
    "taskDefinition": "family:revision-number",
    "cluster": "my-cluster",
    "loadBalancers": [
        {
            "targetGroupArn": "target-group-arn",
            "containerName": "sample-website",
            "containerPort": 80
        }
    ],
    "desiredCount": 1,
    "launchType": "FARGATE",
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "CODE_DEPLOY"
    },
    "networkConfiguration": {
```

```
        "awsvpcConfiguration": {
            "subnets": [
                "subnet-1",
                "subnet-2"
            ],
            "securityGroups": [
                "security-group"
            ],
            "assignPublicIp": "ENABLED"
        }
    }
}
```

2. Führen Sie den Befehl create-service aus und geben Sie dabei die JSON-Datei an:

> **Important**
>
> Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

In diesem Beispiel wird ein Service mit dem Namen `my-service` erstellt.

> **Note**
>
> Dieser Beispielbefehl erstellt einen Service mit dem Namen my-Service. Wenn Sie bereits über einen Service mit diesem Namen verfügen, gibt der Befehl einen Fehler zurück.

```
aws ecs create-service --service-name my-service --cli-input-json file://create-
service.json
```

Die Ausgabe gibt die Beschreibungsfelder für Ihren Service zurück.

3. Führen Sie den Befehl describe-services aus, um zu überprüfen, ob Ihr Service erstellt wurde.

```
aws ecs describe-services --cluster cluster-name --services service-name
```

Schritt 5: Erstellen Sie Ihre CodeDeploy Anwendung und Bereitstellungsgruppe (ECS-Rechenplattform)

Wenn Sie eine CodeDeploy Anwendung und eine Bereitstellungsgruppe für die Amazon ECS-Rechenplattform erstellen, wird die Anwendung während einer Bereitstellung verwendet, um auf die richtige Bereitstellungsgruppe, Zielgruppen, Listener und das richtige Verhalten bei der Umleitung des Datenverkehrs zu verweisen.

Um eine Anwendung zu erstellen CodeDeploy

1. Öffnen Sie die CodeDeploy Konsole und wählen Sie Anwendung erstellen.
2. Geben Sie unter Application name (Anwendungsname) den Namen ein, den Sie verwenden möchten.
3. Wählen Sie unter Compute Platform (Datenverarbeitungsplattform) die Option Amazon ECS aus.
4. Wählen Sie Create application aus.

Um eine CodeDeploy Bereitstellungsgruppe zu erstellen

1. Wählen Sie auf Ihrer Anwendungsseite über die Registerkarte Deployment groups (Bereitstellungsgruppen) die Option Create deployment group (Bereitstellungsgruppe erstellen) aus.
2. Geben Sie im Feld Deployment group name (Name der Bereitstellungsgruppe) einen Namen ein, der die Bereitstellungsgruppe beschreibt.
3. Wählen Sie unter Servicerolle eine Servicerolle aus, die CodeDeploy Zugriff auf Amazon ECS gewährt. Gehen Sie folgendermaßen vor, um eine neue Servicerolle zu erstellen:
   a. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/).
   b. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
   c. Wählen Sie Rolle erstellen aus.
   d. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen AWS-Servicedie Option aus. Wählen Sie unter Anwendungsfall auswählen die Option aus CodeDeploy. Wählen Sie unter Wählen Sie Ihren Anwendungsfall die Option CodeDeploy - ECS aus. Wählen Sie Weiter: Berechtigungen aus. Die `AWSCodeDeployRoleForECS`-verwaltete Richtlinie ist der Rolle bereits zugewiesen.
   e. Wählen Sie Next: Tags (Weiter: Tags) und Next: Review (Weiter: Prüfen) aus.

f. Geben Sie einen Namen für die Rolle ein (beispielsweise **`CodeDeployECSRole`**) und wählen Sie Next (Weiter) aus.

4. Wählen Sie in der Umgebungskonfiguration Ihren Amazon ECS-Clusternamen und Servicenamen aus.
5. Wählen Sie unter Load Balancers den Namen des Load Balancers aus, der den Datenverkehr an Ihren Amazon ECS-Service weiterleitet.
6. Wählen Sie unter Production listener port (Produktions-Listener-Port) den Port und das Protokoll für den Listener aus, der den Produktionsdatenverkehr an Ihren Amazon ECS Service weiterleitet. Wählen Sie unter Test listener port (Test-Listener-Port) den Port und das Protokoll für den Test-Listener aus.
7. Wählen Sie unter Target group 1 name (Name der Zielgruppe 1) und Target group 2 name (Name der Zielgruppe 2) die Zielgruppen für die Weiterleitung des Datenverkehrs während der Bereitstellung aus. Stellen Sie sicher, dass es sich hierbei um die Zielgruppen handelt, die Sie für Ihren Load Balancer erstellt haben.
8. Wählen Sie Traffic sofort umleiten, um zu bestimmen, wie lange nach einer erfolgreichen Bereitstellung der Verkehr zu Ihrer aktualisierten Amazon ECS-Aufgabe umgeleitet werden soll.
9. Wählen Si Create deployment group (Bereitstellungsgruppe erstellen).

Schritt 6: Erstellen Ihrer Pipeline

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine CodeCommit Aktion, bei der die Quellartefakte die Aufgabendefinition und die AppSpec Datei sind.
- Eine Quellstufe mit einer Amazon ECR-Quellaktion, bei der das Quellartefakt die Bilddatei ist.
- Eine Bereitstellungsphase mit einer Amazon ECS-Bereitstellungsaktion, in der die Bereitstellung mit einer CodeDeploy Anwendung und einer Bereitstellungsgruppe ausgeführt wird.

So erstellen Sie eine zweistufige Pipeline mit dem Assistenten

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) aus, oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).

3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyImagePipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie unter Step 2: Add source stage (Schritt 2: Quell-Phase hinzufügen) unter Source provider (Quell-Anbieter) die Option AWS CodeCommit. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, in Schritt 1: Erstellen Sie ein CodeCommit Repository dem Sie es erstellt haben. Wählen Sie in Branch name den Namen des Branch aus, der das neueste Code-Update enthält.

   Wählen Sie Weiter aus.
8. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken. Wählen Sie Weiter aus.
9. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Amazon ECS (Blue/Green) (Amazon ECS (Blau/Grün)) aus. Geben Sie unter Application name (Anwendungsname) den Anwendungsnamen ein oder wählen Sie ihn aus der Liste aus, z. B. `codedeployapp`. Geben Sie unter Deployment group (Bereitstellungsgruppe) den Namen einer Bereitstellungsgruppe ein oder wählen Sie ihn in der Liste aus, z. B. `codedeploydeplgroup`

   > **Note**
   >
   > Der Name "Deploy" (Bereitstellen) ist der Name, der der Stufe in Step 4: Deploy (Schritt 4: bereitstellen) standardmäßig gegeben wird, so wie der ersten Stufe der Pipeline der Namen "Source" (Quelle) gegeben wird.

   b. Wählen Sie unter Amazon ECS-Aufgabendefinition die Option SourceArtifact. Geben Sie im Feld **`taskdef.json`** ein.

c. Wählen Sie unter AWS CodeDeploy AppSpec Datei die Option SourceArtifact. Geben Sie im Feld **`appspec.yaml`** ein.

> **Note**
>
> Geben Sie an diesem Punkt keine Informationen unter Dynamically update task definition image (Aufgabendefinitions-Image dynamisch aktualisieren) ein.

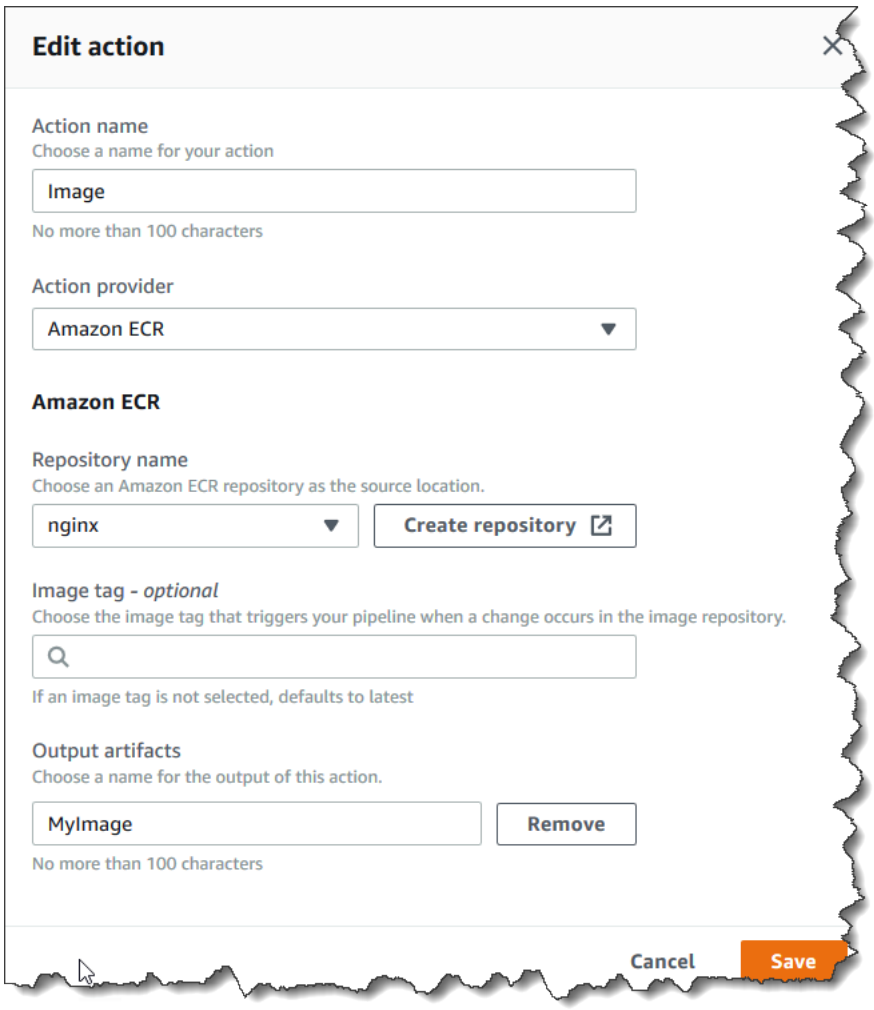
d. Wählen Sie Weiter aus.

10. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.

So fügen Sie Ihrer Pipeline eine Amazon ECR-Quellaktion hinzu

Sehen Sie sich Ihre Pipeline an und fügen Sie Ihrer Pipeline eine Amazon ECR-Quellaktion hinzu.

1. Wählen Sie Ihre Pipeline aus. Wählen Sie links oben Edit (Bearbeiten) aus.
2. Wählen Sie in der Quellstufe die Option Edit stage (Stufe bearbeiten) aus.
3. Fügen Sie eine parallel Aktion hinzu, indem Sie + Aktion hinzufügen neben Ihrer CodeCommit Quellaktion auswählen.
4. Geben Sie im Feld Action name (Name der Aktion) einen Namen ein (zum Beispiel **`Image`**).
5. Wählen Sie unter Action provider (Aktionsanbieter) die Option Amazon ECR aus.



6. Wählen Sie unter Repository-Name den Namen Ihres Amazon ECR-Repositorys aus.
7. Geben Sie im Feld Image tag (Image-Tag) den Image-Namen und die Version an, falls sie nicht LATEST lautet.
8. Wählen Sie im Feld Output artifacts (Ausgabeartefakte) den Standardwert für die Ausgabeartefakte aus (z. B. `MyImage`), der den Image-Namen und Repository-URI-Informationen enthält, die in der nächsten Stufe verwendet werden sollen.
9. Wählen Sie im Aktionsbildschirm Save (Speichern) aus. Wählen Sie im Phasenbilschirm Done (Fertig) aus. Wählen Sie in der Pipeline Save (Speichern) aus. In einer Meldung wird die Amazon CloudWatch Events-Regel angezeigt, die für die Amazon ECR-Quellaktion erstellt werden soll.

So senden Sie Ihre Quellartefakte an die Bereitstellungsaktion

1. Wählen Sie in Ihrer Bereitstellungsphase Bearbeiten und wählen Sie das Symbol, um die Amazon ECS-Aktion (Blau/Grün) zu bearbeiten.

2. Scrollen Sie nach unten bis zum Ende des Bereichs. Wählen Sie unter Input artifacts (Eingabeartefakte) die Option Add (Hinzufügen) aus. Fügen Sie das Quellartefakt aus Ihrem neuen Amazon ECR-Repository hinzu (z. B.`MyImage`).
3. Wählen Sie unter Aufgabendefinition die Option aus SourceArtifact, und geben Sie dann verifizieren **`taskdef.json`** ein.
4. Wählen Sie unter AWS CodeDeploy AppSpec Datei die Option aus SourceArtifact, und bestätigen Sie, **`appspec.yaml`** dass verifizieren eingegeben wurde.
5. Wählen Sie unter Aufgabendefinitionsbild dynamisch aktualisieren unter Eingabeartefakt mit Bild-URI den Platzhaltertext aus MyImage, der in der Datei verwendet wird, und geben Sie dann den Platzhaltertext ein, der in der `taskdef.json` Datei verwendet wird:. **`IMAGE1_NAME`** Wählen Sie Speichern.
6. Wählen Sie im AWS CodePipeline Bereich Pipeline-Änderung speichern und dann Änderung speichern aus. Sehen Sie sich Ihre aktualisierte Pipeline an.

   Nachdem diese Beispiel-Pipeline erstellt wurde, wird die Aktionskonfiguration für die Konsoleneinträge wie folgt in der Pipeline-Struktur angezeigt:

```
"configuration": {
  "AppSpecTemplateArtifact": "SourceArtifact",
  "AppSpecTemplatePath": "appspec.yaml",
  "TaskDefinitionTemplateArtifact": "SourceArtifact",
  "TaskDefinitionTemplatePath": "taskdef.json",
  "ApplicationName": "codedeployapp",
  "DeploymentGroupName": "codedeploydeplgroup",
  "Image1ArtifactName": "MyImage",
  "Image1ContainerName": "IMAGE1_NAME"
},
```

7. Um Ihre Änderungen zu übertragen und einen Pipeline-Build zu starten, wählen Sie Release change (Änderung freigeben) und dann Release (Freigeben).
8. Wählen Sie die Bereitstellungsaktion aus, um sie anzuzeigen CodeDeploy und den Fortschritt der Verkehrsverlagerung zu verfolgen.

> **Note**
>
> Möglicherweise wird ein Bereitstellungsschritt mit einer optionalen Wartezeit angezeigt. Standardmäßig wird nach einer erfolgreichen Bereitstellung eine Stunde CodeDeploy gewartet, bevor der ursprüngliche Tasksatz beendet wird. Sie können diese Zeit nutzen,

um die Aufgabe zurückzusetzen oder zu beenden. Ihre Bereitstellung wird andernfalls abgeschlossen, wenn der Aufgabensatz beendet wird.

Schritt 7: Vornehmen einer Änderung an Ihrer Pipeline und Überprüfen der Bereitstellung

Nehmen Sie eine Änderung an Ihrem Image vor und übertragen Sie die Änderung dann in Ihr Amazon ECR-Repository. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Überprüfen Sie, ob die Änderung Ihrer Image-Quelle bereitgestellt wird.

Tutorial: Erstellen einer Pipeline zum Bereitstellen eines Amazon Alexa-Skills

In diesem Tutorial konfigurieren Sie eine Pipeline, über die Ihre Alexa-Qualifikation kontinuierlich bereitgestellt wird, indem das Alexa Skills Kit auf Ihrer Bereitstellungsstufe als Bereitstellungsanbieter verwendet wird. Die fertige Pipeline erkennt Änderungen an Ihrer Qualifikation, wenn Sie in Ihrem Quell-Repository eine Änderung an den Quelldateien vornehmen. Die Pipeline nutzt dann das Alexa Skills Kit für Bereitstellungen für die Entwicklungsstufe der Alexa-Qualifikation.

> **Note**
>
> Diese Funktion ist in der Region Asien-Pazifik (Hongkong) oder Europa (Mailand) nicht verfügbar. Informationen zur Verwendung anderer Bereitstellungsaktionen, die in dieser Region verfügbar sind, finden Sie unterBereitstellungsaktions-Integrationen.

Informationen zum Erstellen Ihres benutzerdefinierten Skills als Lambda-Funktion finden Sie unter Hosten eines benutzerdefinierten Skills als AWS Lambda-Funktion. Sie können auch eine Pipeline erstellen, die Lambda-Quelldateien und ein CodeBuild Projekt verwendet, um Änderungen an Lambda für Ihre Fähigkeiten bereitzustellen. Wenn Sie beispielsweise eine neue Qualifikation und eine zugehörige Lambda-Funktion erstellen möchten, können Sie ein AWS CodeStar -Projekt erstellen. Informationen hierzu finden Sie unter Erstellen eines Alexa-Skill-Projekts in AWS CodeStar. Für diese Option umfasst die Pipeline eine dritte Build-Phase mit einer CodeBuild Aktion und eine Aktion in der Bereitstellungsphase für. AWS CloudFormation

Voraussetzungen

Sie müssen bereits über Folgendes verfügen:

- Ein CodeCommit Repository. Sie können das AWS CodeCommit Repository verwenden, in dem Sie es erstellt habenTutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository).
- Ein Amazon-Entwicklerkonto. Dies ist das Konto, das im Besitz Ihrer Alexa-Qualifikationen ist. Sie können unter Alexa Skills Kit kostenlos ein Konto erstellen.
- Eine Alexa-Qualifikation. Sie können eine Beispielqualifikation erstellen, indem Sie das Tutorial Get Custom Skill Sample Code (Verwenden von Beispiel-Code für eine benutzerdefinierte Qualifikation) verwenden.
- Installieren Sie die ASK CLI und konfigurieren Sie sie `ask init` mit Ihren AWS Anmeldeinformationen. Siehe Installieren und Initialisieren von ASK CLI.

Schritt 1: Erstellen eines Alexa Developer Services-LWA-Sicherheitsprofils

In diesem Abschnitt erstellen Sie ein Sicherheitsprofil für Login With Amazon (LWA). Falls Sie bereits über ein Profil verfügen, können Sie diesen Schritt überspringen.

- Gehen Sie wie unter beschrieben vor generate-lwa-tokens, um ein Sicherheitsprofil zu erstellen.
- Notieren Sie sich die Client-ID und den Clientschlüssel, nachdem Sie das Profil erstellt haben.
- Achten Sie darauf, dass Sie die URLs unter Allowed Return URLs (Zugelassene Rückgabe-URLs) gemäß der Anleitung eingeben. Mit den URLs kann der ASK-CLI-Befehl Anforderungen von Aktualisierungs-Token umleiten.

Schritt 2: Erstellen Sie Alexa-Skill-Quelldateien und übertragen Sie sie in Ihr CodeCommit Repository

In diesem Abschnitt erstellen Sie die Quelldateien für Ihre Alexa-Qualifikation und übertragen sie per Push an das Repository, das von der Pipeline für Ihre Quellstufe verwendet wird. Für die Qualifikation, die Sie in der Amazon-Entwicklerkonsole erstellt haben, erstellen und übertragen Sie Folgendes:

- Die Datei `skill.json`.
- Einen `interactionModel/custom`-Ordner.

> **Note**
>
> Diese Verzeichnisstruktur entspricht den Anforderungen des Alexa Skills Kit-Skill-Pakets, wie in Skill-Paketformat beschrieben. Wenn Ihre Verzeichnisstruktur nicht das richtige Skill-Paketformat verwendet, werden Änderungen nicht erfolgreich in der Alexa Skills Kit-Konsole bereitgestellt.

So erstellen Sie Quelldateien für Ihre Qualifikation

1. Rufen Sie Ihre Qualifikations-ID aus der Alexa Skills Kit-Entwicklerkonsole ab. Verwenden Sie diesen Befehl:

   ```
   ask api list-skills
   ```

   Suchen Sie anhand des Namens nach Ihrer Qualifikation und kopieren Sie die zugehörige ID anschließend in das Feld `skillId`.

2. Generieren Sie die Datei `skill.json`, die Ihre Qualifikationsdetails enthält. Verwenden Sie diesen Befehl:

   ```
   ask api get-skill -s skill-ID > skill.json
   ```

3. (Optional) Erstellen Sie einen `interactionModel/custom`-Ordner.

   Verwenden Sie diesen Befehl, um die Interaktionsmodelldatei im Ordner zu generieren. Als Gebietsschema wird in diesem Tutorial „en-US“ im Dateinamen verwendet.

   ```
   ask api get-model --skill-id skill-ID --locale locale >
       ./interactionModel/custom/locale.json
   ```

Um Dateien in Ihr CodeCommit Repository zu übertragen

1. Push oder lade die Dateien in dein CodeCommit Repository hoch. Diese Dateien stellen das Quellartefakt dar, das vom Assistenten Create Pipeline (Pipeline erstellen) für Ihre Bereitstellungsaktion in AWS CodePipeline erstellt wurde. Ihre Dateien sollten in Ihrem lokalen Verzeichnis wie folgt aussehen:

```
skill.json
/interactionModel
  /custom
    |en-US.json
```

2. Wählen Sie die Methode aus, mit der Sie Ihre Dateien hochladen möchten:

   a. So verwenden Sie die Git-Befehlszeile aus einem geklonten Repository auf Ihrem lokalen Computer:

      i. Führen Sie den folgenden Befehl aus, um alle Dateien auf einmal zu übertragen:

      ```
      git add -A
      ```

      ii. Führen Sie den folgenden Befehl aus, um einen Commit für Dateien mit einer Commit-Nachricht durchzuführen:

      ```
      git commit -m "Added Alexa skill files"
      ```

      iii. Führen Sie den folgenden Befehl aus, um die Dateien von Ihrem lokalen Repository in Ihr CodeCommit Repository zu übertragen:

      ```
      git push
      ```

   b. So verwenden Sie die CodeCommit Konsole, um Ihre Dateien hochzuladen:

      i. Öffnen Sie die CodeCommit Konsole und wählen Sie Ihr Repository aus der Repository-Liste aus.

      ii. Wählen Sie Add file (Datei hinzufügen) und dann Upload file (Datei hochladen) aus.

      iii. Wählen Sie Choose file (Datei auswählen) aus und navigieren Sie anschließend zu Ihrer Datei. Übernehmen Sie die Änderung, indem Sie Ihren Benutzernamen und Ihre E-Mail-Adresse eingeben. Wählen Sie Commit changes (Änderungen übernehmen) aus.

      iv. Wiederholen Sie diesen Schritt für jede Datei, die Sie hochladen möchten.

Schritt 3: Verwenden von ASK-CLI-Befehlen zum Erstellen eines Aktualisierungs-Tokens

CodePipeline verwendet ein Aktualisierungstoken, das auf der Client-ID und dem geheimen Schlüssel in Ihrem Amazon-Entwicklerkonto basiert, um Aktionen zu autorisieren, die es in Ihrem Namen ausführt. In diesem Abschnitt verwenden Sie die ASK-CLI zum Erstellen des Tokens. Sie nutzen diese Anmeldeinformationen bei der Verwendung des Assistenten Create Pipeline (Pipeline erstellen).

So erstellen Sie ein Aktualisierungs-Token mit den Anmeldeinformationen Ihres Amazon-Entwicklerkontos

1. Verwenden Sie den folgenden Befehl:

   ```
   ask util generate-lwa-tokens
   ```

2. Geben Sie bei entsprechender Aufforderung Ihre Client-ID und den Schlüssel wie in diesem Beispiel ein:

   ```
   ? Please type in the client ID:
   amzn1.application-client.example112233445566
   ? Please type in the client secret:
   example112233445566
   ```

3. Die Browserseite für die Anmeldung wird angezeigt. Melden Sie sich mit den Anmeldeinformationen für Ihr Amazon-Entwicklerkonto an.
4. Wechseln Sie zurück zum Fenster mit der Befehlszeile. Das Zugriffstoken und das Aktualisierungstoken werden in der Ausgabe generiert. Kopieren Sie das zurückgegebene Aktualisierungs-Token, das in der Ausgabe enthalten ist.

Schritt 4: Erstellen Ihrer Pipeline

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellphase mit einer CodeCommit Aktion, bei der es sich bei den Quellartefakten um die Alexa-Skill-Dateien handelt, die Ihren Skill unterstützen.
- Eine Bereitstellungsstufe mit einer Alexa Skills Kit-Bereitstellungsaktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie die AWS Region aus, in der Sie das Projekt und seine Ressourcen erstellen möchten. Die Alexa Skill-Laufzeit ist nur in den folgenden Regionen verfügbar:
   - Asien-Pazifik (Tokio)
   - Europa (Irland)
   - USA Ost (Nord-Virginia)
   - USA West (Oregon)
3. Wählen sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) aus, oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
4. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyAlexaPipeline`** ein.
5. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
6. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
7. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
8. Wählen Sie unter Step 2: Add source stage (Schritt 2: Quell-Phase hinzufügen) unter Source provider (Quell-Anbieter) die Option AWS CodeCommit. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, in Schritt 1: Erstellen Sie ein CodeCommit Repository dem Sie es erstellt haben. Wählen Sie in Branch name den Namen des Branch aus, der das neueste Code-Update enthält.

   Nachdem Sie den Repository-Namen und den Branch ausgewählt haben, wird in einer Meldung die Amazon CloudWatch Events-Regel angezeigt, die für diese Pipeline erstellt werden soll.

   Wählen Sie Weiter aus.
9. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken.

Wählen Sie Weiter aus.

10. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

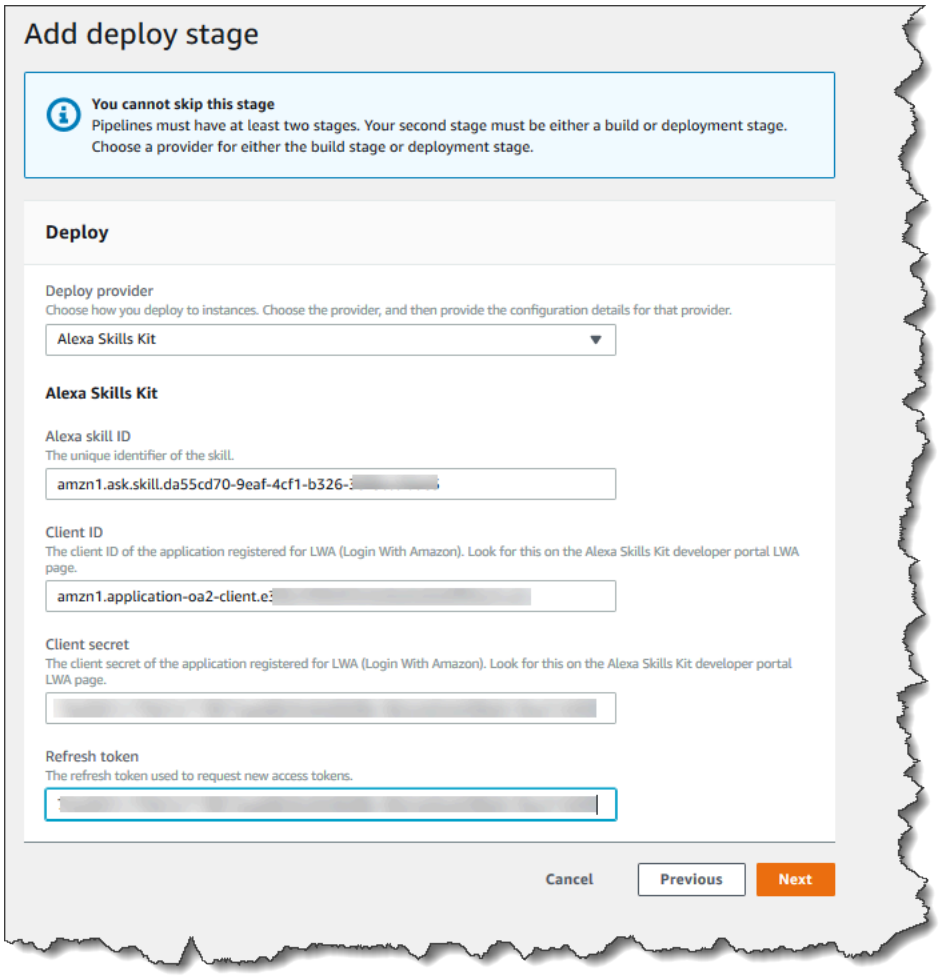
    a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Alexa Skills Kit.

    b. Geben Sie unter Alexa Skill-ID die Qualifikations-ID ein, die Ihrer Qualifikation in der Alexa Skills Kit-Entwicklerkonsole zugewiesen ist.

    c. Geben Sie unter Client-ID die ID der von Ihnen registrierten Anwendung ein.

    d. Geben Sie unter Clientschlüssel den Schlüssel ein, den Sie während der Registrierung gewählt haben.

    e. Geben Sie unter Aktualisierungs-Token das Token ein, das Sie in Schritt 3 generiert haben.



    f. Wählen Sie Weiter aus.

11. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.

Schritt 5: Vornehmen einer Änderung an einer beliebigen Quelldatei und Überprüfen der Bereitstellung

Nehmen Sie eine Änderung an Ihrer Qualifikation vor und übertragen Sie die Änderung per Push in Ihr Repository. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Vergewissern Sie sich, dass Ihre Qualifikation in der Alexa Skills Kit-Entwicklerkonsole aktualisiert wurde.

Tutorial: Erstellen Sie eine Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet

In diesem Tutorial konfigurieren Sie eine Pipeline, die kontinuierlich Dateien mithilfe von Amazon S3 als Bereitstellungsaktionsanbieter in Ihrer Bereitstellungsphase bereitstellt. Die fertige Pipeline erkennt Änderungen, wenn Sie in Ihrem Quell-Repository eine Änderung an den Quelldateien vornehmen. Die Pipeline verwendet dann Amazon S3, um die Dateien in Ihrem Bucket bereitzustellen. Jedes Mal, wenn Sie Ihre Website-Dateien an Ihrem Quellspeicherort ändern oder hinzufügen, erstellt die Bereitstellung die Website mit Ihren neuesten Dateien.

> **Note**
>
> Selbst wenn Sie Dateien aus dem Quell-Repository löschen, löscht die S3-Bereitstellungsaktion keine S3-Objekte, die gelöschten Dateien entsprechen.

In diesem Tutorial haben Sie zwei Optionen:

- Erstellen einer Pipeline, mit der eine statische Website für Ihren öffentlichen S3-Bucket bereitgestellt wird. In diesem Beispiel wird eine Pipeline mit einer AWS CodeCommit Quellaktion und einer Amazon S3 S3-Bereitstellungsaktion erstellt. Siehe Option 1: Statische Website-Dateien auf Amazon S3 bereitstellen.
- Erstellen Sie eine Pipeline, die TypeScript Beispielcode kompiliert JavaScript und das CodeBuild Ausgabeartefakt zur Archivierung in Ihrem S3-Bucket bereitstellt. In diesem Beispiel wird eine Pipeline mit einer Amazon S3 S3-Quellaktion, einer CodeBuild Build-Aktion und einer Amazon S3 S3-Bereitstellungsaktion erstellt. Siehe Option 2: Stellen Sie erstellte Archivdateien aus einem S3-Quell-Bucket in Amazon S3 bereit.

> **Important**
>
> Viele der Aktionen, die Sie in diesem Verfahren zu Ihrer Pipeline hinzufügen, beinhalten AWS Ressourcen, die Sie erstellen müssen, bevor Sie die Pipeline erstellen. AWS Ressourcen für Ihre Quellaktionen müssen immer in derselben AWS Region erstellt werden, in der Sie Ihre Pipeline erstellen. Wenn Sie Ihre Pipeline beispielsweise in der Region USA Ost (Ohio) erstellen, muss sich Ihr CodeCommit Repository in der Region USA Ost (Ohio) befinden. Sie können beim Erstellen Ihrer Pipeline regionsübergreifende Aktionen hinzufügen. AWS Ressourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

Option 1: Statische Website-Dateien auf Amazon S3 bereitstellen

In diesem Beispiel laden Sie die statische Website-Vorlagendatei herunter, laden die Dateien in Ihr AWS CodeCommit Repository hoch, erstellen Ihren Bucket und konfigurieren ihn für das Hosting. Als Nächstes verwenden Sie die AWS CodePipeline Konsole, um Ihre Pipeline zu erstellen und eine Amazon S3 S3-Bereitstellungskonfiguration festzulegen.

Voraussetzungen

Sie müssen bereits über Folgendes verfügen:

- Ein CodeCommit Repository. Sie können das AWS CodeCommit Repository verwenden, in dem Sie es erstellt habenTutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository).
- Quelldateien für Ihre statische Website. Verwenden Sie diesen Link, um ein Beispiel für eine statische Website herunterzuladen. Die heruntergeladene Datei „sample-website.zip“ enthält die folgenden Dateien:
  - Eine `index.html`-Datei
  - Eine `main.css`-Datei
  - Eine `graphic.jpg`-Datei
- Einen S3-Bucket, der für das Website-Hosting konfiguriert ist. Siehe Hosten einer statischen Website auf Amazon S3. Stellen Sie sicher, dass Sie Ihren Bucket in derselben Region wie die Pipeline erstellen.

> **Note**
>
> Für das Hosten einer Website muss Ihr Bucket über öffentlichen Lesezugriff verfügen, damit alle Benutzer Lesezugriff haben. Mit Ausnahme des Website-Hostings sollten Sie die Standardeinstellungen für den Zugriff beibehalten, mit denen der öffentliche Zugriff auf S3-Buckets blockiert wird.

Schritt 1: Pushen Sie die Quelldateien in Ihr CodeCommit Repository

In diesem Abschnitt übertragen Sie Ihre Quelldateien per Push in das Repository, das von der Pipeline für Ihre Quellstufe verwendet wird.

Um Dateien in dein CodeCommit Repository zu übertragen

1. Extrahieren Sie die heruntergeladenen Beispieldateien. Laden Sie die ZIP-Datei nicht in Ihr Repository hoch.
2. Push oder lade die Dateien in dein CodeCommit Repository hoch. Diese Dateien sind das Quellartefakt, das vom Assistenten „Pipeline erstellen“ für Ihre Bereitstellungsaktion in CodePipeline erstellt wurde. Ihre Dateien sollten in Ihrem lokalen Verzeichnis wie folgt aussehen:

   ```
   index.html
   main.css
   graphic.jpg
   ```

3. Du kannst Git oder die CodeCommit Konsole verwenden, um deine Dateien hochzuladen:

   a. So verwenden Sie die Git-Befehlszeile aus einem geklonten Repository auf Ihrem lokalen Computer:

      i. Führen Sie den folgenden Befehl aus, um alle Dateien auf einmal zu übertragen:

      ```
      git add -A
      ```

      ii. Führen Sie den folgenden Befehl aus, um einen Commit für Dateien mit einer Commit-Nachricht durchzuführen:

      ```
      git commit -m "Added static website files"
      ```

iii. Führe den folgenden Befehl aus, um die Dateien von deinem lokalen Repo in dein CodeCommit Repository zu übertragen:

```
git push
```

b. So verwenden Sie die CodeCommit Konsole, um Ihre Dateien hochzuladen:

i. Öffnen Sie die CodeCommit Konsole und wählen Sie Ihr Repository aus der Repository-Liste aus.

ii. Wählen Sie Add file (Datei hinzufügen) und dann Upload file (Datei hochladen) aus.

iii. Wählen Sie Choose file (Datei auswählen) und navigieren Sie dann zu Ihrer Datei. Übernehmen Sie die Änderung, indem Sie Ihren Benutzernamen und Ihre E-Mail-Adresse eingeben. Wählen Sie Commit changes (Änderungen übernehmen) aus.

iv. Wiederholen Sie diesen Schritt für jede Datei, die Sie hochladen möchten.

Schritt 2: Erstellen der Pipeline

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellstufe mit einer CodeCommit Aktion, bei der die Quellartefakte die Dateien für Ihre Website sind.
- Eine Bereitstellungsphase mit einer Amazon S3 S3-Bereitstellungsaktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyS3DeployPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.

6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie unter Step 2: Add source stage (Schritt 2: Quell-Phase hinzufügen) unter Source provider (Quell-Anbieter) die Option AWS CodeCommit. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, in Schritt 1: Erstellen Sie ein CodeCommit Repository dem Sie es erstellt haben. Wählen Sie in Branch name den Namen des Branch aus, der das neueste Code-Update enthält. Sofern Sie keinen anderen Branch für sich erstellt haben, ist nur `main` verfügbar.

   Nachdem Sie den Repository-Namen und den Branch ausgewählt haben, wird die Amazon CloudWatch Events-Regel angezeigt, die für diese Pipeline erstellt werden soll.

   Wählen Sie Weiter aus.
8. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken.

   Wählen Sie Weiter aus.
9. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

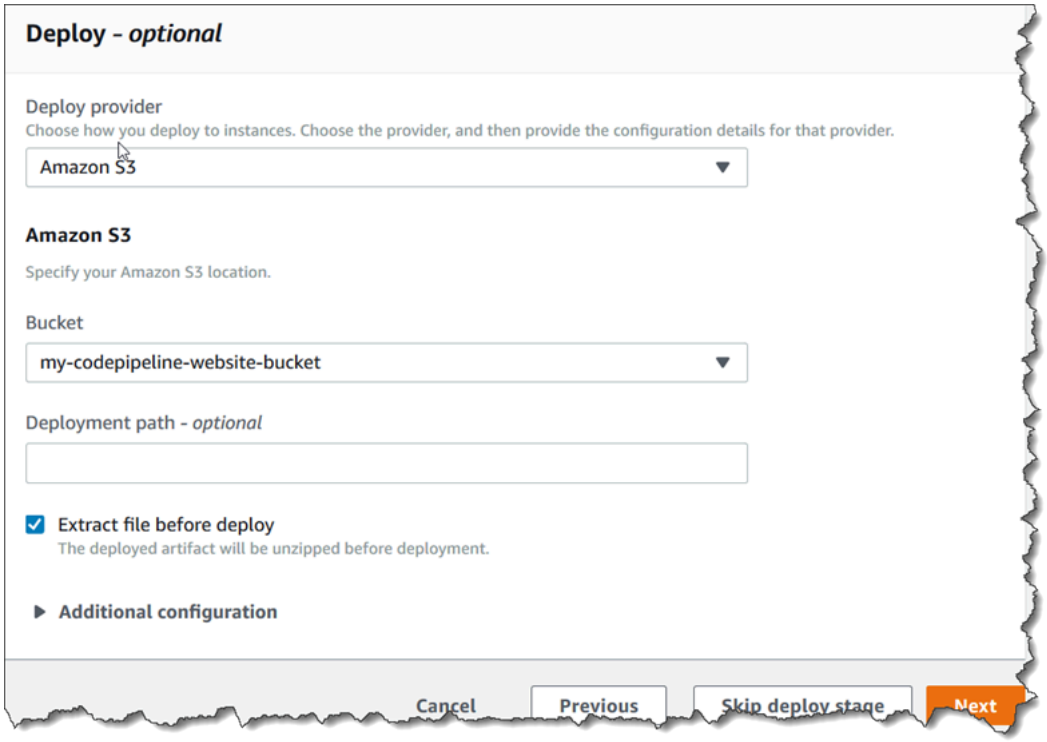
   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Amazon S3.

   b. Geben Sie unter Bucket den Namen Ihres öffentlichen Buckets ein.

   c. Wählen Sie Extract file before deploy (Datei vor Bereitstellung extrahieren).

   > **Note**
   >
   > Die Bereitstellung schlägt fehl, wenn Sie nicht Datei vor der Bereitstellung extrahieren auswählen. Das liegt daran, dass die AWS CodeCommit Aktion in Ihrer Pipeline Quellartefakte zippt und es sich bei Ihrer Datei um eine ZIP-Datei handelt.

   Wenn die Option Extract file before deploy (Datei vor Bereitstellung extrahieren) aktiviert ist, wird die Option Deployment path (Bereitstellungspfad) angezeigt. Geben Sie den Namen des gewünschten Pfads ein. Dadurch wird eine Ordnerstruktur in Amazon S3 erstellt, in die die Dateien extrahiert werden. Lassen Sie dieses Feld für dieses Tutorial leer.



d. (Optional) Unter Canned ACL (Vordefinierte ACL) können Sie einen Satz vordefinierter Berechtigungen, als vordefinierte ACL bezeichnet, auf die hochgeladenen Artefakte anwenden.

e. (Optional) Geben Sie unter Cache control (Cache-Kontrolle) die Caching-Parameter ein. Sie können hiermit das Caching-Verhalten für Anforderungen/Antworten festlegen. Informationen zu gültigen Werten finden Sie im `Cache-Control`-Header-Feld für HTTP-Operationen.

f. Wählen Sie Weiter aus.

10. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.

11. Nachdem Ihre Pipeline erfolgreich ausgeführt wurde, öffnen Sie die Amazon S3 S3-Konsole und überprüfen Sie, ob Ihre Dateien in Ihrem öffentlichen Bucket wie folgt angezeigt werden:

```
index.html
main.css
graphic.jpg
```

12. Greifen Sie auf Ihren Endpunkt zu, um die Website zu testen. Ihr Endpunkt hat das folgende Format: `http://`*`bucket-name`*`.s3-website-`*`region`*`.amazonaws.com/`.

    Beispielendpunkt: `http://my-bucket.s3-website-us-west-2.amazonaws.com/`.

Die Beispiel-Webseite wird angezeigt.

Schritt 3: Vornehmen einer Änderung an einer beliebigen Quelldatei und Überprüfen der Bereitstellung

Nehmen Sie eine Änderung an Ihren Quelldateien vor und übertragen Sie die Änderung per Push in Ihr Repository. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Überprüfen Sie, ob Ihre Website aktualisiert wurde.

Option 2: Stellen Sie erstellte Archivdateien aus einem S3-Quell-Bucket in Amazon S3 bereit

Bei dieser Option kompilieren die Build-Befehle in Ihrer Build-Phase TypeScript Code in JavaScript Code und stellen die Ausgabe in Ihrem S3-Ziel-Bucket in einem separaten Ordner mit Zeitstempel bereit. Zunächst erstellen Sie TypeScript Code und eine buildspec.yml-Datei. Nachdem Sie die Quelldateien in einer ZIP-Datei kombiniert haben, laden Sie die Quell-ZIP-Datei in Ihren S3-Quell-Bucket hoch und verwenden einen CodeBuild Stagingbereich, um eine ZIP-Datei für eine erstellte Anwendung in Ihrem S3-Ziel-Bucket bereitzustellen. Der kompilierte Code wird in Ihrem Ziel-Bucket als Archiv aufbewahrt.

Voraussetzungen

Sie müssen bereits über Folgendes verfügen:

- Ein S3-Quell-Bucket. Sie können den Bucket verwenden, den Sie unter Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) erstellt haben.
- Ein S3-Ziel-Bucket. Siehe Hosten einer statischen Website auf Amazon S3. Stellen Sie sicher, dass Sie Ihren Bucket genauso erstellen AWS-Region wie die Pipeline, die Sie erstellen möchten.

> **Note**
>
> In diesem Beispiel wird die Bereitstellung von Dateien in einem privaten Bucket veranschaulicht. Aktivieren Sie für Ihren Ziel-Bucket nicht das Website-Hosting und fügen Sie keine Richtlinien an, die dazu führen, dass der Bucket öffentlich verfügbar ist.

Schritt 1: Erstellen und Hochladen von Quelldateien in Ihren S3-Quell-Bucket

In diesem Abschnitt erstellen Sie Ihre Quelldateien und laden sie in den Bucket hoch, der von der Pipeline für Ihre Quellstufe verwendet wird. Dieser Abschnitt enthält eine Anleitung zum Erstellen der folgenden Quelldateien:

- Eine `buildspec.yml` Datei, die für CodeBuild Build-Projekte verwendet wird.
- Die Datei `index.ts`.

So erstellen Sie die Datei „buildspec.yml“

- Erstellen Sie eine Datei mit dem Namen `buildspec.yml` und dem folgenden Inhalt. Diese Build-Befehle installieren TypeScript und verwenden den TypeScript Compiler, um den Code in `index.ts` Code umzuschreiben. JavaScript

```
version: 0.2

phases:
  install:
    commands:
      - npm install -g typescript
  build:
    commands:
      - tsc index.ts
artifacts:
  files:
    - index.js
```

So erstellen Sie die Datei „index.ts“

- Erstellen Sie eine Datei mit dem Namen `index.ts` und dem folgenden Inhalt.

```
interface Greeting {
    message: string;
}

class HelloGreeting implements Greeting {
    message = "Hello!";
}
```

```
function greet(greeting: Greeting) {
    console.log(greeting.message);
}

let greeting = new HelloGreeting();

greet(greeting);
```

So laden Sie Dateien in Ihren S3-Quell-Bucket hoch

1. Ihre Dateien sollten in Ihrem lokalen Verzeichnis wie folgt aussehen:

   ```
   buildspec.yml
   index.ts
   ```

   Zippen Sie die Dateien und geben Sie der Datei den Namen `source.zip`.
2. Wählen Sie in der Amazon S3 S3-Konsole für Ihren Quell-Bucket Upload aus. Wählen Sie Dateien hinzufügen und suchen Sie nach der von Ihnen erstellten ZIP-Datei.
3. Klicken Sie auf Hochladen. Diese Dateien sind das Quellartefakt, das vom Assistenten „Pipeline erstellen“ für Ihre Bereitstellungsaktion in CodePipeline erstellt wurde. Ihre Datei sollte in Ihrem Bucket wie folgt aussehen:

   ```
   source.zip
   ```

Schritt 2: Erstellen der Pipeline

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellstufe mit einer Amazon S3 S3-Aktion, bei der die Quellartefakte die Dateien für Ihre herunterladbare Anwendung sind.
- Eine Bereitstellungsphase mit einer Amazon S3 S3-Bereitstellungsaktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyS3DeployPipeline`** ein.
4. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
5. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
6. Wählen Sie in Step 2: Add source stage (Schritt 2: Quellstufe hinzufügen) unter Source provider (Quellanbieter) die Option Amazon S3. Wählen Sie unter Bucket den Namen Ihres Quell-Buckets aus. Geben Sie unter S3 object key (S3-Objektschlüssel) den Namen Ihrer ZIP-Quelldatei ein. Stellen Sie sicher, dass Sie die Dateierweiterung .zip angeben.

   Wählen Sie Weiter aus.
7. Unter Schritt 3: Build-Stufe hinzufügen:

   a. Wählen Sie unter Build provider (Build-Anbieter) die Option CodeBuild aus.
   b. Wählen Sie Create build project (Build-Projekt erstellen) aus. Auf der Seite Projekt erstellen:
   c. Geben Sie unter Project name (Projektname) einen Namen für dieses Build-Projekt ein.
   d. Wählen Sie für Environment (Umgebung) die Option Managed image (Verwaltetes Image) aus. Wählen Sie für Operating system (Betriebssystem) die Option Ubuntu aus.
   e. Wählen Sie unter Runtime (Laufzeit) die Option Standard aus. Wählen Sie für Runtime version (Laufzeitversion) aws/codebuild/standard:1.0 aus.
   f. Wählen Sie unter Image version (Abbildversion) die Option Always use the latest image for this runtime version (Immer aktuelles Abbild für diese Laufzeitversion verwenden).
   g. Wählen Sie unter Servicerolle Ihre CodeBuild Servicerolle aus, oder erstellen Sie eine.
   h. Wählen Sie unter Build specifications (Build-Spezifikationen) die Option Use a buildspec file (Eine buildspec-Datei verwenden).
   i. Wählen Sie Weiter zu CodePipeline. Eine Meldung mit dem Hinweis, ob die Erstellung des Projekts erfolgreich war, wird angezeigt.
   j. Wählen Sie Weiter aus.
8. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

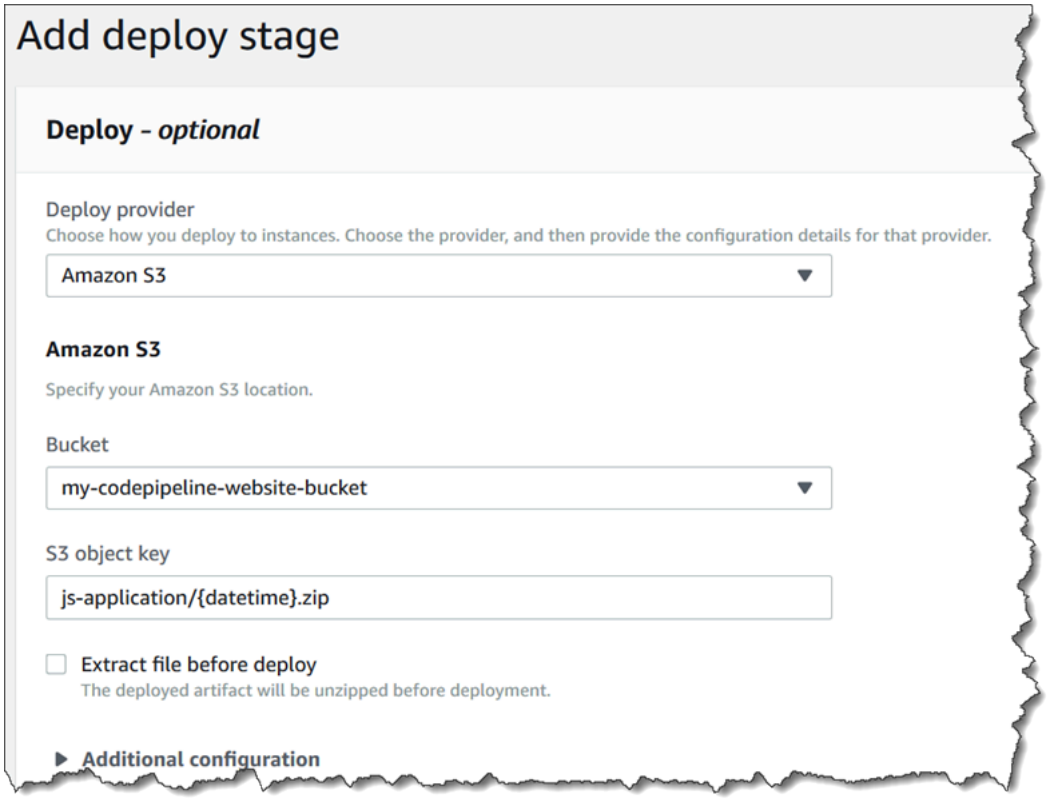
   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Amazon S3.

b. Geben Sie unter Bucket den Namen Ihres S3-Ziel-Buckets ein.

c. Stellen Sie sicher, dass Extract file before deploy (Datei vor Bereitstellung extrahieren) deaktiviert ist.

   Wenn die Option Extract file before deploy (Datei vor Bereitstellung extrahieren) deaktiviert ist, wird die Option S3 object key (S3-Objektschlüssel) angezeigt. Geben Sie den Namen des gewünschten Pfads ein: `js-application/{datetime}.zip`

   Dadurch wird ein `js-application` Ordner in Amazon S3 erstellt, in den die Dateien extrahiert werden. In diesem Ordner wird bei Ausführung Ihrer Pipeline mit der Variablen `{datetime}` ein Zeitstempel für jede Ausgabedatei erstellt.



d. (Optional) Unter Canned ACL (Vordefinierte ACL) können Sie einen Satz vordefinierter Berechtigungen, als vordefinierte ACL bezeichnet, auf die hochgeladenen Artefakte anwenden.

e. (Optional) Geben Sie unter Cache control (Cache-Kontrolle) die Caching-Parameter ein. Sie können hiermit das Caching-Verhalten für Anforderungen/Antworten festlegen. Informationen zu gültigen Werten finden Sie im `Cache-Control`-Header-Feld für HTTP-Operationen.

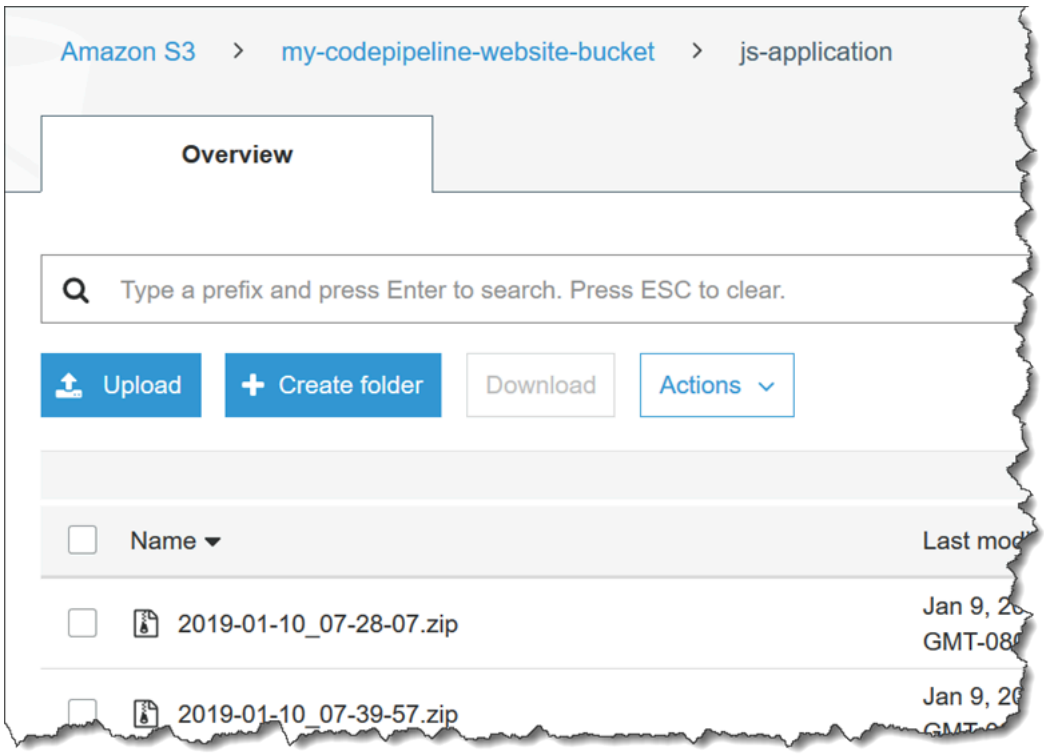
f. Wählen Sie Weiter aus.

9. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.
10. Nachdem Ihre Pipeline erfolgreich ausgeführt wurde, sehen Sie sich Ihren Bucket in der Amazon S3 S3-Konsole an. Vergewissern Sie sich, dass Ihre bereitgestellte ZIP-Datei in Ihrem Ziel-Bucket unter dem Ordner `js-application` angezeigt wird. Die in der JavaScript ZIP-Datei enthaltene Datei sollte sein`index.js`. Die Datei `index.js` enthält die folgende Ausgabe:

```
var HelloGreeting = /** @class */ (function () {
    function HelloGreeting() {
        this.message = "Hello!";
    }
    return HelloGreeting;
}());
function greet(greeting) {
    console.log(greeting.message);
}
var greeting = new HelloGreeting();
greet(greeting);
```

Schritt 3: Vornehmen einer Änderung an einer beliebigen Quelldatei und Überprüfen der Bereitstellung

Nehmen Sie eine Änderung an Ihren Quelldateien vor und laden Sie sie in Ihren Quell-Bucket hoch. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Zeigen Sie Ihren Ziel-Bucket an und vergewissern Sie sich, dass die bereitgestellten Ausgabedateien im Ordner `js-application` enthalten sind (wie hier dargestellt):



Tutorial: Erstellen Sie eine Pipeline, die Ihre serverlose Anwendung auf dem AWS Serverless Application Repository

Sie können AWS CodePipeline es verwenden, um Ihre AWS SAM serverlose Anwendung kontinuierlich an die bereitzustellen. AWS Serverless Application Repository

Dieses Tutorial zeigt, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre serverlose Anwendung zu erstellen, die in gehostet wird, GitHub und sie automatisch auf der AWS Serverless Application Repository zu veröffentlichen. Die Pipeline wird GitHub als Quellanbieter und CodeBuild als Buildanbieter verwendet. Um Ihre serverlose Anwendung auf dem zu veröffentlichen AWS Serverless Application Repository, stellen Sie eine Anwendung (von AWS Serverless Application Repository) bereit und ordnen die von dieser Anwendung erstellte Lambda-Funktion als Invoke-Aktionsanbieter in Ihrer Pipeline zu. Anschließend können Sie kontinuierlich Anwendungsupdates für die bereitstellen AWS Serverless Application Repository, ohne Code schreiben zu müssen.

> **Important**
>
> Viele der Aktionen, die Sie Ihrer Pipeline in diesem Verfahren hinzufügen, beinhalten AWS Ressourcen, die Sie erstellen müssen, bevor Sie die Pipeline erstellen. AWS Ressourcen für Ihre Quellaktionen müssen immer in derselben AWS Region erstellt werden, in der Sie

Ihre Pipeline erstellen. Wenn Sie Ihre Pipeline beispielsweise in der Region USA Ost (Ohio) erstellen, muss sich Ihr CodeCommit Repository in der Region USA Ost (Ohio) befinden. Sie können beim Erstellen Ihrer Pipeline regionsübergreifende Aktionen hinzufügen. AWS Ressourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

Bevor Sie beginnen

In diesem Tutorial wird von Folgendem ausgegangen.

- Sie sind mit AWS Serverless Application Model (AWS SAM) und dem AWS Serverless Application Repository vertraut.
- Sie haben eine serverlose Anwendung gehostet GitHub , die Sie AWS Serverless Application Repository mithilfe der AWS SAM CLI auf der veröffentlicht haben. Informationen zum Veröffentlichen einer Beispielanwendung auf der AWS Serverless Application Repository finden Sie unter Schnellstart: Veröffentlichen von Anwendungen im AWS Serverless Application Repository Entwicklerhandbuch. Informationen zum Veröffentlichen Ihrer eigenen Anwendung auf dem AWS Serverless Application Repository finden Sie unter Veröffentlichen von Anwendungen mit der AWS SAM CLI im AWS Serverless Application Model Entwicklerhandbuch.

Schritt 1: Erstellen einer buildspec.yml-Datei

Erstellen Sie eine `buildspec.yml` Datei mit dem folgenden Inhalt und fügen Sie sie dem GitHub Repository Ihrer serverlosen Anwendung hinzu. Ersetzen Sie *`template.yml`* durch die AWS SAM Vorlage Ihrer Anwendung und *`bucketname`* durch den S3-Bucket, in dem Ihre verpackte Anwendung gespeichert ist.

```
version: 0.2
phases:
  install:
    runtime-versions:
        python: 3.8
  build:
    commands:
      - sam package --template-file template.yml --s3-bucket bucketname --output-
template-file packaged-template.yml
```

```
artifacts:
  files:
    - packaged-template.yml
```

Schritt 2: Erstellen und Konfigurieren Ihrer Pipeline

Gehen Sie wie folgt vor, um Ihre Pipeline in dem Bereich zu erstellen, in AWS-Region dem Sie Ihre serverlose Anwendung veröffentlichen möchten.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wechseln Sie bei Bedarf zu dem AWS-Region Ort, an dem Sie Ihre serverlose Anwendung veröffentlichen möchten.
3. Wählen Sie Create pipeline (Pipeline erstellen) aus. Geben Sie auf der Seite Choose pipeline settings (Auswahl der Pipelineeinstellungen) unter Pipeline name (Pipelinename) den Namen für Ihre Pipeline ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie auf der Seite Quellstufe hinzufügen unter Quellanbieter die Option aus GitHub.
8. Wählen Sie unter Verbindung eine bestehende Verbindung aus, oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.
9. Wählen Sie unter Repository Ihr GitHub Quell-Repository aus.
10. Wählen Sie unter Branch Ihren GitHub Branch aus.
11. Behalten Sie die übrigen Standardeinstellungen für die Quellaktion bei. Wählen Sie Weiter aus.
12. Fügen Sie auf der Seite Add build stage (Build-Phase hinzufügen) eine Build-Phase hinzu:

    a. Wählen Sie unter Build provider (Build-Anbieter) die Option AWS CodeBuild aus. Verwenden Sie als Region die Region der Pipeline.

    b. Wählen Sie Create project (Projekt erstellen) aus.

c. Geben Sie unter Project name (Projektname) einen Namen für dieses Build-Projekt ein.

d. Wählen Sie für Environment image (Umgebungs-Image) die Option Managed image (Verwaltetes Image) aus. Wählen Sie für Operating system (Betriebssystem) die Option Ubuntu aus.

e. Wählen Sie für Runtime (Laufzeit) und Runtime version (Laufzeitversion) die erforderliche Laufzeit und die Version für Ihre serverlose Anwendung.

f. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle) aus.

g. Wählen Sie unter Build specifications (Build-Spezifikationen) die Option Use a buildspec file (Eine buildspec-Datei verwenden).

h. Wählen Sie Weiter zu. CodePipeline Dadurch wird die CodePipeline Konsole geöffnet und ein CodeBuild Projekt erstellt, das die `buildspec.yml` in Ihrem Repository für die Konfiguration verwendet. Das Build-Projekt verwendet eine Servicerolle zur Verwaltung von AWS-Service Berechtigungen. Dieser Vorgang kann einige Minuten dauern.

i. Wählen Sie Weiter aus.

13. Wählen Sie auf der Seite Add deploy stage (Bereitstellungsphase hinzufügen) die Option Skip deploy stage (Bereitstellungsphase übergehen) und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Skip (Übergehen) auswählen. Wählen Sie Weiter aus.

14. Wählen Sie Create pipeline (Pipeline erstellen) aus. Sie sollten ein Diagramm sehen, das Ihre Pipeline-Quell- und -Build-Stufen anzeigt.

15. Erteilen Sie der CodeBuild Servicerolle die Berechtigung, auf den S3-Bucket zuzugreifen, in dem Ihre verpackte Anwendung gespeichert ist.

    a. Wählen Sie in der Build-Phase Ihrer neuen Pipeline CodeBuild.

    b. Wählen Sie die Registerkarte Build details (Build-Details) aus.

    c. Wählen Sie unter Umgebung die CodeBuild Servicerolle aus, um die IAM-Konsole zu öffnen.

    d. Erweitern Sie die Auswahl für `CodeBuildBasePolicy` und wählen Sie Edit policy (Richtlinie bearbeiten).

    e. Wählen Sie JSON.

    f. Fügen Sie eine neue Richtlinienanweisung mit den folgenden Inhalten hinzu. Die Anweisung ermöglicht es CodeBuild , Objekte in den S3-Bucket zu legen, in dem Ihre verpackte Anwendung gespeichert ist. Ersetzen Sie *`bucketname (Bucket-Name)`* durch den Namen Ihres S3-Buckets.

```
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::bucketname/*"
            ],
            "Action": [
                "s3:PutObject"
            ]
        }
```

g. Wählen Sie Richtlinie prüfen.

h. Wählen Sie Änderungen speichern aus.

Schritt 3: Bereitstellen der Veröffentlichungsanwendung

Gehen Sie wie folgt vor, um die Anwendung bereitzustellen, die die Lambda-Funktion enthält, die die Veröffentlichung auf dem AWS Serverless Application Repository durchführt. Diese Anwendung ist aws-serverless-codepipeline-serverlessrepo-publish.

> **Note**
>
> Sie müssen die Anwendung in derselben AWS-Region Pipeline bereitstellen.

1. Gehen Sie zur Seite der Anwendung und wählen Sie Deploy (Bereitstellen).
2. Wählen Sie I acknowledge that this app creates custom IAM roles (Ich bestätige, dass diese Anwendung benutzerdefinierte IAM-Rollen erstellt).
3. Wählen Sie Bereitstellen.
4. Wählen Sie View AWS CloudFormation Stack, um die AWS CloudFormation Konsole zu öffnen.
5. Erweitern Sie den Abschnitt Resources (Ressourcen). Sie sehen ServerlessRepoPublish, welcher vom Typ ist AWS::Lambda::Function. Notieren Sie sich die physische ID dieser Ressource für den nächsten Schritt. Sie verwenden diese physische ID, wenn Sie die neue Veröffentlichungsaktion in erstellen CodePipeline.

Schritt 4: Erstellen der Veröffentlichungsaktion

Führen Sie die folgenden Schritte aus, um die Veröffentlichungsaktion in Ihrer Pipeline zu erstellen.

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie im linken Navigationsbereich die Pipeline aus, die Sie bearbeiten möchten.
3. Wählen Sie Bearbeiten aus.
4. Wählen Sie nach der letzten Phase Ihrer aktuellen Pipeline + Add stage (Phase hinzufügen). Geben Sie unter Stage name (Name der Phase) einen Namen ein, etwa **`Publish`**, und wählen Sie Add stage (Phase hinzufügen).
5. Wählen Sie in der neuen Phase + Add action group (Aktionsgruppe hinzufügen).
6. Geben Sie einen Aktionsnamen ein. Wählen Sie unter Action provider (Aktionsanbieter) in Invoke, die Option AWS Lambda.
7. Wählen Sie unter Eingabeartefakte die Option aus BuildArtifact.
8. Wählen Sie unter Funktionsname die physische ID der Lambda-Funktion aus, die Sie im vorherigen Schritt notiert haben.
9. Wählen Sie für die Aktion Save (Speichern).
10. Wählen Sie für die Phase Done (Fertig) aus.
11. Wählen Sie rechts oben Save (Speichern) aus.
12. Um Ihre Pipeline zu überprüfen, nehmen Sie eine Änderung an Ihrer Anwendung in GitHub vor. Ändern Sie beispielsweise die Beschreibung der Anwendung im `Metadata` Abschnitt Ihrer AWS SAM Vorlagendatei. Übernehmen Sie die Änderung und übertragen Sie sie an Ihre GitHub Filiale. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Wenn die Pipeline fertiggestellt ist, prüfen Sie im AWS Serverless Application Repository, ob Ihre Anwendung mit Ihrer Änderung aktualisiert wurde.

Tutorial: Variablen mit Lambda-Aufrufaktionen verwenden

Eine Lambda-Aufrufaktion kann Variablen aus einer anderen Aktion als Teil ihrer Eingabe verwenden und neue Variablen zusammen mit ihrer Ausgabe zurückgeben. Hinweise zu Variablen für Aktionen in finden Sie CodePipeline unter. Variablen

Am Ende dieses Tutorials werden Sie Folgendes haben::

- Eine Lambda-Aufrufaktion, die:

  - Verbraucht die `CommitId` Variable aus einer Quellaktion CodeCommit
  - drei neue Variablen ausgibt: `dateTime`, `testRunId` und `region`
- Eine manuelle Genehmigungsaktion, die die neuen Variablen aus Ihrer Lambda-Aufrufaktion verwendet, um eine Test-URL und eine Testlauf-ID bereitzustellen
- Eine mit den neuen Aktionen aktualisierte Pipeline

Themen

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Sie können die Pipeline erstellen oder verwenden, in der sich die Quelle befindet. CodeCommit Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository)
- Bearbeiten Sie Ihre bestehende Pipeline so, dass die CodeCommit Quellaktion einen Namespace hat. Weisen Sie der Aktion `SourceVariables` den Namespace zu.

Schritt 1: Erstellen einer Lambda-Funktion

Gehen Sie wie folgt vor, um eine Lambda-Funktion und eine Lambda-Ausführungsrolle zu erstellen. Sie fügen die Lambda-Aktion zu Ihrer Pipeline hinzu, nachdem Sie die Lambda-Funktion erstellt haben.

So erstellen Sie eine Lambda-Funktion und eine Ausführungsrolle

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
2. Wählen Sie Funktion erstellen. Lassen Sie Author from scratch (Von Grund auf erstellen) ausgewählt.
3. Geben Sie unter Funktionsname den Namen Ihrer Funktion ein, z. B. **`myInvokeFunction`**. Lassen Sie in Runtime (Laufzeit) die Standardoption ausgewählt.

4. Erweitern Sie Choose or create an execution role (Ausführungsrolle auswählen oder erstellen). Wählen Sie Create a new role with basic Lambda permissions (Eine neue Rolle mit den grundlegenden Lambda-Berechtigungen erstellen) aus.
5. Wählen Sie Funktion erstellen.
6. Um eine Variable aus einer anderen Aktion zu verwenden, muss sie an die `UserParameters` in der Lambda-Aufrufaktionskonfiguration übergeben werden. Sie werden die Aktion in unserer Pipeline später in diesem Tutorial konfigurieren, Sie fügen jedoch den Code hinzu, vorausgesetzt, dass die Variable übergeben wird.

   ```
   const commitId =
   event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;
   ```

   Um neue Variablen zu erzeugen, setzen Sie eine Eigenschaft mit der Bezeichnung `outputVariables` auf die Eingabe zu `putJobSuccessResult`. Beachten Sie, dass Sie keine Variablen als Teil eines `putJobFailureResult` erzeugen können.

   ```
   const successInput = {
     jobId: jobId,
     outputVariables: {
       testRunId: Math.floor(Math.random() * 1000).toString(),
       dateTime: Date(Date.now()).toString(),
       region: lambdaRegion
     }
   };
   ```

   Lassen Sie in Ihrer neuen Funktion Edit code inline (Code inline bearbeiten) ausgewählt, und fügen Sie den folgenden Beispielcode unter `index.js` ein.

   ```
   var AWS = require('aws-sdk');

   exports.handler = function(event, context) {
       var codepipeline = new AWS.CodePipeline();

       // Retrieve the Job ID from the Lambda action
       var jobId = event["CodePipeline.job"].id;

       // Retrieve the value of UserParameters from the Lambda action configuration in
    CodePipeline,
       // in this case it is the Commit ID of the latest change of the pipeline.
   ```

```
    var params =
event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;

    // The region from where the lambda function is being executed.
    var lambdaRegion = process.env.AWS_REGION;

    // Notify CodePipeline of a successful job
    var putJobSuccess = function(message) {
        var params = {
            jobId: jobId,
            outputVariables: {
                testRunId: Math.floor(Math.random() * 1000).toString(),
                dateTime: Date(Date.now()).toString(),
                region: lambdaRegion
            }
        };
        codepipeline.putJobSuccessResult(params, function(err, data) {
            if(err) {
                context.fail(err);
            } else {
                context.succeed(message);
            }
        });
    };

    // Notify CodePipeline of a failed job
    var putJobFailure = function(message) {
        var params = {
            jobId: jobId,
            failureDetails: {
                message: JSON.stringify(message),
                type: 'JobFailed',
                externalExecutionId: context.invokeid
            }
        };
        codepipeline.putJobFailureResult(params, function(err, data) {
            context.fail(message);
        });
    };

    var sendResult = function() {
        try {
            console.log("Testing commit - " + params);
```

```
            // Your tests here

            // Succeed the job
            putJobSuccess("Tests passed.");
        } catch (ex) {
            // If any of the assertions failed then fail the job
            putJobFailure(ex);
        }
    };

    sendResult();
};
```

7. Wählen Sie Speichern.
8. Kopieren Sie den Amazon-Ressourcenname (ARN) oben im Bildschirm.
9. Öffnen Sie als letzten Schritt die AWS Identity and Access Management (IAM-) Konsole unter https://console.aws.amazon.com/iam/. Ändern Sie die Lambda-Ausführungsrolle, um die folgende Richtlinie hinzuzufügen: AWSCodePipelineCustomActionAccess. Die Schritte zum Erstellen einer Lambda-Ausführungsrolle oder zum Ändern der Rollenrichtlinie finden Sie unter Schritt 2: Lambda-Funktion erstellen .

Schritt 2: Fügen Sie Ihrer Pipeline eine Lambda-Aufrufaktion und eine manuelle Genehmigungsaktion hinzu

In diesem Schritt fügen Sie Ihrer Pipeline eine Lambda-Aufrufaktion hinzu. Sie fügen die Aktion als Teil einer Phase namens Test hinzu. Der Aktionstyp ist eine Aufrufaktion. Anschließend fügen Sie nach der Aufrufaktion eine manuelle Genehmigungsaktion hinzu.

So fügen Sie der Pipeline eine Lambda-Aktion und eine manuelle Genehmigungsaktion hinzu

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Die Namen aller Pipelines, die mit Ihrem AWS Konto verknüpft sind, werden angezeigt. Wählen Sie die Pipeline aus, in der Sie die Aktion hinzufügen möchten.
2. Fügen Sie die Lambda-Testaktion zu Ihrer Pipeline hinzu.

   a. Um die Pipeline zu bearbeiten, wählen Sie Edit (Bearbeiten). Fügen Sie nach der Quellaktion einen Schritt in der vorhandenen Pipeline hinzu. Geben Sie einen Namen für die Phase ein, z. B. **`Test`**.

b. Wählen Sie in der neuen Phase das Symbol zum Hinzufügen einer Aktion aus. Geben Sie unter Action name (Aktionsname) den Namen der Aufrufaktion ein, z. B. **`Test_Commit`**.

c. Wählen Sie unter Aktionsanbieter die Option aus AWS Lambda.

d. Wählen Sie unter Input artifacts (Eingabeartefakte) den Namen des Ausgabeartefakts Ihrer Quellaktion aus, z. B. `SourceArtifact`.

e. Wählen Sie unter Funktionsname den Namen der Lambda-Funktion aus, die Sie erstellt haben.

f. Geben Sie im Feld Benutzerparameter die Variablensyntax für die CodeCommit Commit-ID ein. Dadurch wird die Ausgabevariable erstellt, die jedes Mal überprüft und genehmigt wird, wenn die Pipeline ausgeführt wird.

```
#{SourceVariables.CommitId}
```

g. Fügen Sie unter Variable namespace (Variablennamespace) den Namespace-Namen hinzu, z. B. **`TestVariables`**.

h. Wählen Sie Erledigt aus.

3. Fügen Sie der Pipeline die manuelle Genehmigungsaktion hinzu.

a. Wenn sich Ihre Pipeline noch im Bearbeitungsmodus befindet, fügen Sie nach der Aufrufaktion eine Phase hinzu. Geben Sie einen Namen für die Phase ein, z. B. **`Approval`**.

b. Wählen Sie in der neuen Phase das Symbol zum Hinzufügen einer Aktion aus. Geben Sie unter Action name (Aktionsname) den Namen der Genehmigungsaktion ein, z. B. **`Change_Approval`**.

c. Wählen Sie unter Action provider (Aktionsanbieter) die Option Manual approval (Manuelle Genehmigung).

d. Erstellen Sie unter URL for review (URL zur Überprüfung) die URL, indem Sie die Variablensyntax für die `region`-Variable und die `CommitId`-Variable hinzufügen. Stellen Sie sicher, dass Sie die Namespaces verwenden, die denjenigen Aktionen zugewiesen sind, die die Ausgabevariablen bereitstellen.

In diesem Beispiel hat die URL mit der Variablensyntax für eine CodeCommit Aktion den Standard-Namespace`SourceVariables`. Die Ausgabevariable der Lambda-Region hat den Namespace „`TestVariables`“. Die URL sieht wie folgt aus:

```
https://#{TestVariables.region}.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/commit/#{SourceVariables.CommitId}
```

Erstellen Sie unter Comments (Kommentare) den Text der Genehmigungsmeldung, indem Sie die Variablensyntax für die `testRunId`-Variable hinzufügen. In diesem Beispiel hat die URL mit der Variablensyntax für die `testRunId`-Lambda-Ausgabevariable den Namespace „`TestVariables`“. Geben Sie die folgende Nachricht ein.

```
Make sure to review the code before approving this action. Test Run ID:
 #{TestVariables.testRunId}
```

4. Wählen Sie Done (Fertig) aus, um den Bearbeitungsbildschirm für die Aktion zu schließen. Wählen Sie dann Done (Fertig) aus, um den Bearbeitungsbildschirm für die Phase zu schließen. Zum Speichern der Pipeline wählen Sie Done (Fertig) aus. Die abgeschlossene Pipeline enthält jetzt eine Struktur mit Quell-, Test-, Genehmigungs- und Bereitstellungsschritten.

   Wählen Sie Release change (Änderung freigeben) aus, um die letzte Änderung durch die Pipeline-Struktur auszuführen.

5. Wenn die Pipeline die manuelle Genehmigungsphase erreicht hat, wählen Sie Review (Überprüfen) aus. Die aufgelösten Variablen werden als URL für die Commit-ID angezeigt. Ihr Genehmiger kann die URL auswählen, um das Commit anzuzeigen.
6. Nachdem die Pipeline erfolgreich ausgeführt wurde, können Sie die Variablenwerte auch auf der Seite „Aktionsausführungsverlauf“ anzeigen.

Tutorial: Verwenden Sie eine AWS Step Functions Aufrufaktion in einer Pipeline

Sie können sie verwenden AWS Step Functions , um Zustandsmaschinen zu erstellen und zu konfigurieren. In diesem Tutorial erfahren Sie, wie Sie einer Pipeline eine Aufrufaktion hinzufügen, mit der die Ausführung von Zustandsautomaten über Ihre Pipeline aktiviert wird.

In diesem Tutorial führen Sie folgende Aufgaben aus:

- Erstellen Sie einen Standard-Zustandsmaschine in AWS Step Functions.

- Sie geben die Eingabedaten für den Zustandsautomaten direkt im JSON-Format ein. Sie können die State Machine-Eingabedatei auch in einen Amazon Simple Storage Service (Amazon S3) - Bucket hochladen.
- Sie aktualisieren Ihre Pipeline, indem Sie die Zustandsautomaten-Aktion hinzufügen.

Themen

Voraussetzung: Erstellen oder Auswählen einer einfachen Pipeline

In diesem Tutorial fügen Sie einer vorhandenen Pipeline eine Aufrufaktion hinzu. Sie können die Pipeline verwenden, die Sie in Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) oder Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository) erstellt haben.

Sie verwenden in für diesem Beispiel eine vorhandene Pipeline mit einer Quellaktion und einer mindestens zweistufigen Struktur, jedoch keine Quellartefakte.

> **Note**
>
> Möglicherweise müssen Sie die Servicerolle, die von der Pipeline verwendet wird, mit zusätzlichen Berechtigungen aktualisieren, die zum Ausführen dieser Aktion erforderlich sind. Öffnen Sie dazu die AWS Identity and Access Management (IAM-) Konsole, suchen Sie nach der Rolle und fügen Sie dann die Berechtigungen zur Rollenrichtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.

Schritt 1: Erstellen des Beispielzustandsautomaten

Erstellen Sie in der Step Functions Functions-Konsole mithilfe der `HelloWorld` Beispielvorlage eine Zustandsmaschine. Anweisungen finden Sie unter Create a State Machine im AWS Step Functions Entwicklerhandbuch.

Schritt 2: Fügen Sie Ihrer Pipeline eine Aktion zum Aufrufen von Step Functions hinzu

Fügen Sie Ihrer Pipeline wie folgt eine Aktion zum Aufrufen von Step Functions hinzu:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Auf diese Weise wird eine detaillierte Ansicht der Pipeline geöffnet (einschließlich des Status der Aktionen in den einzelnen Stufen der Pipeline).
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie in der zweiten Phase Ihrer einfachen Pipeline die Option Edit stage (Phase bearbeiten)aus. Wählen Sie Löschen aus. Dadurch wird die zweite Phase gelöscht, die Sie nicht mehr benötigen.
5. Wählen Sie unten im Diagramm + Add stage (+ Stufe hinzufügen) aus.
6. Geben Sie unter Stage Name (Name der Phase) einen Namen für die Phase ein, etwa **`Invoke`**, und wählen Sie dann Add stage (Phase hinzufügen).
7. Wählen Sie + Add action group (Aktionsgruppe hinzufügen).
8. Geben Sie unter Action name (Aktionsname) einen Namen ein, z. B. **`Invoke`**.
9. Wählen Sie unter Aktionsanbieter die Option AWS Step Functions aus. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region.
10. Wählen Sie unter Input artifacts (Eingabeartefakte) die Option `SourceArtifact` aus.
11. Wählen Sie unter State machine ARN (Zustandsautomaten-ARN) den von Ihnen zuvor für den Zustandsautomaten erstellten Amazon-Ressourcennamen (ARN) aus.
12. (Optional) Geben Sie unter Execution name prefix (Präfix des Ausführungsnamens) ein Präfix ein, das der Ausführungs-ID des Zustandsautomaten hinzugefügt werden soll.
13. Wählen Sie unter Input type (Eingabetyp) die Option Literal.
14. Geben Sie unter Input (Eingabe) die Eingabe im JSON-Format ein, den der `HelloWorld` Beispielzustandsautomat erwartet.

> **Note**
>
> Die Eingabe für die State-Machine-Ausführung unterscheidet sich von dem Begriff, der CodePipeline zur Beschreibung von Eingabeartefakten für Aktionen verwendet wird.

Geben Sie in diesem Beispiel den folgenden JSON-Code ein:

```
{"IsHelloWorldExample": true}
```

15. Wählen Sie Erledigt aus.
16. Wählen Sie die Phase aus, die Sie bearbeiten, und wählen Sie dann Done (Fertig). Wählen Sie im AWS CodePipeline -Fenster Save (Speichern) und dann in der Warnmeldung Save (Speichern).
17. Um Ihre Änderungen zu übertragen und eine Pipelineausführung zu starten, wählen Sie Release change (Änderung freigeben) und dann Release (Freigeben).
18. Wählen Sie in Ihrer abgeschlossenen Pipeline in Ihrer Aufrufaktion AWS Step Functions aus. Sehen Sie sich in der AWS Step Functions Konsole Ihre State-Machine-Ausführungs-ID an. Die ID enthält den Namen des Zustandsautomaten `HelloWorld` und die Ausführungs-ID des Zustandsautomaten mit dem Präfix `my-prefix`.

```
arn:aws:states:us-west-2:account-ID:execution:HelloWorld:my-
prefix-0d9a0900-3609-4ebc-925e-83d9618fcca1
```

Tutorial: Eine Pipeline erstellen, die AWS AppConfig als Bereitstellungsanbieter verwendet wird

In diesem Tutorial konfigurieren Sie eine Pipeline, die kontinuierlich Konfigurationsdateien bereitstellt und in Ihrer Bereitstellungsphase AWS AppConfig als Bereitstellungsaktionsanbieter dient.

Themen

- Schritt 3: Erstellen Ihrer Pipeline
- Schritt 4: Nehmen Sie eine Änderung an einer beliebigen Quelldatei vor und überprüfen Sie die Bereitstellung

Voraussetzungen

Bevor Sie beginnen, müssen Sie die folgenden Schritte ausführen:

- In diesem Beispiel wird eine S3-Quelle für Ihre Pipeline verwendet. Erstellen oder verwenden Sie einen Amazon S3 S3-Bucket mit aktivierter Versionierung. Sie können den Anweisungen unter Schritt 1: Erstellen eines S3-Buckets für Ihre Anwendung folgen, um einen S3-Bucket zu erstellen.

Schritt 1: Erstellen Sie Ihre Ressourcen AWS AppConfig

In diesem Abschnitt erstellen Sie die folgenden Ressourcen:

- Eine Anwendung AWS AppConfig ist eine logische Codeeinheit, die Ihren Kunden Funktionen zur Verfügung stellt.
- Eine Umgebung in AWS AppConfig ist eine logische Bereitstellungsgruppe von AppConfig Zielen, z. B. Anwendungen in einer Beta- oder Produktionsumgebung.
- Ein Konfigurationsprofil ist eine Sammlung von Einstellungen, die das Verhalten Ihrer Anwendung beeinflussen. Das Konfigurationsprofil ermöglicht AWS AppConfig den Zugriff auf Ihre Konfiguration an ihrem gespeicherten Speicherort.
- (Optional) Eine Bereitstellungsstrategie AWS AppConfig definiert das Verhalten einer Konfigurationsbereitstellung, z. B. wie viel Prozent der Clients die neu bereitgestellte Konfiguration zu einem bestimmten Zeitpunkt während einer Bereitstellung erhalten sollen.

Um eine Anwendung, eine Umgebung, ein Konfigurationsprofil und eine Bereitstellungsstrategie zu erstellen

1. Melden Sie sich bei der an AWS Management Console.
2. Verwenden Sie die Schritte in den folgenden Themen, um Ihre Ressourcen in zu erstellen AWS AppConfig.

   - Erstellen Sie eine Anwendung.
   - Erstellen Sie eine Umgebung.

- Erstellen Sie ein AWS CodePipeline Konfigurationsprofil.
- (Optional) Wählen Sie eine vordefinierte Bereitstellungsstrategie oder erstellen Sie Ihre eigene.

Schritt 2: Laden Sie Dateien in Ihren S3-Quell-Bucket hoch

Erstellen Sie in diesem Abschnitt Ihre Konfigurationsdatei (en). Dann komprimieren Sie Ihre Quelldateien und übertragen Sie sie in den Bucket, den die Pipeline für Ihre Quellphase verwendet.

Um Konfigurationsdateien zu erstellen

1. Erstellen Sie eine `configuration.json` Datei für jede Konfiguration in jeder Region. Fügen Sie den folgenden Inhalt hinzu:

   ```
   Hello World!
   ```

2. Gehen Sie wie folgt vor, um Ihre Konfigurationsdateien zu komprimieren und hochzuladen.

Um Quelldateien zu komprimieren und hochzuladen

1. Erstellen Sie eine .zip-Datei mit Ihren Dateien und geben Sie der ZIP-Datei einen Namen. `configuration-files.zip` Beispielsweise kann Ihre ZIP-Datei die folgende Struktur haben:

   ```
   .
   ### appconfig-configurations
       ### MyConfigurations
           ### us-east-1
           #   ### configuration.json
           ### us-west-2
               ### configuration.json
   ```

2. Wählen Sie in der Amazon S3 S3-Konsole für Ihren Bucket Upload und folgen Sie den Anweisungen, um Ihre ZIP-Datei hochzuladen.

Schritt 3: Erstellen Ihrer Pipeline

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellstufe mit einer Amazon S3 S3-Aktion, bei der die Quellartefakte die Dateien für Ihre Konfiguration sind.

- Eine Bereitstellungsphase mit einer AppConfig Bereitstellungsaktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyAppConfigPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie in Step 2: Add source stage (Schritt 2: Quellstufe hinzufügen) unter Source provider (Quellanbieter) die Option Amazon S3. Wählen Sie in Bucket den Namen Ihres S3-Quell-Buckets aus.

   Geben Sie im Feld S3-Objektschlüssel den Namen Ihrer .zip-Datei ein:`configuration-files.zip`.

   Wählen Sie Weiter aus.
8. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken.

   Wählen Sie Weiter aus.
9. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option AWS AppConfig aus.

   b. Wählen Sie unter Anwendung den Namen der Anwendung aus, in AWS AppConfig der Sie sie erstellt haben. Das Feld zeigt die ID für Ihre Anwendung.

c. Wählen Sie unter Umgebung den Namen der Umgebung aus, in der Sie erstellt haben AWS AppConfig. Das Feld zeigt die ID für Ihre Umgebung.

d. Wählen Sie unter Konfigurationsprofil den Namen des Konfigurationsprofils aus, in dem Sie es erstellt haben AWS AppConfig. Das Feld zeigt die ID für Ihr Konfigurationsprofil.

e. Wählen Sie unter Bereitstellungsstrategie den Namen Ihrer Bereitstellungsstrategie aus. Dabei kann es sich entweder um eine Bereitstellungsstrategie handeln, die Sie in erstellt haben, AppConfig oder um eine, die Sie unter vordefinierten Bereitstellungsstrategien ausgewählt haben AppConfig. Das Feld zeigt die ID für Ihre Bereitstellungsstrategie.

f. Geben Sie im Feld Konfigurationspfad für das Eingabeartefakt den Dateipfad ein. Stellen Sie sicher, dass Ihr Konfigurationspfad für das Eingabeartefakt mit der Verzeichnisstruktur in Ihrer S3-Bucket-ZIP-Datei übereinstimmt. Geben Sie für dieses Beispiel den folgenden Dateipfad ein:. `appconfig-configurations/MyConfigurations/us-west-2/configuration.json`

g. Wählen Sie Weiter aus.

10. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.

Schritt 4: Nehmen Sie eine Änderung an einer beliebigen Quelldatei vor und überprüfen Sie die Bereitstellung

Nehmen Sie eine Änderung an Ihren Quelldateien vor und laden Sie die Änderung in Ihren Bucket hoch. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Überprüfen Sie anhand der Version, ob Ihre Konfiguration verfügbar ist.

Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden

Sie können die Option „Vollständiges Klonen“ für Ihre GitHub Quellaktion unter auswählen CodePipeline. Verwenden Sie diese Option, um CodeBuild Befehle für Git-Metadaten in Ihrer Pipeline-Build-Aktion auszuführen.

In diesem Tutorial erstellen Sie eine Pipeline, die eine Verbindung zu Ihrem GitHub Repository herstellt, die vollständige Klonoption für Quelldaten verwendet und einen CodeBuild Build ausführen, der Ihr Repository klont und Git-Befehle für das Repository ausführt.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Afrika (Kapstadt), Naher Osten (Bahrain), Europa (Zürich) und AWS GovCloud (USA West) nicht verfügbar. Hinweise zu anderen verfügbaren Aktionen finden Sie unterProdukt- und Serviceintegrationen mit CodePipeline. Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Themen

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Erstellen Sie mit Ihrem GitHub Konto ein GitHub Repository.
- Halten Sie Ihre GitHub Anmeldedaten bereit. Wenn Sie den verwenden AWS Management Console , um eine Verbindung herzustellen, werden Sie aufgefordert, sich mit Ihren GitHub Anmeldeinformationen anzumelden.

Schritt 1: Erstellen Sie eine README-Datei

Nachdem Sie Ihr GitHub Repository erstellt haben, gehen Sie wie folgt vor, um eine README-Datei hinzuzufügen.

1. Loggen Sie sich in Ihr GitHub Repository ein und wählen Sie Ihr Repository aus.
2. Um eine neue Datei zu erstellen, wähle Datei hinzufügen > Neue Datei erstellen. Benennen Sie die Datei`README.md`. Datei und fügen Sie den folgenden Text hinzu.

```
This is a GitHub repository!
```

3. Wählen Sie Commit changes (Änderungen übernehmen) aus.

   Die Datei `README.md` muss sich im Stammverzeichnis Ihres Repositorys befinden.

Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellphase mit einer Verbindung zu Ihrem GitHub Repository und Ihrer Aktion.
- Eine Build-Phase mit einer AWS CodeBuild Build-Aktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich unter https://console.aws.amazon.com/codepipeline/ bei der CodePipeline Konsole an.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyGitHubPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle).

   > **Note**
   >
   > Wenn Sie stattdessen Ihre bestehende CodePipeline Servicerolle verwenden möchten, stellen Sie sicher, dass Sie die `codeconnections:UseConnection` IAM-Berechtigung zu Ihrer Servicerollenrichtlinie hinzugefügt haben. Anweisungen für die CodePipeline Servicerolle finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline Servicerolle.

6. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie unter Artifact store (Artefaktspeicher) die Option Default location (Standardstandort) aus, um den

Standard-Artefakt-Speicherort für die Pipeline in der entsprechenden Region zu verwenden, beispielsweise mit dem Amazon S3-Artefakt-Bucket als Standard.

> **Note**
>
> Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket.

Wählen Sie Next (Weiter).

7. Fügen Sie auf der Seite Step 2: Add source stage (Schritt 2: Quell-Stufe hinzufügen) eine Quellphase hinzu:

   a. Wählen Sie unter Quellanbieter die Option GitHub (Version 2) aus.

   b. Wählen Sie unter Verbindung eine bestehende Verbindung aus, oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.

   c. Wählen Sie unter Repository-Name den Namen Ihres GitHub Repositorys aus.

   d. Wählen Sie unter Branch-Name den Repository-Branch aus, den Sie verwenden möchten.

   e. Stellen Sie sicher, dass die Option Starten der Pipeline bei Änderung des Quellcodes ausgewählt ist.

   f. Wählen Sie unter Ausgabeartefaktformat die Option Vollständiger Klon aus, um die Git-Clone-Option für das Quell-Repository zu aktivieren. Nur Aktionen, die von bereitgestellt werden, CodeBuild können die Git-Clone-Option verwenden. Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um Verbindungen zu verwendenIn diesem Tutorial erfahren Sie, wie Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, sodass Sie diese Option verwenden können.

   Wählen Sie Weiter aus.

8. Fügen Sie unter Add build stage (Build-Phase hinzufügen) eine Build-Phase hinzu:

   a. Wählen Sie unter Build provider (Build-Anbieter) die Option AWS CodeBuild aus. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region.

   b. Wählen Sie Create project (Projekt erstellen) aus.

c. Geben Sie unter Project name (Projektname) einen Namen für dieses Build-Projekt ein.

d. Wählen Sie für Environment image (Umgebungs-Image) die Option Managed image (Verwaltetes Image) aus. Wählen Sie für Operating system (Betriebssystem) die Option Ubuntu aus.

e. Wählen Sie unter Runtime (Laufzeit) die Option Standard aus. Wählen Sie für Image (Abbild) die Option aws/codebuild/standard:5.0 aus.

f. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle) aus.

> **Note**
>
> Notieren Sie sich den Namen Ihrer CodeBuild Servicerolle. Sie benötigen den Rollennamen für den letzten Schritt in diesem Tutorial.

g. Wählen Sie unter BuildSpec bei Build specifications (Build-Spezifikationen) die Option Insert build commands (Build-Befehle einfügen) aus. Wählen Sie Zum Editor wechseln und fügen Sie Folgendes unter Build-Befehle ein.

> **Note**
>
> Stellen Sie im `env` Abschnitt der Build-Spezifikation sicher, dass der Credential-Helper für Git-Befehle aktiviert ist, wie in diesem Beispiel gezeigt.

```
version: 0.2

env:
  git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
      # name: version
    #commands:
      # - command
```

```
      # - command
  pre_build:
    commands:
      - ls -lt
      - cat README.md
  build:
    commands:
      - git log | head -100
      - git status
      - ls
      - git archive --format=zip HEAD > application.zip
  #post_build:
    #commands:
      # - command
      # - command
artifacts:
  files:
    - application.zip
    # - location
  #name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

h. Wählen Sie Weiter zu. CodePipeline Dadurch kehren Sie zur CodePipeline Konsole zurück und erstellen ein CodeBuild Projekt, das Ihre Build-Befehle für die Konfiguration verwendet. Das Build-Projekt verwendet eine Servicerolle zur Verwaltung von AWS-Service Berechtigungen. Dieser Vorgang kann einige Minuten dauern.

i. Wählen Sie Weiter aus.

9. Wählen Sie auf der Seite Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen) die Option Skip deploy stage (Bereitstellungsstufe überspringen) aus und akzeptieren Sie anschließend die Warnmeldung, indem Sie erneut Skip (Überspringen) auswählen. Wählen Sie Weiter aus.

10. Wählen Sie unter Step 5: Review (Schritt 5: Überprüfen) die Option Create pipeline (Pipeline erstellen) aus.

Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um Verbindungen zu verwenden

Der erste Pipeline-Lauf schlägt fehl, da die CodeBuild Servicerolle mit Berechtigungen zur Verwendung von Verbindungen aktualisiert werden muss. Fügen Sie die `codeconnections:UseConnection` IAM-Berechtigung zu Ihrer Servicerollenrichtlinie hinzu. Anweisungen zum Aktualisieren der Richtlinie in der IAM-Konsole finden Sie unter. Fügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, Enterprise Server oder .com GitHub hinzu GitHub GitLab

Schritt 4: Repository-Befehle in der Build-Ausgabe anzeigen

1. Wenn Ihre Servicerolle erfolgreich aktualisiert wurde, wählen Sie in der CodeBuild Phase „Fehlgeschlagen“ die Option „Erneut versuchen“.
2. Wenn die Pipeline erfolgreich ausgeführt wurde, wählen Sie in der erfolgreichen Buildphase die Option Details anzeigen aus.

   Wählen Sie auf der Detailseite die Registerkarte Protokolle aus. Sehen Sie sich die CodeBuild Build-Ausgabe an. Die Befehle geben den Wert der eingegebenen Variablen aus.

   Die Befehle geben den `README.md` Dateiinhalt aus, listen die Dateien im Verzeichnis auf, klonen das Repository, zeigen das Protokoll an und archivieren das Repository als ZIP-Datei.

Tutorial: Vollständigen Klon mit einer CodeCommit Pipeline-Quelle verwenden

Sie können die Option „Vollständiges Klonen“ für Ihre CodeCommit Quellaktion unter auswählen CodePipeline. Verwenden Sie diese Option, um den Zugriff CodeBuild auf Git-Metadaten in Ihrer Pipeline-Build-Aktion zu ermöglichen.

In diesem Tutorial erstellen Sie eine Pipeline, die auf Ihr CodeCommit Repository zugreift, die Option Full Clone für Quelldaten verwendet und einen CodeBuild Build ausführt, der Ihr Repository klont und Git-Befehle für das Repository ausführt.

> **Note**
>
> CodeBuild Aktionen sind die einzigen Downstream-Aktionen, die die Verwendung von Git-Metadaten unterstützen, die mit der Git-Clone-Option verfügbar sind. Außerdem kann deine Pipeline kontoübergreifende Aktionen enthalten, aber die CodeCommit Aktion und die CodeBuild Aktion müssen sich im selben Konto befinden, damit die Option „Vollständiges Klonen“ erfolgreich ist.

Themen

Voraussetzungen

Bevor Sie beginnen, müssen Sie ein CodeCommit Repository in demselben AWS Konto und derselben Region wie Ihre Pipeline erstellen.

Schritt 1: Erstellen Sie eine README-Datei

Gehen Sie wie folgt vor, um Ihrem Quell-Repository eine README-Datei hinzuzufügen. Die README-Datei enthält ein Beispiel für eine Quelldatei, die von der CodeBuild nachgeschalteten Aktion gelesen werden kann.

Um eine README-Datei hinzuzufügen

1. Loggen Sie sich in Ihr Repository ein und wählen Sie Ihr Repository aus.
2. Um eine neue Datei zu erstellen, wähle Datei hinzufügen > Datei erstellen. Benennen Sie die Datei`README.md`. Datei und fügen Sie den folgenden Text hinzu.

   ```
   This is a CodeCommit repository!
   ```

3. Wählen Sie Commit changes (Änderungen übernehmen) aus.

Die Datei `README.md` muss sich im Stammverzeichnis Ihres Repositorys befinden.

Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellphase mit einer CodeCommit Quellaktion.
- Eine Build-Phase mit einer AWS CodeBuild Build-Aktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich unter https://console.aws.amazon.com/codepipeline/ bei der CodePipeline Konsole an.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyCodeCommitPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:
   - Wählen Sie Existing service role (Vorhandene Servicerolle).
   - Wählen Sie Ihre bestehende CodePipeline Servicerolle aus. Diese Rolle muss über die `codecommit:GetRepository` IAM-Berechtigung gemäß Ihrer Servicerollenrichtlinie verfügen. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline Servicerolle.
6. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie Weiter aus.
7. Gehen Sie auf der Seite Schritt 2: Quellstufe hinzufügen wie folgt vor:
   a. Wählen Sie unter Source provider (Quell-Anbieter) die Option CodeCommit.
   b. Wählen Sie unter Repository-Name den Namen Ihres Repositorys aus.
   c. Wählen Sie unter Branchname Ihren Branchennamen aus.
   d. Stellen Sie sicher, dass die Option Starten der Pipeline bei Änderung des Quellcodes ausgewählt ist.

e. Wählen Sie unter Ausgabeartefaktformat die Option Vollständiger Klon aus, um die Git-Clone-Option für das Quell-Repository zu aktivieren. Nur Aktionen, die von bereitgestellt werden, CodeBuild können die Git-Clone-Option verwenden.

Wählen Sie Weiter aus.

8. Gehen Sie in der Phase Build hinzufügen wie folgt vor:

a. Wählen Sie unter Build provider (Build-Anbieter) die Option AWS CodeBuild aus. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region.

b. Wählen Sie Create project (Projekt erstellen) aus.

c. Geben Sie unter Project name (Projektname) einen Namen für dieses Build-Projekt ein.

d. Wählen Sie für Environment image (Umgebungs-Image) die Option Managed image (Verwaltetes Image) aus. Wählen Sie für Operating system (Betriebssystem) die Option Ubuntu aus.

e. Wählen Sie unter Runtime (Laufzeit) die Option Standard aus. Wählen Sie für Image (Abbild) die Option aws/codebuild/standard:5.0 aus.

f. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle) aus.

> **Note**
>
> Notieren Sie sich den Namen Ihrer CodeBuild Servicerolle. Sie benötigen den Rollennamen für den letzten Schritt in diesem Tutorial.

g. Wählen Sie unter BuildSpec bei Build specifications (Build-Spezifikationen) die Option Insert build commands (Build-Befehle einfügen) aus. Wählen Sie Zum Editor wechseln und fügen Sie dann unter Build-Befehle den folgenden Code ein.

```
version: 0.2

env:
  git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
```

```
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
      # name: version
    #commands:
      # - command
      # - command
  pre_build:
    commands:
      - ls -lt
      - cat README.md
  build:
    commands:
      - git log | head -100
      - git status
      - ls
      - git describe --all
  #post_build:
    #commands:
      # - command
      # - command
#artifacts:
  #files:
    # - location
  #name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

h. Wählen Sie Weiter zu CodePipeline. Dadurch kehren Sie zur CodePipeline Konsole zurück und erstellen ein CodeBuild Projekt, das Ihre Build-Befehle für die Konfiguration verwendet. Das Build-Projekt verwendet eine Servicerolle zur Verwaltung von AWS-Service Berechtigungen. Dieser Vorgang kann einige Minuten dauern.

i. Wählen Sie Weiter aus.

9. Wählen Sie auf der Seite Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen) die Option Skip deploy stage (Bereitstellungsstufe überspringen) aus und akzeptieren Sie anschließend die Warnmeldung, indem Sie erneut Skip (Überspringen) auswählen. Wählen Sie Weiter aus.

10. Wählen Sie unter Step 5: Review (Schritt 5: Überprüfen) die Option Create pipeline (Pipeline erstellen) aus.

Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um das Repository zu klonen

Der erste Pipeline-Lauf schlägt fehl, da Sie die CodeBuild Servicerolle mit den Berechtigungen zum Abrufen aus Ihrem Repository aktualisieren müssen.

Fügen Sie die `codecommit:GitPull` IAM-Berechtigung zu Ihrer Servicerollenrichtlinie hinzu. Anweisungen zum Aktualisieren der Richtlinie in der IAM-Konsole finden Sie unter. Fügen Sie CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen hinzu

Schritt 4: Repository-Befehle in der Build-Ausgabe anzeigen

Um die Build-Ausgabe anzuzeigen

1. Wenn Ihre Servicerolle erfolgreich aktualisiert wurde, wählen Sie in der CodeBuild Phase „Fehlgeschlagen“ die Option „Erneut versuchen“.
2. Wenn die Pipeline erfolgreich ausgeführt wurde, wählen Sie in der erfolgreichen Buildphase die Option Details anzeigen aus.

   Wählen Sie auf der Detailseite die Registerkarte Protokolle aus. Sehen Sie sich die CodeBuild Build-Ausgabe an. Die Befehle geben den Wert der eingegebenen Variablen aus.

   Die Befehle geben den `README.md` Dateiinhalt aus, listen die Dateien im Verzeichnis auf, klonen das Repository, zeigen das Protokoll an und werden ausgeführt`git describe --all`.

Tutorial: Eine Pipeline mit AWS CloudFormation StackSets Bereitstellungsaktionen erstellen

In diesem Tutorial verwenden Sie die AWS CodePipeline Konsole, um eine Pipeline mit Bereitstellungsaktionen zum Erstellen eines Stack-Sets und zum Erstellen von Stack-Instances zu erstellen. Wenn die Pipeline ausgeführt wird, erstellt die Vorlage ein Stack-Set und erstellt und aktualisiert auch die Instanzen, in denen das Stack-Set bereitgestellt wird.

Es gibt zwei Möglichkeiten, Berechtigungen für ein Stack-Set zu verwalten: selbstverwaltete und verwaltete AWS IAM-Rollen. Dieses Tutorial enthält Beispiele für selbstverwaltete Berechtigungen.

Um Stacksets am effektivsten einsetzen zu können CodePipeline, sollten Sie ein klares Verständnis der dahinterstehenden Konzepte AWS CloudFormation StackSets und ihrer Funktionsweise haben. Weitere Informationen zu den StackSets Konzepten finden Sie im AWS CloudFormation Benutzerhandbuch.

Themen

Voraussetzungen

Für Stack-Set-Operationen verwenden Sie zwei verschiedene Konten: ein Administratorkonto und ein Zielkonto. Sie erstellen Stack-Sets im Administratorkonto. Sie erstellen einzelne Stacks, die zu einem Stack-Set im Zielkonto gehören.

Um eine Administratorrolle mit Ihrem Administratorkonto zu erstellen

- Folgen Sie den Anweisungen unter Grundberechtigungen für Stack-Set-Operationen einrichten. Ihre Rolle muss benannt werden **`AWSCloudFormationStackSetAdministrationRole`**.

Um eine Servicerolle im Zielkonto zu erstellen

- Erstellen Sie eine Servicerolle im Zielkonto, die dem Administratorkonto vertraut. Folgen Sie den Anweisungen unter Grundberechtigungen für Stack-Set-Operationen einrichten. Ihre Rolle muss benannt werden **`AWSCloudFormationStackSetExecutionRole`**.

Schritt 1: Laden Sie die AWS CloudFormation Beispielvorlage und die Parameterdatei hoch

Erstellen Sie einen Quell-Bucket für Ihre Stackset-Vorlagen- und Parameterdateien. Laden Sie die AWS CloudFormation Beispielvorlagendatei herunter, richten Sie eine Parameterdatei ein und komprimieren Sie dann die Dateien, bevor Sie sie in Ihren S3-Quell-Bucket hochladen.

> **Note**
>
> Stellen Sie sicher, dass Sie die Quelldateien komprimieren, bevor Sie sie in Ihren S3-Quell-Bucket hochladen, auch wenn die einzige Quelldatei die Vorlage ist.

Um einen S3-Quell-Bucket zu erstellen

1. Melden Sie sich bei der Amazon S3 S3-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus.
3. Geben Sie in Bucketname einen Namen für Ihren Bucket ein.

   Wählen Sie unter Region die Region aus, in der Sie Ihre Pipeline erstellen möchten. Wählen Sie Bucket erstellen aus.
4. Nachdem der Bucket erstellt wurde, wird ein Erfolgsbanner angezeigt. Wählen Sie Go to bucket details (Zu Bucket-Details wechseln) aus.
5. Wählen Sie auf der Registerkarte Properties (Eigenschaften) die Option Versioning aus. Wählen Sie Enable versioning (Versioning aktivieren) und dann Save (Speichern)aus.

Um die AWS CloudFormation Vorlagendatei zu erstellen

1. Laden Sie die folgende Beispielvorlagendatei zum Generieren der CloudTrail Konfiguration für Stack-Sets herunter:https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml.
2. Speichern Sie die Datei als `template.yml`.

Um die Datei parameters.txt zu erstellen

1. Erstellen Sie eine Datei mit den Parametern für Ihre Bereitstellung. Parameter sind Werte, die Sie zur Laufzeit in Ihrem Stack aktualisieren möchten. Die folgende Beispieldatei aktualisiert die Vorlagenparameter für Ihr Stack-Set, um die Protokollierung von Validierungen und globalen Ereignissen zu ermöglichen.

```
[
  {
    "ParameterKey": "EnableLogFileValidation",
    "ParameterValue": "true"
  },
  {
    "ParameterKey": "IncludeGlobalEvents",
    "ParameterValue": "true"
  }
]
```

2. Speichern Sie die Datei als `parameters.txt`.

Um die Datei accounts.txt zu erstellen

1. Erstellen Sie eine Datei mit den Konten, für die Sie Instanzen erstellen möchten, wie in der folgenden Beispieldatei gezeigt.

```
[
    "111111222222","333333444444"
]
```

2. Speichern Sie die Datei als `accounts.txt`.

Um Quelldateien zu erstellen und hochzuladen

1. Kombinieren Sie die Dateien zu einer einzigen ZIP-Datei. Ihre Dateien sollten in Ihrer ZIP-Datei so aussehen.

```
template.yml
parameters.txt
accounts.txt
```

2. Laden Sie die ZIP-Datei in Ihren S3-Bucket hoch. Diese Datei ist das Quellartefakt, das vom Assistenten „Pipeline erstellen“ für Ihre Bereitstellungsaktion in CodePipeline erstellt wurde.

Schritt 2: Erstellen der Pipeline

In diesem Abschnitt erstellen Sie eine Pipeline mit den folgenden Aktionen:

- Eine Quellstufe mit einer S3-Quellaktion, bei der das Quellartefakt Ihre Vorlagendatei und alle unterstützenden Quelldateien ist.
- Eine Bereitstellungsphase mit einer AWS CloudFormation Stack-Set-Bereitstellungsaktion, die das Stack-Set erstellt.
- Eine Bereitstellungsphase mit einer Bereitstellungsaktion für AWS CloudFormation Stack-Instances, die die Stacks und Instances innerhalb der Zielkonten erstellt.

Um eine Pipeline mit einer CloudFormationStackSet Aktion zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyStackSetsPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
6. Behalten Sie im Artifact Store die Standardeinstellungen bei.

> **Note**
>
> Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket. Wenn Sie eine Pipeline erstellen oder bearbeiten, benötigen Sie einen Artefakt-Bucket in der Pipeline-Region und einen Artefakt-Bucket pro AWS Region, in der Sie eine Aktion ausführen.

Weitere Informationen finden Sie unter Eingabe- und Ausgabe-Artefakte und CodePipeline Referenz zur Pipeline-Struktur.

Wählen Sie Weiter aus.

7. Auf der Seite Step 2: Add source stage (Schritt 2: Hinzufügen der Quell-Stufe) wählen Sie für Source provider (Quell-Anbieter) die Option Amazon S3.
8. Geben Sie im Feld Bucket den S3-Quell-Bucket ein, den Sie für dieses Tutorial erstellt haben, z. B. `BucketName` Geben Sie im Feld S3-Objektschlüssel den Dateipfad und den Dateinamen für Ihre ZIP-Datei ein, z. `MyFiles.zip` B.
9. Wählen Sie Weiter aus.
10. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken.

    Wählen Sie Weiter aus.
11. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):
    a. Wählen Sie unter Deploy Provider die Option AWS CloudFormation Stack Set aus.
    b. Geben Sie im Feld Name des Stack-Sets einen Namen für das Stack-Set ein. Dies ist der Name des Stack-Sets, das die Vorlage erstellt.

       > **Note**
       >
       > Notieren Sie sich den Namen Ihres Stack-Sets. Sie werden ihn verwenden, wenn Sie Ihrer Pipeline die zweite StackSets Bereitstellungsaktion hinzufügen.

    c. Geben Sie im Feld Vorlagenpfad den Namen des Artefakts und den Dateipfad ein, in den Sie Ihre Vorlagendatei hochgeladen haben. Geben Sie beispielsweise Folgendes ein und verwenden Sie dabei den Standardnamen des Quellartefakts. `SourceArtifact`

       ```
       SourceArtifact::template.yml
       ```

    d. Geben Sie unter Bereitstellungsziele den Artefaktnamen und den Dateipfad ein, in den Sie Ihre Kontodatei hochgeladen haben. Geben Sie beispielsweise Folgendes ein und verwenden Sie dabei den Standardnamen des Quellartefakts. `SourceArtifact`

```
SourceArtifact::accounts.txt
```

e. Geben Sie im Feld Bereitstellungsziel AWS-Regionen eine Region für die Bereitstellung Ihrer ersten Stack-Instance ein, z. B. `us-east-1`

f. Erweitern Sie die Bereitstellungsoptionen. Geben Sie unter Parameter den Namen des Artefakts und den Dateipfad ein, in den Sie Ihre Parameterdatei hochgeladen haben. Geben Sie beispielsweise Folgendes ein und verwenden Sie dabei den Standardnamen des Quellartefakts. `SourceArtifact`

```
SourceArtifact::parameters.txt
```

Um die Parameter als Literaleingabe und nicht als Dateipfad einzugeben, geben Sie Folgendes ein:

```
ParameterKey=EnableLogFileValidation,ParameterValue=true
ParameterKey=IncludeGlobalEvents,ParameterValue=true
```

g. Wählen Sie unter Capabilities die Optionen CAPABILITY_IAM und CAPABILITY_NAMED_IAM aus.

h. Wählen Sie im Berechtigungsmodell die Option SELF_MANAGED aus.

i. Geben Sie im Feld Prozentsatz der Fehlertoleranz den Wert ein. `20`

j. Geben `25` Sie im Feld Maximaler Prozentsatz gleichzeitig den Wert ein.

k. Wählen Sie Weiter aus.

l. Wählen Sie Create pipeline (Pipeline erstellen) aus. Ihre Pipeline wird angezeigt.

m. Lassen Sie die Ausführung Ihrer Pipeline zu.

Schritt 3: Erste Bereitstellung anzeigen

Sehen Sie sich die Ressourcen und den Status Ihrer ersten Bereitstellung an. Nachdem Sie überprüft haben, ob das Deployment Ihr Stack-Set erfolgreich erstellt hat, können Sie die zweite Aktion zu Ihrer Bereitstellungsphase hinzufügen.

Um die Ressourcen anzusehen

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

2. Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
3. Wählen Sie die AWS CloudFormation Aktion für die CloudFormationStackSetAktion in Ihrer Pipeline aus. Die Vorlage, die Ressourcen und Ereignisse für Ihr Stack-Set werden in der AWS CloudFormation Konsole angezeigt.
4. Wählen Sie im linken Navigationsbereich StackSets. Wählen Sie in der Liste das neue Stack-Set aus.
5. Wählen Sie den Tab Stack-Instances. Stellen Sie sicher, dass eine Stack-Instance für jedes von Ihnen angegebene Konto in der Region us-east-1 erstellt wurde. Stellen Sie sicher, dass der Status für jede Stack-Instance lautet. `CURRENT`

Schritt 4: Fügen Sie eine CloudFormationStackInstances Aktion hinzu

Erstellen Sie eine nächste Aktion in Ihrer Pipeline, mit der Sie AWS CloudFormation StackSets die verbleibenden Stack-Instanzen erstellen können.

Um eine nächste Aktion in Ihrer Pipeline zu erstellen

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
2. Wählen Sie diese Option, um die Pipeline zu bearbeiten. Die Pipeline wird im Bearbeitungsmodus angezeigt.
3. Wählen Sie in der Bereitstellungsphase die Option Bearbeiten aus.
4. Wählen Sie unter der Aktion AWS CloudFormation Stack Set Deploy die Option Aktionsgruppe hinzufügen aus.
5. Fügen Sie auf der Seite Aktion bearbeiten die Aktionsdetails hinzu:

   a. Geben Sie im Feld Aktionsname einen Namen für die Aktion ein.

   b. Wählen Sie unter Aktionsanbieter die Option AWS CloudFormation Stack Instances aus.

   c. Wählen Sie unter Eingabeartefakte die Option aus SourceArtifact.

   d. Geben Sie im Feld Name des Stack-Sets den Namen für das Stack-Set ein. Dies ist der Name des Stack-Sets, das Sie in der ersten Aktion angegeben haben.

e. Geben Sie unter Bereitstellungsziele den Namen des Artefakts und den Dateipfad ein, in den Sie Ihre Kontodatei hochgeladen haben. Geben Sie beispielsweise Folgendes ein und verwenden Sie dabei den Standardnamen des Quellartefakts. `SourceArtifact`

```
SourceArtifact::accounts.txt
```

f. Geben Sie im Feld Bereitstellungsziel AWS-Regionen die Regionen für die Bereitstellung Ihrer verbleibenden Stack-Instances ein, z. B. `us-east-2` und `eu-central-1` wie folgt:

```
us-east2, eu-central-1
```

g. Geben Sie im Feld Prozentsatz der Fehlertoleranz den Wert ein`20`.

h. Geben `25` Sie im Feld Maximaler Prozentsatz gleichzeitig den Wert ein.

i. Wählen Sie Speichern.

j. . Geben Sie eine Änderung manuell frei. Ihre aktualisierte Pipeline wird in der Bereitstellungsphase mit zwei Aktionen angezeigt.

Schritt 5: Sehen Sie sich die Stackset-Ressourcen für Ihre Bereitstellung an

Sie können die Ressourcen und den Status für Ihre Stack-Set-Bereitstellung einsehen.

Um die Ressourcen einzusehen

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie unter Pipelines Ihre Pipeline und dann Ansicht aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
3. Wählen Sie die AWS CloudFormation Aktion für die **`AWS CloudFormation Stack Instances`**Aktion in Ihrer Pipeline aus. Die Vorlage, die Ressourcen und Ereignisse für Ihr Stack-Set werden in der AWS CloudFormation Konsole angezeigt.
4. Wählen Sie im linken Navigationsbereich StackSets. Wählen Sie in der Liste Ihr Stack-Set aus.
5. Wählen Sie den Tab Stack-Instances. Stellen Sie sicher, dass alle verbleibenden Stack-Instances für jedes von Ihnen angegebene Konto in den erwarteten Regionen erstellt oder aktualisiert wurden. Stellen Sie sicher, dass der Status für jede Stack-Instance lautet`CURRENT`.

Schritt 6: Nehmen Sie ein Update für Ihr Stack-Set vor

Nehmen Sie ein Update für Ihr Stack-Set vor und stellen Sie das Update für Instances bereit. In diesem Beispiel ändern Sie auch die Bereitstellungsziele, die Sie für das Update festlegen möchten. Die Instanzen, die nicht Teil des Updates sind, erhalten einen veralteten Status.

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie unter Pipelines Ihre Pipeline aus und klicken Sie dann auf Bearbeiten. Wählen Sie in der Bereitstellungsphase die Option Bearbeiten aus.
3. Wählen Sie, ob Sie die Aktion AWS CloudFormation Stack Set in Ihrer Pipeline bearbeiten möchten. Überschreiben Sie unter Beschreibung die bestehende Beschreibung mit einer neuen Beschreibung für das Stack-Set.
4. Wählen Sie, ob Sie die Aktion AWS CloudFormation Stack Instances in Ihrer Pipeline bearbeiten möchten. Löschen Sie im Feld Bereitstellungsziel AWS-Regionen den `us-east-2` Wert, der bei der Erstellung der Aktion eingegeben wurde.
5. Speichern Sie die Änderungen. Wählen Sie Änderung veröffentlichen, um Ihre Pipeline auszuführen.
6. Öffnen Sie Ihre Aktion in AWS CloudFormation. Wählen Sie die Registerkarte „StackSet Informationen“. Vergewissern Sie sich, dass in der StackSet Beschreibung die neue Beschreibung angezeigt wird.
7. Wählen Sie den Tab Stack-Instances aus. Stellen Sie unter Status sicher, dass der Status für die Stack-Instances in us-east-2 lautet. `OUTDATED`

CodePipeline bewährte Verfahren und Anwendungsfälle

In den folgenden Abschnitten werden bewährte Methoden für beschrieben CodePipeline.

Themen

Anwendungsfälle für CodePipeline

Sie können Pipelines erstellen, die sich in andere AWS-Services integrieren lassen. Dies können AWS-Services beispielsweise Amazon S3 oder Produkte von Drittanbietern sein, wie GitHub z. Dieser Abschnitt enthält Beispiele für die Automatisierung Ihrer Code-Releases mithilfe verschiedener Produktintegrationen. CodePipeline Eine vollständige Liste der Integrationen, die nach Aktionstyp CodePipeline geordnet sind, finden Sie unter. CodePipeline Referenz zur Pipeline-Struktur

Themen

Verwendung CodePipeline mit Amazon S3 AWS CodeCommit, und AWS CodeDeploy

Wenn Sie eine Pipeline erstellen, CodePipeline lässt sie sich in AWS Produkte und Services integrieren, die in jeder Phase Ihrer Pipeline als Maßnahmenanbieter agieren. Wenn Sie Stufen im Assistenten auswählen, müssen Sie eine Quellstufe und mindestens eine Build- oder Bereitstellungsstufe auswählen. Der Assistent erstellt die Stufen für Sie. Er vergibt Standardnamen, die Sie nicht ändern können. Dies sind die Stufennamen, die erstellt werden, wenn Sie im Assistenten eine vollständige dreistufige Pipeline erstellen:

- Eine Quellaktionsstufe mit dem Standardnamen „Source“.
- Eine Build-Aktionsstufe mit dem Standardnamen „Build“.
- Eine Bereitstellungsaktionsstufe mit dem Standardnamen „Staging“.

Sie können die Tutorials in diesem Handbuch verwenden, um Pipelines zu erstellen und Stufen anzugeben:

- Die Schritte unter Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) helfen Ihnen, mithilfe des Assistenten eine Pipeline mit zwei Standardstufen zu erstellen: „Source“ und „Staging“, wobei Ihr Amazon S3 S3-Repository der Quellanbieter ist. In diesem Tutorial wird eine Pipeline erstellt, mit der AWS CodeDeploy eine Beispielanwendung aus einem Amazon S3-Bucket auf Amazon EC2 EC2-Instances bereitgestellt wird, auf denen Amazon Linux ausgeführt wird.
- Die nachfolgenden Schritte Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository) helfen Ihnen dabei, mithilfe des Assistenten eine Pipeline mit einer „Source“ -Phase zu erstellen, die Ihr AWS CodeCommit Repository als Quellanbieter verwendet. In diesem Tutorial wird eine Pipeline erstellt, mit AWS CodeDeploy der eine Beispielanwendung aus einem AWS CodeCommit Repository auf einer Amazon EC2 EC2-Instance bereitgestellt wird, auf der Amazon Linux ausgeführt wird.

Verwendung CodePipeline mit Drittanbieter-Aktionsanbietern (GitHubund Jenkins)

Sie können Pipelines erstellen, die sich in Produkte von Drittanbietern wie GitHub Jenkins integrieren lassen. Die Schritte in Tutorial: Erstellen einer vierstufigen Pipeline zeigen, wie Sie eine Pipeline erstellen, die:

- Ruft Quellcode aus einem Repository ab, GitHub
- Jenkins verwendet, um den Quellcode zu erstellen und zu testen,
- Wird verwendet AWS CodeDeploy , um den erstellten und getesteten Quellcode auf Amazon EC2 EC2-Instances bereitzustellen, auf denen Amazon Linux oder Microsoft Windows Server ausgeführt werden.

Verwenden Sie CodePipeline with AWS CodeStar , um eine Pipeline in einem Codeprojekt zu erstellen

AWS CodeStar ist ein cloudbasierter Dienst, der eine einheitliche Benutzeroberfläche für die Verwaltung von Softwareentwicklungsprojekten bietet AWS. AWS CodeStar arbeitet mit CodePipeline , um AWS Ressourcen in einer Toolchain für die Projektentwicklung zu kombinieren. Sie können Ihr AWS CodeStar Dashboard verwenden, um automatisch die Pipeline, die Repositorys, den Quellcode, die Build-Spezifikationsdateien, die Bereitstellungsmethode und die Hosting-Instanzen oder serverlosen Instanzen zu erstellen, die für ein vollständiges Codeprojekt erforderlich sind.

Um Ihr AWS CodeStar Projekt zu erstellen, wählen Sie Ihre Programmiersprache und die Art der Anwendung, die Sie bereitstellen möchten. Sie können die folgenden Projekttypen erstellen: eine Webanwendung, einen Web-Service oder eine Alexa Skill.

Sie können Ihre bevorzugte IDE jederzeit in Ihr AWS CodeStar Dashboard integrieren. Sie können außerdem Teammitglieder hinzufügen und entfernen sowie Berechtigungen für Teammitglieder in Ihrem Projekt verwalten. Ein Tutorial, das Ihnen zeigt, wie Sie eine Beispielpipeline für eine serverlose Anwendung erstellen, finden Sie unter Tutorial: Ein serverloses Projekt erstellen und verwalten in. AWS CodeStar AWS CodeStar

Wird verwendet CodePipeline , um Code zu kompilieren, zu erstellen und zu testen mit CodeBuild

CodeBuild ist ein verwalteter Build-Service in der Cloud, mit dem Sie Ihren Code ohne Server oder System erstellen und testen können. Verwenden Sie CodePipeline with CodeBuild , um die Ausführung von Revisionen über die Pipeline zu automatisieren und so die kontinuierliche Bereitstellung von Software-Builds zu gewährleisten, wenn sich der Quellcode ändert. Weitere Informationen finden Sie unter Verwenden von CodePipeline with, um Code CodeBuild zu testen und Builds auszuführen.

Verwendung CodePipeline mit Amazon ECS für die kontinuierliche Bereitstellung containerbasierter Anwendungen in der Cloud

Amazon ECS ist ein Container-Management-Service, mit dem Sie containerbasierte Anwendungen auf Amazon ECS-Instances in der Cloud bereitstellen können. Verwenden Sie es CodePipeline mit Amazon ECS, um die Ausführung von Revisionen über die Pipeline zu automatisieren und so die kontinuierliche Bereitstellung containerbasierter Anwendungen zu ermöglichen, wann immer sich das Quell-Image-Repository ändert. Weitere Informationen finden Sie unter Tutorial: Kontinuierliche Bereitstellung mit. CodePipeline

Verwendung CodePipeline mit Elastic Beanstalk für die kontinuierliche Bereitstellung von Webanwendungen in der Cloud

Elastic Beanstalk ist ein Rechendienst, mit dem Sie Webanwendungen und Dienste auf Webservern bereitstellen können. Verwenden Sie es CodePipeline zusammen mit Elastic Beanstalk für die kontinuierliche Bereitstellung von Webanwendungen in Ihrer Anwendungsumgebung. Sie können es auch verwenden AWS CodeStar , um eine Pipeline mit einer Elastic Beanstalk Beanstalk-Bereitstellungsaktion zu erstellen.

Verwenden Sie CodePipeline mit AWS Lambda für die kontinuierliche Bereitstellung von Lambda-basierten und serverlosen Anwendungen

Sie können AWS Lambda with verwenden, CodePipeline um eine AWS Lambda Funktion aufzurufen, wie unter Bereitstellen serverloser Anwendungen beschrieben. Sie können auch AWS Lambda und verwenden AWS CodeStar , um eine Pipeline für die Bereitstellung serverloser Anwendungen zu erstellen.

Verwenden Sie es CodePipeline zusammen mit AWS CloudFormation Vorlagen für die kontinuierliche Bereitstellung in der Cloud

Sie können es CodePipeline für Continuous Delivery und Automatisierung verwenden AWS CloudFormation . Weitere Informationen finden Sie unter Continuous Delivery with CodePipeline. AWS CloudFormation wird auch verwendet, um die Vorlagen für Pipelines zu erstellen, die in AWS CodeStar erstellt wurden.

Markieren von Ressourcen

Ein Tag ist eine benutzerdefinierte Attributbezeichnung, die Sie einer AWS Ressource AWS zuweisen oder zuweisen. Jedes AWS Tag besteht aus zwei Teilen:

- einem Tag-Schlüssel (z. B. `CostCenter`, `Environment`, `Project` oder `Secret`). Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.
- einem optionalen Feld, das als Tag-Wert bezeichnet wird (z. B. `111122223333`, `Production` oder ein Team-Name). Ein nicht angegebener Tag-Wert entspricht einer leeren Zeichenfolge. Wie bei Tag-Schlüsseln wird auch bei Tag-Werten zwischen Groß- und Kleinschreibung unterschieden.

Zusammen werden sie als Schlüssel-Wert-Paare bezeichnet.

Mithilfe von Tags können Sie Ihre AWS Ressourcen identifizieren und organisieren. Viele AWS-Services unterstützen Tagging, sodass Sie Ressourcen aus verschiedenen Diensten dasselbe Tag zuweisen können, um anzuzeigen, dass die Ressourcen miteinander verknüpft sind. Sie können beispielsweise einer Pipeline dasselbe Tag zuweisen, das Sie einem Amazon S3 S3-Quell-Bucket zuweisen.

Tipps zur Verwendung von Tags finden Sie unter AWS -Tagging-Strategien im AWS -Antworten-Blog.

Sie können die folgenden Ressourcentypen taggen in CodePipeline:

- Markieren Sie eine Pipeline in CodePipeline
- Kennzeichnen Sie eine benutzerdefinierte Aktion in CodePipeline

Sie können die AWS CLI CodePipeline APIs oder AWS SDKs verwenden, um:

- Hinzufügen von Tags zu einer Pipeline, einer benutzerdefinierten Aktion oder einem Webhook während der Erstellung.
- Hinzufügen, Verwalten und Entfernen von Tags für eine Pipeline, eine benutzerdefinierte Aktion oder einem Webhook.

Sie können auch die Konsole verwenden, um Tags für eine Pipeline hinzuzufügen, zu verwalten und zu entfernen.

Neben der Identifizierung, Organisation und Nachverfolgung Ihrer Ressource mithilfe von Tags können Sie mithilfe von Tags in IAM-Richtlinien steuern, wer Ihre Ressource aufrufen und mit ihnen interagieren kann. Beispiele für Tag-basierte Zugriffsrichtlinien finden Sie unter Verwenden von Tags zur Steuerung des Zugriffs auf Ressourcen CodePipeline.

Verwendung CodePipeline mit Amazon Virtual Private Cloud

AWS CodePipeline unterstützt jetzt Amazon Virtual Private Cloud (Amazon VPC) -Endpunkte, die von betrieben werden. AWS PrivateLink Das bedeutet, dass Sie CodePipeline über einen privaten Endpunkt in Ihrer VPC eine direkte Verbindung herstellen können, sodass der gesamte Datenverkehr innerhalb Ihrer VPC und des AWS Netzwerks bleibt.

Amazon VPC ist eine AWS-Service , mit der Sie AWS Ressourcen in einem von Ihnen definierten virtuellen Netzwerk starten können. Mit einer VPC haben Sie die Kontrolle über Ihre Netzwerkeinstellungen, z. B.:

- IP-Adressbereich
- Subnetze
- Routing-Tabellen
- Netzwerk-Gateways

Schnittstelle, auf der VPC-Endpunkte basieren AWS PrivateLink, eine AWS Technologie, die die private Kommunikation zwischen Benutzern AWS-Services einer elastic network interface mit privaten IP-Adressen ermöglicht. Um Ihre VPC zu verbinden CodePipeline, definieren Sie einen VPC-Schnittstellen-Endpunkt für. CodePipeline Dieser Endpunkttyp ermöglicht es Ihnen, eine Verbindung zu Ihrer VPC herzustellen AWS-Services. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität, CodePipeline ohne dass ein Internet-Gateway, eine NAT-Instanz (Network Address Translation) oder eine VPN-Verbindung erforderlich ist. Weitere Informationen zur Einrichtung einer VPC finden Sie im VPC-Benutzerhandbuch.

Verfügbarkeit

CodePipeline unterstützt derzeit VPC-Endpunkte in den folgenden Bereichen: AWS-Regionen

- US East (Ohio)
- USA Ost (Nord-Virginia)
- USA West (Nordkalifornien)
- USA West (Oregon)
- Kanada (Zentral)
- Europe (Frankfurt)

- Europa (Irland)
- Europe (London)
- Europa (Mailand) *
- Europa (Paris)
- Europa (Stockholm)
- Asien-Pazifik (Hongkong) *
- Asien-Pazifik (Mumbai)
- Asien-Pazifik (Tokio)
- Asien-Pazifik (Seoul)
- Asien-Pazifik (Singapur)
- Asien-Pazifik (Sydney)
- Südamerika (São Paulo)
- AWS GovCloud (US-West)

* Sie müssen diese Region aktivieren, bevor Sie sie verwenden können.

Erstellen eines VPC-Endpunkts für CodePipeline

Sie können die Amazon VPC-Konsole verwenden, um die Datei com.amazonaws zu erstellen. ***Region*** .codepipeline VPC-Endpunkt. In der Konsole ist *Region* die Regionskennung für eine Region, die von AWS-Region unterstützt wird CodePipeline, z. B. `us-east-2` für die Region USA Ost (Ohio). Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts im Amazon VPC Leitfaden.

Der Endpunkt ist bereits mit der Region belegt, die Sie beim Anmelden bei AWS angegeben haben. Wenn Sie sich bei einer anderen Region anmelden, wird der VPC-Endpunkt mit der neuen Region aktualisiert.

> **Note**
>
> Andere AWS-Services , die VPC-Unterstützung bieten und sich in diese integrieren CodePipeline, unterstützen z. B. CodeCommit möglicherweise nicht die Verwendung von Amazon VPC-Endpunkten für diese Integration. Beispielsweise kann der Verkehr zwischen CodePipeline und CodeCommit nicht auf den VPC-Subnetzbereich beschränkt werden.

Fehlerbehebung für Ihre VPC-Einrichtung

Verwenden Sie bei der Fehlerbehebung von VPC-Problemen die in den Fehlermeldungen zur Internetverbindung angegebenen Informationen, um Probleme zu identifizieren, zu diagnostizieren und zu beheben.

1. Stellen Sie sicher, dass Ihr Internet-Gateway Ihrer VPC angefügt ist..
2. Stellen Sie sicher, dass die Routing-Tabelle für Ihr öffentliches Subnetz auf das Internet-Gateway verweist..
3. Stellen Sie sicher, dass Ihre Netzwerk-ACLs den Datenverkehr zulassen.
4. Stellen Sie sicher, dass Ihre Sicherheitsgruppen den Datenverkehr zulassen.
5. Stellen Sie sicher, dass die Routing-Tabelle für private Subnetze auf das Virtual Private Gateway verweist.
6. Stellen Sie sicher, dass die von verwendete Servicerolle CodePipeline über die entsprechenden Berechtigungen verfügt. Wenn Sie beispielsweise CodePipeline nicht über die Amazon EC2 EC2-Berechtigungen verfügen, die für die Arbeit mit einer Amazon VPC erforderlich sind, erhalten Sie möglicherweise die Fehlermeldung „Unerwarteter EC2-Fehler:“. UnauthorizedOperation

Arbeiten mit Pipelines in CodePipeline

Um einen automatisierten Release-Prozess zu definieren AWS CodePipeline, erstellen Sie eine Pipeline. Dabei handelt es sich um ein Workflow-Konstrukt, das beschreibt, wie Softwareänderungen einen Release-Prozess durchlaufen. Eine Pipeline besteht aus mehreren Phasen und Aktionen, die Sie konfigurieren.

> **Note**
>
> Wenn Sie die Phasen Build, Deploy, Test oder Invake hinzufügen, können Sie zusätzlich zu den bereitgestellten Standardoptionen benutzerdefinierte Aktionen auswählen CodePipeline, die Sie bereits für die Verwendung mit Ihren Pipelines erstellt haben. Benutzerdefinierte Aktionen können für das Ausführen von Aufgaben wie beispielsweise des intern entwickelten Erstellungsprozesses oder einer Testsuite eingesetzt werden. Die enthaltenen Versionsbezeichner sollen Ihnen dabei helfen, zwischen den verschiedenen Versionen einer benutzerdefinierten Aktion in den Anbieterlisten zu unterscheiden. Weitere Informationen finden Sie unter Erstellen und fügen Sie eine benutzerdefinierte Aktion hinzu in CodePipeline.

Vor dem Erstellen einer Pipeline müssen Sie die Schritte unter Erste Schritte mit CodePipeline ausführen.

Weitere Informationen zu Pipelines finden Sie unterCodePipeline Konzepte , undCodePipeline Anleitungen, falls Sie die zum Erstellen einer Pipeline verwenden möchten AWS CLI ,. CodePipeline Referenz zur Pipeline-Struktur Eine Liste der Pipelines finden Sie unter Pipelines und Details anzeigen in CodePipeline.

Themen

- Erstellen Sie eine Pipeline CodePipeline , in der Ressourcen von einem anderen AWS Konto verwendet werden
- Migrieren Sie Abfrage-Pipelines, um die ereignisbasierte Änderungserkennung zu nutzen
- Erstellen Sie die CodePipeline Servicerolle
- Markieren Sie eine Pipeline in CodePipeline
- Erstellen einer Benachrichtigungsregel

Starten Sie eine Pipeline in CodePipeline

Jede Pipeline-Ausführung kann auf der Grundlage eines anderen Triggers gestartet werden. Jede Pipeline-Ausführung kann einen anderen Triggertyp haben, je nachdem, wie die Pipeline gestartet wird. Der Triggertyp für jede Ausführung wird im Ausführungsverlauf für eine Pipeline angezeigt. Triggertypen können wie folgt vom Quell-Action-Provider abhängen:

> **Note**
>
> Sie können nicht mehr als einen Trigger pro Quellaktion angeben.

- Pipeline-Erstellung: Wenn eine Pipeline erstellt wird, wird die Pipeline-Ausführung automatisch gestartet. Dies ist der `CreatePipeline` Triggertyp in der Ausführungshistorie.
- Änderungen an überarbeiteten Objekten: Diese Kategorie steht für den `PutActionRevision` Triggertyp in der Ausführungshistorie.
- Erkennung von Änderungen beim Branch und Commit für einen Code-Push: Diese Kategorie steht für den `CloudWatchEvent` Triggertyp in der Ausführungshistorie. Wenn eine Änderung an einem Quell-Commit und einem Branch im Quell-Repository erkannt wird, wird Ihre Pipeline gestartet. Dieser Triggertyp verwendet die automatische Änderungserkennung. Die Anbieter von Quellaktionen, die diesen Triggertyp verwenden, sind S3 und CodeCommit. Dieser Typ wird auch für einen Zeitplan verwendet, mit dem Ihre Pipeline gestartet wird. Siehe Starten Sie eine Pipeline nach einem Zeitplan.
- Abfrage nach Quellenänderungen: Diese Kategorie stellt den **`PollForSourceChanges`** Triggertyp in der Ausführungshistorie dar. Wenn durch Polling eine Änderung an einem Quell-Commit und einem Branch im Quell-Repository erkannt wird, wird Ihre Pipeline gestartet. Dieser Triggertyp wird nicht empfohlen und sollte migriert werden, um die automatische Änderungserkennung

zu verwenden. Die Anbieter von Quellaktionen, die diesen Triggertyp verwenden, sind S3 und CodeCommit.

- Webhook-Ereignisse für Quellen von Drittanbietern: Diese Kategorie stellt den `Webhook` Triggertyp im Ausführungsverlauf dar. Wenn eine Änderung durch ein Webhook-Ereignis erkannt wird, wird Ihre Pipeline gestartet. Dieser Triggertyp verwendet die automatische Änderungserkennung. Die Anbieter von Quellaktionen, die diesen Triggertyp verwenden, sind Verbindungen, die für Code-Push konfiguriert sind (Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltet).
- WebHookV2-Ereignisse für Quellen von Drittanbietern: Diese Kategorie steht für den **`WebhookV2`** Triggertyp im Ausführungsverlauf. Dieser Typ ist für Ausführungen vorgesehen, die auf der Grundlage von Triggern ausgelöst werden, die in der Pipeline-Definition definiert sind. Wenn eine Version mit einem bestimmten Git-Tag erkannt wird, wird Ihre Pipeline gestartet. Mit Git-Tags können Sie einen Commit mit einem Namen oder einer anderen Kennung versehen, damit andere Repository-Benutzer dessen Bedeutung verstehen. Sie können Git-Tags auch dazu verwenden, um einen bestimmten Commit im Repository-Verlauf zu identifizieren. Dieser Triggertyp deaktiviert die automatische Erkennung von Änderungen. Die Anbieter von Quellaktionen, die diesen Triggertyp verwenden, sind Verbindungen, die für Git-Tags konfiguriert sind (Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab .com).
- Manuelles Starten einer Pipeline: Diese Kategorie steht für den `StartPipelineExecution` Triggertyp in der Ausführungshistorie. Sie können die Konsole oder die verwenden AWS CLI , um eine Pipeline manuell zu starten. Weitere Informationen finden Sie unter Manuelles Starten einer Pipeline.
- RollbackStage: Diese Kategorie stellt den `RollbackStage` Triggertyp in der Ausführungshistorie dar. Sie können die Konsole oder die verwenden AWS CLI , um eine Phase manuell oder automatisch rückgängig zu machen. Weitere Informationen finden Sie unter Stage Rollback konfigurieren.

Wenn Sie Ihrer Pipeline eine Quellaktion hinzufügen, die Triggertypen für die automatische Änderungserkennung verwendet, funktionieren die Aktionen mit zusätzlichen Ressourcen. Die Erstellung der einzelnen Quellaktionen wird aufgrund dieser zusätzlichen Ressourcen für die Änderungserkennung in separaten Abschnitten detailliert beschrieben. Einzelheiten zu den einzelnen Quellanbietern und den Methoden zur Änderungserkennung, die für die automatische Änderungserkennung erforderlich sind, finden Sie unter Quellaktionen und Methoden zur Änderungserkennung.

Themen

- Quellaktionen und Methoden zur Änderungserkennung
- Manuelles Starten einer Pipeline
- Starten Sie eine Pipeline nach einem Zeitplan
- Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung

Quellaktionen und Methoden zur Änderungserkennung

Wenn Sie Ihrer Pipeline eine Quellaktion hinzufügen, funktionieren die Aktionen mit zusätzlichen Ressourcen, die in der Tabelle beschrieben sind.

> **Note**
>
> Die Quellaktionen CodeCommit und S3 erfordern entweder eine konfigurierte Ressource zur Änderungserkennung (eine EventBridge Regel) oder verwenden die Option, das Repository nach Quelländerungen abzufragen. Für Pipelines mit einer Bitbucket GitHub - oder GitHub Enterprise Server-Quellaktion musst du weder einen Webhook einrichten noch standardmäßig Polling verwenden. Die Aktion „Verbindungen“ verwaltet die Erkennung von Änderungen für dich.

Quelle	Verwendet zusätzliche Ressourcen?	Schritte
Amazon S3	Diese Quellaktion verwendet zusätzliche Ressourcen. Wenn Sie die CLI verwenden oder CloudFormation diese Aktion erstellen, erstellen und verwalten Sie auch diese Ressourcen.	Siehe Erstellen Sie eine Pipeline in CodePipeline und Amazon S3 S3-Quellaktionen und EventBridge mit AWS CloudTrail
Bitbucket Cloud	Diese Quellaktion verwendet eine Verbindungsressource.	Siehe Bitbucket Cloud-Verbindungen
AWS CodeCommit	Amazon EventBridge (empfohlen). Dies ist die Standardeinstellung für Pipelines mit einer CodeCommit	Siehe Erstellen Sie eine Pipeline in CodePipeline und CodeCommit Quellaktionen und EventBridge

Quelle	Verwendet zusätzliche Ressourcen?	Schritte
	Quelle, die in der Konsole erstellt oder bearbeitet wurde.	
Amazon ECR	Amazon EventBridge. Dies wird vom Assistenten für Pipelines mit einer Amazon ECR-Quelle erstellt, die in der Konsole erstellt oder bearbeitet wurde.	Siehe Erstellen Sie eine Pipeline in CodePipeline und Amazon ECR-Quellaktionen und Ressourcen EventBridge .
GitHub oder Enterprise Cloud GitHub	Diese Quellaktion verwendet eine Verbindungsressource.	Siehe GitHub Verbindungen
GitHub Unternehmensserver	Diese Quellaktion verwendet eine Verbindungsressource und eine Hostressource.	Siehe GitHub Enterprise Server-Verbindungen
GitLab.com	Diese Quellaktion verwendet eine Verbindungsressource.	Siehe GitLab.com-Verbindungen
GitLab selbst verwaltet	Diese Quellaktion verwendet eine Verbindungsressource und eine Hostressource.	Siehe Verbindungen für GitLab selbstverwaltete

Wenn Sie eine Pipeline verwenden, die Abfragen verwendet, können Sie sie aktualisieren, um die empfohlene Erkennungsmethode zu verwenden. Weitere Informationen finden Sie unter Aktivieren von Pipelines für die empfohlenen Methode zur Änderungserkennung.

Informationen zum Deaktivieren der Änderungserkennung für eine Quellaktion, die Verbindungen verwendet, finden Sie unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Manuelles Starten einer Pipeline

Standardmäßig startet eine Pipeline automatisch bei ihrer Erstellung und bei jeder Änderung eines Quellrepositorys. Möglicherweise möchten Sie jedoch die letzte Revision über die Pipeline ein

zweites Mal ausführen. Sie können die CodePipeline Konsole oder den start-pipeline-execution Befehl AWS CLI and verwenden, um die neueste Version manuell über Ihre Pipeline erneut auszuführen.

Themen

Manueller Start einer Pipeline (Konsole)

So starten Sie eine Pipeline und führen die letzte Revision manuell über eine Pipeline aus

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie starten möchten.
3. Wählen Sie auf der Seite mit den Pipeline-Details die Option Änderung veröffentlichen aus. Wenn die Pipeline für die Übergabe von Parametern (Pipeline-Variablen) konfiguriert ist, wird bei Auswahl von Versionsänderung das Fenster Versionsänderung geöffnet. Geben Sie unter Pipeline-Variablen in das Feld oder die Felder für die Variablen auf Pipelineebene den Wert oder die Werte ein, die Sie für diese Pipeline-Ausführung übergeben möchten. Weitere Informationen finden Sie unter Variablen.

   Dadurch wird die letzte Revision gestartet, die in jedem in einer Quellaktion der Pipeline angegebenen Quellspeicherort vorhanden ist.

Manuelles Starten einer Pipeline (CLI)

So starten Sie eine Pipeline und führen die letzte Version eines Artefakts manuell über eine Pipeline aus

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie die, AWS CLI um den start-pipeline-execution Befehl auszuführen. Geben Sie dabei den Namen der Pipeline an, die Sie starten möchten. Um beispielsweise mit der Ausführung der letzten Änderung über eine Pipeline mit dem Namen zu beginnen *MyFirstPipeline*:

```
aws codepipeline start-pipeline-execution --name MyFirstPipeline
```

Um eine Pipeline zu starten, in der Variablen auf Pipelineebene konfiguriert werden, verwenden Sie den start-pipeline-execution Befehl mit dem optionalen --variables Argument, um die Pipeline zu starten und die Variablen hinzuzufügen, die bei der Ausführung verwendet werden. Um beispielsweise eine Variable `var1` mit dem Wert von hinzuzufügen1, verwenden Sie den folgenden Befehl:

```
aws codepipeline start-pipeline-execution --name MyFirstPipeline --variables
 name=var1,value=1
```

2. Zeigen Sie das zurückgegebene Objekt an, um den Erfolg zu überprüfen. Dieser Befehl gibt eine Ausführungs-ID zurück, die in etwa wie folgt aussieht:

```
{
    "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
}
```

> **Note**
>
> Nachdem Sie die Pipeline gestartet haben, können Sie ihren Fortschritt in der CodePipeline Konsole oder durch Ausführen des get-pipeline-state Befehls überwachen. Weitere Informationen finden Sie unter Pipelines anzeigen (Konsole) und Anzeigen von der Pipeline-Details und des Verlaufs (CLI).

Starten Sie eine Pipeline nach einem Zeitplan

Sie können eine Regel einrichten, EventBridge um eine Pipeline nach einem Zeitplan zu starten.

Erstellen Sie eine EventBridge Regel, die den Start Ihrer Pipeline plant (Konsole)

Um eine EventBridge Regel mit einem Zeitplan als Ereignisquelle zu erstellen

1. Öffnen Sie die EventBridge Amazon-Konsole unter https://console.aws.amazon.com/events/.
2. Wählen Sie im Navigationsbereich Regeln aus.
3. Wählen Sie Regel erstellen und dann unter Regeldetails die Option Zeitplan aus.

4. Richten Sie den Zeitplan in einem bestimmten Zeitintervall oder mit bestimmten Ausdrücken ein. Weitere Informationen finden Sie unter Planen von Ausdrücken für Regeln.
5. Wählen Sie unter Ziele die Option aus CodePipeline.
6. Geben Sie den Pipeline-ARN für die Pipeline-Ausführung für diesen Zeitplan ein.

> **Note**
>
> Sie finden den Pipeline-ARN unter Einstellungen in der Konsole. Siehe Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen.

7. Wählen Sie eine der folgenden Optionen, um eine IAM-Dienstrolle zu erstellen oder anzugeben, die EventBridge Berechtigungen zum Aufrufen des mit Ihrer EventBridge Regel verknüpften Ziels erteilt (in diesem Fall ist CodePipeline das Ziel).
   - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen aus, um eine Servicerolle zu erstellen, die EventBridge Berechtigungen zum Starten Ihrer Pipeline-Ausführungen gewährt.
   - Wählen Sie Bestehende Rolle verwenden aus, um eine Servicerolle einzugeben, die EventBridge Berechtigungen zum Starten Ihrer Pipeline-Ausführungen gewährt.
8. Wählen Sie Details konfigurieren.
9. Geben Sie auf der Seite Configure rule details (Regeldetails konfigurieren) einen Namen und eine Beschreibung für die Regel ein und wählen Sie dann State (Status), um die Regel zu aktivieren.
10. Wenn Sie mit der Regel zufrieden sind, klicken Sie auf Create rule (Regel erstellen).

Erstellen Sie eine EventBridge Regel, die den Start Ihrer Pipeline plant (CLI)

Um die zum Erstellen einer Regel AWS CLI zu verwenden, rufen Sie den put-rule Befehl auf und geben Sie Folgendes an:

- Einen Namen, der die von Ihnen erstellte Regel eindeutig bezeichnet. Dieser Name muss für alle Pipelines, die Sie erstellen und die mit Ihrem AWS Konto CodePipeline verknüpft sind, eindeutig sein.
- Den Zeitplanausdruck für die Regel.

Um eine EventBridge Regel mit einem Zeitplan als Ereignisquelle zu erstellen

1. Rufen Sie den Befehl put-rule auf und beziehen Sie die Parameter `--name` und `--schedule-expression` ein.

   Beispiele:

   Mit dem folgenden Beispielbefehl wird --schedule-expression eine Regel mit dem Namen „MyRule2Filter nach EventBridge einem Zeitplan“ erstellt.

   ```
   aws events put-rule --schedule-expression 'cron(15 10 ? * 6L 2002-2005)' --name
    MyRule2
   ```

2. Erteilen Sie Berechtigungen für EventBridge die Verwendung CodePipeline zum Aufrufen der Regel. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie zu erstellen, damit EventBridge Sie die Servicerolle übernehmen können. Geben Sie ihr den Namen `trustpolicyforEB.json`.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

   b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

   ```
   aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
    file://trustpolicyforEB.json
   ```

c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie in diesem Beispiel für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Führen Sie den folgenden Befehl aus, um der erstellten `Role-for-MyRule`-Rolle die neue `CodePipeline-Permissions-Policy-for-EB`-Berechtigungsrichtlinie anzufügen.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforCWE.json
```

Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung

Sie können Overrides verwenden, um eine Pipeline mit einer bestimmten Quell-Revision-ID zu starten, die Sie für die Pipeline-Ausführung angeben. Wenn Sie beispielsweise eine Pipeline starten möchten, die eine bestimmte Commit-ID aus Ihrer CodeCommit Quelle verarbeitet, können Sie die Commit-ID als Override hinzufügen, wenn Sie Ihre Pipeline starten.

Es gibt vier Arten von Quellrevisionen für`revisionType`:

- `COMMIT_ID`
- `IMAGE_DIGEST`
- `S3_OBJECT_VERSION_ID`
- `S3_OBJECT_OBJECT_KEY`

> **Note**
>
> Bei den Quellrevisionen `COMMIT_ID` und den `IMAGE_DIGEST` Typen gilt die Quellrevisions-ID für alle Inhalte im Repository in allen Zweigen.

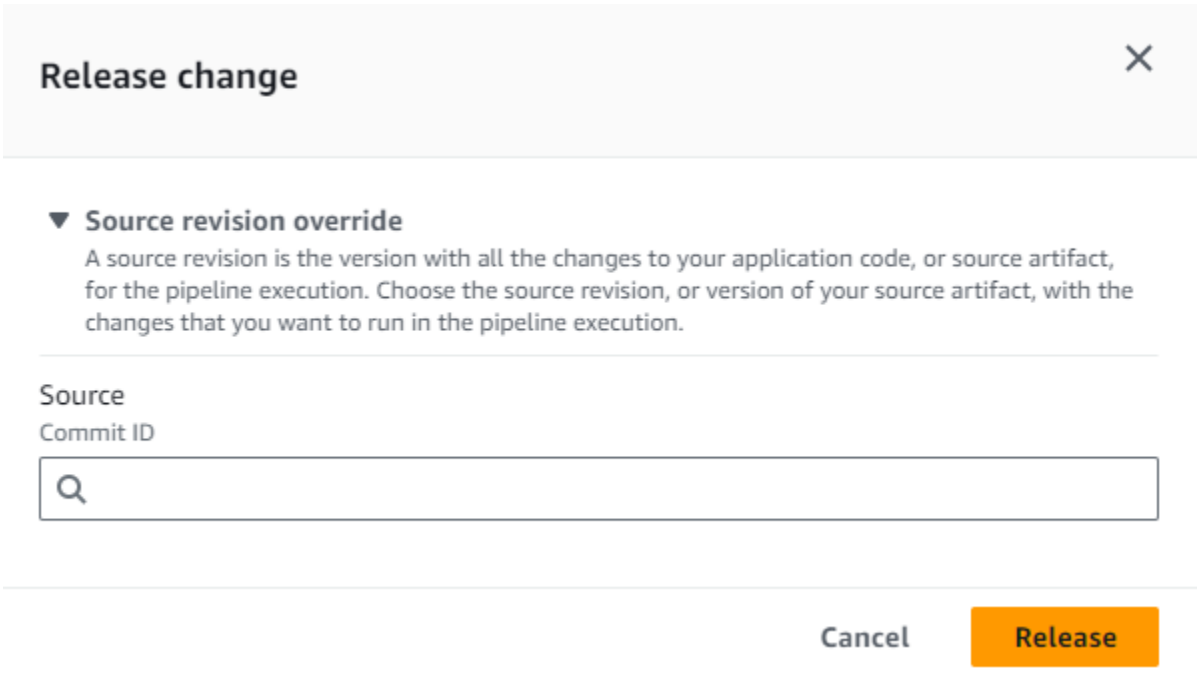
> **Note**
>
> Für die `S3_OBJECT_KEY` Typen `S3_OBJECT_VERSION_ID` und Typen der Quellversionen können beide Typen unabhängig voneinander verwendet werden, oder sie können zusammen verwendet werden, um die Quelle mit einer bestimmten ObjectKey Versions-ID zu überschreiben.

Themen

Startet eine Pipeline mit einer Überschreibung der Quellversion (Konsole)

So starten Sie eine Pipeline und führen die letzte Revision manuell über eine Pipeline aus

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie starten möchten.
3. Wählen Sie auf der Seite mit den Pipeline-Details die Option Änderung veröffentlichen aus. Wenn Sie Versionsänderung auswählen, wird das Fenster Versionsänderung geöffnet. Wählen Sie für Quellrevision überschreiben den Pfeil, um das Feld zu erweitern. Geben Sie im Feld Quelle die Quell-Revision-ID ein. Wenn Ihre Pipeline beispielsweise über eine CodeCommit Quelle verfügt, wählen Sie die Commit-ID aus dem Feld aus, das Sie verwenden möchten.



Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung (CLI)

Um eine Pipeline manuell zu starten und die angegebene Quell-Revision-ID für ein Artefakt über eine Pipeline auszuführen

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie die, AWS CLI um den start-pipeline-execution Befehl auszuführen. Geben Sie dabei den Namen der Pipeline an, die Sie starten möchten. Sie verwenden das --source-revisions Argument auch, um die Quell-Revision-ID anzugeben. Die Quellrevision besteht aus ActionName, RevisionType und RevisionValue. Gültige RevisionType-Werte sind. `COMMIT_ID | IMAGE_DIGEST | S3_OBJECT_VERSION_ID | S3_OBJECT_KEY`

   Um im folgenden Beispiel die Ausführung der angegebenen Änderung über eine Pipeline mit dem Namen zu startencodecommit-pipeline, gibt der folgende Befehl den Namen der Quellaktion Source, den Revisionstyp und die Commit-ID von `COMMIT_ID` an. `78a25c18755ccac3f2a9eec099dEXAMPLE`

   ```
   aws codepipeline start-pipeline-execution --name codecommit-pipeline --source-revisions
    actionName=Source,revisionType=COMMIT_ID,revisionValue=78a25c18755ccac3f2a9eec099dEXAMPLE
    --region us-west-1
   ```

2. Zeigen Sie das zurückgegebene Objekt an, um den Erfolg zu überprüfen. Dieser Befehl gibt eine Ausführungs-ID zurück, die in etwa wie folgt aussieht:

```
{
    "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
}
```

> **Note**
>
> Nachdem Sie die Pipeline gestartet haben, können Sie ihren Fortschritt in der CodePipeline Konsole oder durch Ausführen des get-pipeline-state Befehls überwachen. Weitere Informationen finden Sie unter Pipelines anzeigen (Konsole) und Anzeigen von der Pipeline-Details und des Verlaufs (CLI).

Stoppen Sie eine Pipeline-Ausführung in CodePipeline

Wenn eine Pipeline-Ausführung mit dem Durchlaufen einer Pipeline beginnt, tritt sie in eine Phase nach der anderen ein und sperrt die entsprechende Phase, während alle Aktionsausführungen der Phase ausgeführt werden. Diese laufenden Aktionen müssen so verarbeitet werden, dass beim Anhalten der Pipeline-Ausführung die Aktionen entweder abgeschlossen oder abgebrochen werden können.

Es gibt zwei Möglichkeiten, eine Pipeline-Ausführung anzuhalten:

- Stoppen und warten: AWS CodePipeline Wartet mit dem Beenden der Ausführung, bis alle laufenden Aktionen abgeschlossen sind (d. h., die Aktionen haben den `Failed` Status `Succeeded` oder). Mit dieser Option werden laufende Aktionen beibehalten. Die Ausführung befindet sich in einem `Stopping`-Status, bis die laufenden Aktionen abgeschlossen sind. Dann befindet sich die Ausführung in einem `Stopped`-Status. Die Phase wird freigegeben, nachdem die Aktionen abgeschlossen sind.

  Wenn Sie sich für das Anhalten und Warten entscheiden und Ihre Meinung ändern, während sich Ihre Ausführung noch in einem `Stopping`-Status befindet, können Sie zu diesem Zeitpunkt das Beenden auswählen.
- Stopp und Abbruch: AWS CodePipeline Stoppt die Ausführung, ohne auf den Abschluss der laufenden Aktionen zu warten. Die Ausführung befindet sich für eine sehr kurze Zeit in einem `Stopping`-Status, während die laufenden Aktionen beendet werden. Nachdem die Ausführung

beendet wurde, befindet sich die Aktionsausführung in einem `Abandoned`-Status, während die Pipeline-Ausführung in einem `Stopped`-Status ist. Die Phase wird entsperrt.

Bei einer Pipeline-Ausführung im Status `Stopped` können die Aktionen in der Phase, in der die Ausführung angehalten wurde, erneut ausgeführt werden.

> **Warning**
>
> Diese Option kann zu fehlgeschlagenen oder nicht aufeinander folgenden Aufgaben führen.

Themen

Pipeline-Ausführung beenden (Konsole)

Sie können die Konsole verwenden, um die Ausführung einer Pipeline zu beenden. Wählen Sie eine Ausführung und dann die Methode zum Stoppen der Pipeline-Ausführung aus.

> **Note**
>
> Sie können auch eine Pipeline-Ausführung beenden, bei der es sich um eine eingehende Ausführung handelt. Weitere Informationen zum Stoppen einer eingehenden Ausführung finden Sie unter. Stoppen Sie eine eingehende Ausführung (Konsole)

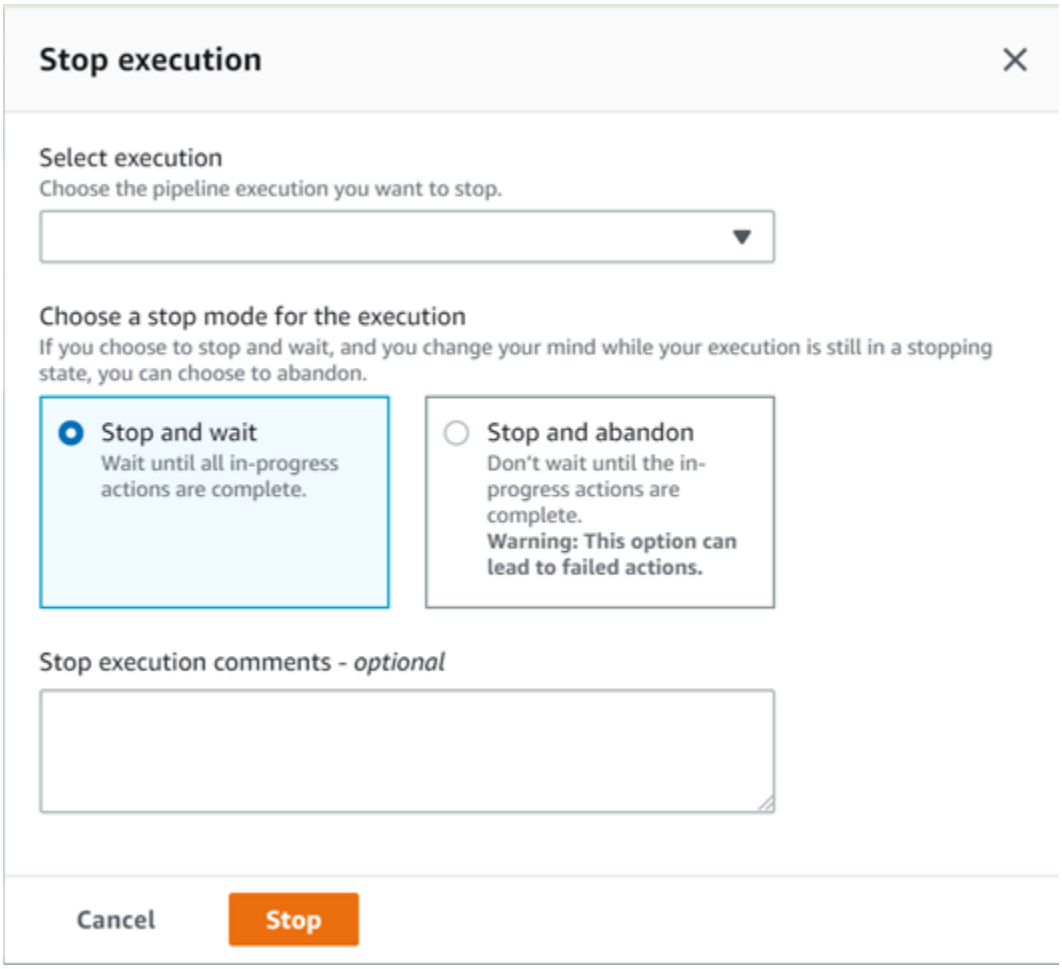
1. Melden Sie sich unter http://console.aws.amazon.com/codesuite/codepipeline/home bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole.
2. Führen Sie eine der folgenden Aktionen aus:

> **Note**
>
> Bevor Sie eine Ausführung beenden, empfehlen wir Ihnen, den Übergang vor der Phase zu deaktivieren. Auf diese Weise akzeptiert die Phase beim Entsperren aufgrund der angehaltenen Ausführung keine anschließende Pipeline-Ausführung.

- Wählen Sie in Name den Namen der Pipeline mit der Ausführung aus, die Sie beenden möchten. Wählen Sie auf der Seite mit den Pipeline-Details Stop execution (Ausführung anhalten) aus.
- Wählen Sie View history (Verlauf anzeigen). Wählen Sie auf der Verlaufsseite Stop execution (Ausführung stoppen) aus.

3. Wählen Sie auf der Seite Stop execution (Ausführung stoppen) unter Select execution (Ausführung auswählen) die Ausführung aus, die Sie beenden möchten.

> **Note**
>
> Die Ausführung wird nur dann angezeigt, wenn sie noch läuft. Bereits abgeschlossene Ausführungen werden nicht angezeigt.



4. Wählen Sie unter Select an action to apply to execution (Aktion für Anwendung auf Ausführung auswählen) eine der folgenden Optionen aus:

   - Um sicherzustellen, dass die Ausführung erst dann beendet wird, wenn alle laufenden Aktionen abgeschlossen sind, wählen Sie Anhalten und warten aus.

     > **Note**
     >
     > Sie können "Anhalten und warten" nicht auswählen, wenn sich die Ausführung bereits im Status Stopping (Wird angehalten) befindet. Sie können jedoch "Anhalten und beenden" auswählen.

   - Um zu beenden, ohne auf den Abschluss laufender Aktionen zu warten, wählen Sie Stop and abandon (Anhalten und beenden) aus.

> **Warning**
>
> Diese Option kann zu fehlgeschlagenen oder nicht aufeinander folgenden Aufgaben führen.

5. (Optional) Geben Sie Kommentare ein. Diese Kommentare werden zusammen mit dem Ausführungsstatus auf der Verlaufsseite für die Ausführung angezeigt.
6. Wählen Sie Beenden aus.

> **Important**
>
> Diese Aktion kann nicht rückgängig gemacht werden.

7. Zeigen Sie den Ausführungsstatus in der Pipeline-Visualisierung wie folgt an:

   - Wenn Sie sich für "Anhalten und Warten" entschieden haben, wird die ausgewählte Ausführung fortgesetzt, bis die laufenden Aktionen abgeschlossen sind.
     - Die Erfolgs-Bannermeldung wird oben in der Konsole angezeigt.
     - In der aktuellen Phase werden die laufenden Aktionen in einem `InProgress`-Status fortgesetzt. Während die Aktionen in Bearbeitung sind, befindet sich die Pipeline-Ausführung in einem `Stopping`-Status.

       Nachdem die Aktionen abgeschlossen sind (d. h. die Aktion schlägt fehl oder ist erfolgreich), wechselt die Pipeline-Ausführung in einen `Stopped`-Status und die Aktion in einen `Failed`- oder `Succeeded`-Status. Sie können den Aktionsstatus auch auf der Seite mit den Ausführungsdetails anzeigen. Sie können den Ausführungsstatus auf der Seite mit dem Ausführungsverlauf oder auf der Seite mit den Ausführungsdetails anzeigen.
     - Die Pipeline-Ausführung wechselt kurz in den Status `Stopping` und dann in den Status `Stopped`. Sie können den Ausführungsstatus auf der Seite mit dem Ausführungsverlauf oder auf der Seite mit den Ausführungsdetails anzeigen.
   - Wenn Sie sich für "Anhalten und beenden" entscheiden, wartet die Ausführung nicht auf den Abschluss der laufenden Aktionen.
     - Die Erfolgs-Bannermeldung wird oben in der Konsole angezeigt.
     - In der aktuellen Phase wechseln die laufenden Aktionen in den Status `Abandoned`. Sie können den Aktionsstatus auch auf der Seite mit den Ausführungsdetails anzeigen.

- Die Pipeline-Ausführung wechselt kurz in den Status `Stopping` und dann in den Status `Stopped`. Sie können den Ausführungsstatus auf der Seite mit dem Ausführungsverlauf oder auf der Seite mit den Ausführungsdetails anzeigen.

Sie können den Pipeline-Ausführungsstatus im Ausführungsverlauf und in der detaillierten Verlaufsansicht anzeigen.

Stoppen Sie eine eingehende Ausführung (Konsole)

Sie können die Konsole verwenden, um eine eingehende Ausführung zu beenden. Eine eingehende Ausführung ist eine Pipeline-Ausführung, die darauf wartet, in eine Phase überzugehen, in der der Übergang deaktiviert wurde. Wenn der Übergang aktiviert ist, befindet sich eine eingehende Ausführung, die `InProgress` weiterhin in die Phase übergeht. Eine eingehende Ausführung, die `Stopped` nicht in die Phase eintritt.

> **Note**
>
> Nachdem eine eingehende Ausführung gestoppt wurde, kann sie nicht erneut versucht werden.

Wenn Sie keine eingehende Ausführung sehen, gibt es in einer deaktivierten Phase des Übergangs keine ausstehenden Ausführungen.

1. Melden Sie sich unter http://console.aws.amazon.com/codesuite/codepipeline/home bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole.

   Die Namen aller Pipelines, die mit Ihrem AWS Konto verknüpft sind, werden angezeigt.

2. Wählen Sie den Namen der Pipeline, für die Sie die eingehende Ausführung beenden möchten, und führen Sie einen der folgenden Schritte aus:

   - Wählen Sie in der Pipeline-Ansicht die Ausführungs-ID für den eingehenden Datenverkehr aus und entscheiden Sie dann, ob die Ausführung gestoppt werden soll.
   - Wählen Sie die Pipeline aus und klicken Sie auf Verlauf anzeigen. Wählen Sie in der Ausführungshistorie die Ausführungs-ID für eingehende Nachrichten aus und wählen Sie dann, ob die Ausführung gestoppt werden soll.

3. Folgen Sie im Modal „Ausführung beenden“ den Schritten im obigen Abschnitt, um die Ausführungs-ID auszuwählen und die Stoppmethode anzugeben.

   Verwenden Sie den get-pipeline-state Befehl, um den Status der eingehenden Ausführung anzuzeigen.

Pipeline-Ausführung anhalten (CLI)

Um eine Pipeline manuell AWS CLI zu stoppen, verwenden Sie den stop-pipeline-execution Befehl mit den folgenden Parametern:

- Ausführungs-ID (erforderlich)
- Kommentare (optional)
- Pipeline-Name (erforderlich)
- Beenden-Flag (optional, der Standardwert ist Falsch)

Befehlsformat:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --pipeline-
execution-id Execution_ID [--abandon | --no-abandon] [--reason STOP_EXECUTION_REASON]
```

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix).
2. Um eine Pipeline-Ausführung zu beenden, wählen Sie eine der folgenden Optionen aus:

   - Um sicherzustellen, dass die Ausführung nicht beendet wird, bis alle laufenden Aktionen abgeschlossen sind, wählen Sie "Anhalten und warten" aus. Sie können dies tun, indem Sie den Parameter no-abandon einschließen. Wenn Sie den Parameter nicht angeben, wird standardmäßig "Anhalten und warten" verwendet. Verwenden Sie den, AWS CLI um den stop-pipeline-execution Befehl auszuführen, und geben Sie dabei den Namen der Pipeline und die Ausführungs-ID an. Um beispielsweise eine Pipeline zu beenden, deren Name *MyFirstPipeline*mit der angegebenen Stop-and-Wait-Option benannt ist:

     ```
     aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
     pipeline-execution-id d-EXAMPLE --no-abandon
     ```

     Um beispielsweise eine Pipeline mit dem Namen zu beenden *MyFirstPipeline*, standardmäßig die Option Stopp und Warten zu verwenden und Kommentare einzubeziehen:

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --reason "Stopping execution after the build
 action is done"
```

> **Note**
>
> Sie können sich nicht für das Anhalten und Warten entscheiden, wenn sich die Ausführung bereits in einem Stopping (Wird angehalten)-Status befindet. Sie können eine Ausführung, die sich bereits in einem Status Stopping (Wird angehalten) befindet, anhalten und beenden.

- Um die Ausführung zu beenden, ohne auf den Abschluss der laufenden Aktionen zu warten, können Sie die Ausführung anhalten und beenden. Schließen Sie den Parameter abandon ein. Verwenden Sie den AWS CLI , um den stop-pipeline-execution Befehl auszuführen, und geben Sie dabei den Namen der Pipeline und die Ausführungs-ID an.

  Um beispielsweise eine Pipeline mit dem Namen zu beenden *MyFirstPipeline*, geben Sie die Abandon-Option an und wählen, ob Kommentare hinzugefügt werden sollen:

  ```
  aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
  pipeline-execution-id d-EXAMPLE --abandon --reason "Stopping execution for a bug
   fix"
  ```

Stoppen einer eingehenden Ausführung (CLI)

Sie können die CLI verwenden, um eine eingehende Ausführung zu stoppen. Eine eingehende Ausführung ist eine Pipeline-Ausführung, die darauf wartet, in eine Phase überzugehen, in der der Übergang deaktiviert wurde. Wenn der Übergang aktiviert ist, befindet sich eine eingehende Ausführung, die `InProgress` weiterhin in die Phase übergeht. Eine eingehende Ausführung, die `Stopped` nicht in die Phase eintritt.

> **Note**
>
> Nachdem eine eingehende Ausführung gestoppt wurde, kann sie nicht erneut versucht werden.

Wenn Sie keine eingehende Ausführung sehen, gibt es in einer deaktivierten Phase des Übergangs keine ausstehenden Ausführungen.

Verwenden Sie AWS CLI den stop-pipeline-execution Befehl mit den folgenden Parametern, um eine eingehende Ausführung manuell zu beenden:

- Ausführungs-ID für eingehenden Datenverkehr (erforderlich)
- Kommentare (optional)
- Pipeline-Name (erforderlich)
- Beenden-Flag (optional, der Standardwert ist Falsch)

Befehlsformat:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --
pipeline-execution-id Inbound_Execution_ID [--abandon | --no-abandon] [--
reason STOP_EXECUTION_REASON]
```

Gehen Sie wie im obigen Verfahren beschrieben vor, um den Befehl einzugeben und die Stoppmethode anzugeben.

Verwenden Sie den get-pipeline-state Befehl, um den Status der eingehenden Ausführung anzuzeigen.

Erstellen Sie eine Pipeline in CodePipeline

Sie können die AWS CodePipeline Konsole oder die verwenden AWS CLI , um eine Pipeline zu erstellen. Pipelines müssen mindestens zwei Stufen enthalten. Die erste Stufe einer Pipeline muss eine Quellstufe sein. Die Pipeline muss mindestens eine weitere Stufe besitzen, bei der es sich um eine Build- oder Bereitstellungsstufe handelt.

Sie können Ihrer Pipeline Aktionen hinzufügen, die sich in einer anderen AWS Region als Ihrer Pipeline befinden. Eine regionsübergreifende Aktion ist eine Aktion, bei der an der Anbieter für eine Aktion AWS-Service ist und sich der Aktionstyp oder der Anbietertyp in einer anderen AWS Region als Ihrer Pipeline befinden. Weitere Informationen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

Sie können auch Pipelines erstellen, die containerbasierte Anwendungen erstellen und bereitstellen, indem Sie Amazon ECS als Bereitstellungsanbieter verwenden. Bevor Sie eine Pipeline erstellen,

die containerbasierte Anwendungen mit Amazon ECS bereitstellt, müssen Sie eine Image-Definitionsdatei erstellen, wie unter beschrieben. Referenz zu Abbild-Definitionsdateien

CodePipeline verwendet Methoden zur Erkennung von Änderungen, um Ihre Pipeline zu starten, wenn eine Quellcodeänderung übertragen wird. Diese Erkennungsmethoden basieren auf dem Quelltyp:

- CodePipeline verwendet Amazon CloudWatch Events, um Änderungen in Ihrem CodeCommit Quell-Repository und Branch oder Ihrem S3-Quell-Bucket zu erkennen.

> **Note**
>
> Wenn Sie die Konsole zum Erstellen oder Ändern einer Pipeline verwenden, werden die Ressourcen für die Änderungserkennung für Sie erstellt. Wenn Sie die AWS CLI verwenden, um die Pipeline zu erstellen, müssen Sie die zusätzlichen Ressourcen selbst erstellen. Weitere Informationen finden Sie unter CodeCommit Quellaktionen und EventBridge.

Themen

Erstellen einer Pipeline (Konsole)

Um eine Pipeline in der Konsole zu erstellen, müssen Sie den Standort der Quelldatei sowie Informationen zu den Providern angeben, die Sie für Ihre Aktionen verwenden werden.

Wenn Sie die Konsole zum Erstellen einer Pipeline verwenden, müssen Sie eine Source-Stufe und eine der folgenden Stufen einschließen:

- Eine Build-Stufe.
- Eine Bereitstellungsstufe.

Wenn Sie den Pipeline-Assistenten verwenden, CodePipeline erstellt er die Namen der Stufen (Source, Build, Staging). Diese Namen können nicht geändert werden. Sie können spezifischere Namen (z. B. BuildToGamma oder DeployToProd) für Stufen verwenden, die Sie später hinzufügen.

Schritt 1: Erstellen Sie Ihre Pipeline und geben Sie ihr einen Namen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Klicken Sie auf der Seite Welcome auf Create pipeline.

   Wenn Sie es zum ersten Mal verwenden CodePipeline, wählen Sie Get Started.
3. Geben Sie auf der Seite Step 1: Choose pipeline settings (Schritt 1: Pipeline-Einstellungen auswählen) unter Pipeline name (Pipeline-Name) den Namen für Ihre Pipeline ein.

   In einem einzelnen AWS Konto muss jede Pipeline, die Sie in einer AWS Region erstellen, einen eindeutigen Namen haben. Namen können für Pipelines in anderen Regionen wiederverwendet werden.

   > **Note**
   >
   > Der Name einer erstellten Pipeline kann nicht nachträglich geändert werden. Weitere Informationen zu Einschränkungen finden Sie unter Kontingente in AWS CodePipeline.
4. Wählen Sie unter Pipeline-Typ eine der folgenden Optionen aus. Die Pipeline-Typen unterscheiden sich in ihren Eigenschaften und im Preis. Weitere Informationen finden Sie unter Pipeline-Typen.

   - Pipelines vom Typ V1 haben eine JSON-Struktur, die Standardparameter auf Pipeline-, Stufen- und Aktionsebene enthält.

- Pipelines vom Typ V2 haben dieselbe Struktur wie Pipelines vom Typ V1 und bieten zusätzliche Parameterunterstützung, z. B. Trigger für Git-Tags und Variablen auf Pipeline-Ebene.

5. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:

   - Wählen Sie Neue Servicerolle aus, um die Erstellung einer neuen Servicerolle in IAM CodePipeline zu ermöglichen.
   - Wählen Sie Existing service role (Vorhandene Servicerolle), um eine Servicerolle zu verwenden, die in IAM bereits erstellt wurde. Wählen Sie unter Role ARN (Rollen-ARN) Ihren Servicerollen-ARN aus der Liste aus.

   > **Note**
   >
   > Je nachdem, wann Ihre Servicerolle erstellt wurde, müssen Sie möglicherweise ihre Berechtigungen aktualisieren, um weitere AWS-Services Funktionen zu unterstützen. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.

   Weitere Informationen über die Service-Rolle und ihre Richtlinienanweisung finden Sie unter Die CodePipeline Servicerolle verwalten.

6. (Optional) Wählen Sie unter Variablen die Option Variable hinzufügen aus, um Variablen auf Pipeline-Ebene hinzuzufügen.

   Weitere Informationen zu Variablen auf Pipeline-Ebene finden Sie unterVariablen. Ein Tutorial mit einer Variablen auf Pipelineebene, die bei der Ausführung der Pipeline übergeben wird, finden Sie unter. Tutorial: Variablen auf Pipeline-Ebene verwenden

   > **Note**
   >
   > Es ist zwar optional, Variablen auf Pipelineebene hinzuzufügen, aber bei einer Pipeline, die mit Variablen auf Pipelineebene spezifiziert ist, für die keine Werte bereitgestellt werden, schlägt die Pipelineausführung fehl.

7. (Optional) Erweitern Sie den Abschnitt Advanced Settings (Erweiterte Einstellungen).
8. Führen Sie unter Artifact store (Artefakt speichern) einen der folgenden Schritte aus:

a. Wählen Sie Standardspeicherort, um den Standardartefaktspeicher, z. B. den als Standard festgelegten S3-Artefakt-Bucket, für Ihre Pipeline in der Pipeline zu verwenden, die AWS-Region Sie für Ihre Pipeline ausgewählt haben.

b. Wählen Sie Custom location (Benutzerdefinierter Speicherort), wenn Sie in derselben Region wie Ihre Pipeline bereits über einen Artefakt-Speicher, wie z. B. einen S3-Artefakt-Bucket, verfügen. Wählen Sie unter Bucket den Bucket-Namen aus.

> **Note**
>
> Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket. Wenn Sie eine Pipeline erstellen oder bearbeiten, benötigen Sie einen Artefakt-Bucket in der Pipeline-Region und einen Artefakt-Bucket pro AWS Region, in der Sie eine Aktion ausführen.
> Weitere Informationen finden Sie unter Eingabe- und Ausgabe-Artefakte und CodePipeline Referenz zur Pipeline-Struktur.

9. Führen Sie unter Encryption key (Verschlüsselungsschlüssel) einen der folgenden Schritte aus:

   a. Um den CodePipeline Standard AWS KMS key für die Verschlüsselung der Daten im Pipeline-Artefaktspeicher (S3-Bucket) zu verwenden, wählen Sie „Verwalteter Standardschlüssel“. AWS

   b. Um Ihren vom Kunden verwalteten Schlüssel zum Verschlüsseln der Daten im Pipeline-Artefaktspeicher (S3-Bucket) zu verwenden, wählen Sie Customer Managed Key aus. Wählen Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN aus.

10. Wählen Sie Weiter aus.

Schritt 2: Erstellen Sie eine Quellstufe

- Wählen Sie auf der Seite Step 2: Add source stage (Schritt 2: Quelle) in der Dropdown-Liste Source provider (Quell-Stufe hinzufügen) den Typ des Repositorys aus, in dem der Quellcode gespeichert ist. Geben Sie die erforderlichen Optionen an und wählen Sie dann Next step (Nächster Schritt).

  - Für Bitbucket Cloud GitHub (Version 2), GitHub Enterprise Server, GitLab .com oder GitLab selbstverwaltet:

1. Wähle unter Verbindung eine bestehende Verbindung aus oder erstelle eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.
2. Wählen Sie das Repository aus, das Sie als Quellverzeichnis für Ihre Pipeline verwenden möchten.

   Wählen Sie, ob Sie einen Trigger hinzufügen oder nach Triggertypen filtern möchten, um Ihre Pipeline zu starten. Weitere Informationen zum Arbeiten mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern. Weitere Informationen zum Filtern mit Glob-Mustern finden Sie unterArbeiten mit Glob-Mustern in der Syntax.
3. Wählen Sie unter Ausgabeartefaktformat das Format für Ihre Artefakte aus.
   - Um die Ausgabeartefakte der GitHub Aktion mit der Standardmethode zu speichern, wählen Sie CodePipelineStandard. Die Aktion greift auf die Dateien aus dem GitHub Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.
   - Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone (Vollständiger Klon). Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

     Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschriebenProblembehebung CodePipeline. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiges Klonen verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

- Für Amazon S3:

1. Geben Sie in Amazon S3 location (Amazon S3-Speicherort) den S3-Bucket-Namen und den Pfad zum Objekt in einem Bucket mit aktiviertem Versioning an. Das Format des Bucket-Namens und der Pfad sehen etwa so aus:

```
s3://bucketName/folderName/objectName
```

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, können Sie Ihre Quelldatei (en) in eine einzige ZIP-Datei komprimieren und die ZIP-Datei in Ihren Quell-

Bucket hochladen. Sie können auch eine einzelne Datei ungezippt hochladen, aber nachgelagerte Aktionen, die eine ZIP-Datei erwarten, schlagen dann fehl.

2. Nachdem Sie den S3-Quell-Bucket ausgewählt haben, CodePipeline erstellt er die Amazon CloudWatch Events-Regel und den AWS CloudTrail Trail, der für diese Pipeline erstellt werden soll. Lassen Sie die Standardwerte in Change detection options (Optionen für die Änderungserkennung) unverändert. Auf diese Weise können CodePipeline Sie Amazon CloudWatch Events verwenden und AWS CloudTrail Änderungen für Ihre neue Pipeline erkennen. Wählen Sie Weiter aus.

- AWS CodeCommit:
  - Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, das Sie als Quellpfad für Ihre Pipeline verwenden möchten. Wählen Sie in Branch name aus der Dropdown-Liste die Verzweigung aus, die Sie verwenden möchten.
  - Wählen Sie unter Ausgabeartefaktformat das Format für Ihre Artefakte aus.
    - Um die Ausgabeartefakte der CodeCommit Aktion mit der Standardmethode zu speichern, wählen Sie CodePipelineStandard. Die Aktion greift auf die Dateien aus dem CodeCommit Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.
    - Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone (Vollständiger Klon). Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

      Wenn Sie diese Option wählen, müssen Sie Ihrer CodeBuild Servicerolle die `codecommit:GitPull` entsprechende Berechtigung hinzufügen, wie unter beschriebenFügen Sie CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen hinzu. Sie müssen außerdem die `codecommit:GetRepository` Berechtigungen zu Ihrer CodePipeline Servicerolle hinzufügen, wie unter beschriebenHinzufügen von Berechtigungen zur CodePipeline-Servicerolle. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiges Klonen verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.
  - Nachdem Sie den CodeCommit Repository-Namen und den Branch ausgewählt haben, wird unter Optionen zur Änderungserkennung eine Meldung mit der Amazon CloudWatch Events-Regel angezeigt, die für diese Pipeline erstellt werden soll. Lassen Sie die Standardwerte in Change detection options (Optionen für die Änderungserkennung) unverändert. Auf diese Weise können CodePipeline Sie Amazon CloudWatch Events verwenden, um Änderungen für Ihre neue Pipeline zu erkennen.

- Für Amazon ECR:
  - Wählen Sie unter Repository-Name den Namen Ihres Amazon ECR-Repositorys aus.
  - Geben Sie im Feld Image tag (Image-Tag) den Image-Namen und die Version an, falls sie nicht LATEST lautet.
  - Wählen Sie unter Ausgabeartefakte die Standardeinstellung für das Ausgabeartefakt aus, z. B. das MyApp Element, das den Bildnamen und die Repository-URI-Informationen enthält, die in der nächsten Phase verwendet werden sollen.

    Ein Tutorial zum Erstellen einer Pipeline für Amazon ECS mit CodeDeploy blaugrünen Bereitstellungen, die eine Amazon ECR-Quellphase beinhaltet, finden Sie unter. Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy

  Wenn Sie eine Amazon ECR-Quellstufe in Ihre Pipeline aufnehmen, generiert die Quellaktion eine `imageDetail.json` Datei als Ausgabeartefakt, wenn Sie eine Änderung festschreiben. Informationen zur `imageDetail.json`-Datei finden Sie unter ImageDetail.json-Datei für Amazon ECS-Bereitstellungsaktionen in Blau/Grün.

> **Note**
>
> Das Objekt und der Dateityp müssen mit dem Bereitstellungssystem kompatibel sein, das Sie verwenden möchten (z. B. Elastic Beanstalk oder CodeDeploy). Beispiele für unterstützte Dateitypen sind ZIP, TAR und TGZ. Weitere Informationen zu den unterstützten Containertypen für Elastic Beanstalk finden Sie unter Anpassen und Konfigurieren von Elastic Beanstalk Beanstalk-Umgebungen und unterstützten Plattformen. Weitere Informationen zur Bereitstellung von Revisionen mit finden Sie unter Upload Your Application Revision CodeDeploy und Prepare a Revision.

Schritt 3: Erstellen Sie eine Build-Stufe

Dieser Schritt ist optional, wenn Sie die Erstellung einer Bereitstellungsstufe planen.

- Führen Sie auf der Seite Step 3:Add build stage (Schritt 3: Build-Stufe hinzufügen) einen der folgenden Schritte aus und klicken Sie dann auf Next (Weiter):

- Wählen Sie Skip build stage (Build-Stufe überspringen), wenn Sie die Erstellung einer Bereitstellungsstufe planen.
- Wählen Sie in Build provider (Build-Anbieter) einen benutzerdefinierten Aktionsanbieter von Build-Services aus und geben Sie die Konfigurationsdetails für diesen Anbieter ein. Ein Beispiel zum Hinzufügen von Jenkins als Build-Anbieter finden Sie unter Tutorial: Erstellen einer vierstufigen Pipeline.
- Wählen Sie unter Build provider (Build-Anbieter) AWS CodeBuild aus.

  Wählen Sie unter Region die AWS Region aus, in der die Ressource vorhanden ist. Das Feld Region gibt an, wo die AWS Ressourcen für diesen Aktionstyp und Anbietertyp erstellt werden. Dieses Feld wird nur für Aktionen angezeigt, bei denen es sich bei dem Aktionsanbieter um einen handelt. AWS-Service Das Feld Region ist standardmäßig auf dieselbe AWS Region wie Ihre Pipeline eingestellt.

  Wählen Sie unter Project name (Projektname) Ihr Build-Projekt aus. Wenn Sie bereits ein Build-Projekt in erstellt haben CodeBuild, wählen Sie es aus. Sie können auch ein Build-Projekt in erstellen CodeBuild und dann zu dieser Aufgabe zurückkehren. Folgen Sie den Anweisungen unter Erstellen einer Pipeline, die verwendet wird CodeBuild im CodeBuildBenutzerhandbuch.

  Um Ihrer Build-Aktion CodeBuild Umgebungsvariablen hinzuzufügen, wählen Sie unter Umgebungsvariablen die Option Umgebungsvariable hinzufügen aus. Jede Variable besteht aus drei Einträgen:

  - Geben Sie in Name (Name) den Namen oder Schlüssel der Umgebungsvariable ein.
  - Geben Sie in Value (Wert) den Wert der Umgebungsvariable ein. Wenn Sie Parameter für den Variablentyp wählen, stellen Sie sicher, dass dieser Wert der Name eines Parameters ist, den Sie bereits im AWS Systems Manager Parameter Store gespeichert haben.

  > **Note**
  >
  > Es wird dringend davon abgeraten, Umgebungsvariablen zum Speichern sensibler Werte, insbesondere von AWS Anmeldeinformationen, zu verwenden. Wenn Sie die CodeBuild Konsole oder AWS CLI verwenden, werden Umgebungsvariablen im Klartext angezeigt. Wenn es sich um vertrauliche Werte handelt, sollten Sie stattdessen den TypParameter (Parameter) verwenden.

- (Optional) Geben Sie in Type (Typ) den Typ der Umgebungsvariablen ein. Gültige Werte sind Plaintext (Klartext) oder Parameter (Parameter). Der Standardwert ist Plaintext (Klartext).

(Optional) Wählen Sie unter Buildtyp eine der folgenden Optionen aus:

- Um jeden Build in einer einzigen Ausführung einer Build-Aktion auszuführen, wählen Sie Single Build aus.
- Um mehrere Builds in derselben Build-Aktionsausführung auszuführen, wählen Sie Batch-Build.

(Optional) Wenn Sie Batch-Builds ausführen möchten, können Sie „Alle Artefakte aus dem Batch an einem einzigen Speicherort zusammenfassen“ wählen, um alle Build-Artefakte in einem einzigen Ausgabeartefakt zu platzieren.

Schritt 4: Erstellen Sie eine Bereitstellungsstufe

Dieser Schritt ist optional, falls Sie bereits eine Build-Stufe erstellt haben.

- Führen Sie auf der Seite Step 4:Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen) einen der folgenden Schritte aus und klicken Sie dann auf Next (Weiter):

  - Wählen Sie Skip deploy stage (Bereitstellungsstufe überspringen), wenn Sie im vorherigen Schritt eine Build-Stufe erstellt haben.

    > **Note**
    >
    > Diese Option wird nicht angezeigt, falls Sie die Build-Stufe bereits übersprungen haben.

  - Wählen Sie unter Deploy provider (Bereitstellungsanbieter) eine benutzerdefinierte Aktion aus, die Sie für einen Bereitstellungsanbieter erstellt haben.

    Wählen Sie unter Region nur für regionsübergreifende Aktionen die AWS Region aus, in der die Ressource erstellt wird. Das Feld Region gibt an, wo die AWS Ressourcen für diesen Aktionstyp und Anbietertyp erstellt werden. Dieses Feld wird nur für Aktionen angezeigt, bei denen es sich bei dem Aktionsanbieter um einen handelt. AWS-Service Das Feld Region ist standardmäßig auf dieselbe AWS Region wie Ihre Pipeline eingestellt.

- Unter Deploy provider (Bereitstellungsanbieter) finden Sie folgende Felder für Standardanbieter:
  - CodeDeploy

    Geben Sie im Feld Anwendungsname den Namen einer vorhandenen CodeDeploy Anwendung ein, oder wählen Sie ihn aus. Geben Sie unter Deployment group (Bereitstellungsgruppe) den Namen einer Bereitstellungsgruppe für die Anwendung ein. Wählen Sie Weiter aus. Sie können in der CodeDeploy Konsole auch eine Anwendung, eine Bereitstellungsgruppe oder beides erstellen.
  - AWS Elastic Beanstalk

    Geben Sie im Feld Anwendungsname den Namen einer vorhandenen Elastic Beanstalk Beanstalk-Anwendung ein, oder wählen Sie ihn aus. Geben Sie unter Environment Name (Umgebungsname) eine Umgebung für die Anwendung ein. Wählen Sie Weiter aus. Sie können in der Elastic Beanstalk Beanstalk-Konsole auch eine Anwendung, eine Umgebung oder beides erstellen.
  - AWS OpsWorks Stacks

    Geben Sie unter Stack den Namen des Stacks ein, den Sie verwenden möchten. Wählen Sie unter Layer (Ebene) die Ebene aus, der Ihre Ziel-Instances angehören. Wählen Sie in App die Anwendung aus, die Sie aktualisieren und bereitstellen möchten. Wenn Sie eine App erstellen müssen, wählen Sie Create a new one in AWS OpsWorks(Eine neue in &OPSlong; erstellen) aus.

    Informationen zum Hinzufügen einer Anwendung zu einem Stack und Layer in AWS OpsWorks finden Sie unter Hinzufügen von Apps im AWS OpsWorks Benutzerhandbuch.

    Ein end-to-end Beispiel dafür, wie Sie eine einfache Pipeline CodePipeline als Quelle für Code verwenden, den Sie auf AWS OpsWorks Ebenen ausführen, finden Sie unter Verwenden CodePipeline mit AWS OpsWorks Stacks.
  - AWS CloudFormation

    Führen Sie eine der folgenden Aktionen aus:
    - Wählen Sie im Aktionsmodus die Option Stapel erstellen oder aktualisieren aus, geben Sie einen Stacknamen und einen Namen der Vorlagendatei ein und wählen Sie dann den Namen einer Rolle aus, die übernommen werden AWS CloudFormation soll.

Geben Sie optional den Namen einer Konfigurationsdatei ein und wählen Sie eine IAM-Fähigkeitsoption aus.

- Wählen Sie im Aktionsmodus die Option Änderungssatz erstellen oder ersetzen aus, geben Sie einen Stacknamen und einen Namen des Änderungssatzes ein und wählen Sie dann den Namen einer Rolle aus, die übernommen werden AWS CloudFormation soll. Geben Sie optional den Namen einer Konfigurationsdatei ein und wählen Sie eine IAM-Fähigkeitsoption.

Informationen zur Integration von AWS CloudFormation Funktionen in eine Pipeline finden Sie unter Continuous Delivery with CodePipeline im AWS CloudFormation Benutzerhandbuch. CodePipeline

- Amazon ECS

Geben Sie im Feld Clustername den Namen eines vorhandenen Amazon ECS-Clusters ein, oder wählen Sie ihn aus. Geben Sie unter Service name (Service-Name) den Namen des auf dem Cluster ausgeführten Services ein oder wählen Sie ihn aus. Sie können auch einen neuen Cluster und einen Service erstellen. Geben Sie unter Image filename (Abbild-Dateiname) den Namen der Abbild-Definitionsdatei ein, die den Container und das Abbild Ihres Services beschreibt.

> **Note**
>
> Für die Amazon ECS-Bereitstellungsaktion ist eine `imagedefinitions.json` Datei als Eingabe für die Bereitstellungsaktion erforderlich. Der Standardname der Datei ist imagedefinitions.json. Wenn Sie einen anderen Dateinamen verwenden, müssen Sie diesen während der Erstellung der Pipeline-Bereitstellungsphase angeben. Weitere Informationen finden Sie unter Datei imagedefinitions.json für Amazon ECS-Standardbereitstellungsaktionen.

Wählen Sie Weiter.

> **Note**
>
> Stellen Sie sicher, dass Ihr Amazon ECS-Cluster mit zwei oder mehr Instances konfiguriert ist. Amazon ECS-Cluster müssen mindestens zwei Instances enthalten,

sodass eine als primäre Instance beibehalten wird und eine weitere für neue Bereitstellungen verwendet wird.

Ein Tutorial zur Bereitstellung containerbasierter Anwendungen mit Ihrer Pipeline finden Sie unter Tutorial: Continuous Deployment with. CodePipeline

- Amazon ECS (Blue/Green) (Amazon ECS (Blau/Grün))

  Geben Sie die CodeDeploy Anwendungs- und Bereitstellungsgruppe, die Amazon ECS-Aufgabendefinition und die AppSpec Dateiinformationen ein, und klicken Sie dann auf Weiter.

  > **Note**
  >
  > Die Aktion Amazon ECS (Blue/Green) (Amazon ECS (Blau/Grün)) erfordert eine imageDetail.json-Datei als Eingabeartefakt für die Bereitstellungsaktion. Da die Amazon ECR-Quellaktion diese Datei erstellt, müssen Pipelines mit einer Amazon ECR-Quellaktion keine Datei bereitstellen. `imageDetail.json` Weitere Informationen finden Sie unter ImageDetail.json-Datei für Amazon ECS-Bereitstellungsaktionen in Blau/Grün.

  Ein Tutorial zum Erstellen einer Pipeline für blaugrüne Bereitstellungen in einem Amazon ECS-Cluster mit CodeDeploy finden Sie unter. Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy

- AWS Service Catalog

  Wählen Sie Enter deployment configuration (Bereitstellungskonfiguration eingeben), wenn Sie Felder in der Konsole zum Angeben Ihrer Konfiguration verwenden möchten, oder Konfigurationsdatei, falls Sie über eine separate Konfigurationsdatei verfügen. Geben Sie die Produkt- und Konfigurationsinformationen ein und klicken Sie anschließend auf Weiter.

  Ein Tutorial zur Bereitstellung von Produktänderungen in Service Catalog mit Ihrer Pipeline finden Sie unterTutorial: Erstellen Sie eine Pipeline, die in Service Catalog bereitgestellt wird.

- Alexa Skills Kit

Geben Sie unter Alexa Skill-ID die Skill-ID für Ihre Alexa-Qualifikation ein. Geben Sie unter Client-ID und Clientschlüssel die Anmeldeinformationen ein, die über ein Login with Amazon-Sicherheitsprofil (LWA) generiert wurden. Geben Sie unter Aktualisierungs-Token das Aktualisierungs-Token ein, das Sie mit dem ASK-CLI-Befehl zum Abrufen eines Aktualisierungs-Tokens generiert haben. Wählen Sie Weiter aus.

Ein Tutorial zur Bereitstellung von Alexa-Qualifikationen mit Ihrer Pipeline und zur Generierung der LWA-Anmeldeinformationen finden Sie unter Tutorial: Erstellen einer Pipeline zum Bereitstellen eines Amazon Alexa-Skills.

- Amazon S3

Geben Sie unter Bucket den Namen des gewünschten S3-Buckets ein. Wählen Sie Extract file before deploy (Datei vor Bereitstellung extrahieren), falls das Eingabeartefakt für Ihre Bereitstellungsstufe eine ZIP-Datei ist. Wenn Extract file before deploy (Datei vor Bereitstellung extrahieren) ausgewählt ist, können Sie optional einen Wert für Deployment path (Bereitstellungspfad) als Speicherort eingeben, an dem Ihre ZIP-Datei entzippt wird. Wenn die Option nicht aktiviert ist, müssen Sie einen Wert unter S3 object key (S3-Objektschlüssel) eingeben.

> **Note**
>
> Die meisten Ausgabeartefakte von Quell- und Build-Stufen werden gezippt. Alle Pipeline-Quellanbieter außer Amazon S3 komprimieren Ihre Quelldateien, bevor Sie sie als Eingabeartefakt für die nächste Aktion bereitstellen.

(Optional) Geben Sie in Canned ACL die gespeicherte ACL ein, die auf das in Amazon S3 bereitgestellte Objekt angewendet werden soll.

> **Note**
>
> Beim Anwenden einer vordefinierten ACL werden alle auf das Objekt angewandten vorhandenen ACLs überschrieben.

(Optional) Geben Sie unter Cache control (Cache-Kontrolle) die Cache-Kontroll-Parameter für Anforderungen zum Herunterladen von Objekten aus dem Bucket an. Eine Liste der

gültigen Werte finden Sie im `Cache-Control`-Header-Feld für HTTP-Operationen. Um mehrere Werte in die Cache-Steuerung einzugeben, verwenden Sie ein Komma zwischen den einzelnen Werten. Sie können nach jedem Komma ein Leerzeichen hinzufügen (optional), wie in diesem Beispiel gezeigt.

Der vorherige Beispieleintrag wird in der CLI wie folgt angezeigt:

```
"CacheControl": "public, max-age=0, no-transform"
```

Wählen Sie Weiter aus.

Ein Tutorial zum Erstellen einer Pipeline mit einem Amazon S3 S3-Bereitstellungsaktionsanbieter finden Sie unterTutorial: Erstellen Sie eine Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet.

Schritt 5: Überprüfen der Pipeline

- Prüfen Sie auf der Seite Step 5: Review die Konfiguration Ihrer Pipeline und wählen Sie dann Create pipeline zur Erstellung der Pipeline oder Previous zum Bearbeiten aus. Um den Assistenten ohne Erstellen einer Pipeline zu beenden, wählen Sie Cancel aus.

Nachdem Sie die Pipeline erstellt haben, können Sie sie in der Konsole anzeigen. Die Pipeline wird nach dem Erstellen ausgeführt. Weitere Informationen finden Sie unter Pipelines und Details anzeigen in CodePipeline. Weitere Informationen über die Durchführung von Änderungen an Ihrer Pipeline finden Sie unter Eine Pipeline bearbeiten in CodePipeline.

Erstellen einer Pipeline (CLI)

Um die zum Erstellen einer Pipeline AWS CLI zu verwenden, erstellen Sie eine JSON-Datei, um die Pipeline-Struktur zu definieren, und führen dann den create-pipeline Befehl mit dem `--cli-input-json` Parameter aus.

> **Important**
>
> Sie können den nicht verwenden AWS CLI , um eine Pipeline zu erstellen, die Partneraktionen enthält. Sie müssen stattdessen die CodePipeline Konsole verwenden.

Weitere Informationen zur Pipeline-Struktur finden Sie unter CodePipeline Referenz zur Pipeline-Struktur und create-pipeline in der CodePipeline API-Referenz.

Zum Erstellen einer JSON-Datei verwenden Sie die Beispiel-Pipeline-JSON-Datei, bearbeiten Sie und rufen diese Datei dann auf, wenn Sie den Befehl create-pipeline ausführen.

Voraussetzungen:

Sie benötigen den ARN der Servicerolle, für die Sie CodePipeline in erstellt habenErste Schritte mit CodePipeline. Sie verwenden die CodePipeline Dienstrolle ARN in der Pipeline-JSON-Datei, wenn Sie den create-pipeline Befehl ausführen. Weitere Informationen zum Erstellen einer Servicerolle finden Sie unter Erstellen Sie die CodePipeline Servicerolle. Im Gegensatz zur Konsole besteht bei der Ausführung des create-pipeline Befehls in der AWS CLI nicht die Möglichkeit, die CodePipeline Servicerolle für Sie zu erstellen. Die Servicerolle muss bereits vorhanden sein.

Sie benötigen den Namen eines S3-Buckets, in dem die Artefakte für die Pipeline gespeichert werden. Dieser Bucket muss sich in derselben Region wie die Pipeline befinden. Sie verwenden den Bucketnamen in der Pipeline-JSON-Datei, wenn Sie den Befehl create-pipeline ausführen. Im Gegensatz zur Konsole wird bei der Ausführung des create-pipeline Befehls in der AWS CLI kein S3-Bucket zum Speichern von Artefakten erstellt. Der Bucket muss bereits vorhanden sein.

> **Note**
>
> Auch mit dem Befehl get-pipeline erhalten Sie eine Kopie der JSON-Struktur dieser Pipeline. Sie können diese dann in einem Texteditor entsprechend modifizieren.

So erstellen Sie die JSON-Datei

1. Erstellen Sie an einem Terminal (Linux, macOS oder Unix) oder einer Befehlszeile (Windows) eine neue Textdatei in einem lokalen Verzeichnis.
2. (Optional) Sie können eine oder mehrere Variablen auf Pipeline-Ebene hinzufügen. Sie können in der Konfiguration von CodePipeline Aktionen auf diesen Wert verweisen. Sie können die

Variablennamen und Werte hinzufügen, wenn Sie die Pipeline erstellen, und Sie können auch festlegen, dass Werte zugewiesen werden, wenn Sie die Pipeline in der Konsole starten.

> **Note**
>
> Es ist zwar optional, Variablen auf Pipeline-Ebene hinzuzufügen, aber bei einer Pipeline, die mit Variablen auf Pipelineebene spezifiziert ist und für die keine Werte bereitgestellt werden, schlägt die Pipeline-Ausführung fehl.

Eine Variable auf Pipelineebene wird zur Laufzeit der Pipeline aufgelöst. Alle Variablen sind unveränderlich, was bedeutet, dass sie nicht aktualisiert werden können, nachdem ein Wert zugewiesen wurde. Variablen auf Pipeline-Ebene mit aufgelösten Werten werden im Verlauf jeder Ausführung angezeigt.

Sie stellen Variablen auf Pipeline-Ebene mithilfe des Variablen-Attributs in der Pipeline-Struktur bereit. Im folgenden Beispiel `Variable1` hat die Variable den Wert`Value1`.

```
"variables": [
     {
         "name": "Timeout",
         "defaultValue": "1000",
         "description": "description"
     }
 ]
```

Fügen Sie diese Struktur im folgenden Schritt zu Ihrem Pipeline-JSON oder zum Beispiel-JSON hinzu. Weitere Informationen zu Variablen, einschließlich Namespace-Informationen, finden Sie unterVariablen.

3. Öffnen Sie die Datei in einem Texteditor und bearbeiten Sie die Werte entsprechend der Struktur, die Sie erstellen möchten. Sie müssen mindestens den Namen der Pipeline ändern. Sie sollten auch überlegen, ob Sie Folgendes ändern möchten:

   - Den S3-Bucket, in dem Artefakte für diese Pipeline gespeichert sind.
   - Den Speicherort für Ihren Code.
   - Den Bereitstellungsanbieter.
   - Art der Bereitstellung des Codes.

- Die Tags für Ihre Pipeline.

Die folgende Beispielstruktur für eine Pipeline mit zwei Phasen zeigt die Werte an, deren Änderung Sie für Ihre Pipeline in Betracht ziehen sollten. Ihre Pipeline enthält wahrscheinlich mehr als zwei Phasen:

```
{
    "pipeline": {
        "roleArn": "arn:aws:iam::80398EXAMPLE::role/AWS-CodePipeline-Service",
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "inputArtifacts": [],
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "S3"
                        },
                        "outputArtifacts": [
                            {
                                "name": "MyApp"
                            }
                        ],
                        "configuration": {
                            "S3Bucket": "awscodepipeline-demobucket-example-date",
                            "S3ObjectKey": "ExampleCodePipelineSampleBundle.zip",
                            "PollForSourceChanges": "false"
                        },
                        "runOrder": 1
                    }
                ]
            },
            {
                "name": "Staging",
                "actions": [
                    {
                        "inputArtifacts": [
                            {
```

```
                            "name": "MyApp"
                        }
                    ],
                    "name": "Deploy-CodeDeploy-Application",
                    "actionTypeId": {
                        "category": "Deploy",
                        "owner": "AWS",
                        "version": "1",
                        "provider": "CodeDeploy"
                    },
                    "outputArtifacts": [],
                    "configuration": {
                        "ApplicationName": "CodePipelineDemoApplication",
                        "DeploymentGroupName": "CodePipelineDemoFleet"
                    },
                    "runOrder": 1
                }
            ]
        }
    ],
    "artifactStore": {
        "type": "S3",
        "location": "codepipeline-us-east-2-250656481468"
    },
    "name": "MyFirstPipeline",
    "version": 1,
    "variables": [
        {
            "name": "Timeout",
            "defaultValue": "1000",
            "description": "description"
        }
      ]
    },
    "triggers": [
        {
            "providerType": "CodeStarSourceConnection",
            "gitConfiguration": {
                "sourceActionName": "Source",
                "push": [
                    {
                        "tags": {
                            "includes": [
                                "v1"
```

```
                                ],
                                "excludes": [
                                    "v2"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline",
        "updated": 1501626591.112,
        "created": 1501626591.112
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
    }]
}
```

In diesem Beispiel wird der Pipeline Tagging hinzugefügt, indem der Pipeline der Tag-Schlüssel `Project` und der Wert `ProjectA` hinzugefügt werden. Weitere Informationen zum Markieren von Ressourcen in finden Sie CodePipeline unter. Markieren von Ressourcen

Stellen Sie sicher, dass der `PollForSourceChanges`-Parameter in Ihrer JSON-Datei wie folgt festgelegt ist:

```
                "PollForSourceChanges": "false",
```

CodePipeline verwendet Amazon CloudWatch Events, um Änderungen in Ihrem CodeCommit Quell-Repository und Branch oder Ihrem S3-Quell-Bucket zu erkennen. Der nächste Schritt enthält Anweisungen zum manuellen Erstellen dieser Ressourcen für Ihre Pipeline. Wenn das Flag auf `false` gesetzt wird, werden regelmäßige Prüfungen deaktiviert, die nicht erforderlich sind, wenn die empfohlenen Methoden zur Änderungserkennung verwendet werden.

4. Um eine Build-, Test- oder Bereitstellungsaktion in einer anderen Region als Ihre Pipeline zu erstellen, müssen Sie Ihrer Pipeline-Struktur Folgendes hinzufügen. Anweisungen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

- Fügen Sie den `Region`-Parameter der Pipeline-Struktur Ihrer Aktion hinzu.
- Verwenden Sie den `artifactStores` Parameter, um einen Artefakt-Bucket für jede AWS Region anzugeben, in der Sie eine Aktion ausführen.

5. Wenn Sie mit der Struktur zufrieden sind, speichern Sie die Datei mit einem Namen wie **`pipeline.json`**.

So erstellen Sie eine Pipeline

1. Führen Sie den Befehl create-pipeline aus und verwenden Sie die Parameter `--cli-input-json`, um die zuvor erstellte JSON-Datei anzugeben.

   Um eine Pipeline *`MySecondPipeline`*mit einer JSON-Datei namens pipeline.json zu erstellen, die den Namen "*`MySecondPipeline`*" als Wert für `name` in der JSON-Datei enthält, würde Ihr Befehl wie folgt aussehen:

   ```
   aws codepipeline create-pipeline --cli-input-json file://pipeline.json
   ```

   > **Important**
   >
   > Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

   Dieser Befehl gibt die Struktur der gesamten erstellten Pipeline zurück.

2. Um die Pipeline anzuzeigen, öffnen Sie entweder die CodePipeline Konsole und wählen Sie sie aus der Pipeline-Liste aus, oder verwenden Sie den Befehl. get-pipeline-state Weitere Informationen finden Sie unter Pipelines und Details anzeigen in CodePipeline.
3. Wenn Sie die CLI zum Erstellen einer Pipeline verwenden, müssen Sie die empfohlenen Änderungserkennungsressourcen für Ihre Pipeline manuell erstellen:

   - Für eine Pipeline mit einem CodeCommit Repository müssen Sie die CloudWatch Ereignisregel manuell erstellen, wie unter beschriebenEine EventBridge Regel für eine CodeCommit Quelle erstellen (CLI).

- Für eine Pipeline mit einer Amazon S3 S3-Quelle müssen Sie die Regel und den AWS CloudTrail Trail für CloudWatch Ereignisse manuell erstellen, wie unter beschriebenAmazon S3 S3-Quellaktionen und EventBridge mit AWS CloudTrail.

Amazon ECR-Quellaktionen und Ressourcen EventBridge

Um eine Amazon ECR-Quellaktion hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Verwenden Sie den Assistenten „Pipeline erstellen“ (Erstellen einer Pipeline (Konsole)) der CodePipeline Konsole oder die Aktionsseite „Aktion bearbeiten“, um die Option Amazon ECR Provider auszuwählen. Die Konsole erstellt eine EventBridge Regel, die Ihre Pipeline startet, wenn sich die Quelle ändert.
- Verwenden Sie die CLI, um die Aktionskonfiguration für die `ECR` Aktion hinzuzufügen und zusätzliche Ressourcen wie folgt zu erstellen:
  - Verwenden Sie die `ECR` Beispiel-Aktionskonfiguration inAmazon ECR, um Ihre Aktion zu erstellen, wie unter gezeigtErstellen einer Pipeline (CLI).
  - Die Methode zur Erkennung von Änderungen startet die Pipeline standardmäßig durch Abfragen der Quelle. Sie sollten die regelmäßigen Prüfungen deaktivieren und die Regel zur Erkennung von Änderungen manuell erstellen. Verwenden Sie eine der folgenden Methoden:Eine EventBridge Regel für eine Amazon ECR-Quelle (Konsole) erstellen,Eine EventBridge Regel für eine Amazon ECR-Quelle (CLI) erstellen, oderEine EventBridge Regel für eine Amazon ECR-Quelle erstellen (AWS CloudFormation Vorlage) .

Themen

Eine EventBridge Regel für eine Amazon ECR-Quelle (Konsole) erstellen

So erstellen Sie eine EventBridge Regel zur Verwendung in CodePipeline Vorgängen (Amazon ECR-Quelle)

1. Öffnen Sie die EventBridge Amazon-Konsole unter https://console.aws.amazon.com/events/.

2. Wählen Sie im Navigationsbereich die Option Events.
3. Wählen Sie Create Rule und dann unter Event source unter Service Name die Option Elastic Container Registry (ECR) aus.
4. Wählen Sie unter Event source (Ereignisquelle) die Option Event Pattern (Ereignismuster) aus.

   Wählen Sie Bearbeiten und fügen Sie dann das folgende Beispiel-Event-Muster in das Fenster „Ereignisquelle“ für ein `eb-test` Repository mit dem folgenden Image-Tag ein: `cli-testing`

```
{
    "detail-type": [
        "ECR Image Action"
    ],
    "source": [
        "aws.ecr"
    ],
    "detail": {
        "action-type": [
            "PUSH"
        ],
        "image-tag": [
            "latest"
        ],
        "repository-name": [
            "eb-test"
        ],
        "result": [
            "SUCCESS"
        ]
    }
}
```

> **Note**
>
> Das vollständige Ereignismuster, das für Amazon ECR-Ereignisse unterstützt wird, finden Sie unter Amazon ECR Events EventBridge und/oder Amazon Elastic Container Registry Events.

5. Wählen Sie Speichern.

   Zeigen Sie die Regel im Bereich Event Pattern Preview an.

6. Wählen Sie unter Ziele die Option. CodePipeline
7. Geben Sie den Pipeline-ARN für die Pipeline ein, die mit dieser Regel gestartet werden soll.

> **Note**
>
> Sie finden den Pipeline-ARN in der Metadatenausgabe, nachdem Sie den Befehl get-pipeline ausgeführt haben. Der Pipeline-ARN wird in folgendem Format erstellt:
> *arn:aws:codepipeline: region: konto: Pipeline-Name*
> Pipeline-Beispiel-ARN:
> `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

8. Erstellen oder spezifizieren Sie eine IAM-Dienstrolle, die EventBridge Berechtigungen zum Aufrufen des mit Ihrer EventBridge Regel verknüpften Ziels gewährt (in diesem Fall ist das Ziel). CodePipeline
   - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen aus, um eine Servicerolle zu erstellen, die Ihnen EventBridge Berechtigungen für den Start Ihrer Pipeline-Ausführungen erteilt.
   - Wählen Sie Bestehende Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihnen EventBridge Berechtigungen für den Start Ihrer Pipeline-Ausführungen erteilt.
9. Überprüfen Sie die eingerichteten Regeln, um sicherzustellen, dass sie Ihren Anforderungen entsprechen.
10. Wählen Sie Details konfigurieren.
11. Geben Sie auf der Seite Configure rule details (Regeldetails konfigurieren) einen Namen und eine Beschreibung für die Regel ein und wählen Sie dann State (Status), um die Regel zu aktivieren.
12. Wenn Sie mit der Regel zufrieden sind, klicken Sie auf Create rule (Regel erstellen).

Eine EventBridge Regel für eine Amazon ECR-Quelle (CLI) erstellen

Rufen Sie den Befehl put-rule mit folgenden Angaben auf:

- Einen Namen, der die von Ihnen erstellte Regel eindeutig bezeichnet. Dieser Name muss für alle Pipelines, die Sie erstellen und die mit Ihrem AWS Konto CodePipeline verknüpft sind, eindeutig sein.

- Das Ereignismuster für die Quelle und von der Regel verwendete Detailfelder. Weitere Informationen finden Sie unter Amazon EventBridge und Event Patterns.

Um eine EventBridge Regel mit Amazon ECR als Ereignisquelle und CodePipeline als Ziel zu erstellen

1. Fügen Sie Berechtigungen hinzu EventBridge , die CodePipeline zum Aufrufen der Regel verwendet werden sollen. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie zu erstellen, die es ermöglicht, die Servicerolle EventBridge zu übernehmen. Geben Sie der Vertrauensrichtlinie den Namen `trustpolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

   c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie in diesem Beispiel für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Verwenden Sie den folgenden Befehl, um die Berechtigungsrichtlinie `CodePipeline-Permissions-Policy-for-EB` der Rolle `Role-for-MyRule` anzufügen.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen dieser Richtlinie zur Rolle werden Berechtigungen für erstellt EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

   Warum nehme ich diese Änderung vor? Sie müssen ein Ereignis mit einer Regel erstellen, die festlegt, wie ein Image-Push durchgeführt werden muss, und einem Ziel, das die Pipeline benennt, die durch das Ereignis gestartet werden soll.

   Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „`MyECRRepoRule`“ erstellt.

```
aws events put-rule --name "MyECRRepoRule" --event-pattern "{\"detail-type\":[\"ECR
 Image Action\"],\"source\":[\"aws.ecr\"],\"detail\":{\"action-type\":[\"PUSH\"],
\"image-tag\":[\"latest\"],\"repository-name\":[\"eb-test\"],\"result\":[\"SUCCESS
\"]}}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

> **Note**
>
> Das vollständige Ereignismuster, das für Amazon ECR-Ereignisse unterstützt wird, finden Sie unter Amazon ECR Events EventBridge und/oder Amazon Elastic Container Registry Events.

3. Um es CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die folgenden Parameter an:

   - Der Parameter `--rule` wird für den `rule_name` verwendet, den Sie mit put-rule erstellt haben.
   - Der Parameter `--targets` wird für die Listen-`Id` des Ziels in der Zielliste und den `ARN` der Ziel-Pipeline verwendet.

   Der folgende Beispielbefehl legt fest, dass für die Regel mit dem Namen `MyECRRepoRule` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dieses Ziel 1 ist. Der Beispielbefehl gibt auch einen Beispiel-`Arn` für die Pipeline und den Beispiel-`RoleArn` für die Regel an. Die Pipeline startet, wenn Änderungen im Repository auftreten.

   ```
   aws events put-targets --rule MyECRRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-
   west-2:80398EXAMPLE:TestPipeline,RoleArn=arn:aws:iam::80398EXAMPLE:role/Role-for-
   MyRule
   ```

Eine EventBridge Regel für eine Amazon ECR-Quelle erstellen (AWS CloudFormation Vorlage)

Um eine Regel AWS CloudFormation zu erstellen, verwenden Sie das Vorlagen-Snippet, wie hier gezeigt.

Um Ihre AWS CloudFormation Pipeline-Vorlage zu aktualisieren und eine Regel zu erstellen EventBridge

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` AWS CloudFormation Ressource`Resources`, um die IAM-Rolle zu konfigurieren, mit der Ihre Veranstaltung Ihre Pipeline starten kann. Dieser Eintrag erstellt eine Rolle mit zwei Richtlinien:

   - Die erste Richtlinie ermöglicht die Übernahme der Rolle.
   - Die zweite Richtlinie stellt Berechtigungen zum Starten der Pipeline bereit.

Warum nehme ich diese Änderung vor? Sie müssen eine Rolle erstellen, die übernommen werden kann, EventBridge um eine Ausführung in unserer Pipeline zu starten.

YAML

```
  EventRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Sub arn:aws:codepipeline:${AWS::Region}:
${AWS::AccountId}:${AppPipeline}
```

JSON

```
{
    "EventRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
```

```
                        "Principal": {
                            "Service": [
                                "events.amazonaws.com"
                            ]
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "eb-pipeline-execution",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "codepipeline:StartPipelineExecution",
                                "Resource": {
                                    "Fn::Sub": "arn:aws:codepipeline:
${AWS::Region}:${AWS::AccountId}:${AppPipeline}"
                                }
                            }
                        ]
                    }
                }
            ]
        }
    }
}
...
```

2. Verwenden Sie in der Vorlage unter die `AWS::Events::Rule` AWS CloudFormation Ressource`Resources`, um eine EventBridge Regel für die Amazon ECR-Quelle hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis, das Commits in Ihrem Repository überwacht. Wenn eine Änderung des Repository-Status EventBridge erkannt wird, wird die Regel in Ihrer Zielpipeline aufgerufen`StartPipelineExecution`.

   Warum nehme ich diese Änderung vor? Sie müssen ein Ereignis mit einer Regel erstellen, die festlegt, wie ein Image-Push durchgeführt werden muss, und einem Ziel, das die Pipeline benennt, die durch das Ereignis gestartet werden soll.

Dieses Snippet verwendet ein Bild mit dem Namen `eb-test` und dem Tag `latest`.

YAML

```
EventRule:
  Type: 'AWS::Events::Rule'
  Properties:
    EventPattern:
      detail:
        action-type: [PUSH]
        image-tag: [latest]
        repository-name: [eb-test]
        result: [SUCCESS]
      detail-type: [ECR Image Action]
      source: [aws.ecr]
    Targets:
      - Arn: !Sub arn:aws:codepipeline:${AWS::Region}:${AWS::AccountId}:
${AppPipeline}
        RoleArn: !GetAtt
          - EventRole
          - Arn
        Id: codepipeline-AppPipeline
```

JSON

```
{
    "EventRule": {
        "Type": "AWS::Events::Rule",
        "Properties": {
            "EventPattern": {
                "detail": {
                    "action-type": [
                        "PUSH"
                    ],
                    "image-tag": [
                        "latest"
                    ],
                    "repository-name": [
                        "eb-test"
                    ],
                    "result": [
```

```
                        "SUCCESS"
                    ]
                },
                "detail-type": [
                    "ECR Image Action"
                ],
                "source": [
                    "aws.ecr"
                ]
            },
            "Targets": [
                {
                    "Arn": {
                        "Fn::Sub": "arn:aws:codepipeline:${AWS::Region}:
${AWS::AccountId}:${AppPipeline}"
                    },
                    "RoleArn": {
                        "Fn::GetAtt": [
                            "EventRole",
                            "Arn"
                        ]
                    },
                    "Id": "codepipeline-AppPipeline"
                }
            ]
        }
    }
},
```

> **Note**
>
> Das vollständige Ereignismuster, das für Amazon ECR-Ereignisse unterstützt wird, finden Sie unter Amazon ECR Events EventBridge und/oder Amazon Elastic Container Registry Events.

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation -Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).

5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Amazon S3 S3-Quellaktionen und EventBridge mit AWS CloudTrail

Um eine Amazon S3 S3-Quellaktion hinzuzufügen CodePipeline, wählen Sie entweder:

- Verwenden Sie den Assistenten „Pipeline erstellen“ (Erstellen einer Pipeline (Konsole)) oder die Aktionsseite „Bearbeiten“ der CodePipeline Konsole, um die Option S3-Anbieter auszuwählen. Die Konsole erstellt eine EventBridge Regel und einen CloudTrail Trail, die Ihre Pipeline starten, wenn sich die Quelle ändert.
- Verwenden Sie den AWS CLI , um die Aktionskonfiguration für die S3 Aktion hinzuzufügen und zusätzliche Ressourcen wie folgt zu erstellen:
  - Verwenden Sie die S3 Beispiel-Aktionskonfiguration inAmazon S3 S3-Quellaktion, um Ihre Aktion zu erstellen, wie unter gezeigtErstellen einer Pipeline (CLI).
  - Die Methode zur Erkennung von Änderungen startet die Pipeline standardmäßig durch Abfragen der Quelle. Sie sollten die regelmäßigen Prüfungen deaktivieren und die Regel und das Protokoll zur Änderungserkennung manuell erstellen. Verwenden Sie eine der folgenden Methoden:Eine EventBridge Regel für eine Amazon S3 S3-Quelle (Konsole) erstellen,Eine EventBridge Regel für eine Amazon S3 S3-Quelle (CLI) erstellen, oderEine EventBridge Regel für eine Amazon S3 S3-Quelle erstellen (AWS CloudFormation Vorlage) .

AWS CloudTrail ist ein Service, der Ereignisse in Ihrem Amazon S3 S3-Quell-Bucket protokolliert und filtert. Der Trail sendet die gefilterten Quelländerungen an die EventBridge Regel. Die EventBridge Regel erkennt die Quellenänderung und startet dann Ihre Pipeline.

Voraussetzungen:

- Wenn Sie keinen Trail erstellen, verwenden Sie einen vorhandenen AWS CloudTrail Trail, um Ereignisse in Ihrem Amazon S3 S3-Quell-Bucket zu protokollieren und gefilterte Ereignisse an die EventBridge Regel zu senden.
- Erstellen oder verwenden Sie einen vorhandenen S3-Bucket, in dem die Protokolldateien gespeichert werden AWS CloudTrail können. AWS CloudTrail muss über die erforderlichen Berechtigungen verfügen, um Protokolldateien an einen Amazon S3 S3-Bucket zu liefern. Der

Bucket kann nicht als Zahlung durch den Anforderer-Bucket konfiguriert werden. Wenn Sie im Rahmen der Erstellung oder Aktualisierung eines Trails in der Konsole einen Amazon S3 S3-Bucket erstellen, AWS CloudTrail fügt die erforderlichen Berechtigungen für Sie einem Bucket hinzu. Weitere Informationen finden Sie unter Amazon S3 S3-Bucket-Richtlinie für CloudTrail.

Eine EventBridge Regel für eine Amazon S3 S3-Quelle (Konsole) erstellen

Bevor Sie eine Regel in einrichten EventBridge, müssen Sie einen AWS CloudTrail Trail erstellen. Weitere Informationen finden Sie unter Erstellen eines Trails in der Konsole.

> **Important**
>
> Wenn Sie die Konsole verwenden, um Ihre Pipeline zu erstellen oder zu bearbeiten, werden Ihre EventBridge Regel und Ihr AWS CloudTrail Trail für Sie erstellt.

Sie erstellen einen Trail wie folgt:

1. Öffnen Sie die AWS CloudTrail Konsole.
2. Wählen Sie im Navigationsbereich Trails aus.
3. Wählen Sie Create Trail (Trail erstellen) aus. Geben Sie unter Trail name (Trail-Name) einen Namen für Ihren Trail ein.
4. Erstellen oder geben Sie unter Storage location (Speicherort) den Bucket an, der zum Speichern der Protokolldateien verwendet werden soll. Standardmäßig werden Amazon-S3-Buckets und -Objekte als privat eingestuft. Nur der Besitzer der Ressource (das AWS Konto, das den Bucket erstellt hat) kann auf den Bucket und seine Objekte zugreifen. Der Bucket muss über eine Ressourcenrichtlinie verfügen, die AWS CloudTrail Berechtigungen für den Zugriff auf die Objekte im Bucket gewährt.
5. Geben Sie unter Trail-Protokoll-Bucket und -Ordner einen Amazon S3 S3-Bucket und das Objektpräfix (Ordnername) an, um Datenereignisse für alle Objekte im Ordner zu protokollieren. Sie können jedem Trail 250 Amazon S3-Objekte hinzufügen. Geben Sie die erforderlichen Informationen zum Verschlüsselungsschlüssel ein und wählen Sie Weiter.
6. Wählen Sie als Ereignistyp die Option Verwaltungsereignisse aus.
7. Wählen Sie für Verwaltungsereignisse die Option Schreiben aus. Der Trail zeichnet Amazon S3 S3-API-Aktivitäten auf Objektebene (z. B. `GetObject` und`PutObject`) für den angegebenen Bucket und das angegebene Präfix auf.

8. Wählen Sie Write (Schreiben) aus.
9. Wenn Sie mit dem Trail zufrieden sind, wählen Sie Create Trail.

Um eine EventBridge Regel zu erstellen, die auf Ihre Pipeline mit einer Amazon S3 S3-Quelle abzielt

1. Öffnen Sie die EventBridge Amazon-Konsole unter https://console.aws.amazon.com/events/.
2. Wählen Sie im Navigationsbereich Regeln aus. Lassen Sie den Standardbus ausgewählt oder wählen Sie einen Event-Bus. Wählen Sie Regel erstellen aus.
3. Geben Sie im Feld Name einen Namen für Ihre Regel ein.
4. Wählen Sie unter Regeltyp die Option Regel mit einem Ereignismuster aus. Wählen Sie Weiter aus.
5. Wählen Sie unter Ereignisquelle AWS Ereignisse oder EventBridge Partnerereignisse aus.
6. Wählen Sie unter Beispiel-Ereignistyp die Option AWS Ereignisse aus.
7. Geben Sie im Feld Beispielereignisse S3 als Schlüsselwort ein, nach dem gefiltert werden soll. Wählen Sie AWS API-Aufruf über CloudTrail.
8. Wählen Sie unter Erstellungsmethode die Option Kundenmuster (JSON-Editor) aus.

   Fügen Sie das unten angegebene Ereignismuster ein. Stellen Sie sicher, dass Sie den Bucket-Namen und den S3-Objektschlüssel (oder den Schlüsselnamen) hinzufügen, mit denen das Objekt im Bucket eindeutig identifiziert wird`requestParameters`. In diesem Beispiel wird eine Regel für einen Bucket mit dem Namen `my-bucket` und dem Objektschlüssel von erstellt`my-files.zip`. Wenn Sie das Fenster Edit (Bearbeiten) zum Angeben von Ressourcen verwenden, wird Ihre Regel für die Verwendung eines benutzerdefinierten Ereignismusters aktualisiert.

   Im Folgenden finden Sie ein Beispiel-Ereignismuster, das Sie kopieren und einfügen können:

```
{
    "source": [
        "aws.s3"
    ],
    "detail-type": [
        "AWS API Call via CloudTrail"
    ],
    "detail": {
        "eventSource": [
            "s3.amazonaws.com"
```

```
        ],
        "eventName": [
            "CopyObject",
            "CompleteMultipartUpload",
            "PutObject"
        ],
        "requestParameters": {
            "bucketName": [
                "my-bucket"
            ],
            "key": [
                "my-files.zip"
            ]
        }
    }
}
```

9. Wählen Sie Weiter aus.
10. Wählen Sie unter Zieltypen die Option AWS Dienst aus.
11. Wählen Sie unter Ziel auswählen die Option aus CodePipeline. Geben Sie im Feld Pipeline-ARN den Pipeline-ARN für die Pipeline ein, die mit dieser Regel gestartet werden soll.

> **Note**
>
> Um den Pipeline-ARN zu erhalten, führen Sie den Befehl get-pipeline aus. Der Pipeline-ARN wird in der Ausgabe angezeigt. Er wird in folgendem Format erstellt:
> *arn:aws:codepipeline: region: konto: Pipeline-Name*
> Pipeline-Beispiel-ARN:
> arn:aws:codepipeline:us-east- 2:80398 BEISPIEL: MyFirstPipeline

12. So erstellen oder spezifizieren Sie eine IAM-Dienstrolle, die EventBridge Berechtigungen zum Aufrufen des mit Ihrer Regel verknüpften Ziels gewährt (in diesem Fall lautet das Ziel): EventBridge CodePipeline

    - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen aus, um eine Servicerolle zu erstellen, die Ihnen EventBridge Berechtigungen für den Start Ihrer Pipeline-Ausführung erteilt.
    - Wählen Sie Bestehende Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihnen EventBridge Berechtigungen für den Start Ihrer Pipeline-Ausführungen erteilt.

13. Wählen Sie Weiter aus.

14. Wählen Sie auf der Seite „Tags“ die Option Weiter aus.
15. Überprüfen Sie auf der Seite Überprüfen und erstellen die Regelkonfiguration. Wenn Sie mit der Regel zufrieden sind, klicken Sie auf Create rule (Regel erstellen).

Eine EventBridge Regel für eine Amazon S3 S3-Quelle (CLI) erstellen

Um einen AWS CloudTrail Trail zu erstellen und die Protokollierung zu aktivieren

Um mit dem einen Trail AWS CLI zu erstellen, rufen Sie den create-trail Befehl auf und geben Sie Folgendes an:

- Den Trail-Namen.
- Der Bucket, auf den Sie bereits die Bucket-Richtlinie für AWS CloudTrail angewendet haben.

Weitere Informationen finden Sie unter Erstellen eines Pfads mit der AWS Befehlszeilenschnittstelle.

1. Rufen Sie den Befehl create-trail auf und beziehen Sie die Parameter `--name` und `--s3-bucket-name` ein.

   Warum nehme ich diese Änderung vor? Dadurch wird der für Ihren S3-Quell-Bucket erforderliche CloudTrail Trail erstellt.

   Der folgende Befehl verwendet `--name` und `--s3-bucket-name` zum Erstellen eines Trails mit dem Namen `my-trail` und eines Buckets mit dem Namen `myBucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name myBucket
   ```

2. Rufen Sie den Befehl start-logging auf und beziehen Sie den `--name`-Parameter ein.

   Warum nehme ich diese Änderung vor? Dieser Befehl startet die CloudTrail Protokollierung für Ihren Quell-Bucket und sendet Ereignisse an EventBridge.

   Beispiel:

   Im folgenden Befehl wird `--name` verwendet, um die Protokollierung auf einem Trail mit der Bezeichnung `my-trail` zu starten.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Rufen Sie den Befehl put-event-selectors auf und beziehen Sie die Parameter `--trail-name` und `--event-selectors` ein. Verwenden Sie Event-Selektoren, um anzugeben, dass Ihr Trail Datenereignisse für Ihren Quell-Bucket protokollieren und die Ereignisse an die EventBridge Regel senden soll.

   Warum nehme ich diese Änderung vor? Dieser Befehl filtert Ereignisse.

   Beispiel:

   Im folgenden Beispielbefehl werden `--trail-name` und `--event-selectors` verwendet, um die Verwaltung von Datenereignissen für einen Quell-Bucket und einen Präfix namens `myBucket/myFolder` anzugeben.

   ```
   aws cloudtrail put-event-selectors --trail-name my-trail --event-selectors
    '[{ "ReadWriteType": "WriteOnly", "IncludeManagementEvents":false,
    "DataResources": [{ "Type": "AWS::S3::Object", "Values": ["arn:aws:s3:::myBucket/
   myFolder/file.zip"] }] }]'
   ```

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Erteilen Sie Berechtigungen EventBridge , die CodePipeline zum Aufrufen der Regel verwendet werden können. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie zu erstellen, damit EventBridge Sie die Servicerolle übernehmen können. Geben Sie ihr den Namen `trustpolicyforEB.json`.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   ```

```
}
```

b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen dieser Vertrauensrichtlinie zur Rolle werden Berechtigungen für erstellt EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie hier für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Führen Sie den folgenden Befehl aus, um der erstellten `Role-for-MyRule`-Rolle die neue `CodePipeline-Permissions-Policy-for-EB`-Berechtigungsrichtlinie anzufügen.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „`MyS3SourceRule`“ erstellt.

```
aws events put-rule --name "MyS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"AWS API Call via CloudTrail\"],\"detail\":
{\"eventSource\":[\"s3.amazonaws.com\"],\"eventName\":[\"CopyObject\",\"PutObject
\",\"CompleteMultipartUpload\"],\"requestParameters\":{\"bucketName\":[\"my-bucket
\"],\"key\":[\"my-key\"]}}}
 --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Um das Objekt CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die `--targets` Parameter `--rule` und an.

   Der folgende Befehl legt fest, dass für die Regel mit dem Namen `MyS3SourceRule` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dies Ziel 1 ist. Der Befehl gibt zudem ein Beispiel ARN für die Pipeline an. Die Pipeline startet, wenn Änderungen im Repository auftreten.

   ```
   aws events put-targets --rule MyS3SourceRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem Texteditor und bearbeiten Sie die Quellphase, indem Sie den Parameter `PollForSourceChanges` für einen Bucket mit dem Namen `storage-bucket`in `false` ändern wie in diesem Beispiel gezeigt.

   Warum nehme ich diese Änderung vor? Durch Festlegen dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

   ```
   "configuration": {
       "S3Bucket": "storage-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Wenn Sie mit einer Pipeline-Struktur arbeiten, die Sie mit dem Befehl get-pipeline abgerufen haben, müssen Sie die `metadata`-Zeilen aus der JSON-Datei entfernen. Andernfalls kann der update-pipeline-Befehl sie nicht nutzen. Entfernen Sie die `"metadata": { }`-Zeilen und die Felder `"created"`, `"pipelineARN"` und `"updated"`.

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

   ```
   "metadata": {
       "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
       "created": "date",
       "updated": "date"
   },
   ```

   Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

   > **Important**
   >
   > Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

   ```
   aws codepipeline update-pipeline --cli-input-json file://pipeline.json
   ```

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den start-pipeline-execution-Befehl, um Ihre Pipeline manuell zu starten.

Eine EventBridge Regel für eine Amazon S3 S3-Quelle erstellen (AWS CloudFormation Vorlage)

Um eine Regel AWS CloudFormation zu erstellen, aktualisieren Sie Ihre Vorlage wie hier gezeigt.

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` AWS CloudFormation Ressource`Resources`, um die IAM-Rolle zu konfigurieren, mit der Ihr Ereignis Ihre Pipeline starten kann. Dieser Eintrag erstellt eine Rolle mit zwei Richtlinien:

   - Die erste Richtlinie ermöglicht die Übernahme der Rolle.
   - Die zweite Richtlinie stellt Berechtigungen zum Starten der Pipeline bereit.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen einer `AWS::IAM::Role` Ressource können AWS CloudFormation Sie Berechtigungen für EventBridge erstellen. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

   ```
   EventRole:
     Type: AWS::IAM::Role
     Properties:
       AssumeRolePolicyDocument:
         Version: 2012-10-17
         Statement:
   ```

```
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
 'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]


...
```

JSON

```
  "EventRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": [
                "events.amazonaws.com"
              ]
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "Path": "/",
      "Policies": [
```

```
        {
          "PolicyName": "eb-pipeline-execution",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Action": "codepipeline:StartPipelineExecution",
                "Resource": {
                  "Fn::Join": [
                    "",
                    [
                      "arn:aws:codepipeline:",
                      {
                        "Ref": "AWS::Region"
                      },
                      ":",
                      {
                        "Ref": "AWS::AccountId"
                      },
                      ":",
                      {
                        "Ref": "AppPipeline"
                      }
                    ]
                  ]

...
```

2. Verwenden Sie die `AWS::Events::Rule` AWS CloudFormation Ressource, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis`CopyObject`, `PutObject` das Ihren `CompleteMultipartUpload` Amazon S3 S3-Quell-Bucket überwacht. Fügen Sie darüber hinaus ein Ziel für Ihre Pipeline ein. Wenn `CopyObject`, `PutObject` oder `CompleteMultipartUpload` auftritt, ruft diese Rolle `StartPipelineExecution` in Ihrer Ziel-Pipeline auf.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::Events::Rule` Ressource kann AWS CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

YAML

```
  EventRule:
    Type: AWS::Events::Rule
    Properties:
      EventPattern:
        source:
          - aws.s3
        detail-type:
          - 'AWS API Call via CloudTrail'
        detail:
          eventSource:
            - s3.amazonaws.com
          eventName:
            - CopyObject
            - PutObject
            - CompleteMultipartUpload
          requestParameters:
            bucketName:
              - !Ref SourceBucket
            key:
              - !Ref SourceObjectKey
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline


...
```

JSON

```
  "EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
      "EventPattern": {
        "source": [
          "aws.s3"
```

```
        ],
        "detail-type": [
          "AWS API Call via CloudTrail"
        ],
        "detail": {
          "eventSource": [
            "s3.amazonaws.com"
          ],
          "eventName": [
            "CopyObject",
            "PutObject",
            "CompleteMultipartUpload"
          ],
          "requestParameters": {
            "bucketName": [
              {
                "Ref": "SourceBucket"
              }
            ],
            "key": [
              {
                "Ref": "SourceObjectKey"
              }
            ]
          }
        }
      },
      "Targets": [
        {
          "Arn": {
            "Fn::Join": [
              "",
              [
                "arn:aws:codepipeline:",
                {
                  "Ref": "AWS::Region"
                },
                ":",
                {
                  "Ref": "AWS::AccountId"
                },
                ":",
                {
                  "Ref": "AppPipeline"
```

```
                }
              ]
            ]
          },
          "RoleArn": {
            "Fn::GetAtt": [
              "EventRole",
              "Arn"
            ]
          },
          "Id": "codepipeline-AppPipeline"
        }
      ]
    }
  }
},

...
```

3. Fügen Sie diesen Ausschnitt zu Ihrer ersten Vorlage hinzu, um Stack-übergreifende Funktionalität zu ermöglichen:

   YAML

```
Outputs:
  SourceBucketARN:
    Description: "S3 bucket ARN that Cloudtrail will use"
    Value: !GetAtt SourceBucket.Arn
    Export:
      Name: SourceBucketARN
```

   JSON

```
  "Outputs" : {
    "SourceBucketARN" : {
      "Description" : "S3 bucket ARN that Cloudtrail will use",
      "Value" : { "Fn::GetAtt": ["SourceBucket", "Arn"] },
      "Export" : {
        "Name" : "SourceBucketARN"
      }
    }
```

```
...
```

4. Speichern Sie Ihre aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation Konsole.
5. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
6. Laden Sie Ihre aktualisierte Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
7. Wählen Sie Execute (Ausführen).

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

  Warum nehme ich diese Änderung vor? Durch Ändern von `PollForSourceChanges` in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

  YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
```

```
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref SourceObjectKey
                PollForSourceChanges: false
              RunOrder: 1
```

JSON

```
{
   "Name": "SourceAction",
   "ActionTypeId": {
     "Category": "Source",
     "Owner": "AWS",
     "Version": 1,
     "Provider": "S3"
   },
   "OutputArtifacts": [
     {
       "Name": "SourceOutput"
     }
   ],
   "Configuration": {
     "S3Bucket": {
       "Ref": "SourceBucket"
     },
     "S3ObjectKey": {
       "Ref": "SourceObjectKey"
     },
     "PollForSourceChanges": false
   },
   "RunOrder": 1
 }
```

Um eine zweite Vorlage für die CloudTrail Ressourcen Ihrer Amazon S3 S3-Pipeline zu erstellen

- Verwenden Sie in einer separaten Vorlage unter`Resources`, die `AWS::CloudTrail::Trail` AWS CloudFormation Ressourcen `AWS::S3::BucketAWS::S3::BucketPolicy`, und, um eine einfache Bucket-Definition und einen Trail für bereitzustellen CloudTrail.

  Warum nehme ich diese Änderung vor? Angesichts des aktuellen Limits von fünf Trails pro Konto muss der CloudTrail Trail separat erstellt und verwaltet werden. (Siehe Grenzwerte unter AWS CloudTrail.) Sie können jedoch viele Amazon S3 S3-Buckets in einen einzigen Trail aufnehmen, sodass Sie den Trail einmal erstellen und dann bei Bedarf Amazon S3 S3-Buckets für andere Pipelines hinzufügen können. Fügen Sie den folgenden Code in Ihre zweite Beispielvorlagendatei ein.

  YAML

```
########################################################################################
# Prerequisites:
#   - S3 SourceBucket and SourceObjectKey must exist
########################################################################################

Parameters:
  SourceObjectKey:
    Description: 'S3 source artifact'
    Type: String
    Default: SampleApp_Linux.zip

Resources:
  AWSCloudTrailBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref AWSCloudTrailBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Sid: AWSCloudTrailAclCheck
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:GetBucketAcl
            Resource: !GetAtt AWSCloudTrailBucket.Arn
```

```
          -
            Sid: AWSCloudTrailWrite
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt AWSCloudTrailBucket.Arn, '/
AWSLogs/', !Ref 'AWS::AccountId', '/*' ] ]
            Condition:
              StringEquals:
                s3:x-amz-acl: bucket-owner-full-control
  AWSCloudTrailBucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
  AwsCloudTrail:
    DependsOn:
      - AWSCloudTrailBucketPolicy
    Type: AWS::CloudTrail::Trail
    Properties:
      S3BucketName: !Ref AWSCloudTrailBucket
      EventSelectors:
        -
          DataResources:
            -
              Type: AWS::S3::Object
              Values:
                - !Join [ '', [ !ImportValue SourceBucketARN, '/', !Ref
 SourceObjectKey ] ]
          ReadWriteType: WriteOnly
          IncludeManagementEvents: false
      IncludeGlobalServiceEvents: true
      IsLogging: true
      IsMultiRegionTrail: true


...
```

JSON

```
{
  "Parameters": {
    "SourceObjectKey": {
```

```
      "Description": "S3 source artifact",
      "Type": "String",
      "Default": "SampleApp_Linux.zip"
    }
  },
  "Resources": {
    "AWSCloudTrailBucket": {
      "Type": "AWS::S3::Bucket",
        "DeletionPolicy": "Retain"
    },
    "AWSCloudTrailBucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {
          "Ref": "AWSCloudTrailBucket"
        },
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "AWSCloudTrailAclCheck",
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "cloudtrail.amazonaws.com"
                ]
              },
              "Action": "s3:GetBucketAcl",
              "Resource": {
                "Fn::GetAtt": [
                  "AWSCloudTrailBucket",
                  "Arn"
                ]
              }
            },
            {
              "Sid": "AWSCloudTrailWrite",
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "cloudtrail.amazonaws.com"
                ]
              },
              "Action": "s3:PutObject",
```

```
            "Resource": {
              "Fn::Join": [
                "",
                [
                  {
                    "Fn::GetAtt": [
                      "AWSCloudTrailBucket",
                      "Arn"
                    ]
                  },
                  "/AWSLogs/",
                  {
                    "Ref": "AWS::AccountId"
                  },
                  "/*"
                ]
              ]
            },
            "Condition": {
              "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control"
              }
            }
          }
        ]
      }
    }
  },
  "AwsCloudTrail": {
    "DependsOn": [
      "AWSCloudTrailBucketPolicy"
    ],
    "Type": "AWS::CloudTrail::Trail",
    "Properties": {
      "S3BucketName": {
        "Ref": "AWSCloudTrailBucket"
      },
      "EventSelectors": [
        {
          "DataResources": [
            {
              "Type": "AWS::S3::Object",
              "Values": [
                {
```

```
                    "Fn::Join": [
                      "",
                      [
                        {
                          "Fn::ImportValue": "SourceBucketARN"
                        },
                        "/",
                        {
                          "Ref": "SourceObjectKey"
                        }
                      ]
                    ]
                  }
                ]
              }
            ],
            "ReadWriteType": "WriteOnly",
            "IncludeManagementEvents": false
          }
        ],
        "IncludeGlobalServiceEvents": true,
        "IsLogging": true,
        "IsMultiRegionTrail": true
      }
    }
  }
}

...
```

Bitbucket Cloud-Verbindungen

Mithilfe von Verbindungen kannst du Konfigurationen autorisieren und einrichten, die deinen Drittanbieter mit deinen AWS Ressourcen verknüpfen. Um Ihr Drittanbieter-Repository als Quelle für Ihre Pipeline zuzuweisen, verwenden Sie eine Verbindung.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Afrika (Kapstadt), Naher Osten (Bahrain), Naher Osten (VAE), Europa (Spanien), Europa

(Zürich), Israel (Tel Aviv) oder AWS GovCloud (US-West) nicht verfügbar. Weitere verfügbare Aktionen finden Sie unter. Produkt- und Serviceintegrationen mit CodePipeline Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Um eine Bitbucket Cloud-Quellaktion hinzuzufügen CodePipeline, kannst du eine der folgenden Optionen wählen:

- Verwende den Assistenten „Pipeline erstellen“ der CodePipeline Konsole oder die Aktionsseite „Aktion bearbeiten“, um die Bitbucket-Provider-Option auszuwählen. Informationen Stelle eine Verbindung zu Bitbucket Cloud (Konsole) her zum Hinzufügen der Aktion findest du unter. Die Konsole hilft Ihnen beim Erstellen einer Verbindungsressource.

  > **Note**
  >
  > Sie können Verbindungen mit einem Bitbucket-Cloud-Repository erstellen. Installierte Bitbucket-Anbietertypen wie Bitbucket Server werden nicht unterstützt.

- Verwenden Sie die CLI, um die Aktionskonfiguration für die `CreateSourceConnection` Aktion mit dem `Bitbucket` Anbieter wie folgt hinzuzufügen:
  - Informationen zum Erstellen Ihrer Verbindungsressourcen finden Sie unter Verbindung zu Bitbucket Cloud (CLI) herstellen So erstellen Sie eine Verbindungsressource mit der CLI.
  - Verwenden Sie die `CreateSourceConnection` Beispiel-Aktionskonfiguration inCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen, um Ihre Aktion hinzuzufügen, wie unter gezeigtErstellen einer Pipeline (CLI).

> **Note**
>
> Sie können auch mithilfe der Developer Tools-Konsole unter Einstellungen eine Verbindung herstellen. Siehe Verbindung herstellen.

Bevor Sie beginnen:

- Du musst ein Konto beim Anbieter des Drittanbieter-Repositorys wie Bitbucket Cloud erstellt haben.
- Du musst bereits ein Code-Repository eines Drittanbieters erstellt haben, z. B. ein Bitbucket Cloud-Repository.

> **Note**
>
> Bitbucket Cloud-Verbindungen bieten nur Zugriff auf Repositorys, die dem Bitbucket Cloud-Konto gehören, mit dem die Verbindung hergestellt wurde.
> Wenn die Anwendung in einem Bitbucket Cloud-Workspace installiert wird, benötigst du die Workspace-Berechtigungen verwalten. Andernfalls wird die Option zum Installieren der App nicht angezeigt.

Themen

Stelle eine Verbindung zu Bitbucket Cloud (Konsole) her

Gehe wie folgt vor, um mit der CodePipeline Konsole eine Verbindungsaktion für dein Bitbucket-Repository hinzuzufügen.

> **Note**
>
> Sie können Verbindungen mit einem Bitbucket-Cloud-Repository erstellen. Installierte Bitbucket-Anbietertypen wie Bitbucket Server werden nicht unterstützt.

Schritt 1: Erstelle oder bearbeite deine Pipeline

Um Ihre Pipeline zu erstellen oder zu bearbeiten

1. Melden Sie sich bei der CodePipeline Konsole an.
2. Wählen Sie eine der folgenden Optionen aus.

- Wählen Sie, ob Sie eine Pipeline erstellen möchten. Folgen Sie den Schritten unter Pipeline erstellen, um den ersten Bildschirm abzuschließen, und wählen Sie Weiter. Wähle auf der Quellseite unter Quellanbieter die Option Bitbucket aus.
- Wähle, ob du eine bestehende Pipeline bearbeiten möchtest. Wählen Sie Bearbeiten und dann Phase bearbeiten aus. Wählen Sie, ob Sie Ihre Quellaktion hinzufügen oder bearbeiten möchten. Geben Sie auf der Seite Aktion bearbeiten unter Aktionsname den Namen für Ihre Aktion ein. Wähle unter Aktionsanbieter die Option Bitbucket aus.

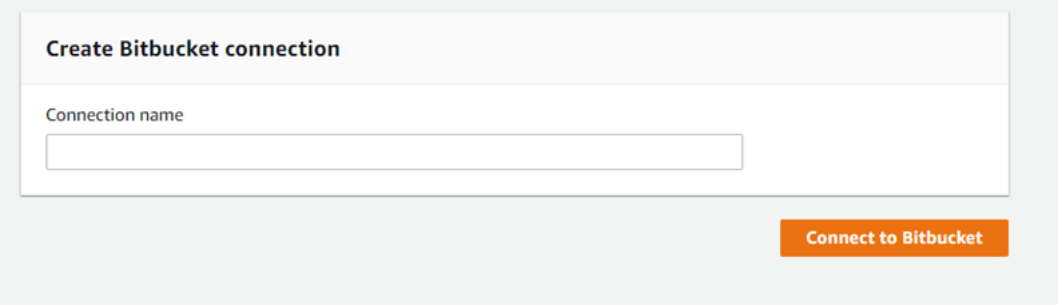
3. Führen Sie eine der folgenden Aktionen aus:

   - Wenn du noch keine Verbindung zu deinem Anbieter hergestellt hast, wähle Connect to Bitbucket aus. Fahren Sie mit Schritt 2 fort: Verbindung zu Bitbucket herstellen.
   - Wenn du unter Verbindung bereits eine Verbindung zu deinem Anbieter hergestellt hast, wähle die Verbindung aus. Fahren Sie mit Schritt 3 fort: Speichern Sie die Quellaktion für Ihre Verbindung.

Schritt 2: Stelle eine Verbindung zu Bitbucket Cloud her

Um eine Verbindung zu Bitbucket Cloud herzustellen

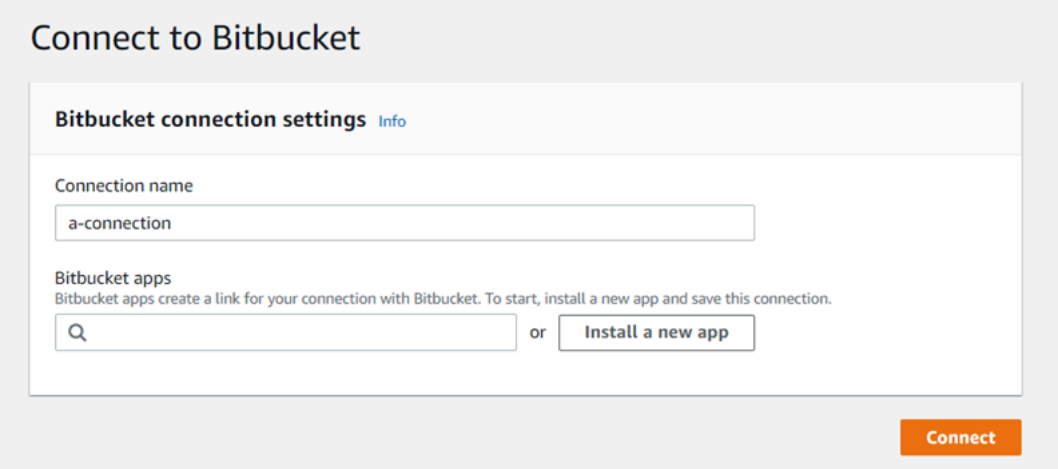
1. Gib auf der Einstellungsseite Connect to Bitbucket deinen Verbindungsnamen ein und wähle Connect to Bitbucket aus.



   Das Feld Bitbucket-Apps wird angezeigt.

2. Wählen Sie unter Bitbucket apps (Bitbucket-Apps) eine App-Installation aus oder wählen Sie Install a new app (Neue App installieren), um eine App zu erstellen.

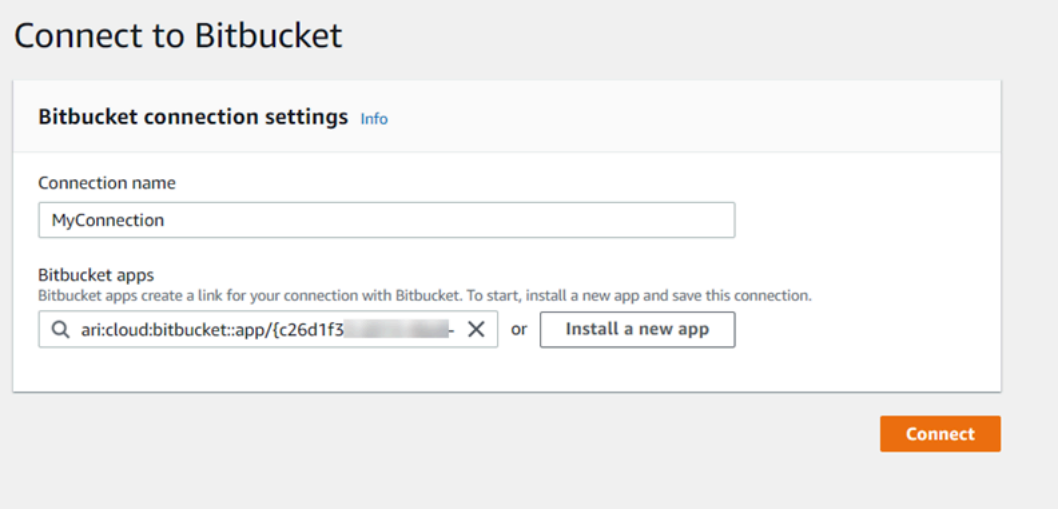
> **Note**
>
> Du installierst die App nur einmal für jeden Bitbucket Cloud-Workspace oder Account. Wenn du die Bitbucket-App bereits installiert hast, wähle sie aus und fahre mit Schritt 4 fort.



3. Wenn die Anmeldeseite für Bitbucket Cloud angezeigt wird, melde dich mit deinen Anmeldedaten an und wähle dann, ob du fortfahren möchtest.
4. Auf der App-Installationsseite wird eine Meldung angezeigt, dass die AWS CodeStar App versucht, eine Verbindung zu deinem Bitbucket-Konto herzustellen.

   Wenn Sie einen Bitbucket-Workspace verwenden, ändern Sie die Option Authorize for (Autorisieren für) auf den Workspace. Es werden nur Workspaces angezeigt, für die Sie über den Administratorzugriff verfügen.

   Wählen Sie Grant access (Zugriff gewähren).
5. In der Bitbucket-App wird die Verbindungs-ID für die neue Installation angezeigt. Wählen Sie Connect aus. Die erstellte Verbindung wird in der Verbindungsliste angezeigt.



Schritt 3: Speichere deine Bitbucket Cloud-Quellaktion

Verwende diese Schritte auf dem Assistenten oder der Aktionsseite „Aktion bearbeiten“, um deine Quellaktion mit deinen Verbindungsinformationen zu speichern.

Um Ihre Quellaktion mit Ihrer Verbindung abzuschließen und zu speichern

1. Wählen Sie unter Repository name (Repository-Name) den Namen Ihres Drittanbieter-Repositorys aus.
2. Unter Pipeline-Trigger können Sie Auslöser hinzufügen, wenn es sich bei Ihrer Aktion um eine CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern.
3. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

   - Um Ausgabeartefakte der Bitbucket Cloud-Aktion mit der Standardmethode zu speichern, wähle CodePipeline Standard. Die Aktion greift auf die Dateien aus dem Bitbucket Cloud-Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.
   - Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone (Vollständiger Klon). Diese Option kann nur von nachgelagerten Aktionen verwendet werden CodeBuild .

Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschriebenFügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, Enterprise Server oder .com GitHub hinzu GitHub GitLab.

4. Wählen Sie im Assistenten Weiter oder auf der Aktionsseite Bearbeiten die Option Speichern aus.

Verbindung zu Bitbucket Cloud (CLI) herstellen

Du kannst das AWS Command Line Interface (AWS CLI) verwenden, um eine Verbindung herzustellen.

> **Note**
>
> Sie können Verbindungen mit einem Bitbucket-Cloud-Repository erstellen. Installierte Bitbucket-Anbietertypen wie Bitbucket Server werden nicht unterstützt.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über AWS CLI oder AWS CloudFormation erstellt wurde, hat standardmäßig `PENDING` den Status. Nachdem Sie eine Verbindung mit der CLI hergestellt haben oder verwenden Sie die Konsole AWS CloudFormation, um die Verbindung so zu bearbeiten, dass sie ihren Status festlegt`AVAILABLE`.

So stellen Sie eine Verbindung her

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie den AWS CLI , um den create-connection Befehl auszuführen, und geben Sie dabei das `--provider-type` und `--connection-name` für Ihre Verbindung an. In diesem Beispiel lautet der Name des Drittanbieters `Bitbucket` und der angegebene Verbindungsname `MyConnection`.

```
aws codestar-connections create-connection --provider-type Bitbucket --connection-
name MyConnection
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er die ARN-Informationen der Verbindung ähnlich der folgenden zurück.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Verwenden Sie die Konsole, um die Verbindung fertigzustellen. Weitere Informationen finden Sie unter Aktualisieren einer ausstehenden Verbindung.
3. Die Pipeline erkennt standardmäßig Änderungen, wenn der Code an das Quell-Repository der Verbindung weitergeleitet wird. Gehen Sie wie folgt vor, um die Pipeline-Trigger-Konfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

   - Um die Pipeline-Trigger-Konfiguration so zu konfigurieren, dass sie nur mit einer manuellen Veröffentlichung beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

     ```
     "DetectChanges": "false",
     ```

   - Weitere Informationen zur Konfiguration der Pipeline-Trigger zum Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern. Im Folgenden werden beispielsweise Git-Tags zur Pipeline-Ebene der Pipeline-JSON-Definition hinzugefügt. In diesem Beispiel `release-v1` sind `release-v0` und die Git-Tags, die eingeschlossen werden sollen, und `release-v2` ist das Git-Tag, das ausgeschlossen werden soll.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
     ```

```
                            "excludes": [
                                "release-v2"
                            ]
                        }
                    }
                ]
            }
        }
    ]
```

CodeCommit Quellaktionen und EventBridge

Um eine CodeCommit Quellaktion hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Wählen Sie auf der CodePipeline Konsole den Assistenten „Pipeline erstellen“ (Erstellen einer Pipeline (Konsole)) oder die Aktionsseite „Bearbeiten“, um die CodeCommitAnbieteroption auszuwählen. Die Konsole erstellt eine EventBridge Regel, die Ihre Pipeline startet, wenn sich die Quelle ändert.
- Verwenden Sie die AWS CLI , um die Aktionskonfiguration für die `CodeCommit` Aktion hinzuzufügen und zusätzliche Ressourcen wie folgt zu erstellen:
  - Verwenden Sie die `CodeCommit` Beispiel-Aktionskonfiguration inCodeCommit, um Ihre Aktion zu erstellen, wie unter gezeigtErstellen einer Pipeline (CLI).
  - Die Methode zur Erkennung von Änderungen startet die Pipeline standardmäßig durch Abfragen der Quelle. Sie sollten die regelmäßigen Prüfungen deaktivieren und die Regel zur Erkennung von Änderungen manuell erstellen. Verwenden Sie eine der folgenden Methoden:Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (Konsole),Eine EventBridge Regel für eine CodeCommit Quelle erstellen (CLI), oderErstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (AWS CloudFormation Vorlage) .

Themen

- Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (Konsole)
- Eine EventBridge Regel für eine CodeCommit Quelle erstellen (CLI)
- Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (AWS CloudFormation Vorlage)

Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (Konsole)

> **Important**
>
> Wenn Sie die Konsole verwenden, um Ihre Pipeline zu erstellen oder zu bearbeiten, wird Ihre EventBridge Regel für Sie erstellt.

Um eine EventBridge Regel zur Verwendung in CodePipeline Vorgängen zu erstellen

1. Öffnen Sie die EventBridge Amazon-Konsole unter https://console.aws.amazon.com/events/.
2. Wählen Sie im Navigationsbereich Regeln aus. Lassen Sie den Standardbus ausgewählt oder wählen Sie einen Event-Bus. Wählen Sie Regel erstellen aus.
3. Geben Sie im Feld Name einen Namen für Ihre Regel ein.
4. Wählen Sie unter Regeltyp die Option Regel mit einem Ereignismuster aus. Wählen Sie Weiter aus.
5. Wählen Sie unter Ereignisquelle AWS Ereignisse oder EventBridge Partnerereignisse aus.
6. Wählen Sie unter Beispiel-Ereignistyp die Option AWS Ereignisse aus.
7. Geben CodeCommit Sie im Feld Beispielereignisse das Schlüsselwort ein, nach dem gefiltert werden soll. Wählen Sie CodeCommit Repository-Statusänderung aus.
8. Wählen Sie unter Erstellungsmethode die Option Kundenmuster (JSON-Editor) aus.

   Fügen Sie das unten angegebene Ereignismuster ein. Im Folgenden finden Sie ein Beispiel für ein CodeCommit Ereignismuster im Ereignisfenster für ein `MyTestRepo` Repository mit einem Branch namens`main`:

```
{
  "source": [
    "aws.codecommit"
  ],
  "detail-type": [
    "CodeCommit Repository State Change"
  ],
  "resources": [
    "arn:aws:codecommit:us-west-2:80398EXAMPLE:MyTestRepo"
  ],
  "detail": {
```

```
    "referenceType": [
      "branch"
    ],
    "referenceName": [
      "main"
    ]
  }
}
```

9. Wählen Sie unter Ziele die Option CodePipeline.
10. Geben Sie den Pipeline-ARN für die Pipeline ein, die mit dieser Regel gestartet werden soll.

> **Note**
>
> Sie finden den Pipeline-ARN in der Metadatenausgabe, nachdem Sie den Befehl get-pipeline ausgeführt haben. Der Pipeline-ARN wird in folgendem Format erstellt: *arn:aws:codepipeline: region: konto: Pipeline-Name*
> Pipeline-Beispiel-ARN:
> `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

11. So erstellen oder spezifizieren Sie eine IAM-Dienstrolle, die EventBridge Berechtigungen zum Aufrufen des mit Ihrer EventBridge Regel verknüpften Ziels gewährt (in diesem Fall lautet das Ziel): CodePipeline

    - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen aus, um eine Servicerolle zu erstellen, die Ihnen EventBridge Berechtigungen für den Start Ihrer Pipeline-Ausführung erteilt.
    - Wählen Sie Bestehende Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihnen EventBridge Berechtigungen für den Start Ihrer Pipeline-Ausführungen erteilt.

12. Wählen Sie Weiter aus.
13. Wählen Sie auf der Seite „Tags“ die Option Weiter aus.
14. Überprüfen Sie auf der Seite Überprüfen und erstellen die Regelkonfiguration. Wenn Sie mit der Regel zufrieden sind, klicken Sie auf Create rule (Regel erstellen).

Eine EventBridge Regel für eine CodeCommit Quelle erstellen (CLI)

Rufen Sie den Befehl put-rule mit folgenden Angaben auf:

- Einen Namen, der die von Ihnen erstellte Regel eindeutig bezeichnet. Dieser Name muss für alle Pipelines, die Sie erstellen und die mit Ihrem AWS Konto CodePipeline verknüpft sind, eindeutig sein.
- Das Ereignismuster für die Quelle und von der Regel verwendete Detailfelder. Weitere Informationen finden Sie unter Amazon EventBridge und Event Patterns.

Um eine EventBridge Regel mit CodeCommit als Ereignisquelle und CodePipeline als Ziel zu erstellen

1. Fügen Sie Berechtigungen hinzu EventBridge , die CodePipeline zum Aufrufen der Regel verwendet werden sollen. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie zu erstellen, die es ermöglicht, die Servicerolle EventBridge zu übernehmen. Geben Sie der Vertrauensrichtlinie den Namen `trustpolicyforEB.json`.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

   b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

   ```
   aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
    file://trustpolicyforEB.json
   ```

   c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie in diesem Beispiel für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Verwenden Sie den folgenden Befehl, um die Berechtigungsrichtlinie `CodePipeline-Permissions-Policy-for-EB` der Rolle `Role-for-MyRule` anzufügen.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen dieser Richtlinie zur Rolle werden Berechtigungen für erstellt EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

Warum nehme ich diese Änderung vor? Dieser Befehl aktiviert AWS CloudFormation , um das Ereignis zu erstellen.

Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „`MyCodeCommitRepoRule`“ erstellt.

```
aws events put-rule --name "MyCodeCommitRepoRule" --event-pattern "{\"source\":
[\"aws.codecommit\"],\"detail-type\":[\"CodeCommit Repository State Change\"],
\"resources\":[\"repository-ARN\"],\"detail\":{\"referenceType\":[\"branch\"],
\"referenceName\":[\"main\"]}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-
MyRule"
```

3. Um das Objekt CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die folgenden Parameter an:

- Der Parameter `--rule` wird für den `rule_name` verwendet, den Sie mit put-rule erstellt haben.
- Der Parameter `--targets` wird für die Listen-`Id` des Ziels in der Zielliste und den ARN der Ziel-Pipeline verwendet.

Der folgende Beispielbefehl legt fest, dass für die Regel mit dem Namen `MyCodeCommitRepoRule` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dieses Ziel 1 ist. Der Beispielbefehl gibt zudem ein Beispiel ARN für die Pipeline an. Die Pipeline startet, wenn Änderungen im Repository auftreten.

```
aws events put-targets --rule MyCodeCommitRepoRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem beliebigen Texteditor und bearbeiten Sie die Quellstufe, indem Sie den Parameter `PollForSourceChanges` in `false` ändern, wie in diesem Beispiel gezeigt.

Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

```
"configuration": {
    "PollForSourceChanges": "false",
    "BranchName": "main",
    "RepositoryName": "MyTestRepo"
},
```

3. Wenn Sie mit einer Pipeline-Struktur arbeiten, die Sie mit dem Befehl get-pipeline abgerufen haben, müssen Sie die `metadata`-Zeilen aus der JSON-Datei entfernen. Andernfalls kann der update-pipeline-Befehl sie nicht nutzen. Entfernen Sie die `"metadata": { }`-Zeilen und die Felder `"created"`, `"pipelineARN"` und `"updated"`.

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

   ```
   "metadata": {
       "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
       "created": "date",
       "updated": "date"
   },
   ```

   Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

   > **Important**
   >
   > Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

   ```
   aws codepipeline update-pipeline --cli-input-json file://pipeline.json
   ```

   Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den **`start-pipeline-execution`**-Befehl, um Ihre Pipeline manuell zu starten.

Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (AWS CloudFormation Vorlage)

Um eine Regel AWS CloudFormation zu erstellen, aktualisieren Sie Ihre Vorlage wie hier gezeigt.

Um Ihre AWS CloudFormation Pipeline-Vorlage zu aktualisieren und eine EventBridge Regel zu erstellen

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` AWS CloudFormation Ressource`Resources`, um die IAM-Rolle zu konfigurieren, mit der Ihre Veranstaltung Ihre Pipeline starten kann. Dieser Eintrag erstellt eine Rolle mit zwei Richtlinien:

   - Die erste Richtlinie ermöglicht die Übernahme der Rolle.
   - Die zweite Richtlinie stellt Berechtigungen zum Starten der Pipeline bereit.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::IAM::Role` Ressource können AWS CloudFormation Sie Berechtigungen für EventBridge erstellen. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

   ```
   EventRole:
     Type: AWS::IAM::Role
     Properties:
       AssumeRolePolicyDocument:
         Version: 2012-10-17
         Statement:
           -
   ```

```
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
 'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
```

JSON

```
"EventRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "events.amazonaws.com"
            ]
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "eb-pipeline-execution",
        "PolicyDocument": {
          "Version": "2012-10-17",
```

```
          "Statement": [
            {
              "Effect": "Allow",
              "Action": "codepipeline:StartPipelineExecution",
              "Resource": {
                "Fn::Join": [
                  "",
                  [
                    "arn:aws:codepipeline:",
                    {
                      "Ref": "AWS::Region"
                    },
                    ":",
                    {
                      "Ref": "AWS::AccountId"
                    },
                    ":",
                    {
                      "Ref": "AppPipeline"
                    }
                  ]

...
```

2. Verwenden Sie in der Vorlage unter die `AWS::Events::Rule` AWS CloudFormation Ressource`Resources`, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis, das Push-Änderungen an Ihrem Repository überwacht. Wenn eine Änderung des Repository-Status EventBridge erkannt wird, wird die Regel in `StartPipelineExecution` Ihrer Zielpipeline aufgerufen.

   Warum nehme ich diese Änderung vor? Durch Hinzufügen der `AWS::Events::Rule` Ressource kann AWS CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

```
  EventRule:
    Type: AWS::Events::Rule
    Properties:
      EventPattern:
        source:
          - aws.codecommit
```

```
        detail-type:
          - 'CodeCommit Repository State Change'
        resources:
          - !Join [ '', [ 'arn:aws:codecommit:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref RepositoryName ] ]
        detail:
          event:
            - referenceCreated
            - referenceUpdated
          referenceType:
            - branch
          referenceName:
            - main
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
```

JSON

```
"EventRule": {
  "Type": "AWS::Events::Rule",
  "Properties": {
    "EventPattern": {
      "source": [
        "aws.codecommit"
      ],
      "detail-type": [
        "CodeCommit Repository State Change"
      ],
      "resources": [
        {
          "Fn::Join": [
            "",
            [
              "arn:aws:codecommit:",
              {
                "Ref": "AWS::Region"
              },
```

```
              ":",
              {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "RepositoryName"
              }
            ]
          ]
        }
      ],
      "detail": {
        "event": [
          "referenceCreated",
          "referenceUpdated"
        ],
        "referenceType": [
          "branch"
        ],
        "referenceName": [
          "main"
        ]
      }
    },
    "Targets": [
      {
        "Arn": {
          "Fn::Join": [
            "",
            [
              "arn:aws:codepipeline:",
              {
                "Ref": "AWS::Region"
              },
              ":",
              {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "AppPipeline"
              }
            ]
```

```
          ]
        },
        "RoleArn": {
          "Fn::GetAtt": [
            "EventRole",
            "Arn"
          ]
        },
        "Id": "codepipeline-AppPipeline"
      }
    ]
  }
},
```

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation -Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> In vielen Fällen ist der Parameter `PollForSourceChanges` „true“, wenn Sie eine Pipeline erstellen. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

YAML

```
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: CodeCommit
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                BranchName: !Ref BranchName
                RepositoryName: !Ref RepositoryName
                PollForSourceChanges: false
              RunOrder: 1
```

JSON

```
{
  "Name": "Source",
  "Actions": [
    {
      "Name": "SourceAction",
      "ActionTypeId": {
        "Category": "Source",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "CodeCommit"
      },
      "OutputArtifacts": [
        {
          "Name": "SourceOutput"
        }
      ],
      "Configuration": {
```

```
        "BranchName": {
          "Ref": "BranchName"
        },
        "RepositoryName": {
          "Ref": "RepositoryName"
        },
        "PollForSourceChanges": false
      },
      "RunOrder": 1
    }
  ]
},
```

GitHub Verbindungen

Sie verwenden Verbindungen, um Konfigurationen zu autorisieren und einzurichten, die Ihren Drittanbieter mit Ihren AWS Ressourcen verknüpfen.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Afrika (Kapstadt), Naher Osten (Bahrain), Naher Osten (VAE), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv) oder AWS GovCloud (US-West) nicht verfügbar. Informationen zu anderen verfügbaren Aktionen finden Sie unter. Produkt- und Serviceintegrationen mit CodePipeline Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Um eine Quellaktion für Ihr GitHub oder Ihr GitHub Enterprise Cloud-Repository in hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Verwenden Sie den Assistenten zum Erstellen von Pipelines in der CodePipeline Konsole oder auf der Aktionsseite Bearbeiten, um die Anbieteroption GitHub (Version 2) auszuwählen. Informationen Stellen Sie eine Verbindung zu GitHub Enterprise Server (Konsole) her zum Hinzufügen der Aktion finden Sie unter. Die Konsole hilft Ihnen beim Erstellen einer Verbindungsressource.

> **Note**
>
> Ein Tutorial, das Ihnen zeigt, wie Sie eine GitHub Verbindung hinzufügen und die Option Vollständiges Klonen in Ihrer Pipeline verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

- Verwenden Sie die CLI, um die Aktionskonfiguration für die `CodeStarSourceConnection` Aktion mit dem `GitHub` Anbieter hinzuzufügen. Verwenden Sie dazu die unter aufgeführten CLI-SchritteErstellen einer Pipeline (CLI).

> **Note**
>
> Sie können auch mithilfe der Developer Tools-Konsole unter Einstellungen eine Verbindung herstellen. Weitere Informationen finden Sie unter Verbindung erstellen.

Bevor Sie beginnen:

- Sie müssen ein Konto bei erstellt haben GitHub.
- Sie müssen bereits ein GitHub Code-Repository erstellt haben.
- Wenn Ihre CodePipeline Servicerolle vor dem 18. Dezember 2019 erstellt wurde, müssen Sie möglicherweise ihre Berechtigungen aktualisieren, um sie `codestar-connections:UseConnection` für AWS CodeStar Verbindungen verwenden zu können. Anweisungen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.

> **Note**
>
> Um die Verbindung herzustellen, müssen Sie der Eigentümer der GitHub Organisation sein. Bei Repositorys, die keiner Organisation angehören, müssen Sie der Repository-Besitzer sein.

Themen

Stellen Sie eine Verbindung zu GitHub (Konsole) her

Gehen Sie wie folgt vor, um mithilfe der CodePipeline Konsole eine Verbindungsaktion für Ihr GitHub oder GitHub Enterprise Cloud-Repository hinzuzufügen.

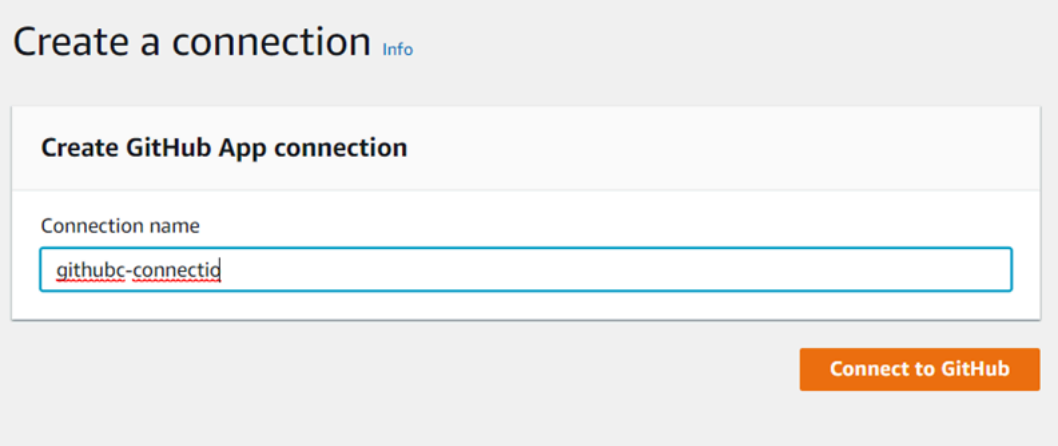
> **Note**
>
> In diesen Schritten können Sie unter Repository-Zugriff bestimmte Repositorys auswählen. Alle Repositorys, die nicht ausgewählt sind, sind für nicht zugänglich oder sichtbar. CodePipeline

Schritt 1: Erstellen oder bearbeiten Sie Ihre Pipeline

1. Melden Sie sich bei der CodePipeline Konsole an.
2. Wählen Sie eine der folgenden Optionen aus.
   - Wählen Sie, ob Sie eine Pipeline erstellen möchten. Folgen Sie den Schritten unter Pipeline erstellen, um den ersten Bildschirm zu vervollständigen, und wählen Sie Weiter. Wählen Sie auf der Quellseite unter Quellanbieter die Option GitHub (Version 2) aus.
   - Wählen Sie, ob Sie eine bestehende Pipeline bearbeiten möchten. Wählen Sie „Bearbeiten“ und dann „Phase bearbeiten“. Wählen Sie, ob Sie Ihre Quellaktion hinzufügen oder bearbeiten möchten. Geben Sie auf der Seite Aktion bearbeiten unter Aktionsname den Namen für Ihre Aktion ein. Wählen Sie unter Aktionsanbieter die Option GitHub (Version 2) aus.
3. Führen Sie eine der folgenden Aktionen aus:
   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Option Connect aus GitHub. Fahren Sie mit Schritt 2 fort: Verbindung herstellen zu GitHub.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie die Verbindung aus. Fahren Sie mit Schritt 3 fort: Speichern Sie die Quellaktion für Ihre Verbindung.

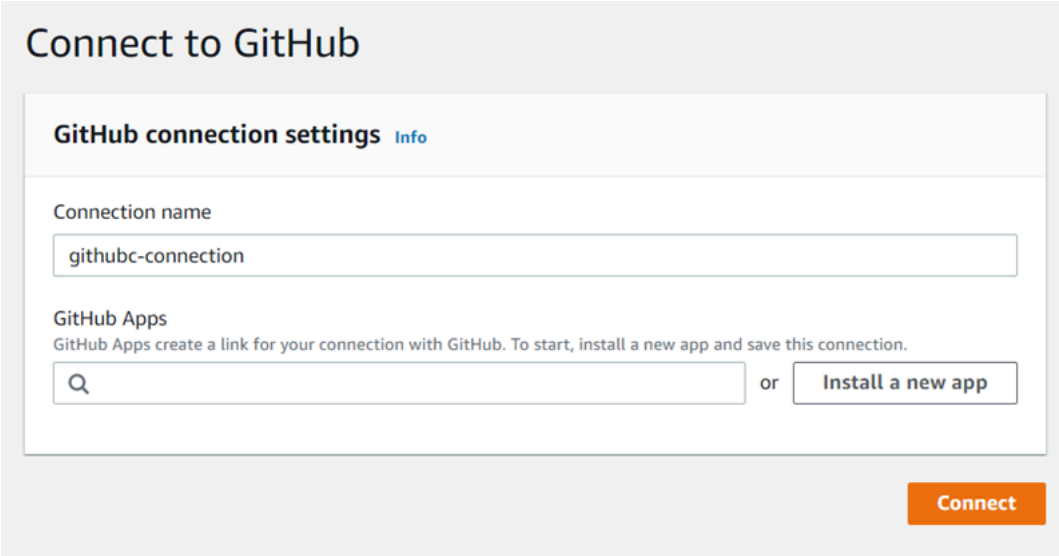
Schritt 2: Stellen Sie eine Verbindung her zu GitHub

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die GitHub Seite Connect angezeigt.



Um eine Verbindung herzustellen zu GitHub

1. Unter GitHub Verbindungseinstellungen wird Ihr Verbindungsname unter Verbindungsname angezeigt. Wählen Sie Connect GitHub. Die Seite für die Zugriffsanforderung wird angezeigt.
2. Wählen Sie AWS Connector autorisieren für GitHub. Auf der Verbindungsseite wird das Feld GitHub Apps angezeigt und angezeigt.



3. Wählen Sie unter GitHub Apps eine App-Installation aus oder wählen Sie Neue App installieren, um eine zu erstellen.

> **Note**
>
> Sie installieren eine App für alle Verbindungen mit einem bestimmten Anbieter. Wenn Sie den AWS Connector für GitHub App bereits installiert haben, wählen Sie ihn aus und überspringen Sie diesen Schritt.

4. Wählen Sie auf der GitHub Seite AWS Connector installieren für das Konto aus, in dem Sie die App installieren möchten.

   > **Note**
   >
   > Sie installieren die App nur einmal für jedes GitHub Konto. Wenn Sie die App schon einmal installiert haben, können Sie Configure (Konfiguration) wählen und mit einer Änderungsseite für die App-Installation fortfahren. Alternativ kommen Sie über die Schaltfläche „Back“ (Zurück) zur Konsole zurück.

5. Behalten Sie auf der GitHub Seite „ AWS Connector installieren für“ die Standardeinstellungen bei und wählen Sie Installieren aus.
6. Auf der GitHub Seite Connect wird die Verbindungs-ID für Ihre neue Installation unter GitHub Apps angezeigt. Wählen Sie Connect aus.

Schritt 3: Speichern Sie Ihre GitHub Quellaktion

Gehen Sie wie folgt auf der Seite Aktion bearbeiten vor, um Ihre Quellaktion mit Ihren Verbindungsinformationen zu speichern.

Um Ihre GitHub Quellaktion zu speichern

1. Wählen Sie unter Repository name (Repository-Name) den Namen Ihres Drittanbieter-Repositorys aus.
2. Unter Pipeline-Trigger können Sie Auslöser hinzufügen, wenn es sich bei Ihrer Aktion um eine CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern.
3. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

   - Um die Ausgabeartefakte der GitHub Aktion mit der Standardmethode zu speichern, wählen Sie CodePipeline Standard. Die Aktion greift auf die Dateien aus dem GitHub Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.
   - Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone

(Vollständiger Klon). Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschriebenFügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, Enterprise Server oder .com GitHub hinzu GitHub GitLab. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiges Klonen verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

4. Wählen Sie im Assistenten Weiter oder auf der Aktionsseite Bearbeiten die Option Speichern aus.

Verbindung herstellen zu GitHub (CLI)

Sie können das AWS Command Line Interface (AWS CLI) verwenden, um eine Verbindung herzustellen.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über AWS CLI oder AWS CloudFormation erstellt wurde, hat standardmäßig `PENDING` den Status. Nachdem Sie eine Verbindung mit der CLI hergestellt haben oder verwenden Sie die Konsole AWS CloudFormation, um die Verbindung so zu bearbeiten, dass sie ihren Status festlegt`AVAILABLE`.

So stellen Sie eine Verbindung her

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie den AWS CLI , um den create-connection Befehl auszuführen, und geben Sie dabei das `--provider-type` und `--connection-name` für Ihre Verbindung an. In diesem Beispiel lautet der Name des Drittanbieters `GitHub` und der angegebene Verbindungsname `MyConnection`.

```
aws codestar-connections create-connection --provider-type GitHub --connection-name
 MyConnection
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er die ARN-Informationen der Verbindung ähnlich der folgenden zurück.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Verwenden Sie die Konsole, um die Verbindung fertigzustellen. Weitere Informationen finden Sie unter Aktualisieren einer ausstehenden Verbindung.
3. Die Pipeline erkennt standardmäßig Änderungen, wenn der Code an das Quell-Repository der Verbindung weitergeleitet wird. Gehen Sie wie folgt vor, um die Pipeline-Trigger-Konfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

   - Um die Pipeline-Trigger-Konfiguration so zu konfigurieren, dass sie nur mit einer manuellen Veröffentlichung beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

     ```
     "DetectChanges": "false",
     ```

   - Weitere Informationen zur Konfiguration der Pipeline-Trigger zum Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern. Im Folgenden wird beispielsweise die Pipeline-Ebene der Pipeline-JSON-Definition erweitert. In diesem Beispiel `release-v1` sind `release-v0` und die Git-Tags, die eingeschlossen werden sollen, und `release-v2` ist das Git-Tag, das ausgeschlossen werden soll.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
     ```

```
                    }
                ]
            }
        }
    ]
```

GitHub Enterprise Server-Verbindungen

Mithilfe von Verbindungen können Sie Konfigurationen autorisieren und einrichten, die Ihren Drittanbieter mit Ihren AWS Ressourcen verknüpfen. Um Ihr Drittanbieter-Repository als Quelle für Ihre Pipeline zuzuweisen, verwenden Sie eine Verbindung.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Afrika (Kapstadt), Naher Osten (Bahrain), Naher Osten (VAE), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv) oder AWS GovCloud (US-West) nicht verfügbar. Informationen zu anderen verfügbaren Aktionen finden Sie unter. Produkt- und Serviceintegrationen mit CodePipeline Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Um eine GitHub Enterprise Server-Quellaktion hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Verwenden Sie in der CodePipeline Konsole den Assistenten zum Erstellen von Pipelines oder die Aktionsseite Bearbeiten, um die Option GitHub Enterprise Server-Provider auszuwählen. Informationen Stellen Sie eine Verbindung zu GitHub Enterprise Server (Konsole) her zum Hinzufügen der Aktion finden Sie unter. Die Konsole hilft Ihnen beim Erstellen einer Hostressource und einer Verbindungsressource.
- Verwenden Sie die CLI, um die Aktionskonfiguration für die `CreateSourceConnection` Aktion mit dem `GitHubEnterpriseServer` Anbieter hinzuzufügen und Ihre Ressourcen zu erstellen:
  - Informationen zum Erstellen Ihrer Verbindungsressourcen finden Sie unter Erstellen Sie einen Host und eine Verbindung zu GitHub Enterprise Server (CLI) So erstellen Sie eine Hostressource und eine Verbindungsressource mit der CLI.

- Verwenden Sie die `CreateSourceConnection` Beispiel-Aktionskonfiguration in[CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen](), um Ihre Aktion hinzuzufügen, wie unter gezeigt[Erstellen einer Pipeline (CLI)]().

> **Note**
>
> Sie können auch mithilfe der Developer Tools-Konsole unter Einstellungen eine Verbindung herstellen. Siehe [Verbindung herstellen]().

Bevor Sie beginnen:

- Sie müssen ein Konto bei GitHub Enterprise Server erstellt und die GitHub Enterprise Server-Instanz in Ihrer Infrastruktur installiert haben.

  > **Note**
  >
  > Jede VPC kann jeweils nur einem Host (GitHub Enterprise Server-Instanz) zugeordnet werden.

- Sie müssen bereits ein Code-Repository mit GitHub Enterprise Server erstellt haben.

Themen

Stellen Sie eine Verbindung zu GitHub Enterprise Server (Konsole) her

Gehen Sie wie folgt vor, um mithilfe der CodePipeline Konsole eine Verbindungsaktion für Ihr GitHub Enterprise Server-Repository hinzuzufügen.

> **Note**
>
> GitHub Enterprise Server-Verbindungen bieten nur Zugriff auf Repositorys, die dem GitHub Enterprise Server-Konto gehören, mit dem die Verbindung hergestellt wurde.

Bevor Sie beginnen:

Für eine Hostverbindung zu GitHub Enterprise Server müssen Sie die Schritte zum Erstellen einer Hostressource für Ihre Verbindung abgeschlossen haben. Siehe Hosts für Verbindungen verwalten.

Schritt 1: Erstellen oder bearbeiten Sie Ihre Pipeline

Um Ihre Pipeline zu erstellen oder zu bearbeiten

1. Melden Sie sich bei der CodePipeline Konsole an.
2. Wählen Sie eine der folgenden Optionen aus.
   - Wählen Sie, ob Sie eine Pipeline erstellen möchten. Folgen Sie den Schritten unter Pipeline erstellen, um den ersten Bildschirm abzuschließen, und wählen Sie Weiter. Wählen Sie auf der Seite Quelle unter Quellanbieter die Option GitHub Enterprise Server aus.
   - Wählen Sie, ob Sie eine bestehende Pipeline bearbeiten möchten. Wählen Sie Bearbeiten und dann Phase bearbeiten aus. Wählen Sie, ob Sie Ihre Quellaktion hinzufügen oder bearbeiten möchten. Geben Sie auf der Seite Aktion bearbeiten unter Aktionsname den Namen für Ihre Aktion ein. Wählen Sie unter Aktionsanbieter die Option GitHub Enterprise Server aus.
3. Führen Sie eine der folgenden Aktionen aus:
   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Option Connect to GitHub Enterprise Server aus. Fahren Sie mit Schritt 2 fort: Verbindung zum GitHub Enterprise Server herstellen.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie die Verbindung aus. Fahren Sie mit Schritt 3 fort: Speichern Sie die Quellaktion für Ihre Verbindung.

Stellen Sie eine Verbindung zu GitHub Enterprise Server her

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die Seite Connect to GitHub Enterprise Server angezeigt.

> **Important**
>
> AWS CodeConnections unterstützt GitHub Enterprise Server Version 2.22.0 aufgrund eines bekannten Problems in der Version nicht. Um eine Verbindung zu erstellen, aktualisieren Sie auf Version 2.22.1 bzw. die neueste verfügbare Version.

Um eine Verbindung zu Enterprise Server herzustellen GitHub

1. Geben Sie unter Connection Name (Verbindungsname) den Namen für die Verbindung ein.
2. Geben Sie unter URL den Endpunkt für Ihren Server ein.

> **Note**
>
> Wenn die angegebene URL bereits verwendet wurde, um einen GitHub Enterprise Server für eine Verbindung einzurichten, werden Sie aufgefordert, den Host-Ressourcen-ARN auszuwählen, der zuvor für diesen Endpunkt erstellt wurde.

3. Wenn Sie Ihren Server in einer Amazon VPC gestartet haben und eine Verbindung mit Ihrer VPC erstellen möchten, wählen Sie Use a VPC (Verwenden einer VPC) aus und geben Sie Folgendes ein.

   a. Wählen Sie unter VPC ID Ihre VPC-ID aus. Stellen Sie sicher, dass Sie die VPC für die Infrastruktur wählen, in der Ihre GitHub Enterprise Server-Instance installiert ist, oder eine VPC mit Zugriff auf Ihre GitHub Enterprise Server-Instance über VPN oder Direct Connect.

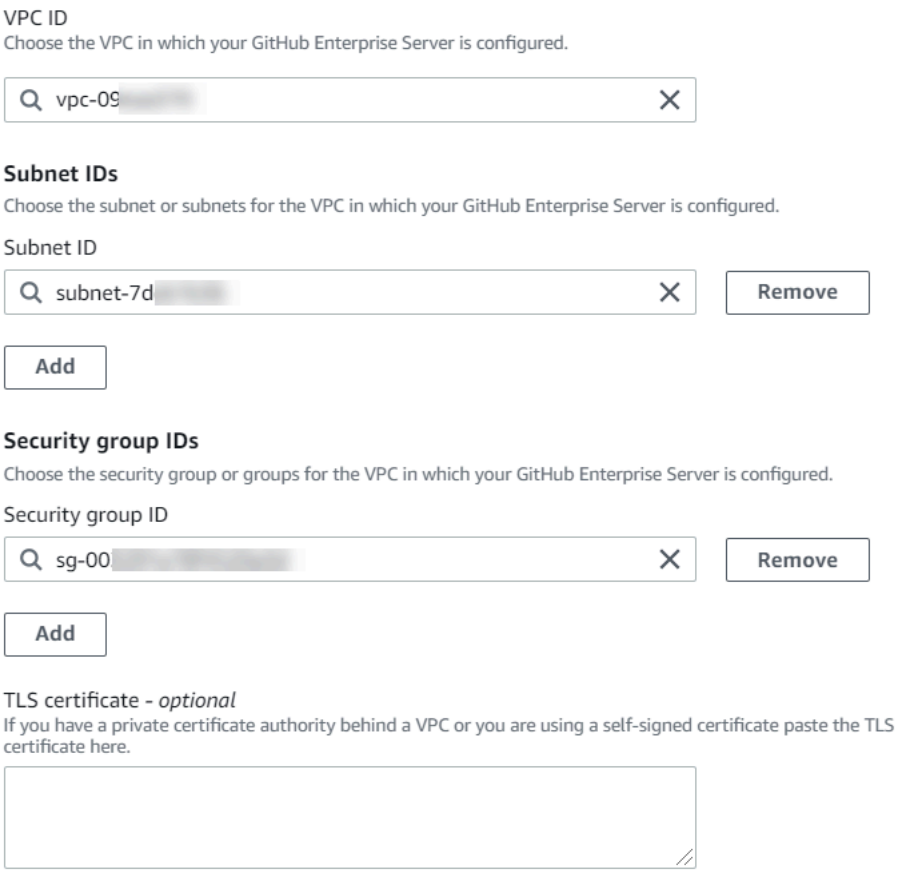
   b. Wählen Sie unter Subnetz-ID (Subnetz-ID) die Option Add (Hinzufügen) aus. Wählen Sie im Feld die Subnetz-ID aus, die Sie für Ihren Host verwenden möchten. Sie können bis zu 10 Subnetze wählen.

      Stellen Sie sicher, dass Sie das Subnetz für die Infrastruktur auswählen, in der Ihre GitHub Enterprise Server-Instanz installiert ist, oder ein Subnetz mit Zugriff auf Ihre installierte GitHub Enterprise Server-Instanz über VPN oder Direct Connect.

   c. Wählen Sie unter Security group IDs (Sicherheitsgruppen-IDs) die Option Add (Hinzufügen) aus. Wählen Sie im Feld die Sicherheitsgruppe aus, die Sie für Ihren Host verwenden möchten. Sie können bis zu 10 Sicherheitsgruppen auswählen.

      Stellen Sie sicher, dass Sie die Sicherheitsgruppe für die Infrastruktur auswählen, in der Ihre GitHub Enterprise Server-Instanz installiert ist, oder eine Sicherheitsgruppe mit Zugriff auf Ihre installierte GitHub Enterprise Server-Instanz über VPN oder Direct Connect.

   d. Wenn Sie eine private VPC konfiguriert haben und Ihre GitHub Enterprise Server-Instanz so konfiguriert haben, dass sie die TLS-Validierung mithilfe einer nicht öffentlichen Zertifizierungsstelle durchführt, geben Sie im Feld TLS-Zertifikat Ihre Zertifikat-ID ein. Der TLS-Zertifikatwert sollte der öffentliche Schlüssel des Zertifikats sein.



4. Wählen Sie Connect to GitHub Enterprise Server. Die erzeugte Verbindung wird mit dem Status Pending (Ausstehend) angezeigt. Für die Verbindung mit den von Ihnen angegebenen Serverinformationen wird eine Hostressource erstellt. Für den Hostnamen wird die URL verwendet.
5. Wählen Sie Update pending connection (Ausstehende aktualisieren) aus.
6. Wenn Sie dazu aufgefordert werden, melden Sie sich auf der GitHub Enterprise-Anmeldeseite mit Ihren GitHub Enterprise-Anmeldeinformationen an.
7. Wählen Sie auf der Seite „ GitHub App erstellen“ einen Namen für Ihre App aus.
8. <app-name>Wählen Sie auf der GitHub Autorisierungsseite die Option Autorisieren aus.
9. Auf der App-Installationsseite wird in einer Meldung angezeigt, dass die Connector-App zur Installation bereit ist. Wenn Sie mehrere Organisationen haben, werden Sie möglicherweise aufgefordert, die Organisation auszuwählen, in der Sie die App installieren möchten.

   Wählen Sie in die Repository-Einstellungen aus, wo Sie die App installieren möchten. Wählen Sie Installieren aus.
10. Die Verbindungsseite zeigt die erstellte Verbindung im Status Available (Verfügbar).

Schritt 3: Speichern Sie Ihre GitHub Enterprise Server-Quellaktion

Verwenden Sie diese Schritte auf dem Assistenten oder auf der Aktionsseite „Aktion bearbeiten“, um Ihre Quellaktion mit Ihren Verbindungsinformationen zu speichern.

Um Ihre Quellaktion mit Ihrer Verbindung abzuschließen und zu speichern

1. Wählen Sie unter Repository name (Repository-Name) den Namen Ihres Drittanbieter-Repositorys aus.
2. Unter Pipeline-Trigger können Sie Auslöser hinzufügen, wenn es sich bei Ihrer Aktion um eine CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern.
3. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

   - Um Ausgabeartefakte der GitHub Enterprise Server-Aktion mit der Standardmethode zu speichern, wählen Sie CodePipelineStandard. Die Aktion greift auf die Dateien aus dem GitHub Enterprise Server-Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.
   - Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone (Vollständiger Klon). Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.
4. Wählen Sie im Assistenten Weiter oder auf der Aktionsseite Bearbeiten die Option Speichern aus.

Erstellen Sie einen Host und eine Verbindung zu GitHub Enterprise Server (CLI)

Sie können das AWS Command Line Interface (AWS CLI) verwenden, um eine Verbindung herzustellen.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über AWS CLI oder AWS CloudFormation erstellt wurde, hat standardmäßig `PENDING` den Status. Nachdem Sie eine Verbindung mit der CLI hergestellt

haben oder verwenden Sie die Konsole AWS CloudFormation, um die Verbindung so zu bearbeiten, dass sie ihren Status festlegt`AVAILABLE`.

Sie können das AWS Command Line Interface (AWS CLI) verwenden, um einen Host für installierte Verbindungen zu erstellen.

> **Note**
>
> Sie erstellen einen Host nur einmal pro GitHub Enterprise Server-Konto. Alle Ihre Verbindungen zu einem bestimmten GitHub Enterprise Server-Konto verwenden denselben Host.

Sie verwenden Sie einen Host, der den Endpunkt für die Infrastruktur darstellt, in der der Drittanbieter installiert ist. Nachdem Sie die Hosterstellung mit der CLI abgeschlossen haben, befindet sich der Host im Status Ausstehend. Anschließend richten Sie den Host ein oder registrieren ihn, um ihn in den Status Verfügbar zu versetzen. Sobald der Host verfügbar ist, führen Sie die Schritte zum Erstellen einer Verbindung aus.

Verwenden Sie dazu den Befehl create-host.

> **Important**
>
> Ein über den erstellter Host AWS CLI befindet sich standardmäßig im `Pending` Status. Nachdem Sie einen Host mit der CLI erstellt haben, verwenden Sie die Konsole oder die CLI, um den Host so einzurichten, dass er seinen Status annimmt`Available`.

Einen Host erstellen

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie den, AWS CLI um den create-host Befehl auszuführen, und geben Sie dabei `--name--provider-type`, und `--provider-endpoint` für Ihre Verbindung an. In diesem Beispiel lautet der Name des Drittanbieters `GitHubEnterpriseServer` und der Endpunkt `my-instance.dev`.

```
aws codestar-connections create-host --name MyHost --provider-type
 GitHubEnterpriseServer --provider-endpoint "https://my-instance.dev"
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er die Amazon-Ressourcenname (ARN)-Informationen zum Host ähnlich der folgenden zurück.

```
{
    "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
Host-28aef605"
}
```

Nach diesem Schritt befindet sich der Host im Status `PENDING` (Ausstehend).

2. Schließen Sie das Host-Setup mit der Konsole ab und ändern Sie den Host-Status zu `Available` (Verfügbar).

Um eine Verbindung zu GitHub Enterprise Server herzustellen

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie den AWS CLI , um den create-connection Befehl auszuführen, und geben Sie dabei `--host-arn` und `--connection-name` für Ihre Verbindung an.

```
aws codestar-connections create-connection --host-arn arn:aws:codestar-
connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
 MyConnection
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er die ARN-Informationen der Verbindung ähnlich der folgenden zurück.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad"
}
```

2. Verwenden Sie die Konsole, um die ausstehende Verbindung einzurichten.
3. Die Pipeline erkennt standardmäßig Änderungen, wenn der Code an das Quell-Repository der Verbindung weitergeleitet wird. Gehen Sie wie folgt vor, um die Pipeline-Trigger-Konfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

- Um die Pipeline-Trigger-Konfiguration so zu konfigurieren, dass sie nur mit einer manuellen Veröffentlichung beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

```
"DetectChanges": "false",
```

- Weitere Informationen zur Konfiguration der Pipeline-Trigger zum Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern. Im Folgenden wird beispielsweise die Pipeline-Ebene der Pipeline-JSON-Definition erweitert. In diesem Beispiel `release-v1` sind `release-v0` und die Git-Tags, die eingeschlossen werden sollen, und `release-v2` ist das Git-Tag, das ausgeschlossen werden soll.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-v0", "release-v1"
                                ],
                                "excludes": [
                                    "release-v2"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
```

GitLab.com-Verbindungen

Mithilfe von Verbindungen können Sie Konfigurationen autorisieren und einrichten, die Ihren Drittanbieter mit Ihren AWS Ressourcen verknüpfen. Um Ihr Drittanbieter-Repository als Quelle für Ihre Pipeline zuzuweisen, verwenden Sie eine Verbindung.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Afrika (Kapstadt), Naher Osten (Bahrain), Naher Osten (VAE), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv) oder AWS GovCloud (US-West) nicht verfügbar. Hinweise zu anderen verfügbaren Aktionen finden Sie unter. Produkt- und Serviceintegrationen mit CodePipeline Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Um GitLab eine.com-Quellaktion hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Verwenden Sie den Assistenten „Pipeline erstellen“ oder die Aktionsseite „Bearbeiten“ der CodePipeline Konsole, um die GitLabAnbieteroption auszuwählen. Informationen Stelle eine Verbindung zu GitLab .com (Konsole) her zum Hinzufügen der Aktion finden Sie unter. Die Konsole hilft Ihnen beim Erstellen einer Verbindungsressource.
- Verwenden Sie die CLI, um die Aktionskonfiguration für die `CreateSourceConnection` Aktion mit dem `GitLab` Anbieter wie folgt hinzuzufügen:
  - Informationen zum Erstellen Ihrer Verbindungsressourcen finden Sie unter Verbindung zu GitLab .com herstellen (CLI) So erstellen Sie eine Verbindungsressource mit der CLI.
  - Verwenden Sie die `CreateSourceConnection` Beispiel-Aktionskonfiguration inCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen, um Ihre Aktion hinzuzufügen, wie unter gezeigtErstellen einer Pipeline (CLI).

> **Note**
>
> Sie können auch mithilfe der Developer Tools-Konsole unter Einstellungen eine Verbindung herstellen. Weitere Informationen finden Sie unter Verbindung erstellen.

> **Note**
>
> Indem Sie diese Verbindungsinstallation auf GitLab .com autorisieren, gewähren Sie unserem Service die Erlaubnis, Ihre Daten zu verarbeiten, indem Sie auf Ihr Konto zugreifen. Sie können die Berechtigungen jederzeit widerrufen, indem Sie die Anwendung deinstallieren.

Bevor Sie beginnen:

- Sie müssen bereits ein Konto bei GitLab .com erstellt haben.

  > **Note**
  >
  > Verbindungen bieten nur Zugriff auf Repositorys, die dem Konto gehören, das zum Erstellen und Autorisieren der Verbindung verwendet wurde.

  > **Note**
  >
  > Sie können Verbindungen zu einem Repository herstellen GitLab, in dem Sie die Rolle des Besitzers haben, und dann kann die Verbindung mit dem Repository mit Ressourcen wie verwendet werden CodePipeline. Bei Repositorys in Gruppen müssen Sie nicht der Gruppenbesitzer sein.

- Um eine Quelle für deine Pipeline anzugeben, musst du bereits ein Repository auf gitlab.com erstellt haben.

Themen

Stelle eine Verbindung zu GitLab .com (Konsole) her

Gehen Sie wie folgt vor, um mithilfe der CodePipeline Konsole eine Verbindungsaktion für Ihr Projekt (Repository) in hinzuzufügen GitLab.

Um Ihre Pipeline zu erstellen oder zu bearbeiten

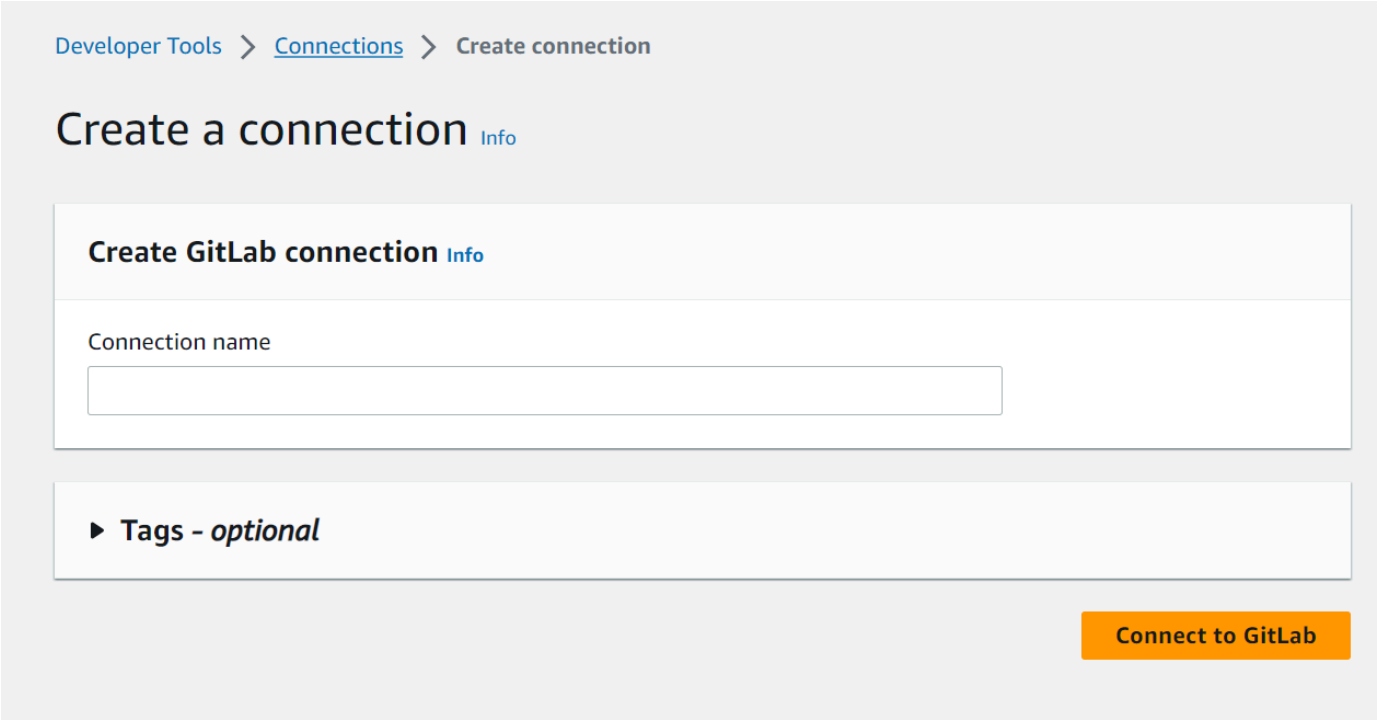
1. Melden Sie sich bei der CodePipeline Konsole an.
2. Wählen Sie eine der folgenden Optionen aus.

   - Wählen Sie, ob Sie eine Pipeline erstellen möchten. Folgen Sie den Schritten unter Pipeline erstellen, um den ersten Bildschirm abzuschließen, und wählen Sie Weiter. Wählen Sie auf der Seite Quelle unter Quellanbieter die Option GitLab.
   - Wählen Sie, ob Sie eine bestehende Pipeline bearbeiten möchten. Wählen Sie Bearbeiten und dann Phase bearbeiten aus. Wählen Sie, ob Sie Ihre Quellaktion hinzufügen oder bearbeiten möchten. Geben Sie auf der Seite Aktion bearbeiten unter Aktionsname den Namen für Ihre Aktion ein. Wählen Sie unter Aktionsanbieter die Option aus GitLab.
3. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Option Connect aus GitLab. Fahren Sie mit Schritt 4 fort, um die Verbindung herzustellen.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie die Verbindung aus. Fahren Sie mit Schritt 9 fort.

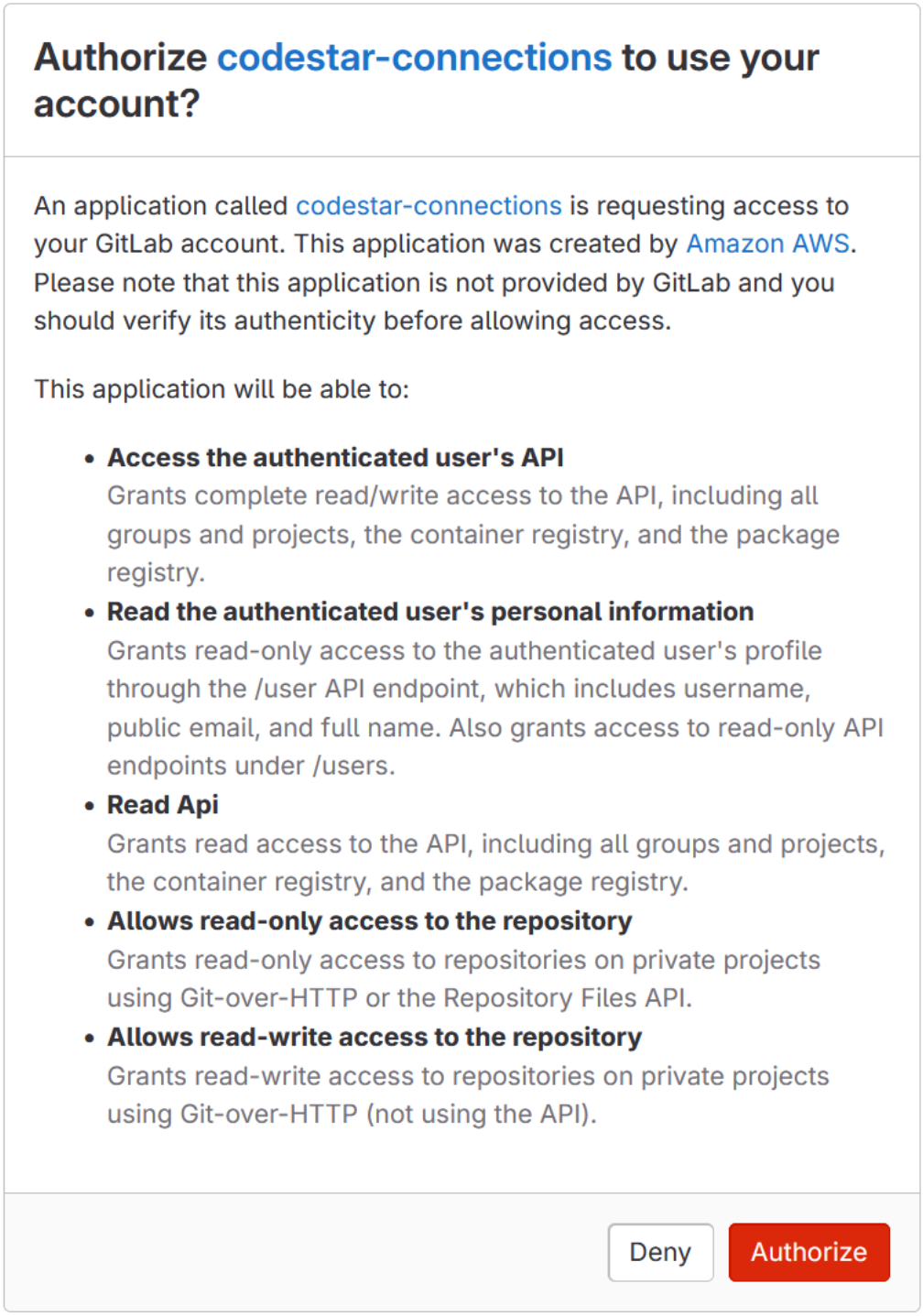
   > **Note**
   >
   > Wenn Sie das Popup-Fenster schließen, bevor eine GitLab .com-Verbindung hergestellt wurde, müssen Sie die Seite aktualisieren.

4. Um eine Verbindung zu einem GitLab .com-Repository herzustellen, wählen Sie unter Anbieter auswählen die Option GitLab. Geben Sie unter Connection name (Verbindungsname) den Namen für die Verbindung ein, die Sie erstellen möchten. Wählen Sie Connect GitLab.



5. Wenn die Anmeldeseite für GitLab .com angezeigt wird, melden Sie sich mit Ihren Anmeldeinformationen an und wählen Sie dann Anmelden aus.
6. Wenn Sie die Verbindung zum ersten Mal autorisieren, wird eine Autorisierungsseite mit einer Meldung angezeigt, in der Sie aufgefordert werden, die Verbindung für den Zugriff auf Ihr GitLab .com-Konto zu autorisieren.

   Klicken Sie auf Authorize.



7. Der Browser kehrt zur Seite der Verbindungskonsole zurück. Unter GitLab Verbindung erstellen wird die neue Verbindung unter Verbindungsname angezeigt.
8. Wählen Sie Connect GitLab.

   Sie kehren zur CodePipeline Konsole zurück.

> **Note**
>
> Nachdem eine GitLab .com-Verbindung erfolgreich hergestellt wurde, wird im Hauptfenster ein Erfolgsbanner angezeigt.
> Wenn Sie sich noch nicht GitLab auf dem aktuellen Computer angemeldet haben, müssen Sie das Popup-Fenster manuell schließen.

9. Wählen Sie unter Repository-Name den Namen Ihres Projekts aus, GitLab indem Sie den Projektpfad mit dem Namespace angeben. Geben Sie beispielsweise für ein Repository auf Gruppenebene den Repository-Namen im folgenden Format ein:. `group-name/repository-name` Weitere Informationen über den Pfad und den Namespace finden Sie in dem `path_with_namespace` Feld unter https://docs.gitlab.com/ee/api/projects.html #.get-single-project Weitere Informationen zum Namespace in finden Sie unter GitLab https://docs.gitlab.com/ee/user/namespace/.

   > **Note**
   >
   > Für Gruppen in GitLab müssen Sie den Projektpfad mit dem Namespace manuell angeben. Geben Sie beispielsweise für ein Repository, das `myrepo` in einer Gruppe benannt ist`mygroup`, Folgendes ein:`mygroup/myrepo`. Sie finden den Projektpfad mit dem Namespace in der URL unter. GitLab

10. Unter Pipeline-Trigger können Sie Auslöser hinzufügen, wenn es sich bei Ihrer Aktion um eine CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern.
11. Wählen Sie unter Branch name (Name der Verzweigung die Verzweigung aus, in der die Pipeline Quelländerungen erkennen soll.

    > **Note**
    >
    > Wenn der Branch-Name nicht automatisch aufgefüllt wird, haben Sie keinen Besitzerzugriff auf das Repository. Entweder ist der Projektname nicht gültig, oder die verwendete Verbindung hat keinen Zugriff auf das Projekt/Repository.

12. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

- Um Ausgabeartefakte der Aktion GitLab .com mit der Standardmethode zu speichern, wählen Sie CodePipeline Standard. Die Aktion greift auf die Dateien aus dem GitLab .com-Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.
- Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone (Vollständiger Klon). Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

  Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschriebenFügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, Enterprise Server oder .com GitHub hinzu GitHub GitLab. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiges Klonen verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

13. Wählen Sie, ob Sie die Quellaktion speichern und fortfahren möchten.

Verbindung zu GitLab .com herstellen (CLI)

Sie können das AWS Command Line Interface (AWS CLI) verwenden, um eine Verbindung herzustellen.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über AWS CLI oder AWS CloudFormation erstellt wurde, hat standardmäßig `PENDING` den Status. Nachdem Sie eine Verbindung mit der CLI hergestellt haben oder verwenden Sie die Konsole AWS CloudFormation, um die Verbindung so zu bearbeiten, dass sie ihren Status festlegt`AVAILABLE`.

So stellen Sie eine Verbindung her

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie den AWS CLI , um den create-connection Befehl auszuführen, und geben Sie dabei das `--provider-type` und `--connection-name` für Ihre Verbindung an. In diesem

Beispiel lautet der Name des Drittanbieters `GitLab` und der angegebene Verbindungsname `MyConnection`.

```
aws codestar-connections create-connection --provider-type GitLab --connection-name
 MyConnection
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er die ARN-Informationen der Verbindung ähnlich der folgenden zurück.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Verwenden Sie die Konsole, um die Verbindung fertigzustellen. Weitere Informationen finden Sie unter Aktualisieren einer ausstehenden Verbindung.
3. Die Pipeline erkennt standardmäßig Änderungen, wenn der Code an das Quell-Repository der Verbindung weitergeleitet wird. Gehen Sie wie folgt vor, um die Pipeline-Trigger-Konfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

   - Um die Pipeline-Trigger-Konfiguration so zu konfigurieren, dass sie nur mit einer manuellen Veröffentlichung beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

     ```
     "DetectChanges": "false",
     ```

   - Weitere Informationen zur Konfiguration der Pipeline-Trigger zum Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern. Im Folgenden wird beispielsweise die Pipeline-Ebene der Pipeline-JSON-Definition erweitert. In diesem Beispiel `release-v1` sind `release-v0` und die Git-Tags, die eingeschlossen werden sollen, und `release-v2` ist das Git-Tag, das ausgeschlossen werden soll.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
     ```

```
                                "release-v0", "release-v1"
                            ],
                            "excludes": [
                                "release-v2"
                            ]
                        }
                    }
                ]
            }
        }
    ]
```

Verbindungen für GitLab selbstverwaltete

Mithilfe von Verbindungen können Sie Konfigurationen autorisieren und einrichten, die Ihren Drittanbieter mit Ihren AWS Ressourcen verknüpfen. Um Ihr Drittanbieter-Repository als Quelle für Ihre Pipeline zuzuweisen, verwenden Sie eine Verbindung.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Afrika (Kapstadt), Naher Osten (Bahrain), Naher Osten (VAE), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv) oder AWS GovCloud (US-West) nicht verfügbar. Informationen zu anderen verfügbaren Aktionen finden Sie unter. Produkt- und Serviceintegrationen mit CodePipeline Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Um eine GitLab selbstverwaltete Quellaktion hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Verwenden Sie in der CodePipeline Konsole den Assistenten zum Erstellen von Pipelines oder die Aktionsseite Bearbeiten, um die Option für den GitLabselbstverwalteten Anbieter auszuwählen. Informationen Stellen Sie eine Verbindung zur GitLab selbstverwalteten (Konsole) her zum Hinzufügen der Aktion finden Sie unter. Die Konsole hilft Ihnen beim Erstellen einer Hostressource und einer Verbindungsressource.

- Verwenden Sie die CLI, um die Aktionskonfiguration für die `CreateSourceConnection` Aktion mit dem `GitLabSelfManaged` Anbieter hinzuzufügen und Ihre Ressourcen zu erstellen:
  - Informationen zum Erstellen Ihrer Verbindungsressourcen finden Sie unter Erstellen Sie einen Host und eine Verbindung zu GitLab Self-Managed (CLI) So erstellen Sie eine Hostressource und eine Verbindungsressource mit der CLI.
  - Verwenden Sie die `CreateSourceConnection` Beispiel-Aktionskonfiguration inCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen, um Ihre Aktion hinzuzufügen, wie unter gezeigtErstellen einer Pipeline (CLI).

> **Note**
>
> Sie können auch mithilfe der Developer Tools-Konsole unter Einstellungen eine Verbindung herstellen. Weitere Informationen finden Sie unter Verbindung erstellen.

Bevor Sie beginnen:

- Sie müssen bereits ein Konto bei GitLab Enterprise Edition oder GitLab Community Edition mit einer selbstverwalteten Installation erstellt haben GitLab und über diese verfügen. Weitere Informationen erhalten Sie unter https://docs.gitlab.com/ee/subscriptions/self_managed/.

  > **Note**
  >
  > Verbindungen bieten nur Zugriff für das Konto, das zum Erstellen und Autorisieren der Verbindung verwendet wurde.

  > **Note**
  >
  > Sie können Verbindungen zu einem Repository erstellen GitLab, in dem Sie die Rolle „Besitzer“ haben. Anschließend können Sie die Verbindung mit Ressourcen wie CodePipeline verwenden. Bei Repositorys in Gruppen müssen Sie nicht der Gruppenbesitzer sein.

- Sie müssen bereits ein GitLab persönliches Zugriffstoken (PAT) mit nur den folgenden eingeschränkten Berechtigungen erstellt haben: api. Weitere Informationen erhalten Sie unter

https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html. Sie müssen Administrator sein, um das PAT erstellen und verwenden zu können.

> **Note**
>
> Ihr PAT wird zur Autorisierung des Hosts verwendet und wird nicht anderweitig gespeichert oder von Verbindungen verwendet. Um einen Host einzurichten, können Sie ein temporäres PAT erstellen. Nachdem Sie den Host eingerichtet haben, können Sie das PAT löschen.

- Sie können sich dafür entscheiden, Ihren Host im Voraus einzurichten. Sie können einen Host mit oder ohne VPC einrichten. Einzelheiten zur VPC-Konfiguration und zusätzliche Informationen zum Erstellen eines Hosts finden Sie unter Host erstellen.

Themen

Stellen Sie eine Verbindung zur GitLab selbstverwalteten (Konsole) her

Gehen Sie wie folgt vor, um mithilfe der CodePipeline Konsole eine Verbindungsaktion für Ihr GitLab selbstverwaltetes Repository hinzuzufügen.

> **Note**
>
> GitLab Selbstverwaltete Verbindungen bieten nur Zugriff auf Repositorys, die dem GitLab selbstverwalteten Konto gehören, mit dem die Verbindung hergestellt wurde.

Bevor Sie beginnen:

Damit eine Hostverbindung zur GitLab selbstverwalteten Verbindung hergestellt werden kann, müssen Sie die Schritte zum Erstellen einer Hostressource für Ihre Verbindung abgeschlossen haben. Informationen zu Verbindungen finden Sie unter Hosts verwalten.

Schritt 1: Erstellen oder bearbeiten Sie Ihre Pipeline

Um Ihre Pipeline zu erstellen oder zu bearbeiten

1. Melden Sie sich bei der CodePipeline Konsole an.
2. Wählen Sie eine der folgenden Optionen aus.
   - Wählen Sie, ob Sie eine Pipeline erstellen möchten. Folgen Sie den Schritten unter Pipeline erstellen, um den ersten Bildschirm abzuschließen, und wählen Sie Weiter. Wählen Sie auf der Seite Quelle unter Quellanbieter die Option GitLab Selbstverwaltet aus.
   - Wählen Sie, ob Sie eine bestehende Pipeline bearbeiten möchten. Wählen Sie Bearbeiten und dann Phase bearbeiten aus. Wählen Sie, ob Sie Ihre Quellaktion hinzufügen oder bearbeiten möchten. Geben Sie auf der Seite Aktion bearbeiten unter Aktionsname den Namen für Ihre Aktion ein. Wählen Sie unter Aktionsanbieter die Option GitLab Selbstverwaltet aus.
3. Führen Sie eine der folgenden Aktionen aus:
   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Option Mit GitLab selbstverwaltetem Connect aus. Fahren Sie mit Schritt 2 fort: Verbindung zu GitLab selbstverwaltetem System herstellen.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie die Verbindung aus. Fahren Sie mit Schritt 3 fort: Speichern Sie die Quellaktion für Ihre Verbindung.

Stellen Sie eine Verbindung zu GitLab Self-Managed her

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die Seite Connect GitLab selbstverwalteter Verbindung herstellen angezeigt.

So stellen Sie eine Verbindung zur selbstverwalteten Website GitLab her

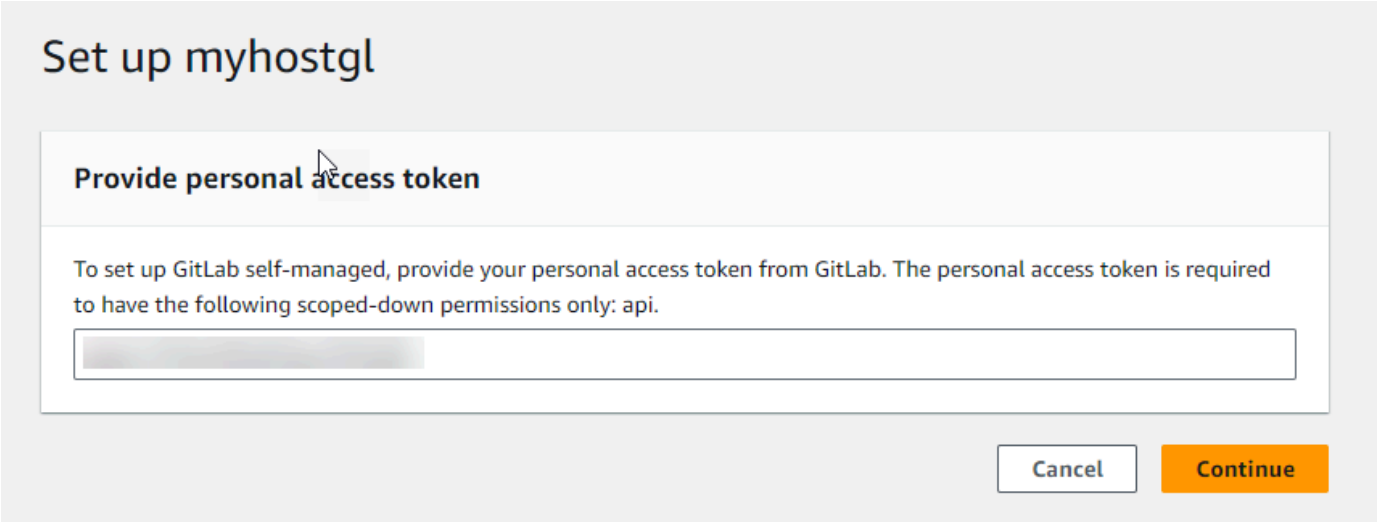
1. Geben Sie unter Connection Name (Verbindungsname) den Namen für die Verbindung ein.
2. Geben Sie unter URL den Endpunkt für Ihren Server ein.

> **Note**
>
> Wenn die angegebene URL bereits verwendet wurde, um einen Host für eine Verbindung einzurichten, werden Sie aufgefordert, den Host-Ressourcen-ARN auszuwählen, der zuvor für diesen Endpunkt erstellt wurde.

3. Wenn Sie Ihren Server in einer Amazon VPC gestartet haben und eine Verbindung zu Ihrer VPC herstellen möchten, wählen Sie Use a VPC aus und geben Sie die Informationen für die VPC ein.
4. Wählen Sie Connect to GitLab self-managed aus. Die erzeugte Verbindung wird mit dem Status Pending (Ausstehend) angezeigt. Für die Verbindung mit den von Ihnen angegebenen Serverinformationen wird eine Hostressource erstellt. Für den Hostnamen wird die URL verwendet.
5. Wählen Sie Update pending connection (Ausstehende aktualisieren) aus.
6. Wenn eine Seite mit einer Weiterleitungsnachricht geöffnet wird, in der bestätigt wird, dass Sie weiter zum Anbieter wechseln möchten, wählen Sie Weiter. Geben Sie die Autorisierung für den Anbieter ein.
7. Eine Seite ***`Hostname`*** einrichten wird angezeigt. Geben Sie unter Persönliches Zugriffstoken bereitstellen Ihrem GitLab PAT nur die folgende eingeschränkte Berechtigung:. `api`

   > **Note**
   >
   > Nur ein Administrator kann das PAT erstellen und verwenden.

   Klicken Sie auf Weiter.



8. Die Verbindungsseite zeigt die erstellte Verbindung im Status Available (Verfügbar).

Schritt 3: Speichern Sie Ihre GitLab selbstverwaltete Quellaktion

Gehen Sie im Assistenten oder auf der Aktionsseite „Aktion bearbeiten“ wie folgt vor, um Ihre Quellaktion mit Ihren Verbindungsinformationen zu speichern.

Um Ihre Quellaktion mit Ihrer Verbindung abzuschließen und zu speichern

1. Wählen Sie unter Repository name (Repository-Name) den Namen Ihres Drittanbieter-Repositorys aus.
2. Unter Pipeline-Trigger können Sie Auslöser hinzufügen, wenn es sich bei Ihrer Aktion um eine CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern.
3. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

   - Um Ausgabeartefakte aus der GitLab selbstverwalteten Aktion mit der Standardmethode zu speichern, wählen Sie CodePipelineStandard. Die Aktion greift auf die Dateien aus dem Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.
   - Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone (Vollständiger Klon). Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

4. Wählen Sie im Assistenten Weiter oder auf der Aktionsseite Bearbeiten die Option Speichern aus.

Erstellen Sie einen Host und eine Verbindung zu GitLab Self-Managed (CLI)

Sie können das AWS Command Line Interface (AWS CLI) verwenden, um eine Verbindung herzustellen.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über AWS CLI oder AWS CloudFormation erstellt wurde, hat standardmäßig `PENDING` den Status. Nachdem Sie eine Verbindung mit der CLI hergestellt haben oder verwenden Sie die Konsole AWS CloudFormation, um die Verbindung so zu bearbeiten, dass sie ihren Status festlegt`AVAILABLE`.

Sie können das AWS Command Line Interface (AWS CLI) verwenden, um einen Host für installierte Verbindungen zu erstellen.

Sie verwenden Sie einen Host, der den Endpunkt für die Infrastruktur darstellt, in der der Drittanbieter installiert ist. Nachdem Sie die Hosterstellung mit der CLI abgeschlossen haben, befindet sich der Host im Status Ausstehend. Anschließend richten Sie den Host ein oder registrieren ihn, um ihn in den Status Verfügbar zu versetzen. Sobald der Host verfügbar ist, führen Sie die Schritte zum Erstellen einer Verbindung aus.

Verwenden Sie dazu den Befehl create-host.

> **Important**
>
> Ein über den erstellter Host AWS CLI befindet sich standardmäßig im `Pending` Status. Nachdem Sie einen Host mit der CLI erstellt haben, verwenden Sie die Konsole oder die CLI, um den Host so einzurichten, dass er seinen Status annimmt`Available`.

Einen Host erstellen

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie den, AWS CLI um den create-host Befehl auszuführen, und geben Sie dabei `--name--provider-type`, und `--provider-endpoint` für Ihre Verbindung an. In diesem Beispiel lautet der Name des Drittanbieters `GitLabSelfManaged` und der Endpunkt `my-instance.dev`.

   ```
   aws codestar-connections create-host --name MyHost --provider-type
    GitLabSelfManaged --provider-endpoint "https://my-instance.dev"
   ```

   Wenn der Befehl erfolgreich ausgeführt wurde, gibt er die Amazon-Ressourcenname (ARN)-Informationen zum Host ähnlich der folgenden zurück.

   ```
   {
       "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
   Host-28aef605"
   }
   ```

   Nach diesem Schritt befindet sich der Host im Status `PENDING` (Ausstehend).

2. Schließen Sie das Host-Setup mit der Konsole ab und ändern Sie den Host-Status zu `Available` (Verfügbar).

Um eine Verbindung zu GitLab Self-Managed herzustellen

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie den AWS CLI , um den create-connection Befehl auszuführen, und geben Sie dabei `--host-arn` und `--connection-name` für Ihre Verbindung an.

   ```
   aws codestar-connections create-connection --host-arn arn:aws:codestar-
   connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
    MyConnection
   ```

   Wenn der Befehl erfolgreich ausgeführt wurde, gibt er die ARN-Informationen der Verbindung ähnlich der folgenden zurück.

   ```
   {
   ```

```
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad"
}
```

2. Verwenden Sie die Konsole, um die ausstehende Verbindung einzurichten.
3. Die Pipeline erkennt standardmäßig Änderungen, wenn der Code an das Quell-Repository der Verbindung weitergeleitet wird. Gehen Sie wie folgt vor, um die Pipeline-Trigger-Konfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

   - Um die Pipeline-Trigger-Konfiguration so zu konfigurieren, dass sie nur mit einer manuellen Veröffentlichung beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

     ```
     "DetectChanges": "false",
     ```

   - Weitere Informationen zur Konfiguration der Pipeline-Trigger zum Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern. Im Folgenden wird beispielsweise die Pipeline-Ebene der Pipeline-JSON-Definition erweitert. In diesem Beispiel `release-v1` sind `release-v0` und die Git-Tags, die eingeschlossen werden sollen, und `release-v2` ist das Git-Tag, das ausgeschlossen werden soll.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
     ```

Eine Pipeline bearbeiten in CodePipeline

Eine Pipeline beschreibt den Release-Prozess, dem Sie folgen AWS CodePipeline möchten, einschließlich der Phasen und Aktionen, die abgeschlossen werden müssen. Sie können eine Pipeline bearbeiten, um diese Elemente hinzuzufügen oder zu entfernen. Wenn Sie jedoch eine Pipeline bearbeiten, können Werte wie der Pipelinename oder die Pipelinemetadaten nicht geändert werden.

Sie können Ihren Pipeline-Typ, Ihre Variablen und Trigger auf der Pipeline-Bearbeitungsseite bearbeiten. Sie können Ihrer Pipeline auch Phasen und Aktionen hinzufügen oder ändern.

Anders als beim Erstellen einer Pipeline wird beim Bearbeiten nicht die letzte Revision über die Pipeline erneut ausgeführt. Wenn Sie die letzte Revision über eine gerade bearbeitete Pipeline ausführen möchten, müssen Sie sie manuell erneut ausführen. Andernfalls wird die bearbeitete Pipeline bei der nächsten Änderung eines Quellstandorts, der in der Quellphase konfiguriert wird, ausgeführt. Weitere Informationen finden Sie unter Manuelles Starten einer Pipeline.

Sie können Ihrer Pipeline Aktionen hinzufügen, die sich in einer anderen AWS Region als Ihrer Pipeline befinden. Wenn an der Anbieter für eine Aktion AWS-Service ist und sich dieser Aktionstyp/dieser Anbietertyp in einer anderen AWS Region als Ihre Pipeline befindet, handelt es sich um eine regionsübergreifende Aktion. Weitere Informationen zu regionsübergreifenden Aktionen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

CodePipeline verwendet Methoden zur Erkennung von Änderungen, um Ihre Pipeline zu starten, wenn eine Quellcodeänderung übertragen wird. Diese Erkennungsmethoden basieren auf dem Quelltyp:

- CodePipeline verwendet Amazon CloudWatch Events, um Änderungen in Ihrem CodeCommit Quell-Repository oder Ihrem Amazon S3 S3-Quell-Bucket zu erkennen.

> **Note**
>
> Ressourcen zur Änderungserkennung werden automatisch erstellt, wenn Sie die Konsole verwenden. Wenn Sie die Konsole zum Erstellen oder Ändern einer Pipeline verwenden, werden die zusätzlichen Ressourcen für Sie erstellt. Wenn Sie AWS CLI die Pipeline mit erstellen, müssen Sie die zusätzlichen Ressourcen selbst erstellen. Weitere Informationen zum Erstellen oder Aktualisieren einer CodeCommit Pipeline finden Sie unterEine EventBridge Regel für eine CodeCommit Quelle erstellen (CLI). Weitere Informationen zur

Verwendung der CLI zum Erstellen oder Aktualisieren einer Amazon S3-Pipeline finden Sie unterEine EventBridge Regel für eine Amazon S3 S3-Quelle (CLI) erstellen.

Themen

- Bearbeiten einer Pipeline (Konsole)
- Bearbeiten einer Pipeline (AWS CLI)

Bearbeiten einer Pipeline (Konsole)

Sie können die CodePipeline Konsole verwenden, um Phasen in einer Pipeline hinzuzufügen, zu bearbeiten oder zu entfernen und Aktionen in einer Phase hinzuzufügen, zu bearbeiten oder zu entfernen.

Wenn Sie eine Pipeline aktualisieren, schließt CodePipeline sie alle laufenden Aktionen ordnungsgemäß ab und schlägt dann die Phasen und Pipeline-Ausführungen fehl, bei denen die laufenden Aktionen abgeschlossen wurden. Wenn eine Pipeline aktualisiert wird, müssen Sie Ihre Pipeline erneut ausführen. Weitere Informationen zum Betrieb einer Pipeline finden Sie unterManuelles Starten einer Pipeline.

Bearbeiten einer Pipeline

1. Melden Sie sich unter http://console.aws.amazon.com/codesuite/codepipeline/home bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Auf diese Weise wird eine detaillierte Ansicht der Pipeline geöffnet (einschließlich des Status der Aktionen in den einzelnen Stufen der Pipeline).
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Um den Pipeline-Typ zu bearbeiten, wählen Sie auf der Karte Bearbeiten: Pipeline-Eigenschaften die Option Bearbeiten aus. Wählen Sie eine der folgenden Optionen und anschließend „Fertig“.

   - Pipelines vom Typ V1 haben eine JSON-Struktur, die Standardparameter auf Pipeline-, Stufen- und Aktionsebene enthält.

- Pipelines vom Typ V2 haben dieselbe Struktur wie Pipelines vom Typ V1 und unterstützen zusätzliche Parameter wie Trigger und Variablen auf Pipeline-Ebene.

Pipeline-Typen unterscheiden sich in ihren Eigenschaften und im Preis. Weitere Informationen finden Sie unter Pipeline-Typen.

5. Um die Pipeline-Variablen zu bearbeiten, wählen Sie auf der Karte Bearbeiten: Variablen die Option Variablen bearbeiten aus. Fügen Sie Variablen für die Pipeline-Ebene hinzu oder ändern Sie sie und wählen Sie dann Fertig aus.

   Weitere Informationen zu Variablen auf Pipeline-Ebene finden Sie unterVariablen. Ein Tutorial mit einer Variablen auf Pipelineebene, die bei der Ausführung der Pipeline übergeben wird, finden Sie unter. Tutorial: Variablen auf Pipeline-Ebene verwenden

   > **Note**
   >
   > Es ist zwar optional, Variablen auf Pipelineebene hinzuzufügen, aber bei einer Pipeline, die mit Variablen auf Pipelineebene spezifiziert ist, für die keine Werte bereitgestellt werden, schlägt die Pipelineausführung fehl.

6. Um die Pipeline-Trigger zu bearbeiten, wählen Sie auf der Karte Bearbeiten: Auslöser die Option Trigger bearbeiten aus. Fügen Sie Trigger hinzu oder ändern Sie sie und wählen Sie dann Fertig.

   Weitere Informationen zum Hinzufügen von Triggern findest du in den Schritten zum Herstellen einer Verbindung zu Bitbucket Cloud GitHub (Version 2), GitHub Enterprise Server, GitLab .com oder GitLab zur Selbstverwaltung, z. B. GitHub Verbindungen

7. Um Stufen und Aktionen auf der Bearbeitungsseite zu bearbeiten, führe einen der folgenden Schritte aus:

   - Um eine Stufe zu bearbeiten, klicken Sie auf Edit stage (Stufe bearbeiten). Sie können Aktionen seriell und parallel zu vorhandenen Aktionen hinzufügen:

     Sie können in dieser Ansicht außerdem Aktionen bearbeiten, indem Sie das Bearbeiten-Symbol für diese Aktionen auswählen. Um eine Aktion zu löschen, wählen Sie das Löschen-Symbol für diese Aktion aus.
   - Um eine Aktion zu bearbeiten, wählen Sie das Bearbeiten-Symbol für diese Aktion aus und ändern dann die Werte unter Edit action. Mit einem Sternchen (*) markierte Elemente sind Pflichtangaben.

  - Für einen CodeCommit Repository-Namen und einen Branch wird eine Meldung mit der Amazon CloudWatch Events-Regel angezeigt, die für diese Pipeline erstellt werden soll. Wenn Sie die CodeCommit Quelle entfernen, erscheint eine Meldung mit der Information, dass die Amazon CloudWatch Events-Regel gelöscht werden soll.
  - Für einen Amazon S3 S3-Quell-Bucket wird eine Meldung mit der Amazon CloudWatch Events-Regel und dem zu erstellenden AWS CloudTrail Trail für diese Pipeline angezeigt. Wenn Sie die Amazon S3 S3-Quelle entfernen, wird eine Meldung mit der Angabe der Amazon CloudWatch Events-Regel und des zu löschenden AWS CloudTrail Trails angezeigt. Wenn der AWS CloudTrail Trail von anderen Pipelines verwendet wird, wird der Trail nicht entfernt und das Datenereignis wird gelöscht.
- Um eine Stufe hinzuzufügen, wählen Sie an dem Punkt der Pipeline + Add stage (Stufe hinzufügen) aus, an dem Sie eine Stufe hinzufügen möchten. Geben Sie einen Namen für die Stufe ein und fügen Sie mindestens eine Aktion hinzu. Mit einem Sternchen (*) markierte Elemente sind Pflichtangaben.
- Um eine Stufe zu löschen, wählen Sie das Löschen-Symbol der Stufe aus. Die Stufe und ihre gesamten Aktionen werden gelöscht.
- Um eine Phase so zu konfigurieren, dass sie bei einem Fehler automatisch zurückgesetzt wird, wählen Sie Phase bearbeiten und anschließend das Kontrollkästchen Automatisches Rollback bei Stufenausfall konfigurieren aus.

Wenn Sie beispielsweise zu einer Stufe in einer Pipeline eine serielle Aktion hinzufügen möchten:

1. Wählen Sie in der Stufe, in der Sie Ihre Aktion hinzufügen möchten, Edit stage (Stufe bearbeiten) und danach + Add action group (+ Aktionsgruppe hinzufügen) aus.
2. Geben im Popup-Fenster Edit action (Aktion bearbeiten) unter Action name (Aktionsname) den Namen Ihrer Aktion ein. Die Liste Action provider (Aktionsanbieter) zeigt die Optionen für Anbieter nach Kategorie an. Suchen Sie die Kategorie (z. B. Deploy (Bereitstellen)). Wählen Sie unter der Kategorie den Anbieter aus (z. B. AWS CodeDeploy). Wählen Sie unter Region die AWS Region aus, in der die Ressource erstellt wird oder in der Sie sie erstellen möchten. Das Feld Region gibt an, wo die AWS Ressourcen für diesen Aktionstyp und Anbietertyp erstellt werden. Dieses Feld wird nur für Aktionen angezeigt, bei denen es sich bei dem Aktionsanbieter um einen handelt. AWS-Service Das Feld Region ist standardmäßig auf dieselbe AWS Region wie Ihre Pipeline eingestellt.

Weitere Informationen zu den Anforderungen für Aktionen in CodePipeline, einschließlich der Namen für Eingabe- und Ausgabeartefakte und deren Verwendung, finden Sie unterAnforderungen an die Aktionsstruktur in CodePipeline. Beispiele zum Hinzufügen von Aktionsanbietern mithilfe der Standard-Felder für die einzelnen Anbieter finden Sie unter Erstellen einer Pipeline (Konsole).

Informationen zum Hinzufügen CodeBuild als Build- oder Testaktion zu einer Phase finden Sie unter CodeBuild Use CodePipeline with to Test Code and Run Builds im CodeBuild Benutzerhandbuch.

> **Note**
>
> Bei einigen Aktionsanbietern, z. B. GitHub, müssen Sie eine Verbindung zur Website des Anbieters herstellen, bevor Sie die Konfiguration der Aktion abschließen können. Stellen Sie bei der Verbindung mit der Website eines Anbieters sicher, dass Sie die Anmeldeinformationen für diese Website verwenden. Verwenden Sie nicht Ihre AWS Anmeldeinformationen.

3. Wenn Sie die Aktionskonfiguration abgeschlossen haben, wählen Sie Save (Speichern).

> **Note**
>
> Sie können eine Phase in der Konsolenansicht nicht umbenennen. Sie können eine Phase mit dem neuen Namen hinzufügen und dann die alte löschen. Stellen Sie sicher, dass Sie alle Aktionen für die Stufe hinzugefügt haben, bevor Sie die alte löschen.

8. Wenn Sie die Bearbeitung der Pipeline abgeschlossen haben, klicken Sie auf Save (Speichern), um zur Übersichtsseite zurückzukehren.

> **Important**
>
> Nachdem Sie Ihre Änderungen gespeichert haben, können Sie sie nicht mehr rückgängig machen. Sie müssen die Pipeline erneut bearbeiten. Wenn beim Speichern Ihrer Änderungen eine Revision in Ihrer Pipeline ausgeführt wird, wird die Ausführung nicht abgeschlossen. Wenn Sie möchten, dass ein bestimmter Commit oder eine Änderung über die bearbeitete Pipeline ausgeführt wird, müssen Sie die Änderung manuell über die

Pipeline ausführen. Andernfalls wird der nächste Commit bzw. die Änderung automatisch über die Pipeline ausgeführt.

9. Um Ihre Aktion zu testen, wählen Sie Änderung veröffentlichen aus, um den Commit über die Pipeline zu verarbeiten und eine Änderung an der Quelle zu übernehmen, die in der Quellphase der Pipeline angegeben wurde. Oder folgen Sie den Schritten Manuelles Starten einer Pipeline unter, AWS CLI um eine Änderung manuell freizugeben.

Bearbeiten einer Pipeline (AWS CLI)

Sie können den Befehl update-pipeline verwenden, um eine Pipeline zu bearbeiten.

Wenn Sie eine Pipeline aktualisieren, werden alle laufenden CodePipeline Aktionen ordnungsgemäß abgeschlossen und anschließend die Phasen und Pipeline-Ausführungen, in denen die laufenden Aktionen abgeschlossen wurden, fehlschlagen. Wenn eine Pipeline aktualisiert wird, müssen Sie Ihre Pipeline erneut ausführen. Weitere Informationen zum Betrieb einer Pipeline finden Sie unterManuelles Starten einer Pipeline.

> **Important**
>
> Sie können die zwar verwenden AWS CLI , um Pipelines zu bearbeiten, die Partneraktionen enthalten, aber Sie dürfen das JSON einer Partneraktion nicht manuell bearbeiten. In diesem Fall wird die Partneraktion nach einem Update der Pipeline fehlschlagen.

Bearbeiten einer Pipeline

1. Öffnen Sie eine Terminalsitzung (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und führen Sie den get-pipeline Befehl aus, um die Pipeline-Struktur in eine JSON-Datei zu kopieren. Geben Sie für eine Pipeline mit dem Namen **`MyFirstPipeline`** den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem beliebigen Texteditor und ändern Sie die Struktur der Datei, um die gewünschten Änderungen an die Pipeline einzufügen. Sie können beispielsweise Stufen hinzufügen oder entfernen oder eine weitere Aktion zu einer vorhandenen Stufe hinzufügen.

   Das folgende Beispiel zeigt, wie Sie der Datei pipeline.json eine weitere Bereitstellungsphase hinzufügen würden. Diese Stufe wird nach der ersten Bereitstellungsstufe mit dem Namen *Staging* ausgeführt.

   > **Note**
   >
   > Dies ist nur ein Teil und nicht die komplette Struktur der Datei. Weitere Informationen finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

```
,
        {
            "name": "Staging",
            "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "MyApp"
                            }
                        ],
                        "name": "Deploy-CodeDeploy-Application",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "CodePipelineDemoApplication",
                            "DeploymentGroupName": "CodePipelineDemoFleet"
                        },
                        "runOrder": 1
                    }
                ]
            },
        {
```

```
            "name": "Production",
            "actions":  [
                    {
                        "inputArtifacts": [
                            {
                                "name": "MyApp"
                            }
                        ],
                        "name": "Deploy-Second-Deployment",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                        "ApplicationName": "CodePipelineDemoApplication",
                        "DeploymentGroupName": "CodePipelineProductionFleet"
                        },
                        "runOrder": 1
                    }
                ]
        }
    ]
}
```

Informationen zum Verwenden der CLI zum Hinzufügen einer Genehmigungsaktion zu einer Pipeline finden Sie unter Fügen Sie einer Pipeline eine manuelle Genehmigungsaktion hinzu in CodePipeline .

Stellen Sie sicher, dass der `PollForSourceChanges`-Parameter in Ihrer JSON-Datei wie folgt festgelegt ist:

```
                "PollForSourceChanges": "false",
```

CodePipeline verwendet Amazon CloudWatch Events, um Änderungen in Ihrem CodeCommit Quell-Repository und Branch oder Ihrem Amazon S3 S3-Quell-Bucket zu erkennen. Der nächste Schritt enthält Anweisungen zum manuellen Erstellen dieser Ressourcen. Wenn das Flag auf `false` gesetzt wird, werden regelmäßigen Prüfungen deaktiviert, die nicht erforderlich sind, wenn die empfohlenen Methoden zur Änderungserkennung verwendet werden.

3. Um eine Build-, Test- oder Bereitstellungsaktion in einer anderen Region als Ihre Pipeline hinzuzufügen, müssen Sie Ihre Pipeline-Struktur um Folgendes ergänzen. Detaillierte Anweisungen finden Sie unter Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline.

   - Fügen Sie den `Region`-Parameter der Pipeline-Struktur Ihrer Aktion hinzu.
   - Verwenden Sie den `artifactStores`-Parameter zum Angeben eines Artefakt-Buckets für jede Region, in der Sie über eine Aktion verfügen.

4. Wenn Sie mit einer Pipeline-Struktur arbeiten, die Sie mit dem Befehl get-pipeline abgerufen haben, müssen Sie die Struktur in der JSON-Datei ändern. Sie müssen die `metadata`-Zeilen aus der Datei entfernen, damit der Befehl update-pipeline sie verwenden kann. Entfernen Sie den Abschnitt aus der Pipeline-Struktur in der JSON-Datei (die `"metadata": { }`-Zeilen und die Fehler `"created"`, `"pipelineARN"` und `"updated"`).

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

   ```
   "metadata": {
     "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
     "created": "date",
     "updated": "date"
     }
   ```

   Speichern Sie die Datei.

5. Wenn Sie die CLI zum Bearbeiten einer Pipeline verwenden, müssen Sie die empfohlenen Änderungserkennungsressourcen für Ihre Pipeline manuell verwalten:

   - Für ein CodeCommit Repository müssen Sie die CloudWatch Ereignisregel erstellen, wie unter beschriebenEine EventBridge Regel für eine CodeCommit Quelle erstellen (CLI).
   - Für eine Amazon S3 S3-Quelle müssen Sie die Regel und den AWS CloudTrail Trail für CloudWatch Ereignisse erstellen, wie unter beschriebenAmazon S3 S3-Quellaktionen und EventBridge mit AWS CloudTrail.

6. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

> **Important**
>
> Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

```
aws codepipeline update-pipeline --cli-input-json file://pipeline.json
```

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen.

7. Öffnen Sie die CodePipeline Konsole und wählen Sie die Pipeline aus, die Sie gerade bearbeitet haben.

   Die Pipeline zeigt die Änderungen an. Wenn Sie das nächste Mal eine Änderung am Quellspeicherort durchführen, führt die Pipeline die Revision über die überarbeitete Struktur der Pipeline aus.

8. Um die letzte Revision manuell über die überarbeitete Struktur der Pipeline auszuführen, nutzen Sie den Befehl start-pipeline-execution. Weitere Informationen finden Sie unter Manuelles Starten einer Pipeline.

Weitere Informationen zur Struktur einer Pipeline und zu den erwarteten Werten finden Sie unter CodePipeline Referenz zur Pipeline-Struktur AWS CodePipeline API-Referenz.

Pipelines und Details anzeigen in CodePipeline

Sie können die AWS CodePipeline Konsole oder die verwenden, um Details AWS CLI zu Pipelines anzuzeigen, die mit Ihrem AWS Konto verknüpft sind.

Themen

Pipelines anzeigen (Konsole)

Sie können den Status, die Übergänge und die Artefaktaktualisierungen für eine Pipeline anzeigen.

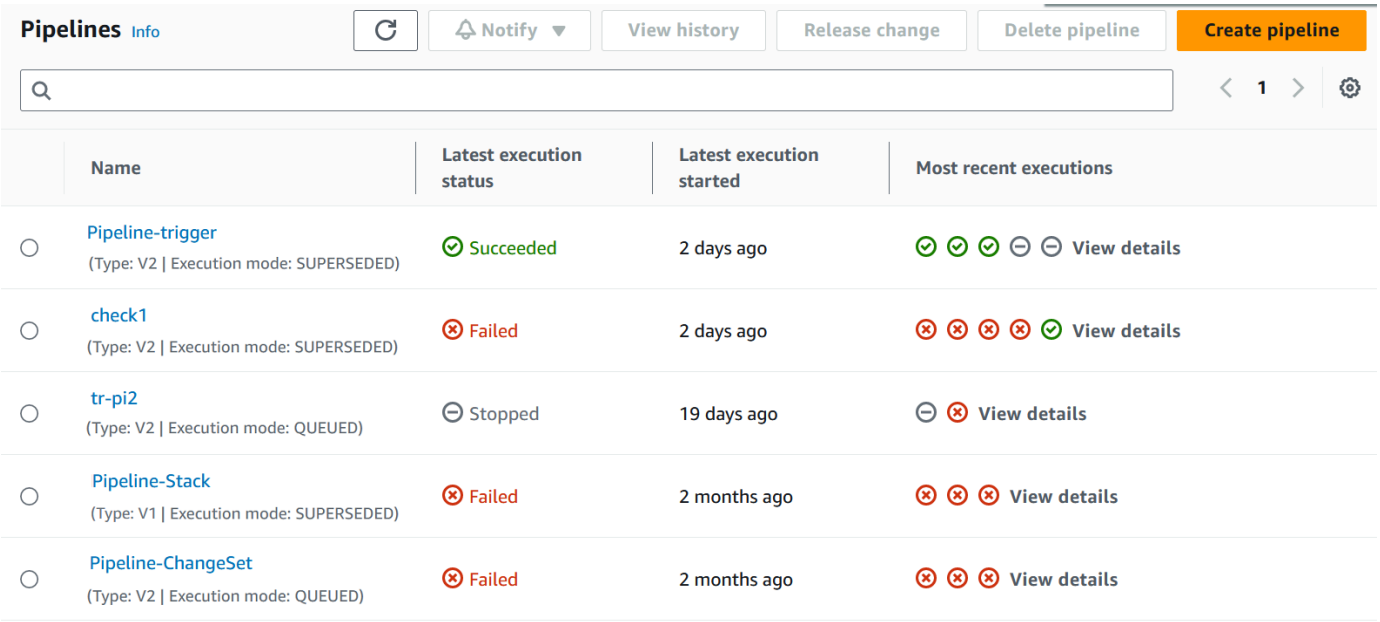
> **Note**
>
> Nach einer Stunde wird die detaillierte Anzeige der Pipeline in Ihrem Browser nicht mehr automatisch aktualisiert. Um aktuelle Informationen anzuzeigen, müssen Sie die Seite aktualisieren.

Um Pipelines anzuzeigen

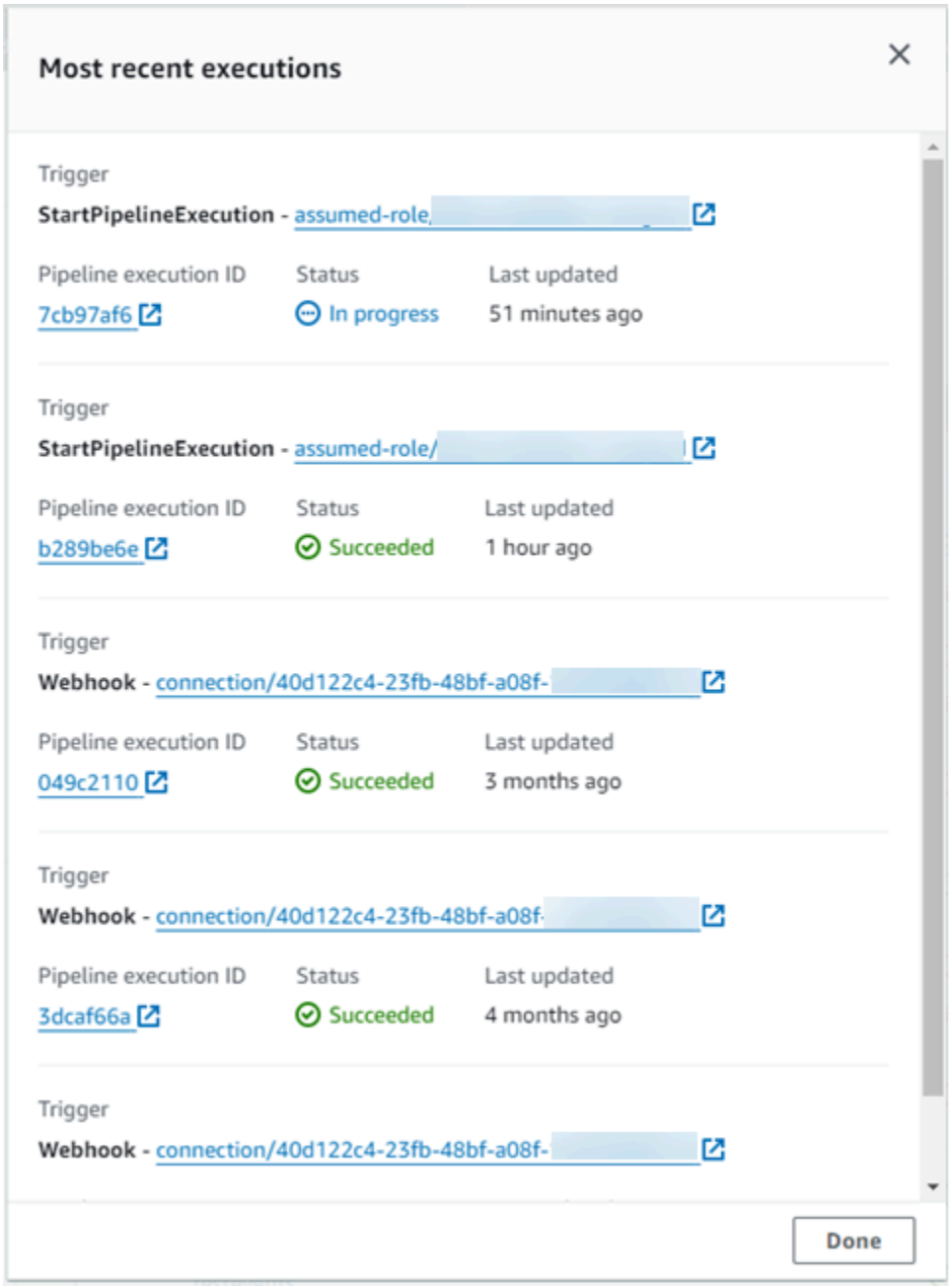
1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Seite „Pipelines“ wird angezeigt. Eine Liste all Ihrer Pipelines für diese Region wird angezeigt.

   Name, Typ, Status, Version, Erstellungsdatum und Datum der letzten Änderung aller mit Ihrem AWS Konto verknüpften Pipelines werden zusammen mit der zuletzt gestarteten Ausführungszeit angezeigt.
2. Der Status der fünf letzten Ausführungen wird angezeigt.



Wählen Sie neben einer bestimmten Zeile die Option Details anzeigen aus, um ein Detaildialogfeld mit den letzten Ausführungen anzuzeigen.

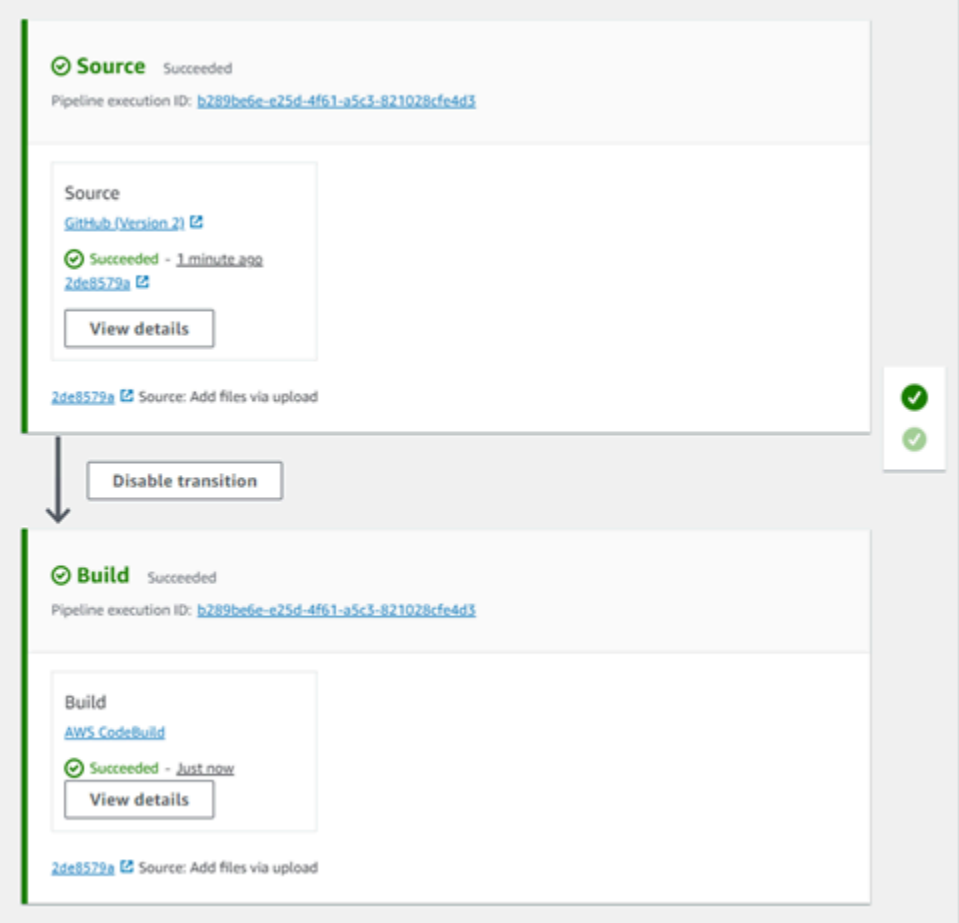


Um Details zu den neuesten Ausführungen für die Pipeline anzuzeigen, wählen Sie View History (Verlauf anzeigen). Die früheren Ausführungen können Sie in den Revisionsdetails zu den Quell-Artefakten sehen (z. B. Ausführungs-IDs, Status, Start- und Endzeiten, Dauer und Commit-IDs und -Nachrichten).

> **Note**
>
> Bei einer Pipeline im PARALLEL-Ausführungsmodus werden in der Hauptansicht der Pipeline weder die Pipeline-Struktur noch die laufenden Ausführungen angezeigt. Bei

einer Pipeline im PARALLEL-Ausführungsmodus greifen Sie auf die Pipeline-Struktur zu, indem Sie auf der Seite mit dem Ausführungsverlauf die ID für die Ausführung auswählen, die Sie anzeigen möchten. Wählen Sie im linken Navigationsbereich Verlauf aus, wählen Sie die Ausführungs-ID für die parallel Ausführung aus, und zeigen Sie dann die Pipeline auf der Registerkarte Visualisierung an.

3. Wenn Sie Details für eine einzelne Pipeline anzeigen möchten, wählen Sie in Name die entsprechende Pipeline aus. Es wird eine detaillierte Ansicht der Pipeline, einschließlich des Status der einzelnen Aktionen in jeder Stufe und des Status der Übergänge, angezeigt.



Die grafische Ansicht zeigt die folgenden Informationen für jede Stufe an:

- Der Name der Stufe.
- Jede für die Stufe konfigurierte Aktion.
- Den Status der Übergänge zwischen den Stufen (aktiviert oder deaktiviert) entsprechend dem Status des Pfeils zwischen den Stufen. Ein aktivierter Übergang wird durch einen Pfeil mit der daneben angeordneten Schaltfläche Disable transition (Übergang deaktivieren)

gekennzeichnet. Ein deaktivierter Übergang wird durch einen Pfeil mit Durchstrich und der daneben angeordneten Schaltfläche Enable transition (Übergang aktivieren) gekennzeichnet.

- Ein Farbbalken zeigt den Status der Stufe an:
  - Grau: Zurzeit keine Ausführungen
  - Blau: In Bearbeitung
  - Grün: Erfolgreich
  - Rot: Fehlgeschlagen

Die grafische Ansicht zeigt außerdem die folgenden Informationen zu Aktionen in den einzelnen Stufen an:

- Den Namen der Aktion.
- Der Anbieter der Aktion, z. CodeDeploy B.
- Wann die Aktion zuletzt ausgeführt wurde.
- Ob die Aktion erfolgreich war oder fehlgeschlagen ist.
- Links zu weiteren Informationen zur letzten Ausführung der Aktion (sofern verfügbar).
- Details zu den Quellrevisionen, die die letzte Pipeline-Ausführung in der Phase durchlaufen, oder, bei CodeDeploy Bereitstellungen, zu den neuesten Quellrevisionen, die auf Ziel-Instances bereitgestellt wurden.
- Eine Schaltfläche „Details anzeigen“, mit der ein Dialogfeld mit Details zur Ausführung der Aktion, zu Protokollen und zur Aktionskonfiguration geöffnet wird.

> **Note**
>
> Die Registerkarte „Protokolle“ ist für CodeBuild AWS CloudFormation Aktionen verfügbar, die im Konto der Pipeline ausgeführt wurden.

4. Um Details zum Anbieter der Aktion anzuzeigen, wählen Sie den Anbieter aus. Wenn Sie beispielsweise in der vorherigen Beispielpipeline entweder die Phase Staging oder die Produktionsphase wählen CodeDeploy , wird die CodeDeploy Konsolenseite für die für diese Phase konfigurierte Bereitstellungsgruppe angezeigt.
5. Um zu sehen, wird der Fortschritt einer Aktion neben einer laufenden Aktion angezeigt (gekennzeichnet durch die Meldung In Bearbeitung). Wenn die Aktion in Bearbeitung ist, werden der inkrementelle Fortschritt und die Schritte oder Aktionen angezeigt.

6. Um für die manuelle Genehmigung konfigurierte Aktionen zu genehmigen oder abzulehnen, wählen Sie Review aus.
7. Um nicht erfolgreich abgeschlossene Aktionen in einer Stufe zu wiederholen, klicken Sie auf Retry.
8. Der Status der letzten Ausführung der Aktion, einschließlich der Ergebnisse dieser Aktion (Erfolgreich oder Fehlgeschlagen), wird angezeigt.

Aktionsdetails in einer Pipeline (Konsole) anzeigen

Sie können Details für eine Pipeline anzeigen, einschließlich Details zu Aktionen in jeder Phase.

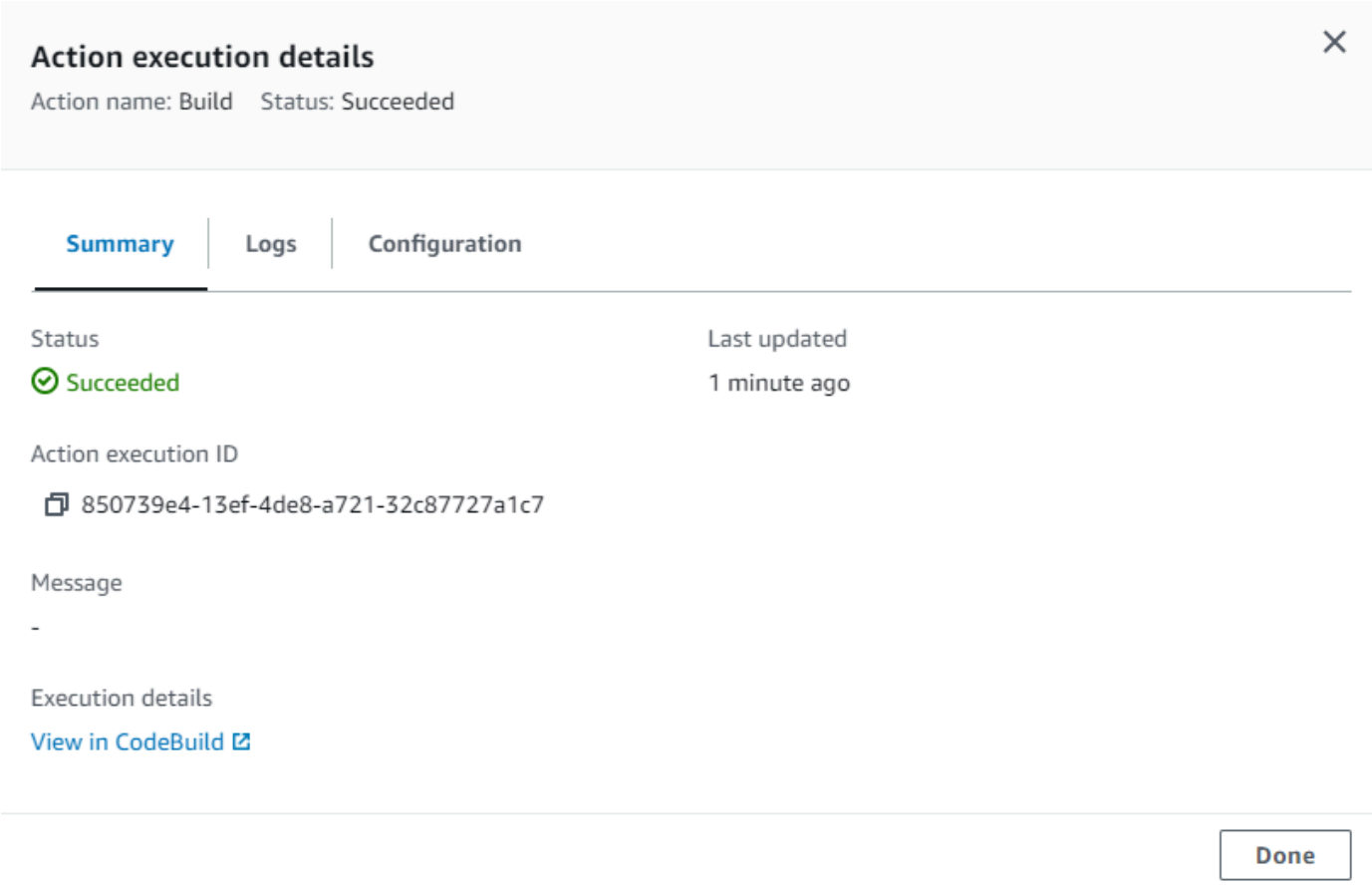
> **Note**
>
> Nach einer Stunde wird die detaillierte Anzeige der Pipeline in Ihrem Browser nicht mehr automatisch aktualisiert. Um aktuelle Informationen anzuzeigen, müssen Sie die Seite aktualisieren.

Um Aktionsdetails in einer Pipeline anzuzeigen

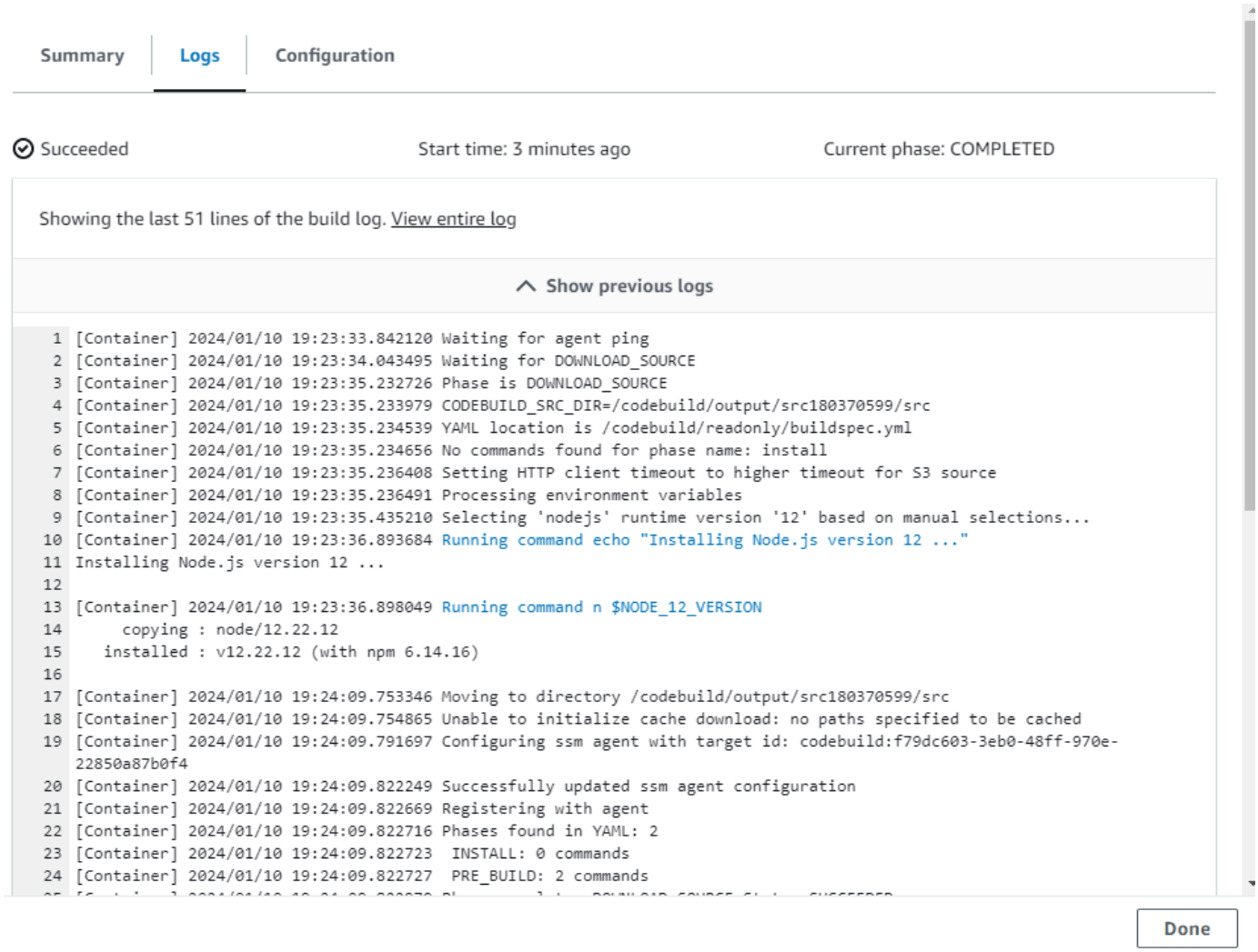
1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Seite „Pipelines“ wird angezeigt.
2. Wählen Sie bei jeder Aktion Details anzeigen aus, um ein Dialogfeld mit Details zur Aktionsausführung, den Protokollen und der Aktionskonfiguration zu öffnen.

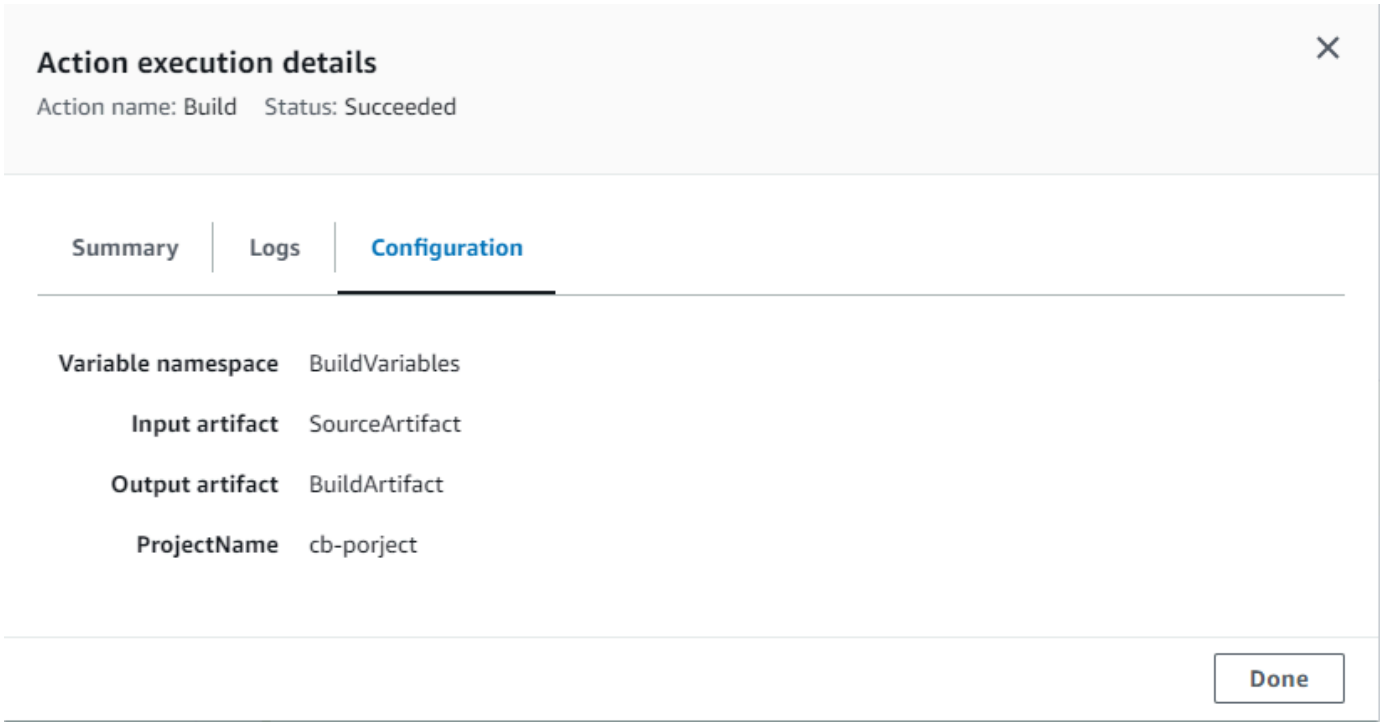
   > **Note**
   >
   > Die Registerkarte Protokolle ist für AWS CloudFormation Aktionen CodeBuild und Aktionen verfügbar.
3. Um die Aktionsübersicht für eine Aktion in einer Phase einer Pipeline anzuzeigen, wählen Sie Details zur Aktion anzeigen und dann die Registerkarte Zusammenfassung aus.



4. Um die Aktionsprotokolle für eine Aktion mit Protokollen anzuzeigen, wählen Sie Details zur Aktion anzeigen und dann die Registerkarte Protokolle aus.



5. Um die Konfigurationsdetails für eine Aktion anzuzeigen, wählen Sie die Registerkarte Konfiguration.



Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen

Sie können die Konsole verwenden, um Pipeline-Einstellungen wie den Pipeline-ARN, den Dienstrollen-ARN und den Pipeline-Artefaktspeicher anzuzeigen.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller Pipelines, die mit Ihrem AWS Konto verknüpft sind, werden angezeigt.
2. Wählen Sie den Namen Ihrer Pipeline und dann im linken Navigationsbereich Einstellungen aus. Auf der Seite wird Folgendes angezeigt:

   - Der Name der Pipeline
   - Die Pipeline Amazon Resource Name (ARN)

     Der Pipeline-ARN wird in folgendem Format erstellt:

     arn:aws:codepipeline: Region: Konto: Pipeline-Name

     Pipeline-Beispiel-ARN:

     `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

- Die CodePipeline Servicerolle ARN für Ihre Pipeline
- Die Pipeline-Version
- Der Name und der Speicherort des Artefaktspeichers für die Pipeline

Anzeigen von der Pipeline-Details und des Verlaufs (CLI)

Sie können die folgenden Befehle verwenden, um weitere Details zu Ihren Pipelines und Pipeline-Ausführungen anzuzeigen:

- list-pipelinesBefehl, um eine Zusammenfassung aller mit Ihrem AWS Konto verknüpften Pipelines anzuzeigen.
- get-pipeline-Befehle, um Details einer einzelnen Pipeline zu überprüfen.
- list-pipeline-executions, um Zusammenfassungen der letzten Ausführungen für eine Pipeline anzuzeigen
- get-pipeline-execution, um Informationen zu einer Ausführung einer Pipeline zurück, einschließlich Details zu Artefakten, Pipeline-Ausführungs-ID und Name, Version und Status der Pipeline anzuzeigen.
- get-pipeline-state-Befehl zum Anzeigen von Pipeline, Phase und Aktionsstatus.
- Aktion list-action-executions zum Anzeigen von Ausführungsdetails für eine Pipeline.

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) und verwenden Sie den AWS CLI , um den list-pipelines Befehl auszuführen:

   ```
   aws codepipeline list-pipelines
   ```

   Dieser Befehl gibt eine Liste aller Pipelines in Ihrem AWS -Konto zurück.

2. Um Informationen zu einer Pipeline anzuzeigen, führen Sie den Befehl get-pipeline unter Angabe des eindeutigen Namens der Pipeline aus. Um beispielsweise Details zu einer Pipeline mit dem Namen anzuzeigen *MyFirstPipeline*, geben Sie Folgendes ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline
   ```

   Dieser Befehl gibt die Struktur der Pipeline zurück.

Löschen Sie eine Pipeline in CodePipeline

Sie können eine Pipeline stets bearbeiten, um ihre Funktionalität zu ändern. Möglicherweise entschließen Sie sich jedoch, sie stattdessen zu löschen. Sie können die AWS CodePipeline Konsole oder den delete-pipeline Befehl in verwenden AWS CLI , um eine Pipeline zu löschen.

Themen

Löschen einer Pipeline (Konsole)

So löschen Sie eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen und der Status aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie löschen möchten.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Klicken Sie auf der Seite Edit auf Delete.
5. Geben Sie zur Bestätigung **`delete`** in das Feld ein und wählen Sie dann Delete (Löschen).

> **Important**
>
> Diese Aktion kann nicht rückgängig gemacht werden.

Löschen einer Pipeline (CLI)

Verwenden Sie den Befehl delete-pipeline, AWS CLI um eine Pipeline manuell zu löschen.

> **Important**
>
> Das Löschen einer Pipeline kann nicht rückgängig gemacht werden. Es wird kein Bestätigungsdialogfeld angezeigt. Nach der Ausführung des Befehls ist die Pipeline gelöscht. Die in der Pipeline verwendeten Ressourcen werden jedoch nicht gelöscht. Dies vereinfacht

das Erstellen einer neuen Pipeline, die diese Ressourcen für die Automatisierung der Veröffentlichung Ihrer Software verwendet.

So löschen Sie eine Pipeline

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie die, AWS CLI um den delete-pipeline Befehl auszuführen. Geben Sie dabei den Namen der Pipeline an, die Sie löschen möchten. Um beispielsweise eine Pipeline mit dem Namen zu löschen *MyFirstPipeline*:

   ```
   aws codepipeline delete-pipeline --name MyFirstPipeline
   ```

   Mit diesem Befehl wird kein Inhalt zurückgegeben.

2. Löschen Sie alle Ressourcen, die Sie nicht mehr benötigen.

   > **Note**
   >
   > Durch das Löschen einer Pipeline werden nicht die in der Pipeline verwendeten Ressourcen gelöscht, z. B. die CodeDeploy Elastic Beanstalk Beanstalk-Anwendung, mit der Sie Ihren Code bereitgestellt haben, oder, falls Sie Ihre Pipeline von der CodePipeline Konsole aus erstellt haben, der Amazon S3 S3-Bucket, der zum Speichern der Artefakte Ihrer Pipelines CodePipeline erstellt wurde. Stellen Sie sicher, dass Sie die Ressourcen löschen, die nicht mehr benötigt werden, sodass keine zukünftigen Kosten für Sie entstehen. Wenn Sie beispielsweise die Konsole zum ersten Mal verwenden, um eine Pipeline zu erstellen, CodePipeline erstellt sie einen Amazon S3 S3-Bucket, in dem alle Artefakte für all Ihre Pipelines gespeichert werden. Wenn Sie alle Pipelines gelöscht haben, führen Sie die Schritte unter Löschen eines Buckets aus.

Erstellen Sie eine Pipeline CodePipeline , in der Ressourcen von einem anderen AWS Konto verwendet werden

Sie möchten möglicherweise eine Pipeline erstellen, die Ressourcen verwendet, die von einem anderen AWS -Konto erstellt oder verwaltet werden. Möglicherweise möchten Sie beispielsweise ein Konto für Ihre Pipeline und ein anderes für Ihre CodeDeploy Ressourcen verwenden.

> **Note**
>
> Beim Erstellen einer Pipeline mit Aktionen aus mehreren Konten müssen Sie Ihre Aktionen so konfigurieren, dass mit ihnen innerhalb der Einschränkungen von kontoübergreifenden Pipelines weiterhin auf Artefakte zugegriffen werden kann. Die folgenden Einschränkungen gelten für kontoübergreifende Aktionen:
>
> - Im Allgemeinen kann eine Aktion nur ein Artefakt verbrauchen, wenn:
>   - Die Aktion befindet sich im selben Konto wie das Pipeline-Konto ODER
>   - das Artefakt im Pipeline-Konto für eine Aktion in einem anderen Konto erstellt wurde ODER
>   - Das Artefakt wurde durch eine frühere Aktion im selben Konto wie die Aktion erzeugt
>
>   Anders gesagt, ist es nicht möglich, ein Artefakt von einem Konto an ein anderes Konto zu übergeben, wenn keines der Konten ein Pipeline-Konto ist.
> - Für die folgenden Aktionstrypen werden keine kontoübergreifenden Aktionen unterstützt:
>   - Jenkins-Build-Aktionen

In diesem Beispiel müssen Sie einen zu verwendenden Schlüssel AWS Key Management Service (AWS KMS) erstellen, den Schlüssel zur Pipeline hinzufügen und Kontorichtlinien und Rollen einrichten, um den kontoübergreifenden Zugriff zu ermöglichen. Für einen AWS KMS-Schlüssel können Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN verwenden.

> **Note**
>
> Aliase werden nur in dem Konto erkannt, das den KMS-Schlüssel erstellt hat. Für kontoübergreifende Aktionen können Sie zum Identifizieren des Schlüssels nur die Schlüssel-ID oder den Schlüssel-ARN vewenden. Bei kontoübergreifenden Aktionen wird die Rolle des anderen Kontos (AccountB) verwendet, sodass bei Angabe der Schlüssel-ID der Schlüssel des anderen Kontos (AccountB) verwendet wird.

In dieser Anleitung und den entsprechenden Beispielen ist `AccountA` das Konto, das ursprünglich zur Erstellung der Pipeline verwendet wurde. Es hat Zugriff auf den Amazon S3 S3-Bucket, der zum Speichern von Pipeline-Artefakten verwendet wird, und auf die Servicerolle, die von verwendet wird

AWS CodePipeline. *AccountB* ist das Konto, mit dem ursprünglich die CodeDeploy Anwendung, die Bereitstellungsgruppe und die Servicerolle erstellt wurden, die von verwendet wurden CodeDeploy.

Damit AccountA eine Pipeline bearbeiten kann, um die von AccountB erstellte CodeDeploy Anwendung zu verwenden, muss AccountA:

- Die ARN oder Konto-ID von *AccountB* anfordern (in dieser Anleitung ist die ID von *AccountB* *012ID_ACCOUNT_B*).
- *Erstellen oder verwenden Sie einen vom AWS KMS Kunden verwalteten Schlüssel in der Region für die Pipeline und gewähren Sie der Servicerolle (CodePipeline_Service_Role) und AccountB Berechtigungen zur Verwendung dieses Schlüssels.*
- Erstellen Sie eine Amazon S3 S3-Bucket-Richtlinie, die *AccountB* Zugriff auf den Amazon S3 S3-Bucket gewährt (z. B. *codepipeline-us-east-2-1234567890*).
- Erstellen Sie eine Richtlinie, die es *AccountA* ermöglicht, eine von *AccountB* konfigurierte Rolle anzunehmen, und fügen Sie diese Richtlinie der Servicerolle (*CodePipeline_Service_Role*) hinzu.
- Bearbeiten Sie die Pipeline so, dass der vom Kunden verwaltete AWS KMS Schlüssel anstelle des Standardschlüssels verwendet wird.

Damit *AccountB* einer in *AccountA* erstellten Pipeline Zugriff auf seine Ressourcen gestattet, muss *AccountB*:

- Die ARN oder Konto-ID von *AccountA* anfordern (in dieser Anleitung ist die ID von *AccountA* *012ID_ACCOUNT_A*).
- Erstellen Sie eine Richtlinie, die auf die Amazon EC2 EC2-Instance-Rolle angewendet wird CodeDeploy , für die der Zugriff auf den Amazon S3 S3-Bucket (*codepipeline-us-east-2-1234567890*) konfiguriert ist.
- *Erstellen Sie eine Richtlinie, die auf die für CodeDeploy die Amazon EC2 EC2-Instance-Rolle konfigurierte Amazon EC2-Instance-Rolle angewendet wird und den Zugriff auf den vom AWS KMS Kunden verwalteten Schlüssel ermöglicht, der zur Verschlüsselung der Pipeline-Artefakte in AccountA verwendet wird.*
- Konfigurieren und fügen Sie eine IAM-Rolle (*CrossAccount_Role*) mit einer Vertrauensbeziehungsrichtlinie hinzu, die es der CodePipeline Servicerolle in *AccountA* ermöglicht, die Rolle zu übernehmen.

- *Erstellen Sie eine Richtlinie, die den Zugriff auf die Bereitstellungsressourcen ermöglicht, die für die Pipeline benötigt werden, und hängen Sie sie an _Role an. CrossAccount*
- *Erstellen Sie eine Richtlinie, die den Zugriff auf den Amazon S3 S3-Bucket (codepipeline-us-east-2-1234567890) ermöglicht, und hängen Sie sie an _Role an. CrossAccount*

Themen

Voraussetzung: Erstellen eines AWS KMS -Verschlüsselungsschlüssels

Vom Kunden verwaltete Schlüssel sind spezifisch für eine Region, ebenso wie alle Schlüssel. AWS KMS Sie müssen Ihren vom Kunden verwalteten AWS KMS Schlüssel in derselben Region erstellen, in der die Pipeline erstellt wurde (z. B.`us-east-2`).

Um einen vom Kunden verwalteten Schlüssel zu erstellen AWS KMS

1. Melden Sie sich AWS Management Console mit *AccountA* an und öffnen Sie die AWS KMS Konsole.
2. Wählen Sie links Customer managed keys (Vom Kunden verwaltete Schlüssel) aus.
3. Klicken Sie auf Create key. Lassen Sie unter Configure key (Schlüssel konfigurieren) die Standardeinstellung Symmetric (Symmetrisch) ausgewählt und wählen Sie Next (Weiter) aus.
4. Geben Sie im Feld Alias einen Alias ein, der für diesen Schlüssel verwendet werden soll (z. B. *PipelineName-Key*). Geben Sie optional eine Beschreibung und die Tags für den Schlüssel ein und wählen Sie dann Next (Weiter) aus.
5. Wählen Sie unter Wichtige Administratorberechtigungen definieren die Rolle oder Rollen aus, die Sie als Administratoren für diesen Schlüssel verwenden möchten, und klicken Sie dann auf Weiter.
6. Wählen Sie unter Schlüsselverwendungsberechtigungen definieren unter Dieses Konto den Namen der Servicerolle für die Pipeline aus (z. B. CodePipeline _Service_Role). Wählen Sie unter Andere AWS Konten die Option Weiteres Konto hinzufügen aus. AWS Geben Sie die Konto-ID für *AccountB* ein, um den ARN abzuschließen, und wählen Sie Next (Weiter) aus.

7. Überprüfen Sie die Richtlinie in Preview Key Policy (Vorschau der Schlüsselrichtlinie) und wählen Sie dann Finish (Beenden) aus.
8. Wählen Sie aus der Liste der Schlüssel den Alias Ihres Schlüssels aus und kopieren Sie dessen ARN (z. B. ***arn:aws:kms:us-east-2:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE***). Sie benötigen diese Informationen, wenn Sie Ihre Pipeline bearbeiten und Richtlinien konfigurieren.

Schritt 1: Einrichten von Kontorichtlinien und Rollen

Nachdem Sie den AWS KMS Schlüssel erstellt haben, müssen Sie Richtlinien erstellen und anhängen, die den kontoübergreifenden Zugriff ermöglichen. Dies erfordert Aktionen sowohl von *AccountA* als auch von *AccountB*.

Themen

- Konfigurieren von Richtlinien und Rollen in dem Konto, das die Pipeline erstellen wird (AccountA)
- Konfigurieren Sie Richtlinien und Rollen in dem Konto, dem die AWS Ressource gehört (AccountB)

Konfigurieren von Richtlinien und Rollen in dem Konto, das die Pipeline erstellen wird (***AccountA***)

Um eine Pipeline zu erstellen, die CodeDeploy Ressourcen verwendet, die mit einem anderen AWS Konto verknüpft sind, muss *AccountA* Richtlinien sowohl für den Amazon S3 S3-Bucket, der zum Speichern von Artefakten verwendet wird, als auch für die Servicerolle für CodePipeline konfigurieren.

Um eine Richtlinie für den Amazon S3 S3-Bucket zu erstellen, der Zugriff auf AccountB (Konsole) gewährt

1. Melden Sie sich AWS Management Console mit *AccountA* an und öffnen Sie die Amazon S3 S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie in der Liste der Amazon S3 S3-Buckets den Amazon S3 S3-Bucket aus, in dem Artefakte für Ihre Pipelines gespeichert sind. *Dieser Bucket ist benanntcodepipeline-region-1234567EXAMPLE, wobei Region die Region ist, in der Sie die AWS Pipeline erstellt haben, und 1234567EXAMPLE eine zehnstellige Zufallszahl ist, die sicherstellt, dass der Bucket-Name eindeutig ist (z. B. -2-1234567890). codepipeline-us-east*

3. Wählen Sie auf der Detailseite für den Amazon S3 S3-Bucket die Option Eigenschaften aus.
4. Erweitern Sie im Eigenschaftenbereich das Feld Permissions und wählen Sie Add bucket policy aus.

> **Note**
>
> Wenn Ihrem Amazon S3 S3-Bucket bereits eine Richtlinie zugeordnet ist, wählen Sie Bucket-Richtlinie bearbeiten aus. Danach können Sie die Anweisungen im folgenden Beispiel zur vorhandenen Richtlinie hinzufügen. Um eine neue Richtlinie hinzuzufügen, wählen Sie den Link und folgen Sie den Anweisungen im AWS Policy Generator. Weitere Informationen finden Sie unter Überblick über die IAM-Richtlinien.

5. Geben Sie die folgende Richtlinie in das Fenster Bucket Policy Editor ein. Dies ermöglicht *AccountB* den Zugriff auf die Pipeline-Artefakte und bietet *AccountB* die Möglichkeit zum Hinzufügen von Ausgabe-Artefakten, falls diese von einer Aktion (z. B. eine benutzerdefinierte Quell- oder Build-Aktion) erstellt werden.

   Im folgenden Beispiel ist für *AccountB* der ARN *012ID_ACCOUNT_B*. Der ARN für den Amazon S3 S3-Bucket lautet *codepipeline-us-east-2-1234567890*. Ersetzen Sie diese ARNs durch den ARN für das Konto, dem Sie Zugriff gewähren möchten, und den ARN für den Amazon S3 S3-Bucket:

```
{
 "Version": "2012-10-17",
 "Id": "SSEAndSSLPolicy",
 "Statement": [
{
  "Sid": "DenyUnEncryptedObjectUploads",
  "Effect": "Deny",
  "Principal": "*",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
  "Condition": {
 "StringNotEquals": {
 "s3:x-amz-server-side-encryption": "aws:kms"
 }
   }
 },
{
  "Sid": "DenyInsecureConnections",
```

```
    "Effect": "Deny",
    "Principal": "*",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
    "Condition": {
   "Bool": {
      "aws:SecureTransport": false
    }
   }
    },
  {
    "Sid": "",
    "Effect": "Allow",
    "Principal": {
   "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
   },
    "Action": [
             "s3:Get*",
             "s3:Put*"
         ],
    "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
  },
  {
    "Sid": "",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
    },
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890"
  }
    ]
}
```

6. Wählen Sie Save und schließen Sie dann den Richtlinien-Editor.
7. Wählen Sie Speichern, um die Berechtigungen für den Amazon S3 S3-Bucket zu speichern.

Um eine Richtlinie für die Servicerolle für CodePipeline (Konsole) zu erstellen

1. Melden Sie sich AWS Management Console mit *AccountA* an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen aus.

3. Wählen Sie in der Rollenliste unter Rollenname den Namen der Servicerolle für aus. CodePipeline
4. Wählen Sie in der Registerkarte Permissions (Berechtigungen) die Option Add inline policy (Inline-Richtlinie hinzufügen).
5. Wählen Sie die Registerkarte JSON und geben Sie die folgende Richtlinie ein, damit *AccountB* die Rolle übernehmen kann. Im folgenden Beispiel ist *012ID_ACCOUNT_B* der ARN für *AccountB*:

```
{
   "Version": "2012-10-17",
   "Statement": {
       "Effect": "Allow",
       "Action": "sts:AssumeRole",
       "Resource": [
           "arn:aws:iam::012ID_ACCOUNT_B:role/*"
       ]
   }
 }
```

6. Wählen Sie Richtlinie prüfen.
7. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Richtlinie erstellen aus.

Konfigurieren Sie Richtlinien und Rollen in dem Konto, dem die AWS Ressource gehört (*AccountB*)

Wenn Sie eine Anwendungs-, Bereitstellungs- und Bereitstellungsgruppe in erstellen CodeDeploy, erstellen Sie auch eine Amazon EC2 EC2-Instance-Rolle. (Diese Rolle wird für Sie erstellt, wenn Sie den Assistenten für die Ausführung der Bereitstellungsanleitung verwenden. Sie können sie jedoch auch manuell erstellen.) Damit eine in *AccountA* erstellte Pipeline CodeDeploy Ressourcen verwenden kann, die in *AccountB* erstellt wurden, müssen Sie:

- Konfigurieren Sie eine Richtlinie für die Instance-Rolle, die ihr den Zugriff auf den Amazon S3 S3-Bucket ermöglicht, in dem Pipeline-Artefakte gespeichert sind.
- Eine zweite Rolle in *AccountB* erstellen, die für den kontoübergreifenden Zugriff konfiguriert ist.

  Diese zweite Rolle muss nicht nur Zugriff auf den Amazon S3 S3-Bucket in *AccountA* haben, sie muss auch eine Richtlinie enthalten, die den Zugriff auf die CodeDeploy Ressourcen ermöglicht,

und eine Vertrauensbeziehungsrichtlinie, die es der CodePipeline Servicerolle in *AccountA* ermöglicht, die Rolle zu übernehmen.

> **Note**
>
> Diese Richtlinien sind spezifisch für die Einrichtung von CodeDeploy Ressourcen, die in einer Pipeline verwendet werden sollen, die mit einem anderen AWS Konto erstellt wurde. Für andere AWS Ressourcen sind Richtlinien erforderlich, die auf ihre jeweiligen Ressourcenanforderungen zugeschnitten sind.

Um eine Richtlinie für die Amazon EC2 EC2-Instance-Rolle zu erstellen, die für CodeDeploy (Konsole) konfiguriert ist

1. Melden Sie sich AWS Management Console mit *AccountB* an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen aus.
3. Wählen Sie in der Rollenliste unter Rollenname den Namen der Servicerolle aus, die als Amazon EC2 EC2-Instance-Rolle für die CodeDeploy Anwendung verwendet wird. Der Rollenname kann unterschiedlich sein. Von einer Bereitstellungsgruppe kann mehr als eine Instance-Rolle verwendet werden. Weitere Informationen finden Sie unter Erstellen eines IAM-Instance-Profils für Ihre Amazon EC2 EC2-Instances.
4. Wählen Sie in der Registerkarte Permissions (Berechtigungen) die Option Add inline policy (Inline-Richtlinie hinzufügen).
5. Wählen Sie die Registerkarte JSON und geben Sie die folgende Richtlinie ein, um Zugriff auf den Amazon S3 S3-Bucket zu gewähren, der von *AccountA* zum Speichern von Artefakten für Pipelines verwendet wird (in diesem Beispiel *codepipeline-us-east-2-1234567890*):

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "s3:Get*"
       ],
       "Resource": [
         "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
```

```
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::codepipeline-us-east-2-1234567890"
      ]
    }
  ]
}
```

6. Wählen Sie Richtlinie prüfen.
7. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Richtlinie erstellen aus.
8. Erstellen Sie eine zweite Richtlinie dafür, AWS KMS wo sich der ARN des vom Kunden verwalteten Schlüssels ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** befindet, der in *AccountA* erstellt und so konfiguriert ist, dass *AccountB ihn* verwenden kann:

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
           "kms:DescribeKey",
           "kms:GenerateDataKey*",
           "kms:Encrypt",
           "kms:ReEncrypt*",
           "kms:Decrypt"
         ],
        "Resource": [
           "arn:aws:kms:us-
east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE"
         ]
      }
   ]
}
```

> **Important**
>
> Sie müssen die Konto-ID von *AccountA* in dieser Richtlinie als Teil des Ressourcen-ARN für den AWS KMS Schlüssel verwenden, wie hier gezeigt, sonst funktioniert die Richtlinie nicht.

9. Wählen Sie Richtlinie prüfen.
10. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Richtlinie erstellen aus.

Erstellen Sie nun eine IAM-Rolle, die für den kontoübergreifenden Zugriff verwendet werden soll, und konfigurieren Sie sie so, dass die CodePipeline Servicerolle in *AccountA* die Rolle übernehmen kann. Diese Rolle muss Richtlinien enthalten, die den Zugriff auf die CodeDeploy Ressourcen und den Amazon S3 S3-Bucket ermöglichen, der zum Speichern von Artefakten in *AccountA* verwendet wird.

Um die kontoübergreifende Rolle in IAM zu konfigurieren

1. Melden Sie sich AWS Management Console mit *AccountB* an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam.
2. Wählen Sie im Navigationsbereich Rollen aus. Wählen Sie Rolle erstellen aus.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option Weiteres AWS -Konto aus. Geben Sie unter Konten angeben, die diese Rolle verwenden können, im Feld Konto-ID die AWS Konto-ID für das Konto ein, das die Pipeline in CodePipeline (*AccountA*) erstellen soll, und wählen Sie dann Weiter: Berechtigungen aus.

   > **Important**
   >
   > In diesem Schritt wird die Vertrauensstellungsrichtlinie zwischen *AccountA* und *AccountB* erstellt. *Dadurch wird jedoch Zugriff auf das Konto auf Root-Ebene gewährt und es wird CodePipeline empfohlen, diesen Zugriff auf die CodePipeline Servicerolle in AccountA zu beschränken.* Folgen Sie Schritt 16, um die Berechtigungen einzuschränken.

4. Wählen Sie unter Richtlinien zum Anhängen von Berechtigungen die Option ReadOnlyAccess AmazonS3 und dann Weiter: Tags aus.

> **Note**
>
> Dies ist nicht die Richtlinie, die Sie verwenden werden. Sie müssen eine Richtlinie auswählen, um den Assistenten abzuschließen.

5. Wählen Sie Weiter: Prüfen aus. Geben Sie im Feld Rollenname einen Namen für diese Rolle ein (z. B. *CrossAccount_Role*). Sie können dieser Rolle einen beliebigen Namen geben, solange sie den Benennungskonventionen in IAM entspricht. Geben Sie der Rolle einen Namen, der ihren Zweck eindeutig beschreibt. Wählen Sie Create Role (Rolle erstellen) aus.
6. Wählen Sie aus der Rollenliste die Rolle aus, die Sie gerade erstellt haben (z. B. *CrossAccount_Role), um die **Übersichtsseite** für diese Rolle* zu öffnen.
7. Wählen Sie in der Registerkarte Permissions (Berechtigungen) die Option Add inline policy (Inline-Richtlinie hinzufügen).
8. Wählen Sie die Registerkarte JSON und geben Sie die folgende Richtlinie ein, um den Zugriff auf CodeDeploy Ressourcen zu gewähren:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:GetApplicationRevision",
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource": "*"
    }
  ]
}
```

9. Wählen Sie Richtlinie prüfen.
10. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Richtlinie erstellen aus.
11. Wählen Sie in der Registerkarte Permissions (Berechtigungen) die Option Add inline policy (Inline-Richtlinie hinzufügen).

12. Wählen Sie die Registerkarte JSON und geben Sie die folgende Richtlinie ein, damit diese Rolle Eingabeartefakte aus dem Amazon S3 S3-Bucket in *AccountA* abrufen und Ausgabeartefakte dort ablegen kann:

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "s3:GetObject*",
         "s3:PutObject",
         "s3:PutObjectAcl"
       ],
       "Resource": [
         "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
       ]
     }
   ]
}
```

13. Wählen Sie Richtlinie prüfen.
14. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Richtlinie erstellen aus.
15. Suchen Sie auf der Registerkarte Berechtigungen ReadOnlyAccess in der Liste der Richtlinien unter Richtlinienname nach AmazonS3 und wählen Sie das Löschsymbol (X) neben der Richtlinie aus. Wählen Sie nach Aufforderung Detach aus.
16. Wählen Sie die Registerkarte Vertrauensverhältnis und dann Vertrauensrichtlinie bearbeiten aus. Wählen Sie in der linken Spalte die Option Principal hinzufügen aus. Wählen Sie als Principaltyp die Option IAM-Rollen aus, und geben Sie dann den ARN für die CodePipeline Servicerolle in *AccountA* ein. Entfernen Sie es **`arn:aws:iam::Account_A:root`** aus der Liste für AWS Principals und wählen Sie dann Richtlinie aktualisieren aus.

Schritt 2: Bearbeiten der Pipeline

Sie können die CodePipeline Konsole nicht verwenden, um eine Pipeline zu erstellen oder zu bearbeiten, die Ressourcen verwendet, die mit einem anderen AWS Konto verknüpft sind. Sie können jedoch die Konsole verwenden, um die allgemeine Struktur der Pipeline zu erstellen und dann die Pipeline AWS CLI zu bearbeiten und diese Ressourcen hinzuzufügen. Alternativ können Sie die Struktur einer vorhandenen Pipeline verwenden und dieser die Ressourcen manuell hinzufügen.

Um die Ressourcen hinzuzufügen, die einem anderen AWS Konto zugeordnet sind (AWS CLI)

1. Führen Sie an einem Terminal (Linux, macOS oder Unix) oder einer Befehlszeile (Windows) den get-pipeline Befehl für die Pipeline aus, zu der Sie Ressourcen hinzufügen möchten. Kopieren Sie die Ausgabe des Befehls in eine JSON-Datei. Für eine Pipeline mit dem Namen würden Sie MyFirstPipeline beispielsweise etwas Ähnliches wie das Folgende eingeben:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Die Ausgabe wird an die Datei *pipeline.json* gesendet.

2. Öffnen Sie die JSON-Datei in einem beliebigen Texteditor. Fügen Sie anschließend `"type": "S3"` im Artefaktspeicher den KMS-Verschlüsselungsschlüssel, die ID und die Typinformationen hinzu, wobei *codepipeline-us-east-2-1234567890* der Name des Amazon S3 S3-Buckets ist, der zum Speichern von Artefakten für die Pipeline verwendet wird, und der ARN des vom Kunden verwalteten Schlüssels ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** ist, den Sie gerade erstellt haben:

   ```
   {
     "artifactStore”: {
       "location": "codepipeline-us-east-2-1234567890",
       "type": "S3",
       "encryptionKey": {
         "id": "arn:aws:kms:us-
   east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE",
         "type": "KMS"
       }
     },
   ```

3. Fügen Sie in einer Phase eine Bereitstellungsaktion hinzu, um die mit *AccountB* verknüpften CodeDeploy Ressourcen zu verwenden, einschließlich der `roleArn` Werte für die von Ihnen erstellte kontenübergreifende Rolle (*CrossAccount_Role*).

   Das folgende Beispiel zeigt JSON, das eine Bereitstellungsaktion mit dem Namen hinzufügt. *ExternalDeploy* Es verwendet die in *AccountB* erstellten CodeDeploy Ressourcen in einer Phase namens *Staging*. Im folgenden Beispiel lautet der ARN für *AccountB* *012ID_ACCOUNT_B*:

   ```
   ,
   ```

```
            {
                "name": "Staging",
                "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "MyAppBuild"
                            }
                        ],
                        "name": "ExternalDeploy",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "AccountBApplicationName",
                            "DeploymentGroupName": "AccountBApplicationGroupName"
                        },
                        "runOrder": 1,
                        "roleArn":
"arn:aws:iam::012ID_ACCOUNT_B:role/CrossAccount_Role"
                    }
                ]
            }
```

> **Note**
>
> Dies ist nicht der JSON-Code für die gesamte Pipeline. Es handelt sich nur um die Struktur für die Aktion in einer Stufe.

4. Sie müssen die `metadata`-Zeilen aus der Datei entfernen, damit der Befehl update-pipeline sie verwenden kann. Entfernen Sie den Abschnitt aus der Pipeline-Struktur in der JSON-Datei (die `"metadata": { }`-Zeilen und die Fehler `"created"`, `"pipelineARN"` und `"updated"`).

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

```
"metadata": {
  "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
  "created": "date",
```

```
  "updated": "date"
  }
```

Speichern Sie die Datei.

5. Führen Sie den Befehl update-pipeline aus, um die Änderungen zu übernehmen. Geben Sie die Pipeline-JSON-Datei dabei folgendermaßen an:

> **Important**
>
> Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

```
aws codepipeline update-pipeline --cli-input-json file://pipeline.json
```

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

Um die Pipeline zu testen, die Ressourcen verwendet, die einem anderen Konto zugeordnet sind AWS

1. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer Befehlszeile (Windows) aus und geben Sie dabei den Namen der Pipeline an, ähnlich wie im Folgenden: start-pipeline-execution

```
aws codepipeline start-pipeline-execution --name MyFirstPipeline
```

Weitere Informationen finden Sie unter Manuelles Starten einer Pipeline.

2. Melden Sie sich AWS Management Console mit *AccountA* an und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.

3. Wählen Sie im Feld Name den Namen der Pipeline, die Sie soeben bearbeitet haben. Auf diese Weise wird eine detaillierte Ansicht der Pipeline geöffnet (einschließlich des Status der einzelnen Aktionen in den einzelnen Stufen der Pipeline).
4. Sehen Sie sich den Fortschritt in der Pipeline an. Warten Sie auf eine Erfolgsmeldung für die Aktion, die die mit einem anderen AWS Konto verknüpfte Ressource verwendet.

> **Note**
>
> Sie erhalten eine Fehlermeldung, wenn Sie versuchen, Details für eine Aktion anzuzeigen, während Sie mit *AccountA* angemeldet sind. Melden Sie sich ab und melden Sie sich dann mit *AccountB* an, um die Bereitstellungsdetails unter CodeDeploy einzusehen.

Migrieren Sie Abfrage-Pipelines, um die ereignisbasierte Änderungserkennung zu nutzen

AWS CodePipeline unterstützt die vollständige, end-to-end kontinuierliche Bereitstellung, was das Starten Ihrer Pipeline bei jeder Codeänderung einschließt. Es gibt zwei unterstützte Methoden, um Ihre Pipeline nach einer Codeänderung zu starten: ereignisbasierte Änderungserkennung und Abfrage. Wir empfehlen, die ereignisbasierte Änderungserkennung für Pipelines zu verwenden.

Verwenden Sie die hier aufgeführten Verfahren, um Ihre Abfrage-Pipelines auf die ereignisbasierte Methode zur Änderungserkennung für Ihre Pipeline zu migrieren (zu aktualisieren).

Die empfohlene Methode zur ereignisbasierten Änderungserkennung für Pipelines wird durch die Pipeline-Quelle bestimmt, z. B. CodeCommit In diesem Fall müsste die Polling-Pipeline beispielsweise auf die ereignisbasierte Änderungserkennung mit migrieren. EventBridge

Wie migriert man Polling-Pipelines

Um Abfrage-Pipelines zu migrieren, ermitteln Sie Ihre Abfrage-Pipelines und legen Sie dann die empfohlene Methode zur ereignisbasierten Änderungserkennung fest:

- Verwenden Sie die unter beschriebenen Schritte, um Ihre Abfrage-Pipelines Polling-Pipelines in Ihrem Konto anzeigen zu ermitteln.
- Suchen Sie in der Tabelle nach Ihrem Pipeline-Quelltyp und wählen Sie dann das Verfahren mit der Implementierung aus, die Sie für die Migration Ihrer Polling-Pipeline verwenden möchten. Jeder Abschnitt enthält mehrere Migrationsmethoden, z. B. die Verwendung der CLI oder AWS CloudFormation.

Migrieren von Pipelines zur empfohlenen Methode zur Änderungserkennung		
Pipeline-Quelle	Empfohlene ereignisbasierte Erkennungsmethode	Migrationsverfahren
AWS CodeCommit	EventBridge (empfohlen).	Siehe Migrieren Sie Abfrage-Pipelines mit einer Quelle CodeCommit .
Amazon S3	EventBridge und Bucket für Ereignisbenachrichtigungen aktiviert (empfohlen).	Siehe Migrieren Sie Abfrage-Pipelines mit einer S3-Quelle, die für Ereignisse aktiviert ist.
Amazon S3	EventBridge und eine AWS CloudTrail Spur.	Siehe Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail .
GitHub Version 1	Verbindungen (empfohlen)	Siehe Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Verbindungen.
GitHub Version 1	Webhooks	Siehe Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Webhooks.

Important

Für entsprechende Aktualisierungen der Konfiguration von Pipeline-Aktionen, wie z. B. Pipelines mit einer Aktion der GitHub Version 1, müssen Sie den `PollForSourceChanges` Parameter in der Konfiguration Ihrer Source-Aktion explizit auf false setzen, um zu verhindern, dass eine Pipeline Abfragen durchführt. Daher ist es möglich, eine Pipeline fälschlicherweise sowohl mit ereignisbasierter Änderungserkennung als auch mit Abfrage zu konfigurieren, indem Sie beispielsweise eine EventBridge Regel konfigurieren und gleichzeitig den Parameter weglassen. `PollForSourceChanges` Dies führt zu duplizierten Pipelineausführungen, wobei die Pipeline bei der maximalen Gesamtzahl von Abfrage-Pipelines berücksichtigt wird. Standardmäßig handelt es sich hierbei um einen sehr viel

niedrigeren Wert als für ereignisbasierte Pipelines. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline.

Polling-Pipelines in Ihrem Konto anzeigen

Verwenden Sie als ersten Schritt eines der folgenden Skripts, um zu ermitteln, welche Pipelines in Ihrem Konto für Abfragen konfiguriert sind. Dies sind die Pipelines für die Migration zur ereignisbasierten Änderungserkennung.

Abfrage-Pipelines in Ihrem Konto anzeigen (Skript)

Gehen Sie wie folgt vor, um mithilfe eines Skripts die Pipelines in Ihrem Konto zu ermitteln, die Polling verwenden.

1. Öffnen Sie ein Terminalfenster und führen Sie dann einen der folgenden Schritte aus:

   - Führen Sie den folgenden Befehl aus, um ein neues Skript mit dem Namen PollingPipelinesExtractor.sh zu erstellen.

     ```
     vi PollingPipelinesExtractor.sh
     ```

   - Um ein Python-Skript zu verwenden, führen Sie den folgenden Befehl aus, um ein neues Python-Skript namens PollingPipelinesExtractor.py zu erstellen.

     ```
     vi PollingPipelinesExtractor.py
     ```

2. Kopieren Sie den folgenden Code und fügen Sie ihn in das PollingPipelinesExtractorSkript ein. Führen Sie eine der folgenden Aktionen aus:

   - Kopieren Sie den folgenden Code und fügen Sie ihn in das PollingPipelinesExtractor.sh-Skript ein.

     ```
     #!/bin/bash

     set +x

     POLLING_PIPELINES=()
     LAST_EXECUTED_DATES=()
     NEXT_TOKEN=null
     ```

```
HAS_NEXT_TOKEN=true
if [[ $# -eq 0 ]] ; then
    echo 'Please provide region name'
    exit 0
fi
REGION=$1


while [ "$HAS_NEXT_TOKEN" != "false" ]; do
    if [ "$NEXT_TOKEN" != "null" ];
        then
            LIST_PIPELINES_RESPONSE=$(aws codepipeline list-pipelines --region
 $REGION --next-token $NEXT_TOKEN)
        else
            LIST_PIPELINES_RESPONSE=$(aws codepipeline list-pipelines --region
 $REGION)
    fi
    LIST_PIPELINES=$(jq -r '.pipelines[].name' <<< "$LIST_PIPELINES_RESPONSE")
    NEXT_TOKEN=$(jq -r '.nextToken' <<< "$LIST_PIPELINES_RESPONSE")
    if [ "$NEXT_TOKEN" == "null" ];
        then
            HAS_NEXT_TOKEN=false
    fi

    for pipline_name in $LIST_PIPELINES
    do
        PIPELINE=$(aws codepipeline get-pipeline --name $pipline_name --region
 $REGION)
        HAS_POLLABLE_ACTIONS=$(jq '.pipeline.stages[].actions[] |
 select(.actionTypeId.category == "Source") | select(.actionTypeId.owner
 == ("ThirdParty","AWS")) | select(.actionTypeId.provider ==
 ("GitHub","S3","CodeCommit")) | select(.configuration.PollForSourceChanges ==
 ("true",null))' <<< "$PIPELINE")
        if [ ! -z "$HAS_POLLABLE_ACTIONS" ];
        then
            POLLING_PIPELINES+=("$pipline_name")
            PIPELINE_EXECUTIONS=$(aws codepipeline list-pipeline-executions --
pipeline-name $pipline_name --region $REGION)
            LAST_EXECUTION=$(jq -r '.pipelineExecutionSummaries[0]' <<<
 "$PIPELINE_EXECUTIONS")
            if [ "$LAST_EXECUTION" != "null" ];
                then
                    LAST_EXECUTED_TIMESTAMP=$(jq -r '.startTime' <<<
 "$LAST_EXECUTION")
```

```
                    LAST_EXECUTED_DATE="$(date -r ${LAST_EXECUTED_TIMESTAMP%.*})"
                else
                    LAST_EXECUTED_DATE="Not executed in last year"
            fi
            LAST_EXECUTED_DATES+=("$LAST_EXECUTED_DATE")
        fi
    done

done

fileName=$REGION-$(date +%s)
printf "| %-30s | %-30s |\n" "Polling Pipeline Name" "Last Executed Time"
printf "| %-30s | %-30s |\n" "_____________________" "___________________"
for i in "${!POLLING_PIPELINES[@]}"; do
  printf "| %-30s | %-30s |\n" "${POLLING_PIPELINES[i]}"
 "${LAST_EXECUTED_DATES[i]}"
  printf "${POLLING_PIPELINES[i]}," >> $fileName.csv
done

printf "\nSaving Polling Pipeline Names to file $fileName.csv."
```

- Kopieren Sie den folgenden Code und fügen Sie ihn in das PollingPipelinesExtractor.py-Skript ein.

```
import boto3
import sys
import time
import math

hasNextToken = True
nextToken = ""
pollablePipelines = []
lastExecutedTimes = []
if len(sys.argv) == 1:
    raise Exception("Please provide region name.")
session = boto3.Session(profile_name='default', region_name=sys.argv[1])
codepipeline = session.client('codepipeline')

def is_pollable_action(action):
    actionTypeId = action['actionTypeId']
    configuration = action['configuration']
    return actionTypeId['owner'] in {"AWS", "ThirdParty"}
 and actionTypeId['provider'] in {"GitHub", "CodeCommit",
```

```
 "S3"} and ('PollForSourceChanges' not in configuration or
 configuration['PollForSourceChanges'] == 'true')

def has_pollable_actions(pipeline):
    hasPollableAction = False
    pipelineDefinition = codepipeline.get_pipeline(name=pipeline['name'])
['pipeline']
    for action in pipelineDefinition['stages'][0]['actions']:
        hasPollableAction = is_pollable_action(action)
        if hasPollableAction:
            break
    return hasPollableAction

def get_last_executed_time(pipelineName):

 pipelineExecutions=codepipeline.list_pipeline_executions(pipelineName=pipelineName)
['pipelineExecutionSummaries']
    if pipelineExecutions:
        return pipelineExecutions[0]['startTime'].strftime("%A %m/%d/%Y, %H:%M:
%S")
    else:
        return "Not executed in last year"

while hasNextToken:
    if nextToken=="":
        list_pipelines_response = codepipeline.list_pipelines()
    else:
        list_pipelines_response =
 codepipeline.list_pipelines(nextToken=nextToken)
    if 'nextToken' in list_pipelines_response:
        nextToken = list_pipelines_response['nextToken']
    else:
        hasNextToken= False
    for pipeline in list_pipelines_response['pipelines']:
        if has_pollable_actions(pipeline):
            pollablePipelines.append(pipeline['name'])
            lastExecutedTimes.append(get_last_executed_time(pipeline['name']))

fileName="{region}-
{timeNow}.csv".format(region=sys.argv[1],timeNow=math.trunc(time.time()))
file = open(fileName, 'w')

print ("{:<30} {:<30} {:<30}".format('Polling Pipeline Name', '|','Last Executed
 Time'))
```

```
print ("{:<30} {:<30} {:<30}".format('_____________________',
 '|','___________________'))
for i in range(len(pollablePipelines)):
    print("{:<30} {:<30} {:<30}".format(pollablePipelines[i], '|',
 lastExecutedTimes[i]))
    file.write("{pipeline},".format(pipeline=pollablePipelines[i]))
file.close()
print("\nSaving Polling Pipeline Names to file
 {fileName}".format(fileName=fileName))
```

3. Für jede Region, in der Sie Pipelines haben, müssen Sie das Skript für diese Region ausführen. Gehen Sie wie folgt vor, um das Skript auszuführen:

   - Führen Sie den folgenden Befehl aus, um das Skript mit dem Namen „PollingPipelinesExtractor.sh“ auszuführen. In diesem Beispiel ist die Region us-west-2.

     ```
     ./PollingPipelinesExtractor.sh us-west-2
     ```

   - Führen Sie für das Python-Skript den folgenden Befehl aus, um das Python-Skript mit dem Namen PollingPipelinesExtractor .py auszuführen. In diesem Beispiel ist die Region us-west-2.

     ```
     python3 PollingPipelinesExtractor.py us-west-2
     ```

   In der folgenden Beispielausgabe des Skripts gab die Region us-west-2 eine Liste von Polling-Pipelines zurück und zeigt die letzte Ausführungszeit für jede Pipeline an.

   ```
    % ./pollingPipelineExtractor.sh us-west-2

   | Polling Pipeline Name | Last Executed Time           |
   | _____________________ | ___________________          |
   | myCodeBuildPipeline   | Wed Mar 8 09:35:49 PST 2023  |
   | myCodeCommitPipeline  | Mon Apr 24 22:32:32 PDT 2023 |
   | TestPipeline          | Not executed in last year    |

   Saving list of polling pipeline names to us-west-2-1682496174.csv...%
   ```

   Analysieren Sie die Skriptausgabe und aktualisieren Sie für jede Pipeline in der Liste die Abfragequelle auf die empfohlene ereignisbasierte Methode zur Erkennung von Änderungen.

> **Note**
>
> Ihre Abfrage-Pipelines werden durch die Aktionskonfiguration der Pipeline für den Parameter bestimmt. `PollForSourceChanges` Wenn in der Konfiguration der Pipeline-Quelle der `PollForSourceChanges` Parameter weggelassen wurde, wird CodePipeline standardmäßig Ihr Repository nach Quelländerungen abgefragt. Dieses Verhalten ist dasselbe, als ob `PollForSourceChanges` es enthalten und auf true gesetzt wäre. Weitere Informationen finden Sie in den Konfigurationsparametern für die Quellaktion Ihrer Pipeline, z. B. in den Konfigurationsparametern der Amazon S3 S3-Quellaktion unterAmazon S3 S3-Quellaktion.

Beachten Sie, dass dieses Skript auch eine .csv-Datei generiert, die die Liste der Abfrage-Pipelines in Ihrem Konto enthält, und die CSV-Datei im aktuellen Arbeitsordner speichert.

Migrieren Sie Abfrage-Pipelines mit einer Quelle CodeCommit

Sie können Ihre Polling-Pipeline migrieren, um sie EventBridge zur Erkennung von Änderungen in Ihrem CodeCommit Quell-Repository oder Ihrem Amazon S3 S3-Quell-Bucket zu verwenden.

CodeCommit-- Bei einer Pipeline mit einer CodeCommit Quelle ändern Sie die Pipeline so, dass die Änderungserkennung automatisiert wird. EventBridge Wählen Sie aus den folgenden Methoden, um die Migration zu implementieren:

- Konsole: Migrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole)
- CLI: Migrieren von Abfrage-Pipelines (CodeCommit Quelle) (CLI)
- AWS CloudFormation: Migrieren Sie Abfrage-Pipelines (CodeCommit Quelle) (AWS CloudFormation Vorlage)

Migrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole)

Sie können die CodePipeline Konsole verwenden, um Ihre Pipeline zu aktualisieren, um Änderungen in Ihrem CodeCommit Quell-Repository oder Ihrem Amazon S3 S3-Quell-Bucket zu erkennen. EventBridge

> **Note**
>
> Wenn Sie die Konsole verwenden, um eine Pipeline zu bearbeiten, die über ein CodeCommit Quell-Repository oder einen Amazon S3 S3-Quell-Bucket verfügt, werden die Regel und die IAM-Rolle für Sie erstellt. Wenn Sie die AWS CLI Pipeline bearbeiten, müssen Sie die EventBridge Regel und die IAM-Rolle selbst erstellen. Weitere Informationen finden Sie unter CodeCommit Quellaktionen und EventBridge.

Führen Sie die folgenden Schritte aus, um eine Pipeline zu bearbeiten, die periodische Prüfungen verwendet. Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen Sie eine Pipeline in CodePipeline.

So bearbeiten Sie die Quellphase einer Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Auf diese Weise wird eine detaillierte Ansicht der Pipeline geöffnet (einschließlich des Status der Aktionen in den einzelnen Stufen der Pipeline).
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie in der Stufe Edit stage (Stufe bearbeiten) das Bearbeitungssymbol für die Quellaktion aus.
5. Erweitern Sie die Optionen zur Änderungserkennung und wählen Sie CloudWatch Ereignisse verwenden, um meine Pipeline automatisch zu starten, wenn eine Änderung eintritt (empfohlen).

   Es wird eine Meldung mit der EventBridge Regel angezeigt, die für diese Pipeline erstellt werden soll. Wählen Sie Aktualisieren.

   Wenn Sie eine Pipeline aktualisieren, die über eine Amazon S3 S3-Quelle verfügt, wird die folgende Meldung angezeigt. Wählen Sie Aktualisieren.
6. Wenn Sie die Bearbeitung der Pipeline abgeschlossen haben, wählen Sie Save pipeline changes aus, um zur Übersichtsseite zurückzukehren.

   In einer Meldung wird der Name der EventBridge Regel angezeigt, die für Ihre Pipeline erstellt werden soll. Wählen Sie Save and continue aus.

7. Um Ihre Aktion zu testen, geben Sie eine Änderung frei, indem Sie mit dem AWS CLI eine Änderung an der Quelle festschreiben, die in der Quellphase der Pipeline angegeben wurde.

Migrieren von Abfrage-Pipelines (CodeCommit Quelle) (CLI)

Gehen Sie wie folgt vor, um eine Pipeline zu bearbeiten, die Abfragen (regelmäßige Prüfungen) verwendet, um eine EventBridge Regel zum Starten der Pipeline zu verwenden. Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen Sie eine Pipeline in CodePipeline.

Um eine ereignisgesteuerte Pipeline mit zu erstellen CodeCommit, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und erstellen dann die folgenden Ressourcen:

- EventBridge Ereignis
- IAM-Rolle, damit dieses Ereignis Ihre Pipeline starten kann

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.
2. Öffnen Sie die JSON-Datei in einem beliebigen Texteditor und bearbeiten Sie die Quellstufe, indem Sie den Parameter `PollForSourceChanges` in `false` ändern, wie in diesem Beispiel gezeigt.

Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

```
"configuration": {
    "PollForSourceChanges": "false",
    "BranchName": "main",
    "RepositoryName": "MyTestRepo"
},
```

3. Wenn Sie mit einer Pipeline-Struktur arbeiten, die Sie mit dem Befehl get-pipeline abgerufen haben, müssen Sie die `metadata`-Zeilen aus der JSON-Datei entfernen. Andernfalls kann der update-pipeline-Befehl sie nicht nutzen. Entfernen Sie die `"metadata": { }`-Zeilen und die Felder `"created"`, `"pipelineARN"` und `"updated"`.

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

   ```
   "metadata": {
       "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
       "created": "date",
       "updated": "date"
   },
   ```

   Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

   > **Important**
   >
   > Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

   ```
   aws codepipeline update-pipeline --cli-input-json file://pipeline.json
   ```

   Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den **`start-pipeline-execution`**-Befehl, um Ihre Pipeline manuell zu starten.

Um eine EventBridge Regel mit CodeCommit als Ereignisquelle und CodePipeline als Ziel zu erstellen

1. Fügen Sie Berechtigungen hinzu EventBridge , die CodePipeline zum Aufrufen der Regel verwendet werden sollen. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie zu erstellen, die es ermöglicht, die Servicerolle EventBridge zu übernehmen. Geben Sie der Vertrauensrichtlinie den Namen `trustpolicyforEB.json`.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

   b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

   ```
   aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
    file://trustpolicyforEB.json
   ```

c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie in diesem Beispiel für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Verwenden Sie den folgenden Befehl, um die Berechtigungsrichtlinie `CodePipeline-Permissions-Policy-for-EB` der Rolle `Role-for-MyRule` anzufügen.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen dieser Richtlinie zur Rolle werden Berechtigungen für erstellt EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

Warum nehme ich diese Änderung vor? Dieser Befehl aktiviert AWS CloudFormation , um das Ereignis zu erstellen.

Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „`MyCodeCommitRepoRule`“ erstellt.

```
aws events put-rule --name "MyCodeCommitRepoRule" --event-pattern "{\"source\":
[\"aws.codecommit\"],\"detail-type\":[\"CodeCommit Repository State Change\"],
\"resources\":[\"repository-ARN\"],\"detail\":{\"referenceType\":[\"branch\"],
```

```
\"referenceName\":[\"main\"]}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-
MyRule"
```

3. Um das Objekt CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die folgenden Parameter an:

   - Der Parameter `--rule` wird für den `rule_name` verwendet, den Sie mit put-rule erstellt haben.
   - Der Parameter `--targets` wird für die Listen-`Id` des Ziels in der Zielliste und den ARN der Ziel-Pipeline verwendet.

   Der folgende Beispielbefehl legt fest, dass für die Regel mit dem Namen `MyCodeCommitRepoRule` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dieses Ziel 1 ist. Der Beispielbefehl gibt zudem ein Beispiel ARN für die Pipeline an. Die Pipeline startet, wenn Änderungen im Repository auftreten.

   ```
   aws events put-targets --rule MyCodeCommitRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

Migrieren Sie Abfrage-Pipelines (CodeCommit Quelle) (AWS CloudFormation Vorlage)

Um eine ereignisgesteuerte Pipeline mit zu erstellen AWS CodeCommit, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und fügen dann Ihrer Vorlage die folgenden Ressourcen hinzu:

- Eine Regel EventBridge
- Eine IAM-Rolle für Ihre Regel EventBridge

Wenn Sie AWS CloudFormation Ihre Pipelines erstellen und verwalten, enthält Ihre Vorlage Inhalte wie die folgenden.

> **Note**
>
> Die `Configuration`-Eigenschaft in der Quellstufe mit dem Namen `PollForSourceChanges`. Wenn diese Eigenschaft in Ihrer Vorlage nicht enthalten ist, dann wird `PollForSourceChanges` standardmäßig auf `true` festgelegt.

YAML

```
Resources:
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      Name: codecommit-polling-pipeline
      RoleArn:
        !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: CodeCommit
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                BranchName: !Ref BranchName
                RepositoryName: !Ref RepositoryName
                PollForSourceChanges: true
              RunOrder: 1
```

JSON

```
"Stages": [
    {
        "Name": "Source",
  "Actions": [{
      "Name": "SourceAction",
      "ActionTypeId": {
    "Category": "Source",
    "Owner": "AWS",
    "Version": 1,
    "Provider": "CodeCommit"
      },
      "OutputArtifacts": [{
          "Name": "SourceOutput"
```

```
      }],
      "Configuration": {
          "BranchName": {
        "Ref": "BranchName"
    },
    "RepositoryName": {
        "Ref": "RepositoryName"
    },
    "PollForSourceChanges": true
            },
            "RunOrder": 1
        }]
    },
```

Um Ihre AWS CloudFormation Pipeline-Vorlage zu aktualisieren und eine Regel zu erstellen EventBridge

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` AWS CloudFormation Ressource`Resources`, um die IAM-Rolle zu konfigurieren, mit der Ihre Veranstaltung Ihre Pipeline starten kann. Dieser Eintrag erstellt eine Rolle mit zwei Richtlinien:

   - Die erste Richtlinie ermöglicht die Übernahme der Rolle.
   - Die zweite Richtlinie stellt Berechtigungen zum Starten der Pipeline bereit.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::IAM::Role` Ressource können AWS CloudFormation Sie Berechtigungen für EventBridge erstellen. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

   ```
   EventRole:
     Type: AWS::IAM::Role
     Properties:
       AssumeRolePolicyDocument:
         Version: 2012-10-17
         Statement:
           -
             Effect: Allow
             Principal:
               Service:
   ```

```
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
 'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
```

JSON

```
"EventRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "events.amazonaws.com"
            ]
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "eb-pipeline-execution",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
```

```
              "Action": "codepipeline:StartPipelineExecution",
              "Resource": {
                "Fn::Join": [
                  "",
                  [
                    "arn:aws:codepipeline:",
                    {
                      "Ref": "AWS::Region"
                    },
                    ":",
                    {
                      "Ref": "AWS::AccountId"
                    },
                    ":",
                    {
                      "Ref": "AppPipeline"
                    }
                  ]

...
```

2. Verwenden Sie in der Vorlage unter die `AWS::Events::Rule` AWS CloudFormation Ressource`Resources`, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis, das Push-Änderungen an Ihrem Repository überwacht. Wenn eine Änderung des Repository-Status EventBridge erkannt wird, wird die Regel in `StartPipelineExecution` Ihrer Zielpipeline aufgerufen.

   Warum nehme ich diese Änderung vor? Durch Hinzufügen der `AWS::Events::Rule` Ressource kann AWS CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

```
  EventRule:
    Type: AWS::Events::Rule
    Properties:
      EventPattern:
        source:
          - aws.codecommit
        detail-type:
          - 'CodeCommit Repository State Change'
        resources:
```

```
          - !Join [ '', [ 'arn:aws:codecommit:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref RepositoryName ] ]
        detail:
          event:
            - referenceCreated
            - referenceUpdated
          referenceType:
            - branch
          referenceName:
            - main
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
```

JSON

```
"EventRule": {
  "Type": "AWS::Events::Rule",
  "Properties": {
    "EventPattern": {
      "source": [
        "aws.codecommit"
      ],
      "detail-type": [
        "CodeCommit Repository State Change"
      ],
      "resources": [
        {
          "Fn::Join": [
            "",
            [
              "arn:aws:codecommit:",
              {
                "Ref": "AWS::Region"
              },
              ":",
              {
                "Ref": "AWS::AccountId"
```

```
            },
            ":",
            {
              "Ref": "RepositoryName"
            }
          ]
        ]
      }
    ],
    "detail": {
      "event": [
        "referenceCreated",
        "referenceUpdated"
      ],
      "referenceType": [
        "branch"
      ],
      "referenceName": [
        "main"
      ]
    }
  },
  "Targets": [
    {
      "Arn": {
        "Fn::Join": [
          "",
          [
            "arn:aws:codepipeline:",
            {
              "Ref": "AWS::Region"
            },
            ":",
            {
              "Ref": "AWS::AccountId"
            },
            ":",
            {
              "Ref": "AppPipeline"
            }
          ]
        ]
      },
      "RoleArn": {
```

```
            "Fn::GetAtt": [
              "EventRole",
              "Arn"
            ]
          },
          "Id": "codepipeline-AppPipeline"
        }
      ]
    }
},
```

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation -Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> In vielen Fällen ist der Parameter `PollForSourceChanges` „true“, wenn Sie eine Pipeline erstellen. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

  Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: CodeCommit
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              BranchName: !Ref BranchName
              RepositoryName: !Ref RepositoryName
              PollForSourceChanges: false
            RunOrder: 1
```

JSON

```
{
  "Name": "Source",
  "Actions": [
    {
      "Name": "SourceAction",
      "ActionTypeId": {
        "Category": "Source",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "CodeCommit"
      },
      "OutputArtifacts": [
        {
          "Name": "SourceOutput"
        }
      ],
      "Configuration": {
        "BranchName": {
          "Ref": "BranchName"
        },
        "RepositoryName": {
```

```
          "Ref": "RepositoryName"
        },
        "PollForSourceChanges": false
      },
      "RunOrder": 1
    }
  ]
},
```

Example

Wenn Sie diese Ressourcen mit erstellen AWS CloudFormation, wird Ihre Pipeline ausgelöst, wenn Dateien in Ihrem Repository erstellt oder aktualisiert werden. Im Folgenden finden Sie den endgültigen Vorlagenausschnitt:

YAML

```
Resources:
  EventRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
```

```
              Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
 EventRule:
   Type: AWS::Events::Rule
   Properties:
     EventPattern:
       source:
         - aws.codecommit
       detail-type:
         - 'CodeCommit Repository State Change'
       resources:
         - !Join [ '', [ 'arn:aws:codecommit:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref RepositoryName ] ]
       detail:
         event:
           - referenceCreated
           - referenceUpdated
         referenceType:
           - branch
         referenceName:
           - main
     Targets:
       -
         Arn:
           !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
         RoleArn: !GetAtt EventRole.Arn
         Id: codepipeline-AppPipeline
 AppPipeline:
   Type: AWS::CodePipeline::Pipeline
   Properties:
     Name: codecommit-events-pipeline
     RoleArn:
       !GetAtt CodePipelineServiceRole.Arn
     Stages:
       -
         Name: Source
         Actions:
           -
             Name: SourceAction
             ActionTypeId:
               Category: Source
               Owner: AWS
               Version: 1
```

```
                Provider: CodeCommit
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                BranchName: !Ref BranchName
                RepositoryName: !Ref RepositoryName
                PollForSourceChanges: false
              RunOrder: 1


...
```

JSON

```
    "Resources": {

...

        "EventRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "events.amazonaws.com"
                                ]
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "Path": "/",
                "Policies": [
                    {
                        "PolicyName": "eb-pipeline-execution",
                        "PolicyDocument": {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
```

```
                                "Effect": "Allow",
                                "Action": "codepipeline:StartPipelineExecution",
                                "Resource": {
                                    "Fn::Join": [
                                        "",
                                        [
                                            "arn:aws:codepipeline:",
                                            {
                                                "Ref": "AWS::Region"
                                            },
                                            ":",
                                            {
                                                "Ref": "AWS::AccountId"
                                            },
                                            ":",
                                            {
                                                "Ref": "AppPipeline"
                                            }
                                        ]
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
        }
    },
    "EventRule": {
        "Type": "AWS::Events::Rule",
        "Properties": {
            "EventPattern": {
                "source": [
                    "aws.codecommit"
                ],
                "detail-type": [
                    "CodeCommit Repository State Change"
                ],
                "resources": [
                    {
                        "Fn::Join": [
                            "",
                            [
                                "arn:aws:codecommit:",
```

```
                                {
                                    "Ref": "AWS::Region"
                                },
                                ":",
                                {
                                    "Ref": "AWS::AccountId"
                                },
                                ":",
                                {
                                    "Ref": "RepositoryName"
                                }
                            ]
                        ]
                    }
                ],
                "detail": {
                    "event": [
                        "referenceCreated",
                        "referenceUpdated"
                    ],
                    "referenceType": [
                        "branch"
                    ],
                    "referenceName": [
                        "main"
                    ]
                }
            },
            "Targets": [
                {
                    "Arn": {
                        "Fn::Join": [
                            "",
                            [
                                "arn:aws:codepipeline:",
                                {
                                    "Ref": "AWS::Region"
                                },
                                ":",
                                {
                                    "Ref": "AWS::AccountId"
                                },
                                ":",
                                {
```

```
                                    "Ref": "AppPipeline"
                                }
                            ]
                        ]
                    },
                    "RoleArn": {
                        "Fn::GetAtt": [
                            "EventRole",
                            "Arn"
                        ]
                    },
                    "Id": "codepipeline-AppPipeline"
                }
            ]
        }
    },
    "AppPipeline": {
        "Type": "AWS::CodePipeline::Pipeline",
        "Properties": {
            "Name": "codecommit-events-pipeline",
            "RoleArn": {
                "Fn::GetAtt": [
                    "CodePipelineServiceRole",
                    "Arn"
                ]
            },
            "Stages": [
                {
                    "Name": "Source",
                    "Actions": [
                        {
                            "Name": "SourceAction",
                            "ActionTypeId": {
                                "Category": "Source",
                                "Owner": "AWS",
                                "Version": 1,
                                "Provider": "CodeCommit"
                            },
                            "OutputArtifacts": [
                                {
                                    "Name": "SourceOutput"
                                }
                            ],
                            "Configuration": {
```

```
                                "BranchName": {
                                    "Ref": "BranchName"
                                },
                                "RepositoryName": {
                                    "Ref": "RepositoryName"
                                },
                                "PollForSourceChanges": false
                            },
                            "RunOrder": 1
                        }
                    ]
                },

...
```

Migrieren Sie Abfrage-Pipelines mit einer S3-Quelle, die für Ereignisse aktiviert ist

Bei einer Pipeline mit einer Amazon S3 S3-Quelle ändern Sie die Pipeline so, dass die Änderungserkennung über EventBridge und mit einem Quell-Bucket, der für Ereignisbenachrichtigungen aktiviert ist, automatisiert wird. Dies ist die empfohlene Methode, wenn Sie die CLI verwenden oder Ihre Pipeline migrieren AWS CloudFormation möchten.

> **Note**
>
> Dazu gehört die Verwendung eines Buckets, der für Ereignisbenachrichtigungen aktiviert ist, sodass Sie keinen separaten CloudTrail Trail erstellen müssen. Wenn Sie die Konsole verwenden, werden eine Ereignisregel und ein CloudTrail Trail für Sie eingerichtet. Informationen zu diesen Schritten finden Sie unterMigrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail .

- CLI: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und CloudTrail -Trail (CLI)
- AWS CloudFormation: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (Vorlage)AWS CloudFormation

Migrieren Sie Abfrage-Pipelines mit einer für Ereignisse aktivierten S3-Quelle (CLI)

Gehen Sie wie folgt vor, um eine Pipeline zu bearbeiten, die Polling (regelmäßige Prüfungen) verwendet, um stattdessen ein Ereignis in zu verwenden. EventBridge Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen Sie eine Pipeline in CodePipeline.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und erstellen dann die folgenden Ressourcen:

- EventBridge Ereignisregel
- IAM-Rolle, damit das EventBridge Ereignis Ihre Pipeline starten kann

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Erteilen Sie Berechtigungen EventBridge , die CodePipeline zum Aufrufen der Regel verwendet werden können. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie zu erstellen, damit EventBridge Sie die Servicerolle übernehmen können. Geben Sie ihr den Namen `trustpolicyforEB.json`.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

   b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen dieser Vertrauensrichtlinie zur Rolle werden Berechtigungen für erstellt EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie hier für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Führen Sie den folgenden Befehl aus, um der erstellten `Role-for-MyRule`-Rolle die neue `CodePipeline-Permissions-Policy-for-EB`-Berechtigungsrichtlinie anzufügen.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

   Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „`EnabledS3SourceRule`“ erstellt.

```
aws events put-rule --name "EnabledS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"Object Created\"],\"detail\":{\"bucket\":{\"name\":
[\"my-bucket\"]}}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Um das Objekt CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die `--targets` Parameter `--rule` und an.

   Der folgende Befehl legt fest, dass für die Regel mit dem Namen `EnabledS3SourceRule` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dies Ziel 1 ist. Der Befehl gibt zudem ein Beispiel ARN für die Pipeline an. Die Pipeline startet, wenn Änderungen im Repository auftreten.

   ```
   aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
   AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem Texteditor und bearbeiten Sie die Quellphase, indem Sie den Parameter `PollForSourceChanges` für einen Bucket mit dem Namen `storage-bucket`in `false` ändern wie in diesem Beispiel gezeigt.

   Warum nehme ich diese Änderung vor? Durch Festlegen dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

```
"configuration": {
    "S3Bucket": "storage-bucket",
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

3. Wenn Sie mit einer Pipeline-Struktur arbeiten, die Sie mit dem Befehl get-pipeline abgerufen haben, müssen Sie die `metadata`-Zeilen aus der JSON-Datei entfernen. Andernfalls kann der update-pipeline-Befehl sie nicht nutzen. Entfernen Sie die `"metadata": { }`-Zeilen und die Felder `"created"`, `"pipelineARN"` und `"updated"`.

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

   ```
   "metadata": {
       "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
       "created": "date",
       "updated": "date"
   },
   ```

   Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

   > **Important**
   >
   > Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

   ```
   aws codepipeline update-pipeline --cli-input-json file://pipeline.json
   ```

   Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

   > **Note**
   >
   > Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision

über die aktualisierte Pipeline auszuführen. Verwenden Sie den start-pipeline-execution-Befehl, um Ihre Pipeline manuell zu starten.

Migrieren Sie Abfrage-Pipelines mit einer für Ereignisse aktivierten S3-Quelle (Vorlage)AWS CloudFormation

Dieses Verfahren gilt für eine Pipeline, bei der im Quell-Bucket Ereignisse aktiviert sind.

Gehen Sie wie folgt vor, um Ihre Pipeline mit einer Amazon S3 S3-Quelle von der Abfrage bis zur ereignisbasierten Änderungserkennung zu bearbeiten.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und fügen dann die folgenden Ressourcen zu Ihrer Vorlage hinzu:

- EventBridge Regel und IAM-Rolle, damit dieses Ereignis Ihre Pipeline starten kann.

Wenn Sie Ihre Pipelines AWS CloudFormation zum Erstellen und Verwalten verwenden, enthält Ihre Vorlage Inhalte wie den folgenden.

> **Note**
>
> Die `Configuration`-Eigenschaft in der Quellstufe mit dem Namen `PollForSourceChanges`. Wenn diese Eigenschaft in Ihrer Vorlage nicht enthalten ist, dann wird `PollForSourceChanges` standardmäßig auf `true` festgelegt.

YAML

```
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    RoleArn: !GetAtt CodePipelineServiceRole.Arn
    Stages:
      -
        Name: Source
        Actions:
          -
```

```
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                -
                  Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref S3SourceObjectKey
                PollForSourceChanges: true
              RunOrder: 1


...
```

JSON

```
        "AppPipeline": {
            "Type": "AWS::CodePipeline::Pipeline",
            "Properties": {
                "RoleArn": {
                    "Fn::GetAtt": ["CodePipelineServiceRole", "Arn"]
                },
                "Stages": [
                    {
                        "Name": "Source",
                        "Actions": [
                            {
                                "Name": "SourceAction",
                                "ActionTypeId": {
                                    "Category": "Source",
                                    "Owner": "AWS",
                                    "Version": 1,
                                    "Provider": "S3"
                                },
                                "OutputArtifacts": [
                                    {
                                        "Name": "SourceOutput"
                                    }
                                ],
```

```
                                "Configuration": {
                                    "S3Bucket": {
                                        "Ref": "SourceBucket"
                                    },
                                    "S3ObjectKey": {
                                        "Ref": "SourceObjectKey"
                                    },
                                    "PollForSourceChanges": true
                                },
                                "RunOrder": 1
                            }
                        ]
                    },


...
```

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` AWS CloudFormation Ressource`Resources`, um die IAM-Rolle zu konfigurieren, mit der Ihr Ereignis Ihre Pipeline starten kann. Dieser Eintrag erstellt eine Rolle mit zwei Richtlinien:

   - Die erste Richtlinie ermöglicht die Übernahme der Rolle.
   - Die zweite Richtlinie stellt Berechtigungen zum Starten der Pipeline bereit.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen einer `AWS::IAM::Role` Ressource können AWS CloudFormation Sie Berechtigungen für EventBridge erstellen. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

   ```
   EventRole:
     Type: AWS::IAM::Role
     Properties:
       AssumeRolePolicyDocument:
         Version: 2012-10-17
         Statement:
           -
   ```

```
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
 'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]


...
```

JSON

```
  "EventRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": [
                "events.amazonaws.com"
              ]
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "Path": "/",
      "Policies": [
        {
```

```
          "PolicyName": "eb-pipeline-execution",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Action": "codepipeline:StartPipelineExecution",
                "Resource": {
                  "Fn::Join": [
                    "",
                    [
                      "arn:aws:codepipeline:",
                      {
                        "Ref": "AWS::Region"
                      },
                      ":",
                      {
                        "Ref": "AWS::AccountId"
                      },
                      ":",
                      {
                        "Ref": "AppPipeline"
                      }
                    ]
                  ]

...
```

2. Verwenden Sie die `AWS::Events::Rule` AWS CloudFormation Ressource, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis, das die Erstellung oder Löschung von Objekten in Ihrem Amazon S3 S3-Quell-Bucket überwacht. Fügen Sie darüber hinaus ein Ziel für Ihre Pipeline ein. Wenn ein Objekt erstellt wird, wird diese Regel in `StartPipelineExecution` Ihrer Zielpipeline aufgerufen.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::Events::Rule` Ressource kann AWS CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

   ```
   EventRule:
     Type: AWS::Events::Rule
   ```

```
    Properties:
      EventBusName: default
      EventPattern:
        source:
          - aws.s3
        detail-type:
          - Object Created
        detail:
          bucket:
            name:
              - !Ref SourceBucket
      Name: EnabledS3SourceRule
      State: ENABLED
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline


...
```

JSON

```
  "EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
  "EventBusName": "default",
  "EventPattern": {
      "source": [
   "aws.s3"
      ],
      "detail-type": [
    "Object Created"
      ],
      "detail": {
    "bucket": {
        "name": [
      "s3-pipeline-source-fra-bucket"
        ]
```

```
        }
            }
  },
  "Name": "EnabledS3SourceRule",
        "State": "ENABLED",
        "Targets": [
        {
          "Arn": {
            "Fn::Join": [
              "",
              [
                "arn:aws:codepipeline:",
                {
                  "Ref": "AWS::Region"
                },
                ":",
                {
                  "Ref": "AWS::AccountId"
                },
                ":",
                {
                  "Ref": "AppPipeline"
                }
              ]
            ]
          },
          "RoleArn": {
            "Fn::GetAtt": [
              "EventRole",
              "Arn"
            ]
          },
          "Id": "codepipeline-AppPipeline"
        }
      ]
    }
  }
},

...
```

3. Speichern Sie Ihre aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation -Konsole.

4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie Ihre aktualisierte Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

  Warum nehme ich diese Änderung vor? Durch Ändern von `PollForSourceChanges` in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

  YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
```

```
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
```

JSON

```
{
   "Name": "SourceAction",
   "ActionTypeId": {
     "Category": "Source",
     "Owner": "AWS",
     "Version": 1,
     "Provider": "S3"
   },
   "OutputArtifacts": [
     {
       "Name": "SourceOutput"
     }
   ],
   "Configuration": {
     "S3Bucket": {
       "Ref": "SourceBucket"
     },
     "S3ObjectKey": {
       "Ref": "SourceObjectKey"
     },
     "PollForSourceChanges": false
   },
   "RunOrder": 1
 }
```

Example

Wenn Sie diese Ressourcen erstellen, wird Ihre Pipeline ausgelöst, wenn Dateien in Ihrem Repository erstellt oder aktualisiert werden. AWS CloudFormation

> **Note**
>
> Hören Sie hier nicht auf. Obwohl Ihre Pipeline erstellt wurde, müssen Sie eine zweite AWS CloudFormation Vorlage für Ihre Amazon S3 S3-Pipeline erstellen. Wenn Sie die zweite Vorlage nicht erstellen, enthält Ihre Pipeline keine Funktionalität für die Änderungserkennung.

YAML

```
Parameters:
  SourceObjectKey:
    Description: 'S3 source artifact'
    Type: String
    Default: SampleApp_Linux.zip
  ApplicationName:
    Description: 'CodeDeploy application name'
    Type: String
    Default: DemoApplication
  BetaFleet:
    Description: 'Fleet configured in CodeDeploy'
    Type: String
    Default: DemoFleet

Resources:
  SourceBucket:
    Type: AWS::S3::Bucket
    Properties:
      NotificationConfiguration:
        EventBridgeConfiguration:
          EventBridgeEnabled: true
      VersioningConfiguration:
        Status: Enabled
  CodePipelineArtifactStoreBucket:
    Type: AWS::S3::Bucket
  CodePipelineArtifactStoreBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref CodePipelineArtifactStoreBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          -
```

```
            Sid: DenyUnEncryptedObjectUploads
            Effect: Deny
            Principal: '*'
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*' ] ]
            Condition:
              StringNotEquals:
                s3:x-amz-server-side-encryption: aws:kms
          -
            Sid: DenyInsecureConnections
            Effect: Deny
            Principal: '*'
            Action: s3:*
            Resource: !Sub ${CodePipelineArtifactStoreBucket.Arn}/*
            Condition:
              Bool:
                aws:SecureTransport: false
  CodePipelineServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - codepipeline.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: AWS-CodePipeline-Service-3
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action:
                  - codecommit:CancelUploadArchive
                  - codecommit:GetBranch
                  - codecommit:GetCommit
                  - codecommit:GetUploadArchiveStatus
```

```
        - codecommit:UploadArchive
      Resource: 'resource_ARN'
    -
      Effect: Allow
      Action:
        - codedeploy:CreateDeployment
        - codedeploy:GetApplicationRevision
        - codedeploy:GetDeployment
        - codedeploy:GetDeploymentConfig
        - codedeploy:RegisterApplicationRevision
      Resource: 'resource_ARN'
    -
      Effect: Allow
      Action:
        - codebuild:BatchGetBuilds
        - codebuild:StartBuild
      Resource: 'resource_ARN'
    -
      Effect: Allow
      Action:
        - devicefarm:ListProjects
        - devicefarm:ListDevicePools
        - devicefarm:GetRun
        - devicefarm:GetUpload
        - devicefarm:CreateUpload
        - devicefarm:ScheduleRun
      Resource: 'resource_ARN'
    -
      Effect: Allow
      Action:
        - lambda:InvokeFunction
        - lambda:ListFunctions
      Resource: 'resource_ARN'
    -
      Effect: Allow
      Action:
        - iam:PassRole
      Resource: 'resource_ARN'
    -
      Effect: Allow
      Action:
        - elasticbeanstalk:*
        - ec2:*
        - elasticloadbalancing:*
```

```
                - autoscaling:*
                - cloudwatch:*
                - s3:*
                - sns:*
                - cloudformation:*
                - rds:*
                - sqs:*
                - ecs:*
              Resource: 'resource_ARN'
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    Name: s3-events-pipeline
    RoleArn:
      !GetAtt CodePipelineServiceRole.Arn
    Stages:
      -
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
      -
        Name: Beta
        Actions:
          -
            Name: BetaAction
            InputArtifacts:
              - Name: SourceOutput
            ActionTypeId:
              Category: Deploy
              Owner: AWS
              Version: 1
```

```
              Provider: CodeDeploy
            Configuration:
              ApplicationName: !Ref ApplicationName
              DeploymentGroupName: !Ref BetaFleet
            RunOrder: 1
    ArtifactStore:
      Type: S3
      Location: !Ref CodePipelineArtifactStoreBucket
 EventRole:
   Type: AWS::IAM::Role
   Properties:
     AssumeRolePolicyDocument:
       Version: 2012-10-17
       Statement:
         -
           Effect: Allow
           Principal:
             Service:
               - events.amazonaws.com
           Action: sts:AssumeRole
     Path: /
     Policies:
       -
         PolicyName: eb-pipeline-execution
         PolicyDocument:
           Version: 2012-10-17
           Statement:
             -
               Effect: Allow
               Action: codepipeline:StartPipelineExecution
               Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
 EventRule:
   Type: AWS::Events::Rule
   Properties:
     EventBusName: default
     EventPattern:
       source:
         - aws.s3
       detail-type:
         - Object Created
       detail:
         bucket:
           name:
```

```
              - !Ref SourceBucket
      Name: EnabledS3SourceRule
      State: ENABLED
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
```

JSON

```
{
    "Parameters": {
        "SourceObjectKey": {
            "Description": "S3 source artifact",
            "Type": "String",
            "Default": "SampleApp_Linux.zip"
        },
        "ApplicationName": {
            "Description": "CodeDeploy application name",
            "Type": "String",
            "Default": "DemoApplication"
        },
        "BetaFleet": {
            "Description": "Fleet configured in CodeDeploy",
            "Type": "String",
            "Default": "DemoFleet"
        }
    },
    "Resources": {
        "SourceBucket": {
            "Type": "AWS::S3::Bucket",
              "Properties": {
                "NotificationConfiguration": {
                    "EventBridgeConfiguration": {
                        "EventBridgeEnabled": true
                    }
                },
                "VersioningConfiguration": {
                    "Status": "Enabled"
```

```
                }
            }
        },
        "CodePipelineArtifactStoreBucket": {
            "Type": "AWS::S3::Bucket"
        },
        "CodePipelineArtifactStoreBucketPolicy": {
            "Type": "AWS::S3::BucketPolicy",
            "Properties": {
                "Bucket": {
                    "Ref": "CodePipelineArtifactStoreBucket"
                },
                "PolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Sid": "DenyUnEncryptedObjectUploads",
                            "Effect": "Deny",
                            "Principal": "*",
                            "Action": "s3:PutObject",
                            "Resource": {
                                "Fn::Join": [
                                    "",
                                    [
                                        {
                                            "Fn::GetAtt": [
                                                "CodePipelineArtifactStoreBucket",
                                                "Arn"
                                            ]
                                        },
                                        "/*"
                                    ]
                                ]
                            },
                            "Condition": {
                                "StringNotEquals": {
                                    "s3:x-amz-server-side-encryption": "aws:kms"
                                }
                            }
                        },
                        {
                            "Sid": "DenyInsecureConnections",
                            "Effect": "Deny",
                            "Principal": "*",
```

```
                        "Action": "s3:*",
                        "Resource": {
                            "Fn::Join": [
                                "",
                                [
                                    {
                                        "Fn::GetAtt": [
                                            "CodePipelineArtifactStoreBucket",
                                            "Arn"
                                        ]
                                    },
                                    "/*"
                                ]
                            ]
                        },
                        "Condition": {
                            "Bool": {
                                "aws:SecureTransport": false
                            }
                        }
                    }
                ]
            }
        }
    },
    "CodePipelineServiceRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "codepipeline.amazonaws.com"
                            ]
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "Path": "/",
            "Policies": [
```

```
                {
                    "PolicyName": "AWS-CodePipeline-Service-3",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "codecommit:CancelUploadArchive",
                                    "codecommit:GetBranch",
                                    "codecommit:GetCommit",
                                    "codecommit:GetUploadArchiveStatus",
                                    "codecommit:UploadArchive"
                                ],
                                "Resource": "resource_ARN"
                            },
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "codedeploy:CreateDeployment",
                                    "codedeploy:GetApplicationRevision",
                                    "codedeploy:GetDeployment",
                                    "codedeploy:GetDeploymentConfig",
                                    "codedeploy:RegisterApplicationRevision"
                                ],
                                "Resource": "resource_ARN"
                            },
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "codebuild:BatchGetBuilds",
                                    "codebuild:StartBuild"
                                ],
                                "Resource": "resource_ARN"
                            },
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "devicefarm:ListProjects",
                                    "devicefarm:ListDevicePools",
                                    "devicefarm:GetRun",
                                    "devicefarm:GetUpload",
                                    "devicefarm:CreateUpload",
                                    "devicefarm:ScheduleRun"
```

```
                                ],
                                "Resource": "resource_ARN"
                            },
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "lambda:InvokeFunction",
                                    "lambda:ListFunctions"
                                ],
                                "Resource": "resource_ARN"
                            },
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "iam:PassRole"
                                ],
                                "Resource": "resource_ARN"
                            },
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "elasticbeanstalk:*",
                                    "ec2:*",
                                    "elasticloadbalancing:*",
                                    "autoscaling:*",
                                    "cloudwatch:*",
                                    "s3:*",
                                    "sns:*",
                                    "cloudformation:*",
                                    "rds:*",
                                    "sqs:*",
                                    "ecs:*"
                                ],
                                "Resource": "resource_ARN"
                            }
                        ]
                    }
                }
            ]
        }
    },
    "AppPipeline": {
        "Type": "AWS::CodePipeline::Pipeline",
        "Properties": {
```

```
                "Name": "s3-events-pipeline",
                "RoleArn": {
                    "Fn::GetAtt": [
                        "CodePipelineServiceRole",
                        "Arn"
                    ]
                },
                "Stages": [
                    {
                        "Name": "Source",
                        "Actions": [
                            {
                                "Name": "SourceAction",
                                "ActionTypeId": {
                                    "Category": "Source",
                                    "Owner": "AWS",
                                    "Version": 1,
                                    "Provider": "S3"
                                },
                                "OutputArtifacts": [
                                    {
                                        "Name": "SourceOutput"
                                    }
                                ],
                                "Configuration": {
                                    "S3Bucket": {
                                        "Ref": "SourceBucket"
                                    },
                                    "S3ObjectKey": {
                                        "Ref": "SourceObjectKey"
                                    },
                                    "PollForSourceChanges": false
                                },
                                "RunOrder": 1
                            }
                        ]
                    },
                    {
                        "Name": "Beta",
                        "Actions": [
                            {
                                "Name": "BetaAction",
                                "InputArtifacts": [
                                    {
```

```
                                    "Name": "SourceOutput"
                                }
                            ],
                            "ActionTypeId": {
                                "Category": "Deploy",
                                "Owner": "AWS",
                                "Version": 1,
                                "Provider": "CodeDeploy"
                            },
                            "Configuration": {
                                "ApplicationName": {
                                    "Ref": "ApplicationName"
                                },
                                "DeploymentGroupName": {
                                    "Ref": "BetaFleet"
                                }
                            },
                            "RunOrder": 1
                        }
                    ]
                }
            ],
            "ArtifactStore": {
                "Type": "S3",
                "Location": {
                    "Ref": "CodePipelineArtifactStoreBucket"
                }
            }
        }
    },
    "EventRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "events.amazonaws.com"
                            ]
                        },
                        "Action": "sts:AssumeRole"
```

```
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "eb-pipeline-execution",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "codepipeline:StartPipelineExecution",
                                "Resource": {
                                    "Fn::Join": [
                                        "",
                                        [
                                            "arn:aws:codepipeline:",
                                            {
                                                "Ref": "AWS::Region"
                                            },
                                            ":",
                                            {
                                                "Ref": "AWS::AccountId"
                                            },
                                            ":",
                                            {
                                                "Ref": "AppPipeline"
                                            }
                                        ]
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
        }
    },
    "EventRule": {
        "Type": "AWS::Events::Rule",

        "Properties": {
            "EventBusName": "default",
```

```
          "EventPattern": {
              "source": [
            "aws.s3"
          ],
                "detail-type": [
        "Object Created"
          ],
          "detail": {
        "bucket": {
                         "name": [
                             {
                             "Ref": "SourceBucket"
                             }
                         ]
                     }
                }
    },
    "Name": "EnabledS3SourceRule",
          "State": "ENABLED",
                 "Targets": [
                     {
                         "Arn": {
                             "Fn::Join": [
                                 "",
                                 [
                                     "arn:aws:codepipeline:",
                                     {
                                         "Ref": "AWS::Region"
                                     },
                                     ":",
                                     {
                                         "Ref": "AWS::AccountId"
                                     },
                                     ":",
                                     {
                                         "Ref": "AppPipeline"
                                     }
                                 ]
                             ]
                         },
                         "RoleArn": {
                             "Fn::GetAtt": [
                                 "EventRole",
                                 "Arn"
```

```
                            ]
                        },
                        "Id": "codepipeline-AppPipeline"
                    }
                ]
            }
        }
    }
    }
}
```

Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail

Bei einer Pipeline mit einer Amazon S3 S3-Quelle ändern Sie die Pipeline so, dass die Änderungserkennung automatisiert wird EventBridge. Wählen Sie aus den folgenden Methoden, um die Migration zu implementieren:

- Konsole: Migrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole)
- CLI: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und CloudTrail -Trail (CLI)
- AWS CloudFormation: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (Vorlage)AWS CloudFormation

Migrieren Sie Polling-Pipelines mit einer S3-Quelle und CloudTrail -Trail (CLI)

Gehen Sie wie folgt vor, um eine Pipeline zu bearbeiten, die Polling (regelmäßige Prüfungen) verwendet, um stattdessen ein Ereignis in zu verwenden. EventBridge Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen Sie eine Pipeline in CodePipeline.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und erstellen dann die folgenden Ressourcen:

- AWS CloudTrail Trail-, Bucket- und Bucket-Richtlinie, die Amazon S3 zur Protokollierung der Ereignisse verwenden kann.
- EventBridge Ereignis
- IAM-Rolle, damit das EventBridge Ereignis Ihre Pipeline starten kann

Um einen AWS CloudTrail Trail zu erstellen und die Protokollierung zu aktivieren

Um mit dem einen Trail AWS CLI zu erstellen, rufen Sie den create-trail Befehl auf und geben Sie Folgendes an:

- Den Trail-Namen.
- Der Bucket, auf den Sie bereits die Bucket-Richtlinie für AWS CloudTrail angewendet haben.

Weitere Informationen finden Sie unter Erstellen eines Pfads mit der AWS Befehlszeilenschnittstelle.

1. Rufen Sie den Befehl create-trail auf und beziehen Sie die Parameter `--name` und `--s3-bucket-name` ein.

   Warum nehme ich diese Änderung vor? Dadurch wird der für Ihren S3-Quell-Bucket erforderliche CloudTrail Trail erstellt.

   Der folgende Befehl verwendet `--name` und `--s3-bucket-name` zum Erstellen eines Trails mit dem Namen `my-trail` und eines Buckets mit dem Namen `myBucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name myBucket
   ```

2. Rufen Sie den Befehl start-logging auf und beziehen Sie den `--name`-Parameter ein.

   Warum nehme ich diese Änderung vor? Dieser Befehl startet die CloudTrail Protokollierung für Ihren Quell-Bucket und sendet Ereignisse an EventBridge.

   Beispiel:

   Im folgenden Befehl wird `--name` verwendet, um die Protokollierung auf einem Trail mit der Bezeichnung `my-trail` zu starten.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Rufen Sie den Befehl put-event-selectors auf und beziehen Sie die Parameter `--trail-name` und `--event-selectors` ein. Verwenden Sie Event-Selektoren, um anzugeben, dass Ihr Trail Datenereignisse für Ihren Quell-Bucket protokollieren und die Ereignisse an die EventBridge Regel senden soll.

   Warum nehme ich diese Änderung vor? Dieser Befehl filtert Ereignisse.

Beispiel:

Im folgenden Beispielbefehl werden `--trail-name` und `--event-selectors` verwendet, um die Verwaltung von Datenereignissen für einen Quell-Bucket und einen Präfix namens `myBucket/myFolder` anzugeben.

```
aws cloudtrail put-event-selectors --trail-name my-trail --event-selectors
 '[{ "ReadWriteType": "WriteOnly", "IncludeManagementEvents":false,
 "DataResources": [{ "Type": "AWS::S3::Object", "Values": ["arn:aws:s3:::myBucket/
myFolder/file.zip"] }] }]'
```

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Erteilen Sie Berechtigungen EventBridge , die CodePipeline zum Aufrufen der Regel verwendet werden können. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie zu erstellen, damit EventBridge Sie die Servicerolle übernehmen können. Geben Sie ihr den Namen `trustpolicyforEB.json`.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

   b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen dieser Vertrauensrichtlinie zur Rolle werden Berechtigungen für erstellt EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie hier für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Führen Sie den folgenden Befehl aus, um der erstellten `Role-for-MyRule`-Rolle die neue `CodePipeline-Permissions-Policy-for-EB`-Berechtigungsrichtlinie anzufügen.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

   Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „`MyS3SourceRule`“ erstellt.

```
aws events put-rule --name "MyS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"AWS API Call via CloudTrail\"],\"detail\":
{\"eventSource\":[\"s3.amazonaws.com\"],\"eventName\":[\"CopyObject\",\"PutObject
\",\"CompleteMultipartUpload\"],\"requestParameters\":{\"bucketName\":[\"my-bucket
\"],\"key\":[\"my-key\"]}}}
```

```
  --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Um das Objekt CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die `--targets` Parameter `--rule` und an.

   Der folgende Befehl legt fest, dass für die Regel mit dem Namen `MyS3SourceRule` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dies Ziel 1 ist. Der Befehl gibt zudem ein Beispiel ARN für die Pipeline an. Die Pipeline startet, wenn Änderungen im Repository auftreten.

   ```
   aws events put-targets --rule MyS3SourceRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem Texteditor und bearbeiten Sie die Quellphase, indem Sie den Parameter `PollForSourceChanges` für einen Bucket mit dem Namen `storage-bucket`in `false` ändern wie in diesem Beispiel gezeigt.

Warum nehme ich diese Änderung vor? Durch Festlegen dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

```
"configuration": {
    "S3Bucket": "storage-bucket",
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

3. Wenn Sie mit einer Pipeline-Struktur arbeiten, die Sie mit dem Befehl get-pipeline abgerufen haben, müssen Sie die `metadata`-Zeilen aus der JSON-Datei entfernen. Andernfalls kann der update-pipeline-Befehl sie nicht nutzen. Entfernen Sie die `"metadata": { }`-Zeilen und die Felder `"created"`, `"pipelineARN"` und `"updated"`.

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

   ```
   "metadata": {
       "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
       "created": "date",
       "updated": "date"
   },
   ```

   Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

   > **Important**
   >
   > Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

   ```
   aws codepipeline update-pipeline --cli-input-json file://pipeline.json
   ```

   Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den start-pipeline-execution-Befehl, um Ihre Pipeline manuell zu starten.

Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (Vorlage)AWS CloudFormation

Gehen Sie wie folgt vor, um Ihre Pipeline mit einer Amazon S3 S3-Quelle von der Abfrage bis zur ereignisbasierten Änderungserkennung zu bearbeiten.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und fügen dann die folgenden Ressourcen zu Ihrer Vorlage hinzu:

- EventBridge erfordert, dass alle Amazon S3 S3-Ereignisse protokolliert werden müssen. Sie müssen eine AWS CloudTrail Trail-, Bucket- und Bucket-Richtlinie erstellen, die Amazon S3 verwenden kann, um die auftretenden Ereignisse zu protokollieren. Weitere Informationen finden Sie unter Datenereignisse für Trails protokollieren und Verwaltungsereignisse für Trails protokollieren.
- EventBridge Regel und IAM-Rolle, damit dieses Ereignis unsere Pipeline starten kann.

Wenn Sie Ihre Pipelines AWS CloudFormation zur Erstellung und Verwaltung verwenden, enthält Ihre Vorlage Inhalte wie den folgenden.

> **Note**
>
> Die `Configuration`-Eigenschaft in der Quellstufe mit dem Namen `PollForSourceChanges`. Wenn diese Eigenschaft in Ihrer Vorlage nicht enthalten ist, dann wird `PollForSourceChanges` standardmäßig auf `true` festgelegt.

YAML

```
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn: !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                -
                  Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref S3SourceObjectKey
                PollForSourceChanges: true
              RunOrder: 1


...
```

JSON

```
       "AppPipeline": {
           "Type": "AWS::CodePipeline::Pipeline",
           "Properties": {
               "RoleArn": {
                   "Fn::GetAtt": ["CodePipelineServiceRole", "Arn"]
               },
               "Stages": [
                   {
                       "Name": "Source",
                       "Actions": [
                           {
                               "Name": "SourceAction",
```

```
                            "ActionTypeId": {
                                "Category": "Source",
                                "Owner": "AWS",
                                "Version": 1,
                                "Provider": "S3"
                            },
                            "OutputArtifacts": [
                                {
                                    "Name": "SourceOutput"
                                }
                            ],
                            "Configuration": {
                                "S3Bucket": {
                                    "Ref": "SourceBucket"
                                },
                                "S3ObjectKey": {
                                    "Ref": "SourceObjectKey"
                                },
                                "PollForSourceChanges": true
                            },
                            "RunOrder": 1
                        }
                    ]
                },


...
```

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` AWS CloudFormation Ressource`Resources`, um die IAM-Rolle zu konfigurieren, mit der Ihr Ereignis Ihre Pipeline starten kann. Dieser Eintrag erstellt eine Rolle mit zwei Richtlinien:

   - Die erste Richtlinie ermöglicht die Übernahme der Rolle.
   - Die zweite Richtlinie stellt Berechtigungen zum Starten der Pipeline bereit.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen einer `AWS::IAM::Role` Ressource können AWS CloudFormation Sie Berechtigungen für EventBridge erstellen. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

YAML

```
  EventRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
 'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]


...
```

JSON

```
  "EventRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
```

```
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": [
                "events.amazonaws.com"
              ]
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "Path": "/",
      "Policies": [
        {
          "PolicyName": "eb-pipeline-execution",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Action": "codepipeline:StartPipelineExecution",
                "Resource": {
                  "Fn::Join": [
                    "",
                    [
                      "arn:aws:codepipeline:",
                      {
                        "Ref": "AWS::Region"
                      },
                      ":",
                      {
                        "Ref": "AWS::AccountId"
                      },
                      ":",
                      {
                        "Ref": "AppPipeline"
                      }
                    ]
                  ]

...
```

2. Verwenden Sie die `AWS::Events::Rule` AWS CloudFormation Ressource, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis`CopyObject`, `PutObject` das Ihren `CompleteMultipartUpload` Amazon S3 S3-Quell-Bucket überwacht. Fügen Sie darüber hinaus ein Ziel für Ihre Pipeline ein. Wenn `CopyObject`, `PutObject` oder `CompleteMultipartUpload` auftritt, ruft diese Rolle `StartPipelineExecution` in Ihrer Ziel-Pipeline auf.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::Events::Rule` Ressource kann AWS CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

```
EventRule:
  Type: AWS::Events::Rule
  Properties:
    EventPattern:
      source:
        - aws.s3
      detail-type:
        - 'AWS API Call via CloudTrail'
      detail:
        eventSource:
          - s3.amazonaws.com
        eventName:
          - CopyObject
          - PutObject
          - CompleteMultipartUpload
        requestParameters:
          bucketName:
            - !Ref SourceBucket
          key:
            - !Ref SourceObjectKey
    Targets:
      -
        Arn:
          !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
        RoleArn: !GetAtt EventRole.Arn
        Id: codepipeline-AppPipeline
```

```
...
```

JSON

```
"EventRule": {
  "Type": "AWS::Events::Rule",
  "Properties": {
    "EventPattern": {
      "source": [
        "aws.s3"
      ],
      "detail-type": [
        "AWS API Call via CloudTrail"
      ],
      "detail": {
        "eventSource": [
          "s3.amazonaws.com"
        ],
        "eventName": [
          "CopyObject",
          "PutObject",
          "CompleteMultipartUpload"
        ],
        "requestParameters": {
          "bucketName": [
            {
              "Ref": "SourceBucket"
            }
          ],
          "key": [
            {
              "Ref": "SourceObjectKey"
            }
          ]
        }
      }
    },
    "Targets": [
      {
        "Arn": {
          "Fn::Join": [
            "",
```

```
              [
                "arn:aws:codepipeline:",
                {
                  "Ref": "AWS::Region"
                },
                ":",
                {
                  "Ref": "AWS::AccountId"
                },
                ":",
                {
                  "Ref": "AppPipeline"
                }
              ]
            ]
          },
          "RoleArn": {
            "Fn::GetAtt": [
              "EventRole",
              "Arn"
            ]
          },
          "Id": "codepipeline-AppPipeline"
        }
      ]
    }
  }
},

...
```

3. Fügen Sie diesen Ausschnitt zu Ihrer ersten Vorlage hinzu, um Stack-übergreifende Funktionalität zu ermöglichen:

   YAML

```
Outputs:
  SourceBucketARN:
    Description: "S3 bucket ARN that Cloudtrail will use"
    Value: !GetAtt SourceBucket.Arn
    Export:
      Name: SourceBucketARN
```

JSON

```
  "Outputs" : {
    "SourceBucketARN" : {
      "Description" : "S3 bucket ARN that Cloudtrail will use",
      "Value" : { "Fn::GetAtt": ["SourceBucket", "Arn"] },
      "Export" : {
        "Name" : "SourceBucketARN"
      }
    }

...
```

4. Speichern Sie Ihre aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation Konsole.
5. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
6. Laden Sie Ihre aktualisierte Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
7. Wählen Sie Execute (Ausführen).

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

Warum nehme ich diese Änderung vor? Durch Ändern von `PollForSourceChanges` in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
```

JSON

```
{
   "Name": "SourceAction",
   "ActionTypeId": {
     "Category": "Source",
     "Owner": "AWS",
     "Version": 1,
     "Provider": "S3"
   },
   "OutputArtifacts": [
     {
       "Name": "SourceOutput"
     }
   ],
   "Configuration": {
     "S3Bucket": {
       "Ref": "SourceBucket"
     },
```

```
      "S3ObjectKey": {
        "Ref": "SourceObjectKey"
      },
      "PollForSourceChanges": false
    },
    "RunOrder": 1
  }
```

Um eine zweite Vorlage für die CloudTrail Ressourcen Ihrer Amazon S3 S3-Pipeline zu erstellen

- Verwenden Sie in einer separaten Vorlage unter`Resources`, die `AWS::CloudTrail::Trail` AWS CloudFormation Ressourcen `AWS::S3::BucketAWS::S3::BucketPolicy`, und, um eine einfache Bucket-Definition und einen Trail für bereitzustellen CloudTrail.

  Warum nehme ich diese Änderung vor? Angesichts des aktuellen Limits von fünf Trails pro Konto muss der CloudTrail Trail separat erstellt und verwaltet werden. (Siehe Grenzwerte unter AWS CloudTrail.) Sie können jedoch viele Amazon S3 S3-Buckets in einen einzigen Trail aufnehmen, sodass Sie den Trail einmal erstellen und dann bei Bedarf Amazon S3 S3-Buckets für andere Pipelines hinzufügen können. Fügen Sie den folgenden Code in Ihre zweite Beispielvorlagendatei ein.

  YAML

```
###############################################################################
# Prerequisites:
#   - S3 SourceBucket and SourceObjectKey must exist
###############################################################################

Parameters:
  SourceObjectKey:
    Description: 'S3 source artifact'
    Type: String
    Default: SampleApp_Linux.zip

Resources:
  AWSCloudTrailBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref AWSCloudTrailBucket
      PolicyDocument:
```

```
        Version: 2012-10-17
        Statement:
          -
            Sid: AWSCloudTrailAclCheck
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:GetBucketAcl
            Resource: !GetAtt AWSCloudTrailBucket.Arn
          -
            Sid: AWSCloudTrailWrite
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt AWSCloudTrailBucket.Arn, '/
AWSLogs/', !Ref 'AWS::AccountId', '/*' ] ]
            Condition:
              StringEquals:
                s3:x-amz-acl: bucket-owner-full-control
  AWSCloudTrailBucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
  AwsCloudTrail:
    DependsOn:
      - AWSCloudTrailBucketPolicy
    Type: AWS::CloudTrail::Trail
    Properties:
      S3BucketName: !Ref AWSCloudTrailBucket
      EventSelectors:
        -
          DataResources:
            -
              Type: AWS::S3::Object
              Values:
                - !Join [ '', [ !ImportValue SourceBucketARN, '/', !Ref
 SourceObjectKey ] ]
          ReadWriteType: WriteOnly
          IncludeManagementEvents: false
      IncludeGlobalServiceEvents: true
      IsLogging: true
      IsMultiRegionTrail: true
```

```
...
```

JSON

```
{
  "Parameters": {
    "SourceObjectKey": {
      "Description": "S3 source artifact",
      "Type": "String",
      "Default": "SampleApp_Linux.zip"
    }
  },
  "Resources": {
    "AWSCloudTrailBucket": {
      "Type": "AWS::S3::Bucket",
        "DeletionPolicy": "Retain"
    },
    "AWSCloudTrailBucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {
          "Ref": "AWSCloudTrailBucket"
        },
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "AWSCloudTrailAclCheck",
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "cloudtrail.amazonaws.com"
                ]
              },
              "Action": "s3:GetBucketAcl",
              "Resource": {
                "Fn::GetAtt": [
                  "AWSCloudTrailBucket",
                  "Arn"
                ]
              }
```

```
            },
            {
              "Sid": "AWSCloudTrailWrite",
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "cloudtrail.amazonaws.com"
                ]
              },
              "Action": "s3:PutObject",
              "Resource": {
                "Fn::Join": [
                  "",
                  [
                    {
                      "Fn::GetAtt": [
                        "AWSCloudTrailBucket",
                        "Arn"
                      ]
                    },
                    "/AWSLogs/",
                    {
                      "Ref": "AWS::AccountId"
                    },
                    "/*"
                  ]
                ]
              },
              "Condition": {
                "StringEquals": {
                  "s3:x-amz-acl": "bucket-owner-full-control"
                }
              }
            }
          ]
        }
      }
    },
    "AwsCloudTrail": {
      "DependsOn": [
        "AWSCloudTrailBucketPolicy"
      ],
      "Type": "AWS::CloudTrail::Trail",
      "Properties": {
```

```
        "S3BucketName": {
          "Ref": "AWSCloudTrailBucket"
        },
        "EventSelectors": [
          {
            "DataResources": [
              {
                "Type": "AWS::S3::Object",
                "Values": [
                  {
                    "Fn::Join": [
                      "",
                      [
                        {
                          "Fn::ImportValue": "SourceBucketARN"
                        },
                        "/",
                        {
                          "Ref": "SourceObjectKey"
                        }
                      ]
                    ]
                  }
                ]
              }
            ],
            "ReadWriteType": "WriteOnly",
            "IncludeManagementEvents": false
          }
        ],
        "IncludeGlobalServiceEvents": true,
        "IsLogging": true,
        "IsMultiRegionTrail": true
      }
    }
  }
}

...
```

Example

Wenn Sie diese Ressourcen erstellen, wird Ihre Pipeline ausgelöst, wenn Dateien in Ihrem Repository erstellt oder aktualisiert werden. AWS CloudFormation

> **Note**
>
> Hören Sie hier nicht auf. Obwohl Ihre Pipeline erstellt wurde, müssen Sie eine zweite AWS CloudFormation Vorlage für Ihre Amazon S3 S3-Pipeline erstellen. Wenn Sie die zweite Vorlage nicht erstellen, enthält Ihre Pipeline keine Funktionalität für die Änderungserkennung.

YAML

```
Resources:
  SourceBucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
  CodePipelineArtifactStoreBucket:
    Type: AWS::S3::Bucket
  CodePipelineArtifactStoreBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref CodePipelineArtifactStoreBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Sid: DenyUnEncryptedObjectUploads
            Effect: Deny
            Principal: '*'
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*' ] ]
            Condition:
              StringNotEquals:
                s3:x-amz-server-side-encryption: aws:kms
          -
            Sid: DenyInsecureConnections
            Effect: Deny
```

```
            Principal: '*'
            Action: s3:*
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*' ] ]
            Condition:
              Bool:
                aws:SecureTransport: false
  CodePipelineServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - codepipeline.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: AWS-CodePipeline-Service-3
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action:
                  - codecommit:CancelUploadArchive
                  - codecommit:GetBranch
                  - codecommit:GetCommit
                  - codecommit:GetUploadArchiveStatus
                  - codecommit:UploadArchive
                Resource: 'resource_ARN'
              -
                Effect: Allow
                Action:
                  - codedeploy:CreateDeployment
                  - codedeploy:GetApplicationRevision
                  - codedeploy:GetDeployment
                  - codedeploy:GetDeploymentConfig
                  - codedeploy:RegisterApplicationRevision
                Resource: 'resource_ARN'
```

```
            -
              Effect: Allow
              Action:
                - codebuild:BatchGetBuilds
                - codebuild:StartBuild
              Resource: 'resource_ARN'
            -
              Effect: Allow
              Action:
                - devicefarm:ListProjects
                - devicefarm:ListDevicePools
                - devicefarm:GetRun
                - devicefarm:GetUpload
                - devicefarm:CreateUpload
                - devicefarm:ScheduleRun
              Resource: 'resource_ARN'
            -
              Effect: Allow
              Action:
                - lambda:InvokeFunction
                - lambda:ListFunctions
              Resource: 'resource_ARN'
            -
              Effect: Allow
              Action:
                - iam:PassRole
              Resource: 'resource_ARN'
            -
              Effect: Allow
              Action:
                - elasticbeanstalk:*
                - ec2:*
                - elasticloadbalancing:*
                - autoscaling:*
                - cloudwatch:*
                - s3:*
                - sns:*
                - cloudformation:*
                - rds:*
                - sqs:*
                - ecs:*
              Resource: 'resource_ARN'
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
```

```
    Properties:
      Name: s3-events-pipeline
      RoleArn:
        !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref SourceObjectKey
                PollForSourceChanges: false
              RunOrder: 1
        -
          Name: Beta
          Actions:
            -
              Name: BetaAction
              InputArtifacts:
                - Name: SourceOutput
              ActionTypeId:
                Category: Deploy
                Owner: AWS
                Version: 1
                Provider: CodeDeploy
              Configuration:
                ApplicationName: !Ref ApplicationName
                DeploymentGroupName: !Ref BetaFleet
              RunOrder: 1
      ArtifactStore:
        Type: S3
        Location: !Ref CodePipelineArtifactStoreBucket
  EventRole:
    Type: AWS::IAM::Role
    Properties:
```

```
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
 EventRule:
    Type: AWS::Events::Rule
    Properties:
      EventPattern:
        source:
          - aws.s3
        detail-type:
          - 'AWS API Call via CloudTrail'
        detail:
          eventSource:
            - s3.amazonaws.com
          eventName:
            - PutObject
            - CompleteMultipartUpload
          resources:
            ARN:
              - !Join [ '', [ !GetAtt SourceBucket.Arn, '/', !Ref
SourceObjectKey ] ]
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
```

```
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline

Outputs:
  SourceBucketARN:
    Description: "S3 bucket ARN that Cloudtrail will use"
    Value: !GetAtt SourceBucket.Arn
    Export:
      Name: SourceBucketARN
```

JSON

```
    "Resources": {
        "SourceBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "VersioningConfiguration": {
                    "Status": "Enabled"
                }
            }
        },
        "CodePipelineArtifactStoreBucket": {
            "Type": "AWS::S3::Bucket"
        },
        "CodePipelineArtifactStoreBucketPolicy": {
            "Type": "AWS::S3::BucketPolicy",
            "Properties": {
                "Bucket": {
                    "Ref": "CodePipelineArtifactStoreBucket"
                },
                "PolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Sid": "DenyUnEncryptedObjectUploads",
                            "Effect": "Deny",
                            "Principal": "*",
                            "Action": "s3:PutObject",
                            "Resource": {
                                "Fn::Join": [
                                    "",
                                    [
                                        {
```

```
                                        "Fn::GetAtt": [
                                            "CodePipelineArtifactStoreBucket",
                                            "Arn"
                                        ]
                                    },
                                    "/*"
                                ]
                            ]
                        },
                        "Condition": {
                            "StringNotEquals": {
                                "s3:x-amz-server-side-encryption": "aws:kms"
                            }
                        }
                    },
                    {
                        "Sid": "DenyInsecureConnections",
                        "Effect": "Deny",
                        "Principal": "*",
                        "Action": "s3:*",
                        "Resource": {
                            "Fn::Join": [
                                "",
                                [
                                    {
                                        "Fn::GetAtt": [
                                            "CodePipelineArtifactStoreBucket",
                                            "Arn"
                                        ]
                                    },
                                    "/*"
                                ]
                            ]
                        },
                        "Condition": {
                            "Bool": {
                                "aws:SecureTransport": false
                            }
                        }
                    }
                ]
            }
        }
    },
```

```
        "CodePipelineServiceRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "codepipeline.amazonaws.com"
                                ]
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "Path": "/",
                "Policies": [
                    {
                        "PolicyName": "AWS-CodePipeline-Service-3",
                        "PolicyDocument": {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "codecommit:CancelUploadArchive",
                                        "codecommit:GetBranch",
                                        "codecommit:GetCommit",
                                        "codecommit:GetUploadArchiveStatus",
                                        "codecommit:UploadArchive"
                                    ],
                                    "Resource": "resource_ARN"
                                },
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "codedeploy:CreateDeployment",
                                        "codedeploy:GetApplicationRevision",
                                        "codedeploy:GetDeployment",
                                        "codedeploy:GetDeploymentConfig",
                                        "codedeploy:RegisterApplicationRevision"
                                    ],
```

```
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:BatchGetBuilds",
                "codebuild:StartBuild"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "devicefarm:ListProjects",
                "devicefarm:ListDevicePools",
                "devicefarm:GetRun",
                "devicefarm:GetUpload",
                "devicefarm:CreateUpload",
                "devicefarm:ScheduleRun"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:ListFunctions"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticbeanstalk:*",
                "ec2:*",
                "elasticloadbalancing:*",
                "autoscaling:*",
```

```
                                "cloudwatch:*",
                                "s3:*",
                                "sns:*",
                                "cloudformation:*",
                                "rds:*",
                                "sqs:*",
                                "ecs:*"
                            ],
                            "Resource": "resource_ARN"
                        }
                    ]
                }
            }
        ]
    }
},
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
    "Properties": {
        "Name": "s3-events-pipeline",
        "RoleArn": {
            "Fn::GetAtt": [
                "CodePipelineServiceRole",
                "Arn"
            ]
        },
        "Stages": [
            {
                "Name": "Source",
                "Actions": [
                    {
                        "Name": "SourceAction",
                        "ActionTypeId": {
                            "Category": "Source",
                            "Owner": "AWS",
                            "Version": 1,
                            "Provider": "S3"
                        },
                        "OutputArtifacts": [
                            {
                                "Name": "SourceOutput"
                            }
                        ],
                        "Configuration": {
```

```
                            "S3Bucket": {
                                "Ref": "SourceBucket"
                            },
                            "S3ObjectKey": {
                                "Ref": "SourceObjectKey"
                            },
                            "PollForSourceChanges": false
                        },
                        "RunOrder": 1
                    }
                ]
            },
            {
                "Name": "Beta",
                "Actions": [
                    {
                        "Name": "BetaAction",
                        "InputArtifacts": [
                            {
                                "Name": "SourceOutput"
                            }
                        ],
                        "ActionTypeId": {
                            "Category": "Deploy",
                            "Owner": "AWS",
                            "Version": 1,
                            "Provider": "CodeDeploy"
                        },
                        "Configuration": {
                            "ApplicationName": {
                                "Ref": "ApplicationName"
                            },
                            "DeploymentGroupName": {
                                "Ref": "BetaFleet"
                            }
                        },
                        "RunOrder": 1
                    }
                ]
            }
        ],
        "ArtifactStore": {
            "Type": "S3",
            "Location": {
```

```
                    "Ref": "CodePipelineArtifactStoreBucket"
                }
            }
        }
    },
    "EventRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "events.amazonaws.com"
                            ]
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "eb-pipeline-execution",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "codepipeline:StartPipelineExecution",
                                "Resource": {
                                    "Fn::Join": [
                                        "",
                                        [
                                            "arn:aws:codepipeline:",
                                            {
                                                "Ref": "AWS::Region"
                                            },
                                            ":",
                                            {
                                                "Ref": "AWS::AccountId"
                                            },
```

```
                                        ":",
                                        {
                                            "Ref": "AppPipeline"
                                        }
                                    ]
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    }
},
"EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
        "EventPattern": {
            "source": [
                "aws.s3"
            ],
            "detail-type": [
                "AWS API Call via CloudTrail"
            ],
            "detail": {
                "eventSource": [
                    "s3.amazonaws.com"
                ],
                "eventName": [
                    "PutObject",
                    "CompleteMultipartUpload"
                ],
                "resources": {
                    "ARN": [
                        {
                            "Fn::Join": [
                                "",
                                [
                                    {
                                        "Fn::GetAtt": [
                                            "SourceBucket",
                                            "Arn"
                                        ]
                                    },
```

```
                                        "/",
                                        {
                                            "Ref": "SourceObjectKey"
                                        }
                                    ]
                                ]
                            }
                        ]
                    }
                }
            },
            "Targets": [
                {
                    "Arn": {
                        "Fn::Join": [
                            "",
                            [
                                "arn:aws:codepipeline:",
                                {
                                    "Ref": "AWS::Region"
                                },
                                ":",
                                {
                                    "Ref": "AWS::AccountId"
                                },
                                ":",
                                {
                                    "Ref": "AppPipeline"
                                }
                            ]
                        ]
                    },
                    "RoleArn": {
                        "Fn::GetAtt": [
                            "EventRole",
                            "Arn"
                        ]
                    },
                    "Id": "codepipeline-AppPipeline"
                }
            ]
        }
    }
},
```

```
    "Outputs" : {
        "SourceBucketARN" : {
            "Description" : "S3 bucket ARN that Cloudtrail will use",
            "Value" : { "Fn::GetAtt": ["SourceBucket", "Arn"] },
            "Export" : {
                "Name" : "SourceBucketARN"
            }
        }
    }
}


...
```

Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Verbindungen

Sie können eine Quellaktion der GitHub Version 1 migrieren, um Verbindungen für Ihr externes Repository zu verwenden. Dies ist die empfohlene Methode zur Erkennung von Änderungen für Pipelines mit einer Quellaktion der GitHub Version 1.

Für eine Pipeline mit einer Quellaktion für GitHub Version 1 empfehlen wir, die Pipeline so zu ändern, dass eine Aktion der GitHub Version 2 verwendet wird, sodass die Änderungserkennung automatisiert wird. AWS CodeConnections Weitere Informationen zum Arbeiten mit Verbindungen finden Sie unterGitHub Verbindungen.

Stellen Sie eine Verbindung zu GitHub (Konsole) her

Sie können die Konsole verwenden, um eine Verbindung zu herzustellen GitHub.

Schritt 1: Ersetzen Sie Ihre GitHub Aktion aus Version 1

Verwenden Sie die Pipeline-Bearbeitungsseite, um Ihre Aktion aus Version 1 durch eine GitHub Aktion aus Version 2 GitHub zu ersetzen.

Um Ihre GitHub Aktion aus Version 1 zu ersetzen

1. Melden Sie sich bei der CodePipeline Konsole an.

2. Wählen Sie Ihre Pipeline und dann Bearbeiten aus. Wählen Sie in Ihrer Quellstufe die Option Phase bearbeiten aus. Es wird eine Meldung angezeigt, in der empfohlen wird, Ihre Aktion zu aktualisieren.
3. Wählen Sie unter Aktionsanbieter die Option GitHub (Version 2) aus.
4. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Option Connect aus GitHub. Fahren Sie mit Schritt 2 fort: Verbindung herstellen zu GitHub.
   - Wenn Sie bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Verbindung aus. Fahren Sie mit Schritt 3 fort: Speichern Sie die Quellaktion für Ihre Verbindung.

Schritt 2: Stellen Sie eine Verbindung her zu GitHub

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die GitHub Seite Connect angezeigt.

Um eine Verbindung herzustellen zu GitHub

1. Unter GitHub Verbindungseinstellungen wird Ihr Verbindungsname unter Verbindungsname angezeigt.

   Wählen Sie unter GitHub Apps eine App-Installation aus oder wählen Sie Neue App installieren, um eine zu erstellen.

   > **Note**
   >
   > Sie installieren eine App für alle Verbindungen mit einem bestimmten Anbieter. Wenn Sie die GitHub App bereits installiert haben, wählen Sie sie aus und überspringen Sie diesen Schritt.

2. Wenn die Autorisierungsseite für GitHub angezeigt wird, melden Sie sich mit Ihren Anmeldeinformationen an und wählen Sie dann, ob Sie fortfahren möchten.
3. Auf der App-Installationsseite wird eine Meldung angezeigt, dass die AWS CodeStar App versucht, eine Verbindung zu Ihrem GitHub Konto herzustellen.

> **Note**
>
> Sie installieren die App nur einmal für jedes GitHub Konto. Wenn Sie die App schon einmal installiert haben, können Sie Configure (Konfiguration) wählen und mit einer Änderungsseite für die App-Installation fortfahren. Alternativ kommen Sie über die Schaltfläche „Back“ (Zurück) zur Konsole zurück.

4. Wählen Sie auf der AWS CodeStar Seite „Installieren“ die Option „Installieren“.
5. Auf der GitHub Seite Connect wird die Verbindungs-ID für Ihre neue Installation angezeigt. Wählen Sie Connect aus.

Schritt 3: Speichern Sie Ihre GitHub Quellaktion

Vervollständigen Sie Ihre Aktualisierungen auf der Seite Aktion bearbeiten, um Ihre neue Quellaktion zu speichern.

Um Ihre GitHub Quellaktion zu speichern

1. Geben Sie unter Repository den Namen Ihres Drittanbieter-Repositorys ein. Geben Sie im Feld Branch den Branch ein, in dem Ihre Pipeline Quelländerungen erkennen soll.

> **Note**
>
> Geben Sie im Feld Repository `owner-name/repository-name` wie in diesem Beispiel gezeigt ein:
>
> ```
> my-account/my-repository
> ```

2. Wählen Sie unter Ausgabeartefaktformat das Format für Ihre Artefakte aus.
   - Um die Ausgabeartefakte der GitHub Aktion mit der Standardmethode zu speichern, wählen Sie CodePipelineStandard. Die Aktion greift auf die Dateien aus dem GitHub Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.
   - Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone (Vollständiger Klon). Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschriebenFügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, Enterprise Server oder .com GitHub hinzu GitHub GitLab. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiges Klonen verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

3. Unter Ausgabeartefakte können Sie den Namen des Ausgabeartefakts für diese Aktion beibehalten, z. B. `SourceArtifact` Wählen Sie Fertig, um die Aktionsseite Bearbeiten zu schließen.
4. Wählen Sie „Fertig“, um die Seite zur Bearbeitung der Phase zu schließen. Wählen Sie Speichern, um die Seite zur Bearbeitung der Pipeline zu schließen.

Verbindung herstellen zu GitHub (CLI)

Sie können das AWS Command Line Interface (AWS CLI) verwenden, um eine Verbindung zu herzustellen GitHub.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über AWS CLI oder AWS CloudFormation erstellt wurde, hat standardmäßig `PENDING` den Status. Nachdem Sie eine Verbindung mit der CLI hergestellt haben oder verwenden Sie die Konsole AWS CloudFormation, um die Verbindung so zu bearbeiten, dass sie ihren Status festlegt`AVAILABLE`.

Um eine Verbindung herzustellen zu GitHub

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie den AWS CLI , um den create-connection Befehl auszuführen, und geben Sie dabei `--provider-type` und `--connection-name` für Ihre Verbindung an. In diesem Beispiel lautet der Name des Drittanbieters `GitHub` und der angegebene Verbindungsname `MyConnection`.

```
aws codeconnections create-connection --provider-type GitHub --connection-name
 MyConnection
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er die ARN-Informationen der Verbindung ähnlich der folgenden zurück.

```
{
    "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Verwenden Sie die Konsole, um die Verbindung fertigzustellen.

Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Webhooks

Sie können Ihre Pipeline migrieren, um Webhooks zu verwenden, um Änderungen in Ihrem Quell-Repository zu erkennen. GitHub Diese Migration zu Webhooks gilt nur für die Aktion GitHub Version 1.

- Konsole: Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (Konsole)
- CLI: Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (CLI)
- AWS CloudFormation: Aktualisierungspipelines für Push-Ereignisse (Quellaktionen der GitHub Version 1) (AWS CloudFormation Vorlage)

Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (Konsole)

Sie können die CodePipeline Konsole verwenden, um Ihre Pipeline so zu aktualisieren, dass Webhooks verwendet werden, um Änderungen in Ihrem Quell-Repository zu erkennen. CodeCommit

Gehen Sie wie folgt vor, um eine Pipeline zu bearbeiten, die stattdessen Polling (regelmäßige Prüfungen) verwendet EventBridge . Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen Sie eine Pipeline in CodePipeline.

Wenn Sie die Konsole verwenden, wird der Parameter `PollForSourceChanges` für Ihre Pipeline für Sie geändert. Der GitHub Webhook wird für Sie erstellt und registriert.

So bearbeiten Sie die Quellphase einer Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Auf diese Weise wird eine detaillierte Ansicht der Pipeline geöffnet (einschließlich des Status der Aktionen in den einzelnen Stufen der Pipeline).
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie in der Stufe Edit stage (Stufe bearbeiten) das Bearbeitungssymbol für die Quellaktion aus.
5. Erweitern Sie die Optionen zur Änderungserkennung und wählen Sie Amazon CloudWatch Events verwenden, um meine Pipeline automatisch zu starten, wenn eine Änderung eintritt (empfohlen).

   Es wird eine Meldung mit dem Hinweis angezeigt, dass ein Webhook CodePipeline erstellt AWS CodePipeline wird GitHub , um Quellenänderungen zu erkennen: erstellt einen Webhook für Sie. Sie können sich in den folgenden Optionen abmelden. Wählen Sie Aktualisieren. CodePipeline Erzeugt zusätzlich zum Webhook Folgendes:

   - Ein Geheimnis, das nach dem Zufallsprinzip generiert wurde und zur Autorisierung der Verbindung verwendet wird. GitHub
   - Die Webhook-URL wird generiert, wobei der öffentliche Endpunkt für die Region verwendet wird.

   CodePipeline registriert den Webhook bei. GitHub Damit wird die URL für den Empfang von Repository-Ereignissen abonniert.
6. Wenn Sie die Bearbeitung der Pipeline abgeschlossen haben, wählen Sie Save pipeline changes aus, um zur Übersichtsseite zurückzukehren.

   Es wird eine Meldung mit dem Namen des für Ihre Pipeline zu erstellenden Webhooks angezeigt. Wählen Sie Save and continue aus.
7. Um Ihre Aktion zu testen, geben Sie eine Änderung frei, indem Sie den verwenden AWS CLI , um eine Änderung an die Quelle zu übertragen, die in der Quellphase der Pipeline angegeben wurde.

Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (CLI)

Führen Sie die folgenden Schritte aus, um eine Pipeline, die Abfragen (periodische Prüfungen) verwendet, zu bearbeiten, damit sie stattdessen einen Webhook verwendet. Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen Sie eine Pipeline in CodePipeline.

Um eine ereignisgesteuerte Pipeline zu erstellen, bearbeiten Sie den Parameter `PollForSourceChanges` Ihrer Pipeline und erstellen anschließend die folgenden Ressourcen manuell:

- GitHub Webhook und Autorisierungsparameter

So erstellen und registrieren Sie Ihren Webhook

> **Note**
>
> Wenn Sie die CLI verwenden oder AWS CloudFormation eine Pipeline erstellen und einen Webhook hinzufügen, müssen Sie regelmäßige Prüfungen deaktivieren. Um die periodischen Prüfungen zu deaktivieren, müssen Sie explizit den `PollForSourceChanges`-Parameter hinzufügen und auf „false“ setzen, wie in der abschließenden Prozedur unten. Andernfalls ist die Standardeinstellung für eine CLI oder AWS CloudFormation Pipeline der `PollForSourceChanges` Standardwert true und wird nicht in der Ausgabe der Pipeline-Struktur angezeigt. Weitere Hinweise zu PollForSourceChanges Standardwerten finden Sie unter. Standardeinstellungen für den Parameter PollForSourceChanges

1. Erstellen Sie in einem Texteditor eine JSON-Datei für den Webhook, den Sie erstellen möchten, und speichern Sie sie. Verwenden Sie diese Beispieldatei für einen Webhook mit dem Namen `my-webhook`:

```
{
    "webhook": {
        "name": "my-webhook",
  "targetPipeline": "pipeline_name",
  "targetAction": "source_action_name",
  "filters": [{
      "jsonPath": "$.ref",
      "matchEquals": "refs/heads/{Branch}"
```

```
  }],
  "authentication": "GITHUB_HMAC",
  "authenticationConfiguration": {
      "SecretToken": "secret"
  }
    }
}
```

2. Rufen Sie den Befehl put-webhook auf und beziehen Sie die Parameter `--cli-input` und `--region` ein.

   Der folgende Beispielbefehl erstellt einen Webhook mit der JSON-Datei `webhook_json`.

```
aws codepipeline put-webhook --cli-input-json file://webhook_json.json --region
 "eu-central-1"
```

3. In der in diesem Beispiel gezeigten Ausgabe für einen Webhook mit dem Namen `my-webhook` werden die URL und der ARN zurückgegeben.

```
{
    "webhook": {
        "url": "https://webhooks.domain.com/
trigger111111111EXAMPLE1111111111111111",
        "definition": {
            "authenticationConfiguration": {
                "SecretToken": "secret"
            },
            "name": "my-webhook",
            "authentication": "GITHUB_HMAC",
            "targetPipeline": "pipeline_name",
            "targetAction": "Source",
            "filters": [
                {
                    "jsonPath": "$.ref",
                    "matchEquals": "refs/heads/{Branch}"
                }
            ]
        },
        "arn": "arn:aws:codepipeline:eu-central-1:ACCOUNT_ID:webhook:my-webhook"
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
```

```
    }]
}
```

In diesem Beispiel wird dem Webhook Tagging hinzugefügt, indem dem Webhook der Tag-Schlüssel `Project` und der Wert `ProjectA` hinzugefügt werden. Weitere Informationen zum Markieren von Ressourcen in finden Sie CodePipeline unter. Markieren von Ressourcen

4. Rufen Sie den Befehl register-webhook-with-third-party auf und beziehen Sie den `--webhook-name`-Parameter ein.

   Der folgende Beispielbefehl registriert einen Webhook mit dem Namen `my-webhook`.

   ```
   aws codepipeline register-webhook-with-third-party --webhook-name my-webhook
   ```

So bearbeiten Sie den Parameter Ihrer Pipeline PollForSourceChanges

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Für eine Pipeline mit dem Namen `MyFirstPipeline` würden Sie beispielsweise den folgenden Befehl verwenden:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.
2. Öffnen Sie die JSON-Datei in einem beliebigen Texteditor und bearbeiten Sie die Quellstufe, indem Sie den Parameter `PollForSourceChanges` ändern oder hinzufügen. In diesem Beispiel für ein Repository mit dem Namen `UserGitHubRepo` ist der Parameter auf `false` festgelegt.

Warum nehme ich diese Änderung vor? Wenn Sie diesen Parameter ändern, werden regelmäßige Überprüfungen deaktiviert, sodass Sie nur die ereignisbasierte Änderungserkennung verwenden können.

```
"configuration": {
    "Owner": "name",
    "Repo": "UserGitHubRepo",
    "PollForSourceChanges": "false",
    "Branch": "main",
    "OAuthToken": "****"
},
```

3. Falls Sie mit der Pipeline-Struktur arbeiten, die mithilfe des get-pipeline-Befehls abgerufen wurde, müssen Sie die Struktur in der JSON-Datei bearbeiten, indem Sie die `metadata`-Zeilen aus der Datei entfernen. Andernfalls kann der update-pipeline-Befehl sie nicht nutzen. Entfernen Sie den Abschnitt `"metadata"` aus der Pipeline-Struktur in der JSON-Datei, einschließlich `{ }` und der Felder `"created"`, `"pipelineARN"` und `"updated"`.

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

   ```
   "metadata": {
       "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
       "created": "date",
       "updated": "date"
   },
   ```

   Speichern Sie die Datei.

4. Führen Sie den Befehl update-pipeline aus, um die Änderungen zu übernehmen. Geben Sie die Pipeline-JSON-Datei dabei folgendermaßen an:

   > **Important**
   >
   > Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

   ```
   aws codepipeline update-pipeline --cli-input-json file://pipeline.json
   ```

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den start-pipeline-execution-Befehl, um Ihre Pipeline manuell zu starten.

Aktualisierungspipelines für Push-Ereignisse (Quellaktionen der GitHub Version 1) (AWS CloudFormation Vorlage)

Gehen Sie wie folgt vor, um Ihre Pipeline (mit einer GitHub Quelle) von regelmäßigen Überprüfungen (Polling) bis hin zur ereignisbasierten Änderungserkennung mithilfe von Webhooks zu aktualisieren.

Um eine ereignisgesteuerte Pipeline mit zu erstellen AWS CodeCommit, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und fügen dann eine GitHub Webhook-Ressource zu Ihrer Vorlage hinzu.

Wenn Sie AWS CloudFormation Ihre Pipelines erstellen und verwalten, hat Ihre Vorlage Inhalte wie den folgenden.

> **Note**
>
> Beachten Sie die `PollForSourceChanges`-Konfigurationseigenschaft in der Quellstufe. Wenn diese Eigenschaft in Ihrer Vorlage nicht enthalten ist, dann wird `PollForSourceChanges` standardmäßig auf `true` festgelegt.

YAML

```
Resources:
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      Name: github-polling-pipeline
```

```
      RoleArn:
        !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: ThirdParty
                Version: 1
                Provider: GitHub
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                Owner: !Ref GitHubOwner
                Repo: !Ref RepositoryName
                Branch: !Ref BranchName
                OAuthToken:
 {{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
 PollForSourceChanges: true
              RunOrder: 1


...
```

JSON

```
        "AppPipeline": {
            "Type": "AWS::CodePipeline::Pipeline",
            "Properties": {
                "Name": "github-polling-pipeline",
                "RoleArn": {
                    "Fn::GetAtt": [
                        "CodePipelineServiceRole",
                        "Arn"
                    ]
                },
                "Stages": [
                    {
                        "Name": "Source",
                        "Actions": [
```

```
                            {
                                "Name": "SourceAction",
                                "ActionTypeId": {
                                    "Category": "Source",
                                    "Owner": "ThirdParty",
                                    "Version": 1,
                                    "Provider": "GitHub"
                                },
                                "OutputArtifacts": [
                                    {
                                        "Name": "SourceOutput"
                                    }
                                ],
                                "Configuration": {
                                    "Owner": {
                                        "Ref": "GitHubOwner"
                                    },
                                    "Repo": {
                                        "Ref": "RepositoryName"
                                    },
                                    "Branch": {
                                        "Ref": "BranchName"
                                    },
                                    "OAuthToken":
 "{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}",
                                    "PollForSourceChanges": true
                                },
                                "RunOrder": 1
                            }
                        ]
                    },


...
```

So fügen Sie Parameter hinzu und erstellen einen Webhook in Ihrer Vorlage

Wir empfehlen dringend, dass Sie diese AWS Secrets Manager zum Speichern Ihrer Anmeldeinformationen verwenden. Wenn Sie Secrets Manager verwenden, müssen Sie Ihre geheimen Parameter bereits in Secrets Manager konfiguriert und gespeichert haben. In diesem Beispiel werden dynamische Verweise auf Secrets Manager für die GitHub Anmeldeinformationen

für Ihren Webhook verwendet. Weitere Informationen finden Sie unter Verwenden von dynamischen Referenzen zum Angeben von Vorlagenwerten.

> **Important**
>
> Geben Sie bei der Übergabe von Secret-Parametern den Wert nicht direkt in die Vorlage ein. Der Wert wird als Klartext gerendert und ist daher lesbar. Verwenden Sie aus Sicherheitsgründen keinen Klartext in Ihrer AWS CloudFormation Vorlage, um Ihre Anmeldeinformationen zu speichern.

Wenn Sie die CLI verwenden oder AWS CloudFormation eine Pipeline erstellen und einen Webhook hinzufügen, müssen Sie regelmäßige Prüfungen deaktivieren.

> **Note**
>
> Um die periodischen Prüfungen zu deaktivieren, müssen Sie explizit den `PollForSourceChanges`-Parameter hinzufügen und auf „false“ setzen, wie in der abschließenden Prozedur unten. Andernfalls ist die Standardeinstellung für eine CLI oder AWS CloudFormation Pipeline der `PollForSourceChanges` Standardwert true und wird nicht in der Ausgabe der Pipeline-Struktur angezeigt. Weitere Hinweise zu PollForSourceChanges Standardwerten finden Sie unter. Standardeinstellungen für den Parameter PollForSourceChanges

1. Fügen Sie in der Vorlage unter `Resources` Ihre Parameter hinzu:

   YAML

   ```
   Parameters:
           GitHubOwner:
             Type: String

   ...
   ```

   JSON

   ```
   {
    "Parameters": {
   ```

```
  "BranchName": {
   "Description": "GitHub branch name",
   "Type": "String",
   "Default": "main"
  },
  "GitHubOwner": {
   "Type": "String"
  },

...
```

2. Verwenden Sie die `AWS::CodePipeline::Webhook` AWS CloudFormation Ressource, um einen Webhook hinzuzufügen.

> **Note**
>
> Die angegebene `TargetAction` muss mit der in der Pipeline definierten `Name`-Eigenschaft Ihrer Quellaktion übereinstimmen.

Wenn auf gesetzt `RegisterWithThirdParty` ist`true`, stellen Sie sicher, dass der mit dem verknüpfte Benutzer die erforderlichen Bereiche festlegen `OAuthToken` kann. GitHub Das Token und der Webhook erfordern die folgenden Bereiche: GitHub

- `repo`: Wird für die vollständige Kontrolle in Bezug auf das Lesen und Einfügen von Artefakten aus öffentlichen und privaten Repositorys in eine Pipeline verwendet.
- `admin:repo_hook`: Wird für die vollständige Kontrolle in Bezug auf Repository-Hooks verwendet.

Andernfalls wird ein GitHub 404-Fehler zurückgegeben. Weitere Informationen zum zurückgegebenen Fehler 404 erhalten Sie unter https://help.github.com/articles/about-webhooks.

YAML

```
AppPipelineWebhook:
  Type: AWS::CodePipeline::Webhook
  Properties:
    Authentication: GITHUB_HMAC
    AuthenticationConfiguration:
```

```
        SecretToken:
 {{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
      Filters:
        -
          JsonPath: "$.ref"
          MatchEquals: refs/heads/{Branch}
      TargetPipeline: !Ref AppPipeline
      TargetAction: SourceAction
      Name: AppPipelineWebhook
      TargetPipelineVersion: !GetAtt AppPipeline.Version
      RegisterWithThirdParty: true


...
```

JSON

```
"AppPipelineWebhook": {
    "Type": "AWS::CodePipeline::Webhook",
    "Properties": {
        "Authentication": "GITHUB_HMAC",
        "AuthenticationConfiguration": {
            "SecretToken":
 "{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}"
        },
        "Filters": [{
            "JsonPath": "$.ref",
            "MatchEquals": "refs/heads/{Branch}"
        }],
        "TargetPipeline": {
            "Ref": "AppPipeline"
        },
        "TargetAction": "SourceAction",
        "Name": "AppPipelineWebhook",
        "TargetPipelineVersion": {
            "Fn::GetAtt": [
              "AppPipeline",
              "Version"
            ]
        },
        "RegisterWithThirdParty": true
    }
},
```

```
...
```

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie dann die AWS CloudFormation Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf "false" fest.

  Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

  YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
```

```
              ActionTypeId:
                Category: Source
                Owner: ThirdParty
                Version: 1
                Provider: GitHub
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                Owner: !Ref GitHubOwner
                Repo: !Ref RepositoryName
                Branch: !Ref BranchName
                OAuthToken:
{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
                PollForSourceChanges: false
              RunOrder: 1
```

JSON

```
{
   "Name": "Source",
   "Actions": [{
 "Name": "SourceAction",
 "ActionTypeId": {
     "Category": "Source",
     "Owner": "ThirdParty",
     "Version": 1,
     "Provider": "GitHub"
},
"OutputArtifacts": [{
    "Name": "SourceOutput"
}],
"Configuration": {
    "Owner": {
  "Ref": "GitHubOwner"
    },
    "Repo": {
  "Ref": "RepositoryName"
    },
    "Branch": {
        "Ref": "BranchName"
    },
    "OAuthToken":
"{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}",
```

```
        PollForSourceChanges: false
    },
    "RunOrder": 1
       }]
```

Example

Wenn Sie diese Ressourcen mit erstellen AWS CloudFormation, wird der definierte Webhook im angegebenen GitHub Repository erstellt. Ihre Pipeline wird bei einem Commit ausgelöst.

YAML

```
Parameters:
  GitHubOwner:
    Type: String

Resources:
  AppPipelineWebhook:
    Type: AWS::CodePipeline::Webhook
    Properties:
      Authentication: GITHUB_HMAC
      AuthenticationConfiguration:
        SecretToken: {{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
      Filters:
        -
          JsonPath: "$.ref"
          MatchEquals: refs/heads/{Branch}
      TargetPipeline: !Ref AppPipeline
      TargetAction: SourceAction
      Name: AppPipelineWebhook
      TargetPipelineVersion: !GetAtt AppPipeline.Version
      RegisterWithThirdParty: true
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      Name: github-events-pipeline
      RoleArn:
        !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
```

```
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: ThirdParty
                Version: 1
                Provider: GitHub
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                Owner: !Ref GitHubOwner
                Repo: !Ref RepositoryName
                Branch: !Ref BranchName
                OAuthToken:
 {{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
                PollForSourceChanges: false
              RunOrder: 1

...
```

JSON

```
{
    "Parameters": {
        "BranchName": {
            "Description": "GitHub branch name",
            "Type": "String",
            "Default": "main"
        },
        "RepositoryName": {
            "Description": "GitHub repository name",
            "Type": "String",
            "Default": "test"
        },
        "GitHubOwner": {
            "Type": "String"
        },
        "ApplicationName": {
            "Description": "CodeDeploy application name",
            "Type": "String",
            "Default": "DemoApplication"
        },
        "BetaFleet": {
```

```
            "Description": "Fleet configured in CodeDeploy",
            "Type": "String",
            "Default": "DemoFleet"
        }
    },
    "Resources": {

...

        },
        "AppPipelineWebhook": {
            "Type": "AWS::CodePipeline::Webhook",
            "Properties": {
                "Authentication": "GITHUB_HMAC",
                "AuthenticationConfiguration": {
                    "SecretToken": {

 "{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}"
                    }
                },
                "Filters": [
                    {
                        "JsonPath": "$.ref",
                        "MatchEquals": "refs/heads/{Branch}"
                    }
                ],
                "TargetPipeline": {
                    "Ref": "AppPipeline"
                },
                "TargetAction": "SourceAction",
                "Name": "AppPipelineWebhook",
                "TargetPipelineVersion": {
                    "Fn::GetAtt": [
                        "AppPipeline",
                        "Version"
                    ]
                },
                "RegisterWithThirdParty": true
            }
        },
        "AppPipeline": {
            "Type": "AWS::CodePipeline::Pipeline",
            "Properties": {
                "Name": "github-events-pipeline",
```

```
                "RoleArn": {
                    "Fn::GetAtt": [
                        "CodePipelineServiceRole",
                        "Arn"
                    ]
                },
                "Stages": [
                    {
                        "Name": "Source",
                        "Actions": [
                            {
                                "Name": "SourceAction",
                                "ActionTypeId": {
                                    "Category": "Source",
                                    "Owner": "ThirdParty",
                                    "Version": 1,
                                    "Provider": "GitHub"
                                },
                                "OutputArtifacts": [
                                    {
                                        "Name": "SourceOutput"
                                    }
                                ],
                                "Configuration": {
                                    "Owner": {
                                        "Ref": "GitHubOwner"
                                    },
                                    "Repo": {
                                        "Ref": "RepositoryName"
                                    },
                                    "Branch": {
                                        "Ref": "BranchName"
                                    },
                                    "OAuthToken":
 "{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}",
                                    "PollForSourceChanges": false
                                },
                                "RunOrder": 1

...
```

Erstellen Sie die CodePipeline Servicerolle

Wenn Sie eine Pipeline erstellen, erstellen Sie eine Servicerolle oder verwenden eine vorhandene Servicerolle.

Sie können die CodePipeline Konsole oder die verwenden AWS CLI , um eine CodePipeline Servicerolle zu erstellen. Eine Servicerolle ist erforderlich, um eine Pipeline zu erstellen, und die Pipeline ist immer dieser Servicerolle zugeordnet.

Bevor Sie eine Pipeline mit der AWS CLI erstellen, müssen Sie eine CodePipeline Servicerolle für Ihre Pipeline erstellen. Eine AWS CloudFormation Beispielvorlage mit der angegebenen Servicerolle und Richtlinie finden Sie in den Tutorials unterTutorial: Eine Pipeline erstellen, die Variablen aus AWS CloudFormation Bereitstellungsaktionen verwendet.

Bei der Servicerolle handelt es sich nicht um eine AWS verwaltete Rolle, sondern sie wird zunächst für die Pipelineerstellung erstellt. Wenn dann der Richtlinie für die Servicerolle neue Berechtigungen hinzugefügt werden, müssen Sie möglicherweise die Servicerolle für Ihre Pipeline aktualisieren. Sobald Ihre Pipeline mit einer Servicerolle erstellt wurde, können Sie für diese Pipeline keine andere Servicerolle mehr verwenden. Fügen Sie der Servicerolle die empfohlene Richtlinie hinzu.

Weitere Informationen über die Service-Rolle finden Sie unter Die CodePipeline Servicerolle verwalten.

Erstellen Sie die CodePipeline Servicerolle (Konsole)

Wenn Sie die Konsole verwenden, um eine Pipeline zu erstellen, erstellen Sie die CodePipeline Servicerolle mit dem Assistenten zum Erstellen von Pipelines.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Wählen Sie Create pipeline (Pipeline erstellen), und führen Sie die Aktionen auf der Seite Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipelineeinstellungen) im Assistenten zur Pipelineerstellung durch.

   > **Note**
   >
   > Der Name einer erstellten Pipeline kann nicht nachträglich geändert werden. Weitere Informationen zu Einschränkungen finden Sie unter Kontingente in AWS CodePipeline.

2. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, CodePipeline damit Sie eine neue Servicerolle in IAM erstellen können.
3. Führen Sie die Pipeline-Erstellung durch. Ihre Pipeline-Servicerolle kann in Ihrer Liste der IAM-Rollen angezeigt werden, und Sie können den einer Pipeline zugewiesenen Servicerollen-ARN anzeigen, indem Sie den `get-pipeline` Befehl mit der AWS CLI ausführen.

Erstellen Sie die CodePipeline Servicerolle (CLI)

Bevor Sie eine Pipeline mit der AWS CLI oder erstellen AWS CloudFormation, müssen Sie eine CodePipeline Servicerolle für Ihre Pipeline erstellen und die Servicerollenrichtlinie und die Vertrauensrichtlinie anhängen. Um die CLI zum Erstellen Ihrer Servicerolle zu verwenden, gehen Sie wie folgt vor, um zunächst eine Vertrauensrichtlinien-JSON und die Rollenrichtlinien-JSON als separate Dateien in dem Verzeichnis zu erstellen, in dem Sie die CLI-Befehle ausführen werden.

> **Note**
>
> Wir empfehlen, dass Sie nur Administratorbenutzern erlauben, eine Servicerolle zu erstellen. Eine Person mit Berechtigungen zum Erstellen einer Rolle und zum Anfügen einer Richtlinie kann ihre eigenen Berechtigungen eskalieren. Erstellen Sie stattdessen eine Richtlinie, die es ihnen ermöglicht, nur die Rollen zu erstellen, die sie benötigen. Oder lassen Sie einen Administrator die Servicerolle in ihrem Namen erstellen.

1. Geben Sie in einem Terminalfenster den folgenden Befehl ein, um eine Datei mit dem Namen zu erstellen`TrustPolicy.json`, in die Sie die Rollenrichtlinie JSON einfügen werden. In diesem Beispiel wird VIM verwendet.

   ```
   vim TrustPolicy.json
   ```

2. Fügen Sie den folgenden JSON-Code in die Datei ein.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "codepipeline.amazonaws.com"
   ```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Geben Sie den folgenden VIM-Befehl ein, um die Datei zu speichern und zu beenden:

```
:wq
```

3. Geben Sie in einem Terminalfenster den folgenden Befehl ein, um eine Datei mit dem Namen zu erstellen`RolePolicy.json`, in die Sie die Rollenrichtlinie JSON einfügen werden. In diesem Beispiel wird VIM verwendet.

```
vim RolePolicy.json
```

4. Fügen Sie den folgenden JSON-Code in die Datei ein. Achten Sie darauf, die Berechtigungen so weit wie möglich einzuschränken, indem Sie den Amazon-Ressourcennamen (ARN) für Ihre Pipeline in das `Resource` Feld Policy Statement eingeben.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:CancelUploadArchive",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetUploadArchiveStatus",
        "codecommit:UploadArchive"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetApplicationRevision",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
```

```
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "devicefarm:ListProjects",
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:ListFunctions"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
```

```
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "cloudformation:*",
        "rds:*",
        "sqs:*",
        "ecs:*"
      ],
      "Resource": "resource_ARN"
    }
  ]
}
```

Geben Sie den folgenden VIM-Befehl ein, um die Datei zu speichern und zu beenden:

```
:wq
```

5. Geben Sie den folgenden Befehl ein, um die Rolle zu erstellen und die Vertrauensrollenrichtlinie anzuhängen. Das Format für den Namen der Richtlinie entspricht normalerweise dem für den Namen der Rolle. In diesem Beispiel werden der Rollenname `MyRole` und die Richtlinie verwendet`TrustPolicy`, die als separate Datei erstellt wurden.

```
aws iam create-role --role-name MyRole --assume-role-policy-document file://
TrustPolicy.json
```

6. Geben Sie den folgenden Befehl ein, um die Rollenrichtlinie zu erstellen und sie der Rolle anzuhängen. Das Format für den Namen der Richtlinie entspricht normalerweise dem für den Namen der Rolle. In diesem Beispiel werden der Rollenname `MyRole` und die Richtlinie verwendet`MyRole`, die als separate Datei erstellt wurden.

```
aws iam put-role-policy --role-name MyRole --policy-name RolePolicy --policy-
document file://RolePolicy.json
```

7. Um den erstellten Rollennamen und die Vertrauensrichtlinie anzuzeigen, geben Sie den folgenden Befehl für die angegebene Rolle ein`MyRole`:

```
aws iam get-role --role-name MyRole
```

8. Verwenden Sie die Servicerolle ARN, wenn Sie Ihre Pipeline mit der AWS CLI oder erstellen AWS CloudFormation.

Markieren Sie eine Pipeline in CodePipeline

Tags sind Schlüssel-Wert-Paare, die Ressourcen zugeordnet AWS sind. Sie können Tags auf Ihre Pipelines in anwenden. CodePipeline Informationen zur Kennzeichnung von CodePipeline Ressourcen, zu Anwendungsfällen, Einschränkungen für Tagschlüssel und -werte sowie zu unterstützten Ressourcentypen finden Sie unter. Markieren von Ressourcen

Sie können Tags beim Erstellen einer Pipeline über die CLI angeben. Zum Hinzufügen oder Entfernen von Tags sowie zum Aktualisieren der Werte der Tags in einer Pipeline können Sie die Konsole oder CLI verwenden. Sie können bis zu 50 Tags für jede Pipeline hinzufügen.

Themen

Markieren von Pipelines (Konsole)

Sie können Ressourcen über die Konsole oder CLI mit Tags markieren. Pipelines sind die einzige CodePipeline Ressource, die entweder mit der Konsole oder der CLI verwaltet werden kann.

Themen

Hinzufügen von Tags zu einer Pipeline (Konsole)

Sie können die Konsole verwenden, um Tags zu einer vorhandenen Pipeline hinzuzufügen.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Pipelines die Pipeline aus, für die Sie Tags hinzufügen möchten.
3. Wählen Sie im Navigationsbereich Settings (Einstellungen) aus.
4. Wählen Sie unter Pipeline Tags (Pipeline-Tags) die Option Edit (Berabeiten) aus.

5. Geben Sie in die Felder Key (Schlüssel) und Value (Wert) ein Schlüsselpaar für jeden hinzuzufügenden Tag-Satz ein. (Das Feld Value (Wert) ist optional.) Geben Sie beispielsweise für Key (Schlüssel) **`Project`** ein. Geben Sie unter Value (Wert) **`ProjectA`** ein.
6. (Optional) Wählen Sie Add tag (Tag hinzufügen) aus, um weitere Zeilen hinzuzufügen und weitere Tags einzugeben.
7. Wählen Sie Submit (Absenden) aus. Die Tags werden unter den Pipeline Einstellungen aufgeführt.

Anzeigen von Tags für eine Pipeline (Konsole)

Sie können über die Konsole Tags für vorhandene Pipelines auflisten.

1. Melden Sie sich bei http://console.aws.amazon.com/codesuite/codepipeline/home an AWS Management Console und öffnen Sie die CodePipeline Konsole.
2. Wählen Sie auf der Seite Pipelines die Pipeline aus, für die Sie Tags anzeigen möchten.
3. Wählen Sie im Navigationsbereich Settings (Einstellungen) aus.
4. Zeigen Sie unter Tags die Tags für die Pipeline in den Spalten Key (Schlüssel) und Value (Wert) an.

Bearbeiten von Tags für eine Pipeline (Konsole)

Sie können über die Konsole Tags bearbeiten, die Pipelines hinzugefügt wurden.

1. Melden Sie sich bei http://console.aws.amazon.com/codesuite/codepipeline/home an AWS Management Console und öffnen Sie die CodePipeline Konsole.
2. Wählen Sie auf der Seite Pipelines die Pipeline aus, für die Sie Tags aktualisieren möchten.
3. Wählen Sie im Navigationsbereich Settings (Einstellungen) aus.
4. Wählen Sie unter Pipeline Tags (Pipeline-Tags) die Option Edit (Berabeiten) aus.
5. Aktualisieren Sie in den Feldern Key (Schlüssel) und Value (Wert) die Werte nach Bedarf. Ändern Sie beispielsweise für den Schlüssel **`Project`** unter Value (Wert) die Angabe **`ProjectA`** in **`ProjectB`**.
6. Wählen Sie Absenden aus.

Entfernen von Tags aus einer Pipeline (Konsole)

Sie können über die Konsole Tags aus Pipelines löschen. Wenn Sie Tags aus der zugehörigen Ressource entfernen, werden die Tags gelöscht.

1. Melden Sie sich bei http://console.aws.amazon.com/codesuite/codepipeline/home an AWS Management Console und öffnen Sie die CodePipeline Konsole.
2. Wählen Sie auf der Seite Pipelines die Pipeline aus, für die Sie Tags entfernen möchten.
3. Wählen Sie im Navigationsbereich Settings (Einstellungen) aus.
4. Wählen Sie unter Pipeline Tags (Pipeline-Tags) die Option Edit (Berabeiten) aus.
5. Wählen Sie neben dem Schlüssel und Wert für jedes zu löschende Tag Remove tag (Tag entfernen) aus.
6. Wählen Sie Absenden aus.

Markieren von Pipelines (CLI)

Sie können über die CLI Ressourcen mit Tags markieren. Für die Verwaltung von Tags in Pipelines müssen Sie die Konsole verwenden.

Themen

- Hinzufügen von Tags zu einer Pipeline (CLI)
- Anzeigen von Tags für eine Pipeline (CLI)
- Bearbeiten von Tags für eine Pipeline (CLI)
- Entfernen von Tags aus einer Pipeline (CLI)

Hinzufügen von Tags zu einer Pipeline (CLI)

Sie können die Konsole oder die verwenden, um Pipelines AWS CLI zu taggen.

Informationen darüber, wie Sie beim Erstellen einer Pipeline ein Tag hinzufügen können, finden Sie unter Erstellen Sie eine Pipeline in CodePipeline.

Bei diesen Schritten wird davon ausgegangen, dass Sie bereits eine aktuelle Version der AWS CLI installiert oder eine Aktualisierung auf die aktuelle Version vorgenommen haben. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface.

Führen Sie am Terminal oder in der Befehlszeile den Befehl tag-resource aus und geben Sie dabei den Amazon-Ressourcennamen (ARN) der Pipeline an, für die Sie Tags hinzufügen möchten, sowie den Schlüssel und Wert des hinzuzufügenden Tags. Sie können einer Pipeline mehrere Tags hinzufügen. Um beispielsweise eine Pipeline *MyPipeline*mit zwei Tags zu taggen, einem Tag-Schlüssel *DeploymentEnvironment*mit dem Tag-Wert *Test* und einem Tag-Schlüssel *IscontainerBased*mit dem Tag-Wert *true*:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tags key=Project,value=ProjectA key=IscontainerBased,value=true
```

Bei erfolgreicher Ausführung gibt dieser Befehl nichts zurück.

Anzeigen von Tags für eine Pipeline (CLI)

Gehen Sie wie folgt vor AWS CLI , um die AWS Tags für eine Pipeline anzuzeigen. Wenn keine Tags hinzugefügt wurden, ist die zurückgegebene Liste leer.

Führen Sie am Terminal oder über die Befehlszeile den Befehl list-tags-for-resource aus. Um beispielsweise eine Liste von Tag-Schlüsseln und Tag-Werten für eine Pipeline anzuzeigen, die *MyPipeline*mit dem `arn:aws:codepipeline:`*us-west-2*`:`*account-id*`:`*MyPipeline* ARN-Wert benannt ist:

```
aws codepipeline list-tags-for-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline
```

Bei erfolgreicher Ausführung gibt dieser Befehl etwa wie folgt aussehende Informationen zurück:

```
{
    "tags": {
        "Project": "ProjectA",
        "IscontainerBased": "true"
    }
}
```

Bearbeiten von Tags für eine Pipeline (CLI)

Gehen Sie wie folgt vor, um AWS CLI mit dem ein Tag für eine Pipeline zu bearbeiten. Sie können den Wert für einen vorhandenen Schlüssel ändern oder einen anderen Schlüssel hinzufügen. Sie können auch Tags aus einer Pipeline entfernen, wie im nächsten Abschnitt erläutert.

Führen Sie am Terminal oder über die Befehlszeile den Befehl tag-resource aus und geben Sie dabei den ARN der Pipeline an, für die Sie ein Tag aktualisieren möchten, sowie den Tag-Schlüssel und Tag-Wert:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tags key=Project,value=ProjectA
```

Bei erfolgreicher Ausführung gibt dieser Befehl nichts zurück.

Entfernen von Tags aus einer Pipeline (CLI)

Gehen Sie wie folgt vor, um AWS CLI mit dem ein Tag aus einer Pipeline zu entfernen. Wenn Sie Tags aus der zugehörigen Ressource entfernen, werden die Tags gelöscht.

> **Note**
>
> Wenn Sie eine Pipeline löschen, werden alle Tag-Zuordnungen aus der gelöschten Pipeline entfernt. Sie müssen keine Tags entfernen, bevor Sie eine Pipeline löschen.

Führen Sie am Terminal oder über die Befehlszeile den Befehl untag-resource aus und geben Sie dabei den ARN der Pipeline an, für die Sie Tags entfernen möchten, sowie den Tag-Schlüssel des zu entfernenden Tags. Um beispielsweise mehrere Tags in einer Pipeline zu entfernen, die *MyPipeline*mit den Tag-Schlüsseln *Project* und benannt ist *IscontainerBased*:

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tag-keys Project IscontainerBased
```

Bei erfolgreicher Ausführung gibt dieser Befehl nichts zurück. Um die der Pipeline zugeordneten Tags zu überprüfen, führen Sie den Befehl list-tags-for-resource aus.

Erstellen einer Benachrichtigungsregel

Sie können Benachrichtigungsregeln verwenden, um Benutzer über wichtige Änderungen zu informieren, z. B. wenn eine Pipeline eine Ausführung startet. Die Benachrichtigungsregeln spezifizieren sowohl die Ereignisse als auch das Amazon SNS SNS-Thema, das zum Senden von Benachrichtigungen verwendet wird. Weitere Informationen finden Sie unter Was sind Benachrichtigungen?.

Sie können die Konsole oder die verwenden, um Benachrichtigungsregeln AWS CLI für zu erstellen. AWS CodePipeline

So erstellen Sie eine Benachrichtigungsregel (Konsole):

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie Pipelines (Pipelines) und anschließend die Pipeline aus, der Sie Benachrichtigungen hinzufügen möchten.
3. Wählen Sie auf der Seite für Pipelines Notify (Benachrichtigen) und anschließend Create notification rule (Benachrichtigungsregel erstellen) aus. Sie können auch zur Seite Settings (Einstellungen) für die Pipeline navigieren und Create notification rule (Benachrichtigungsregel erstellen)auswählen.
4. Geben Sie unter Notification name (Benachrichtigungsname) einen Namen für die Regel ein.
5. Wählen Sie unter Detailtyp die Option Basic aus, wenn Sie möchten, dass nur die Informationen, die Amazon zur Verfügung gestellt wurden, in der Benachrichtigung EventBridge enthalten sind. Wählen Sie Vollständig, wenn Sie Informationen, die Amazon zur Verfügung gestellt wurden, EventBridge und Informationen, die möglicherweise vom CodePipeline oder vom Notification Manager bereitgestellt wurden, einbeziehen möchten.

   Weitere Informationen finden Sie unter Informationen zu Inhalten und Sicherheit von Benachrichtigungen.
6. Wählen Sie unter Events that trigger notifications (Ereignisse, die Benachrichtigungen auslösen) die Ereignisse aus, für die Sie Benachrichtigungen senden möchten. Weitere Informationen finden Sie unter Ereignisse für Benachrichtigungsregeln in Pipelines.
7. Führen Sie unter Targets (Ziele) einen der folgenden Schritte aus:

   - Wenn Sie bereits eine Ressource zur Verwendung mit Benachrichtigungen konfiguriert haben, wählen Sie unter Choose target type (Zieltyp wählen) entweder AWS Chatbot (Slack) oder SNS topic (SNS-Thema)aus. Wählen Sie unter Ziel auswählen den Namen des Clients (für einen in konfigurierten Slack-Client AWS Chatbot) oder den Amazon-Ressourcennamen (ARN) des Amazon SNS-Themas (für Amazon SNS SNS-Themen, die bereits mit der für Benachrichtigungen erforderlichen Richtlinie konfiguriert wurden).
   - Wenn Sie keine Ressource für die Verwendung mit Benachrichtigungen konfiguriert haben, wählen Sie Create target (Ziel erstellen) und dann SNS topic (SNS-Thema) aus. Geben Sie

nach codestar-notifications- einen Namen für das Thema an und wählen Sie dann Create (Erstellen).

> **Note**
>
> - Wenn Sie das Amazon-SNS-Thema im Rahmen des Erstellens der Benachrichtigungsregel erstellen, wird die Richtlinie, die es ermöglicht, Ereignisse in dem Thema zu veröffentlichen, für Sie angewendet. Durch die Verwendung eines Themas, das für Benachrichtigungsregeln erstellt wurde, kann sichergestellt werden, dass Sie das Thema nur für die Benutzer abonnieren, die Benachrichtigungen zu dieser Ressource erhalten sollen.
> - Sie können im Rahmen der Erstellung einer AWS Chatbot Benachrichtigungsregel keinen Client erstellen. Wenn du AWS Chatbot (Slack) auswählst, siehst du eine Schaltfläche, in der du einen Client konfigurieren kannst. AWS Chatbot Wenn Sie diese Option wählen, wird die AWS Chatbot Konsole geöffnet. Weitere Informationen finden Sie unter Integrationen zwischen Benachrichtigungen konfigurieren und AWS Chatbot.
> - Wenn Sie ein vorhandenes Amazon SNS SNS-Thema als Ziel verwenden möchten, müssen Sie die erforderliche Richtlinie für AWS CodeStar Benachrichtigungen zusätzlich zu allen anderen Richtlinien hinzufügen, die möglicherweise für dieses Thema existieren. Weitere Informationen finden Sie unter Konfigurieren vorhandener Amazon SNS-Themen für Benachrichtigungen und Informationen zu Inhalten und Sicherheit von Benachrichtigungen.

8. Um die Erstellung der Regel abzuschließen, wählen Sie Submit (Absenden) aus.
9. Sie müssen das Amazon SNS SNS-Thema für die Regel abonnieren, bevor sie Benachrichtigungen erhalten können. Weitere Informationen finden Sie unter Amazon SNS SNS-Themen abonnieren, die Ziele sind. Sie können auch die Integration zwischen Benachrichtigungen einrichten und AWS Chatbot Benachrichtigungen an Amazon Chime-Chatrooms oder Slack-Kanäle senden. Weitere Informationen finden Sie unter Konfiguration der Integration zwischen Benachrichtigungen und. AWS Chatbot

So erstellen Sie eine Benachrichtigungsregel (AWS CLI):

1. Führen Sie in einem Terminal oder einer Eingabeaufforderung den Befehl create-notification rule aus, um das JSON-Skelett zu generieren:

   ```
   aws codestar-notifications create-notification-rule --generate-cli-skeleton
    > rule.json
   ```

   Sie können die Datei beliebig benennen. In diesem Beispiel trägt die Datei den Namen *rule.json*.

2. Öffnen Sie die JSON-Datei in einem Texteditor, und bearbeiten Sie sie so, dass sie die Ressource, die Ereignistypen und das gewünschte Ziel für die Regel enthält. Das folgende Beispiel zeigt eine Benachrichtigungsregel, die **MyNotificationRule** nach einer Pipeline benannt ist, die *MyDemoPipeline*in einem AWS Konto mit der ID *123456789012* benannt ist. Benachrichtigungen werden mit dem vollständigen Detailtyp an ein Amazon SNS SNS-Thema namens *codestar-notifications* gesendet, MyNotificationTopic wenn Pipeline-Ausführungen beginnen:

   ```
   {
       "Name": "MyNotificationRule",
       "EventTypeIds": [
           "codepipeline-pipeline-pipeline-execution-started"
       ],
       "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyDemoPipeline",
       "Targets": [
           {
               "TargetType": "SNS",
               "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
   notifications-MyNotificationTopic"
           }
       ],
       "Status": "ENABLED",
       "DetailType": "FULL"
   }
   ```

   Speichern Sie die Datei.

3. Führen Sie unter Verwendung der soeben bearbeiteten Datei am Terminal oder in der Befehlszeile erneut den Befehl create-notification-rule aus, um die Benachrichtigungsregel zu erstellen:

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. Bei Erfolg gibt der Befehl den ARN der Benachrichtigungsregel zurück, der ähnlich wie im Folgenden dargestellt aussieht:

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

Arbeiten mit Triggern in CodePipeline

Mit Triggern können Sie Ihre Pipeline so konfigurieren, dass sie bei einem bestimmten Ereignistyp oder einem gefilterten Ereignistyp startet, z. B. wenn eine Änderung an einem bestimmten Branch oder einer Pull-Anforderung erkannt wird. Trigger können für Quellaktionen mit Verbindungen konfiguriert werden, die die `CodeStarSourceConnection` Aktion in verwenden CodePipeline GitHub, wie Bitbucket und GitLab.

Quellaktionen wie CodeCommit und S3 verwenden die Änderungserkennung, wie in diesem Abschnitt über das Starten von Pipelines beschrieben.

Sie können Ihrer Pipeline einen Trigger hinzufügen und den Trigger so konfigurieren, dass er nach bestimmten Ereignissen filtert

Sie geben Trigger über die Konsole oder CLI an.

Trigger für Code-Push- oder Pull-Anfragen filtern

Sie können Filter für Pipeline-Trigger so konfigurieren, dass Pipeline-Ausführungen für verschiedene Git-Ereignisse gestartet werden, z. B. Tag- oder Branch-Push, Änderungen an bestimmten Dateipfaden, eine Pull-Anfrage, die in einem bestimmten Branch geöffnet wird, usw. Sie können die AWS CodePipeline Konsole oder die update-pipeline Befehle create-pipeline und in der verwenden, AWS CLI um die Filter der Trigger zu konfigurieren.

Sie können Filter für die folgenden Triggertypen angeben:

- Push

  Ein Push-Trigger startet eine Pipeline, wenn eine Änderung in Ihr Quell-Repository übertragen wird. Bei der Ausführung wird der Commit aus dem Branch verwendet, in den Sie pushen (das ist der Ziel-Branch). Sie können Push-Trigger nach Branches, Dateipfaden oder Git-Tags filtern.

- Pull-Anfrage

  Ein Pull-Request-Trigger startet eine Pipeline, wenn ein Pull-Request in Ihrem Quell-Repository geöffnet, aktualisiert oder geschlossen wird. Bei der Ausführung wird der Commit aus dem Quell-Branch verwendet, aus dem Sie einen Pull abrufen (das ist der Quell-Branch). Sie können Pull-Request-Trigger nach Branches und Dateipfaden filtern.

> **Note**
>
> Die unterstützten Ereignistypen für Pull-Requests sind geöffnet, aktualisiert oder geschlossen (zusammengeführt). Alle anderen Pull-Request-Ereignisse werden ignoriert.

Die Pipeline-Definition ermöglicht es Ihnen, verschiedene Filter innerhalb derselben Push-Trigger-Konfiguration zu kombinieren. Einzelheiten zur Pipeline-Definition finden Sie unterTriggerfilterung in der Pipeline-JSON (CLI). Gültige Kombinationen sind:

- tags
- Geäst
- Zweige + Dateipfade

Sie geben Filtertypen wie folgt an:

- Kein Filter

  Diese Trigger-Konfiguration startet Ihre Pipeline bei jedem Push zu dem Standardzweig, der als Teil der Aktionskonfiguration angegeben wurde.
- Filter angeben

  Sie fügen einen Filter hinzu, der Ihre Pipeline mit einem bestimmten Filter startet, z. B. bei Branch-Namen für einen Code-Push, und den exakten Commit abruft. Dadurch wird die Pipeline auch so konfiguriert, dass sie bei jeder Änderung nicht automatisch gestartet wird.
- Erkennt keine Änderungen

  Dadurch wird kein Trigger hinzugefügt und die Pipeline wird bei jeder Änderung nicht automatisch gestartet.

Die folgende Tabelle enthält gültige Filteroptionen für jeden Ereignistyp. Die Tabelle zeigt auch, welche Triggerkonfigurationen für die automatische Erkennung von Änderungen in der Aktionskonfiguration standardmäßig auf „true“ oder „false“ gesetzt sind.

Konfiguration des Triggers	Ereignistyp	Filteroptionen	Ermitteln Sie Änderungen
Fügen Sie einen Auslöser hinzu — kein Filter	Keine	Keine	true
Einen Auslöser hinzufügen — Filter bei Code-Push	Ereignis pushen	Git-Tags, Zweige, Dateipfade	false
Füge einen Trigger hinzu — Filter für Pull-Requests	Pull-Anfragen	Zweige, Dateipfade	false
Kein Auslöser — nicht erkennen	Keine	Keine	false

> **Note**
>
> Dieser Triggertyp verwendet die automatische Änderungserkennung (als `Webhook` Triggertyp). Die Anbieter von Quellaktionen, die diesen Triggertyp verwenden, sind Verbindungen, die für Code-Push konfiguriert sind (Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltet).

Für die Filterung werden Muster regulärer Ausdrücke im Glob-Format unterstützt, wie unter beschrieben. Arbeiten mit Glob-Mustern in der Syntax

> **Note**
>
> In bestimmten Fällen startet die Pipeline bei Pipelines mit Triggern, die nach Dateipfaden gefiltert werden, möglicherweise nicht, wenn ein Zweig mit einem Dateipfadfilter zum ersten Mal erstellt wird. Weitere Informationen finden Sie unter Pipelines mit Verbindungen, die Triggerfilterung nach Dateipfaden verwenden, beginnen möglicherweise nicht bei der Erstellung von Zweigen.

Themen

Überlegungen zu Triggerfiltern

Die folgenden Überlegungen gelten für die Verwendung von Triggern.

- Bei einem Trigger mit Verzweigungs- und Dateipfadfiltern wird die Pipeline nicht ausgeführt, wenn der Branch zum ersten Mal übertragen wird, da kein Zugriff auf die Liste der Dateien besteht, die für den neu erstellten Branch geändert wurden.
- Das Zusammenführen eines Pull-Requests kann zwei Pipeline-Ausführungen auslösen, wenn sich die Konfigurationen der Trigger Push (Branches Filter) und Pull Request (Branches Filter) überschneiden.

Beispiele für Triggerfilter

Bei einer Git-Konfiguration mit Filtern für Push- und Pull-Request-Ereignistypen können die angegebenen Filter miteinander in Konflikt geraten. Im Folgenden finden Sie Beispiele für gültige Filterkombinationen für Push- und Pull-Request-Ereignisse.

Wenn Kunden Filter in einem einzigen Konfigurationsobjekt kombinieren, verwenden diese Filter eine AND-Operation, was bedeutet, dass nur eine vollständige Übereinstimmung die Pipeline startet. Das folgende Beispiel zeigt die Git-Konfiguration:

```
{
  "filePaths": {
    "includes": ["common/**/*.js"]
  },
  "branches": {
    "includes": ["feature/**"]
  }
```

```
}
```

Mit der obigen Git-Konfiguration zeigt dieses Beispiel ein Ereignis, das die Pipeline-Ausführung startet, weil die AND-Operation erfolgreich ist.

```
{
  "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "common/app.js"
      ]
      ...
    }
  ]
}
```

Dieses Beispiel zeigt ein Ereignis, das die Pipeline-Ausführung nicht startet, weil der Branch filtern kann, der Dateipfad jedoch nicht.

```
{
   "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "src/Main.java"
      ]
      ...
    }
  ]
}
```

Gleichzeitig verwenden Trigger-Konfigurationsobjekte innerhalb des Push-Arrays eine OR-Operation. Auf diese Weise können Sie mehrere Trigger konfigurieren, um die Ausführung für dieselbe Pipeline zu starten. Eine Liste der Felddefinitionen in der JSON-Struktur finden Sie unterTriggerfilterung in der Pipeline-JSON (CLI).

Nach Push-Ereignissen filtern (Konsole)

Sie können die Konsole verwenden, um Filter für Push-Ereignisse hinzuzufügen und Branches oder Dateipfade ein- oder auszuschließen.

Nach Push-Ereignissen filtern (Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen und der Status aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Verwenden Sie andernfalls diese Schritte im Assistenten zum Erstellen von Pipelines.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie auf der Seite Bearbeiten die Quellaktion aus, die Sie bearbeiten möchten. Wählen Sie Trigger bearbeiten aus. Wählen Sie „Filter angeben“.
5. Wählen Sie unter Ereignistyp die Option Push aus den folgenden Optionen aus.

   - Wählen Sie Push, um die Pipeline zu starten, wenn eine Änderung in Ihr Quell-Repository übertragen wird. Wenn Sie diese Option wählen, können die Felder Filter für Branches und Dateipfade oder Git-Tags angeben.
   - Wählen Sie Pull-Anfrage, um die Pipeline zu starten, wenn eine Pull-Anfrage in Ihrem Quell-Repository geöffnet, aktualisiert oder geschlossen wird. Wenn Sie diese Option wählen, können die Felder Filter für Ziel-Branches und Dateipfade angeben.
6. Wählen Sie unter Filtertyp eine der folgenden Optionen aus.

   - Wählen Sie Branch, um die Branches in Ihrem Quell-Repository anzugeben, die der Trigger überwacht, damit er weiß, wann eine Workflow-Ausführung gestartet werden muss. Geben Sie unter Include Muster für Branch-Namen im Glob-Format ein, die Sie für die Trigger-Konfiguration angeben möchten, damit Ihre Pipeline bei Änderungen in den angegebenen Branches gestartet wird. Geben Sie im Feld Exclude die Regex-Muster für Zweignamen im Glob-Format ein, die Sie angeben möchten, damit die Trigger-Konfiguration sie ignoriert und Ihre Pipeline nicht bei Änderungen in den angegebenen Zweigen startet. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

> **Note**
>
> Wenn Include und Exclude beide dasselbe Muster haben, wird das Muster standardmäßig ausgeschlossen.

Sie können Regex-Muster im Glob-Format verwenden, um Ihre Branchennamen zu definieren. Verwenden Sie dies beispielsweise, um alle Zweige `main.*` abzugleichen, die mit beginnen. `main.*` Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

Geben Sie für einen Push-Trigger die Zweige an, auf die Sie pushen möchten, d. h. die Zielzweige. Geben Sie für einen Pull-Request-Trigger die Ziel-Branches an, für die Sie den Pull-Request öffnen möchten.

- (Optional) Geben Sie unter Dateipfade die Dateipfade für Ihren Trigger an. Geben Sie die entsprechenden Namen in die Felder Einschließen und Ausschließen ein.

  Sie können Regex-Muster im Glob-Format verwenden, um Ihre Dateipfadnamen zu definieren. Verwenden Sie dies beispielsweise, um alle Dateipfade `prod.*` abzugleichen, die mit beginnen. `prod.*` Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.
- Wählen Sie Tags, um die Pipeline-Trigger-Konfiguration so zu konfigurieren, dass sie mit Git-Tags beginnt. Geben Sie unter Include Muster für Tag-Namen im Glob-Format ein, die Sie für die Trigger-Konfiguration angeben möchten, damit Ihre Pipeline bei der Veröffentlichung des oder der angegebenen Tags gestartet wird. Geben Sie im Feld Exclude die Regex-Muster für Tagnamen im Glob-Format ein, die Sie angeben möchten, damit die Trigger-Konfiguration sie ignoriert und Ihre Pipeline nicht startet, wenn das angegebene Tag oder die angegebenen Tags veröffentlicht werden. Wenn sowohl Include als auch Exclude dasselbe Tag-Muster haben, wird standardmäßig das Tag-Muster ausgeschlossen.

Filterung nach Pull-Requests (Konsole)

Sie können die Konsole verwenden, um Filter für Pull-Requests mit bestimmten Ereignissen hinzuzufügen und Branches oder Dateipfade ein- oder auszuschließen.

Filterung nach Pull-Requests (Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen und der Status aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Verwenden Sie andernfalls diese Schritte im Assistenten zum Erstellen von Pipelines.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie auf der Seite Bearbeiten die Quellaktion aus, die Sie bearbeiten möchten. Wählen Sie Trigger bearbeiten aus. Wählen Sie „Filter angeben“.
5. Wählen Sie unter Ereignistyp die Option Pull-Anfrage aus den folgenden Optionen aus.

   - Wählen Sie Push, um die Pipeline zu starten, wenn eine Änderung in Ihr Quell-Repository übertragen wird. Wenn Sie diese Option wählen, können die Felder Filter für Branches und Dateipfade oder Git-Tags angeben.
   - Wählen Sie Pull-Anfrage, um die Pipeline zu starten, wenn eine Pull-Anfrage für die angegebenen Ziel-Branches geöffnet, aktualisiert oder geschlossen wird. Wenn Sie diese Option wählen, können die Felder Filter für Branches und Dateipfade angeben.

     Sie können optional die folgenden Pull-Request-Ereignisse zum Filtern angeben:

     - Eine Pull-Anfrage wird erstellt
     - Der Pull-Request wurde neu überarbeitet
     - Die Pull-Anfrage ist geschlossen
6. Wählen Sie unter Filtertyp eine der folgenden Optionen aus.

   - Wählen Sie Branch, um die Branches in Ihrem Quell-Repository anzugeben, die der Trigger überwacht, damit er weiß, wann eine Workflow-Ausführung gestartet werden muss. Geben Sie unter Include Muster für Branch-Namen im Glob-Format ein, die Sie für die Trigger-Konfiguration angeben möchten, damit Ihre Pipeline bei Änderungen in den angegebenen Branches gestartet wird. Geben Sie im Feld Exclude die Regex-Muster für Zweignamen im Glob-Format ein, die Sie angeben möchten, damit die Trigger-Konfiguration sie ignoriert und Ihre Pipeline nicht bei Änderungen in den angegebenen Zweigen startet. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

> **Note**
>
> Wenn Include und Exclude beide dasselbe Muster haben, wird das Muster standardmäßig ausgeschlossen.

Sie können Regex-Muster im Glob-Format verwenden, um Ihre Branchennamen zu definieren. Verwenden Sie dies beispielsweise, um alle Zweige `main.*` abzugleichen, die mit beginnen. `main.*` Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

Geben Sie für einen Push-Trigger die Zweige an, auf die Sie pushen möchten, d. h. die Zielzweige. Geben Sie für einen Pull-Request-Trigger die Ziel-Branches an, für die Sie den Pull-Request öffnen möchten.

- (Optional) Geben Sie unter Dateipfade die Dateipfadnamen für Ihren Trigger an. Geben Sie die entsprechenden Namen in die Felder Einschließen und Ausschließen ein.

  Sie können Regex-Muster im Glob-Format verwenden, um Ihre Dateipfadnamen zu definieren. Verwenden Sie dies beispielsweise, um alle Dateipfade `prod.*` abzugleichen, die mit beginnen. `prod.*` Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

Triggerfilterung in der Pipeline-JSON (CLI)

Sie können das Pipeline-JSON aktualisieren, um Filter für Trigger hinzuzufügen.

Um die AWS CLI zum Erstellen oder Aktualisieren Ihrer Pipeline zu verwenden, verwenden Sie den `update-pipeline` Befehl `create-pipeline` or.

Die folgende Beispiel-JSON-Struktur bietet eine Referenz für die Felddefinitionen unter`create-pipeline`.

```
{
    "pipeline": {
        "name": "MyServicePipeline",
        "triggers": [
            {
                "provider": "Connection",
                "gitConfiguration": {
```

```
                    "sourceActionName": "ApplicationSource",
                    "push": [
                        {
                            "filePaths": {
                                "includes": [
                                    "projectA/**",
                                    "common/**/*.js"
                                ],
                                "excludes": [
                                    "**/README.md",
                                    "**/LICENSE",
                                    "**/CONTRIBUTING.md"
                                ]
                            },
                            "branches": {
                                "includes": [
                                    "feature/**",
                                    "release/**"
                                ],
                                "excludes": [
                                    "mainline"
                                ]
                            },
                            "tags": {
                                "includes": [
                                    "release-v0", "release-v1"
                                ],
                                "excludes": [
                                    "release-v2"
                                ]
                            }
                        }
                    ],
                    "pullRequest": [
                        {
                            "events": [
                                "CLOSED"
                            ],
                            "branches": {
                                "includes": [
                                    "feature/**",
                                    "release/**"
                                ],
                                "excludes": [
```

```
                                    "mainline"
                                ]
                            },
                            "filePaths": {
                                "includes": [
                                    "projectA/**",
                                    "common/**/*.js"
                                ],
                                "excludes": [
                                    "**/README.md",
                                    "**/LICENSE",
                                    "**/CONTRIBUTING.md"
                                ]
                            }
                        }
                    ]
                }
            }
        ],
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "ApplicationSource",
                        "configuration": {
                            "BranchName": "mainline",
                            "ConnectionArn": "arn:aws:codestar-connections:eu-
central-1:111122223333:connection/fe9ff2e8-ee25-40c9-829e-65f8EXAMPLE",
                            "FullRepositoryId": "monorepo-example",
                            "OutputArtifactFormat": "CODE_ZIP"
                        }
                    }
                ]
            }
        ]
    }
}
```

Die Felder in der JSON-Struktur sind wie folgt definiert:

- `sourceActionName`: Der Name der Pipeline-Quellaktion mit der Git-Konfiguration.

- `push`: Push-Ereignisse mit Filterung. Diese Ereignisse verwenden eine ODER-Operation zwischen verschiedenen Push-Filtern und eine UND-Operation innerhalb von Filtern.
  - `branches`: Die Zweige, nach denen gefiltert werden soll. Zweige verwenden eine UND-Operation zwischen Ein- und Ausschlüssen.
    - `includes`: Muster, nach denen nach Zweigen gefiltert wird, die aufgenommen werden sollen. Beinhaltet die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Zweigen gefiltert werden soll, die ausgeschlossen werden. Schließt die Verwendung einer ODER-Operation aus.
  - `filePaths`: Die Dateipfadnamen, nach denen gefiltert werden soll.
    - `includes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die eingeschlossen werden sollen. Beinhaltet die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die ausgeschlossen werden. Schließt die Verwendung einer ODER-Operation aus.
  - `tags`: Die Tag-Namen, nach denen gefiltert werden soll.
    - `includes`: Muster, nach denen nach Tags gefiltert werden soll, die aufgenommen werden sollen. Beinhaltet die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Tags gefiltert werden soll, die ausgeschlossen werden. Schließt die Verwendung einer ODER-Operation aus.
- `pullRequest`: Pull-Request-Ereignisse mit Filterung nach Pull-Request-Ereignissen und Pull-Request-Filtern.
  - `events`: Filtert nach offenen, aktualisierten oder geschlossenen Pull-Request-Ereignissen wie angegeben.
  - `branches`: Die Branches, nach denen gefiltert werden soll. Zweige verwenden eine UND-Operation zwischen Ein- und Ausschlüssen.
    - `includes`: Muster, nach denen nach Zweigen gefiltert wird, die aufgenommen werden sollen. Beinhaltet die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Zweigen gefiltert werden soll, die ausgeschlossen werden. Schließt die Verwendung einer ODER-Operation aus.
  - `filePaths`: Die Dateipfadnamen, nach denen gefiltert werden soll.
    - `includes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die eingeschlossen werden sollen. Beinhaltet die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die ausgeschlossen werden. Schließt die Verwendung einer ODER-Operation aus.

Löst die Filterung in Vorlagen AWS CloudFormation aus

Sie können die Pipeline-Ressource aktualisieren AWS CloudFormation , um Triggerfilterung hinzuzufügen.

Das folgende Beispiel für einen Vorlagenausschnitt enthält eine YAML-Referenz für Trigger-Felddefinitionen. Eine Liste der Felddefinitionen finden Sie unter. Triggerfilterung in der Pipeline-JSON (CLI)

```
pipeline:
  name: MyServicePipeline
  executionMode: PARALLEL
  triggers:
    - provider: CodeConnection
      gitConfiguration:
        sourceActionName: ApplicationSource
        push:
          - filePaths:
              includes:
                - projectA/**
                - common/**/*.js
              excludes:
                - '**/README.md'
                - '**/LICENSE'
                - '**/CONTRIBUTING.md'
            branches:
              includes:
                - feature/**
                - release/**
              excludes:
                - mainline
          - tags:
              includes:
                - release-v0
                - release-v1
              excludes:
                - release-v2
        pullRequest:
          - events:
              - CLOSED
            branches:
              includes:
                - feature/**
```

```
                - release/**
              excludes:
                - mainline
            filePaths:
              includes:
                - projectA/**
                - common/**/*.js
              excludes:
                - '**/README.md'
                - '**/LICENSE'
                - '**/CONTRIBUTING.md'
  stages:
    - name: Source
      actions:
        - name: ApplicationSource
          configuration:
            BranchName: mainline
            ConnectionArn: arn:aws:codestar-connections:eu-
central-1:111122223333:connection/fe9ff2e8-ee25-40c9-829e-65f85EXAMPLE
            FullRepositoryId: monorepo-example
            OutputArtifactFormat: CODE_ZIP
```

Hinrichtungen verwalten in CodePipeline

Um den Pipeline-Fortschritt zu analysieren, können Sie Fehlerprotokolle einsehen, den Verlauf der Pipeline- und Aktionsausführung einsehen und fehlgeschlagene Phasen oder Aktionen erneut versuchen.

Themen

Hinrichtungen anzeigen in CodePipeline

Sie können die AWS CodePipeline Konsole oder die verwenden, AWS CLI um den Ausführungsstatus und den Ausführungsverlauf anzuzeigen und fehlgeschlagene Phasen oder Aktionen erneut zu versuchen.

Themen

Ausführungsverlauf einer Pipeline anzeigen (Konsole)

Sie können die CodePipeline Konsole verwenden, um eine Liste aller Pipelines in Ihrem Konto anzuzeigen. Sie können auch Details zu jeder Pipeline anzeigen, beispielsweise den Zeitpunkt der letzten Ausführung einer Aktion in einer Pipeline, ob ein Übergang zwischen Phasen aktiviert oder deaktiviert ist, ob Aktionen fehlschlugen und weitere Informationen. Sie können auch eine

Verlaufsseite mit Details zu allen Pipelineausführungen anzeigen, für die ein Verlauf aufgezeichnet wurde.

> **Note**
>
> Wenn Sie zwischen bestimmten Ausführungsmodi wechseln, können sich die Pipeline-Ansicht und der Verlauf ändern. Weitere Informationen finden Sie unter Legen Sie den Pipeline-Ausführungsmodus fest oder ändern Sie ihn.

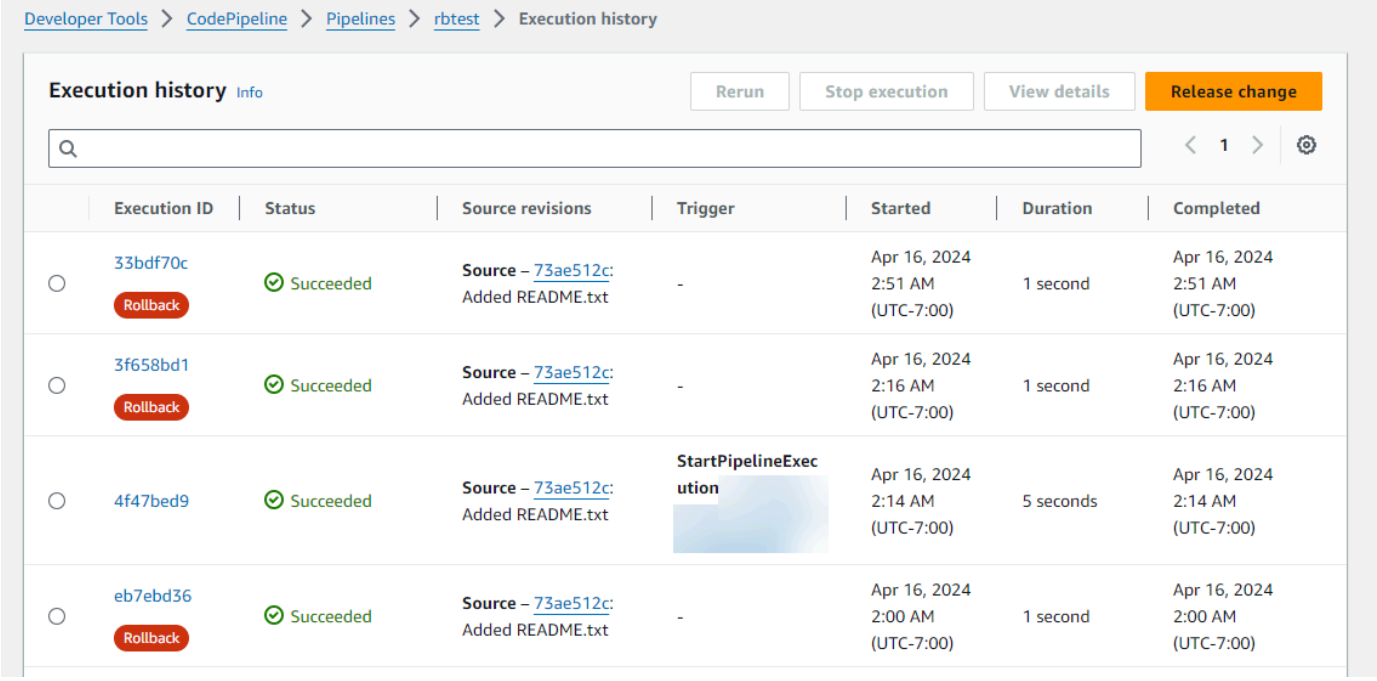
Der Ausführungsverlauf wird bis zu 12 Monate lang aufbewahrt.

Sie können die Konsole verwenden, um den Ausführungsverlauf in einer Pipeline anzuzeigen, einschließlich Status, Quellversionen und Timingdetails für jede Ausführung.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden zusammen mit ihrem Status angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline.
3. Wählen Sie View history (Verlauf anzeigen).

> **Note**
>
> Bei einer Pipeline im PARALLEL-Ausführungsmodus werden in der Pipeline-Hauptansicht weder die Pipeline-Struktur noch die laufenden Ausführungen angezeigt. Bei einer Pipeline im PARALLEL-Ausführungsmodus greifen Sie auf die Pipeline-Struktur zu, indem Sie auf der Seite mit dem Ausführungsverlauf die ID für die Ausführung auswählen, die Sie anzeigen möchten. Wählen Sie im linken Navigationsbereich Verlauf aus, wählen Sie die Ausführungs-ID für die parallel Ausführung aus, und zeigen Sie dann die Pipeline auf der Registerkarte Visualisierung an.



4. Zeigen Sie Status, Quellrevisionen, Änderungsdetails und Auslöser für jede Ausführung Ihrer Pipeline an. Bei Pipeline-Ausführungen, für die ein Rollback vorgenommen wurde, wird auf dem Detailbildschirm der Konsole der Ausführungstyp Rollback angezeigt. Für die fehlgeschlagene Ausführung, die das automatische Rollback ausgelöst hat, wird die ID der fehlgeschlagenen Ausführung angezeigt.
5. Wählen Sie eine Ausführung aus. In der Detailansicht werden Ausführungsdetails, die Registerkarte Zeitleiste, die Registerkarte Visualisierung und die Registerkarte Variablen angezeigt. Variablenwerte für Variablen auf Pipeline-Ebene werden zum Zeitpunkt der Pipeline-Ausführung aufgelöst und können im Ausführungsverlauf für jede Ausführung eingesehen werden.

> **Note**
>
> Ausgabevariablen von Pipeline-Aktionen können auf der Registerkarte Ausgabevariablen unter dem Verlauf für jede Aktionsausführung angezeigt werden.

Anzeige des Ausführungsstatus (Konsole)

Sie können den Pipeline-Status unter Status auf der Ausführungsverlaufsseite anzeigen. Wählen Sie einen Ausführungs-ID-Link, und zeigen Sie dann den Aktionsstatus an.

Die folgenden Status gelten für Pipelines, Phasen und Aktionen:

> **Note**
>
> Die folgenden Pipeline-Status gelten auch für eine Pipeline-Ausführung, bei der es sich um eine eingehende Ausführung handelt. Informationen zur Anzeige einer eingehenden Ausführung und ihres Status finden Sie unter. Eine eingehende Ausführung anzeigen (Konsole)

Status auf Pipelineebene

Pipelinestatus	Beschreibung
InProgress	Die Pipelineausführung wird derzeit ausgeführt.
Wird angehalten	Die Pipeline-Ausführung wird aufgrund einer Anforderung mit "Anhalten und beenden" oder "Anhalten und warten" angehalten.
Angehalten	Der Anhaltevorgang ist abgeschlossen, und die Pipeline-Ausführung wird angehalten.
Erfolgreich	Die Pipelineausführung wurde erfolgreich abgeschlossen.
Ersetzt	Während auf den Abschluss der nächsten Stufe dieser Pipelinea usführung gewartet wurde, wurde stattdessen eine neuere Pipelinea usführung über die Pipeline fortgesetzt.
Fehlgeschlagen	Die Pipelineausführung wurde nicht erfolgreich abgeschlossen.

Status auf Stufenebene

Stufenstatus	Beschreibung
InProgress	Die Stufe wird derzeit ausgeführt.
Wird angehalten	Die Ausführung der Phase wird aufgrund einer Anforderung zum Anhalten und Warten oder zum Anhalten und Beenden der Pipeline-Ausführung gestoppt.

Stufenstatus	Beschreibung
Angehalten	Der Anhaltevorgang ist abgeschlossen, und die Phasenausführung wird angehalten.
Erfolgreich	Die Stufe wurde erfolgreich abgeschlossen.
Fehlgeschlagen	Die Stufe wurde nicht erfolgreich abgeschlossen.

Status auf Aktionsebene

Aktionsstatus	Beschreibung
InProgress	Die Aktion wird derzeit ausgeführt.
Abandoned (Abgebrochen)	Die Aktion wird aufgrund einer Anforderung mit "Anhalten und Beenden" für die Pipeline-Ausführung beendet.
Erfolgreich	Die Aktion wurde erfolgreich abgeschlossen.
Fehlgeschlagen	Bei Genehmigungsaktionen bedeutet der Status FAILED (FEHLGESCHLAGEN), dass die Aktion entweder vom Prüfer abgelehnt wurde oder aufgrund einer falschen Aktionskonfiguration fehlgeschlagen ist.

Eine eingehende Ausführung anzeigen (Konsole)

Sie können die Konsole verwenden, um den Status und die Details einer eingehenden Ausführung anzuzeigen. Wenn der Übergang aktiviert ist oder die Phase verfügbar wird, wird eine eingehende Ausführung `InProgress` fortgesetzt und tritt in die Phase ein. Eine eingehende Ausführung mit einem `Stopped` Status tritt nicht in die Phase ein. Der Status einer eingehenden Ausführung ändert sich in, `Failed` wenn die Pipeline bearbeitet wird. Wenn Sie eine Pipeline bearbeiten, werden alle laufenden Ausführungen nicht fortgesetzt, und der Ausführungsstatus ändert sich auf. `Failed`

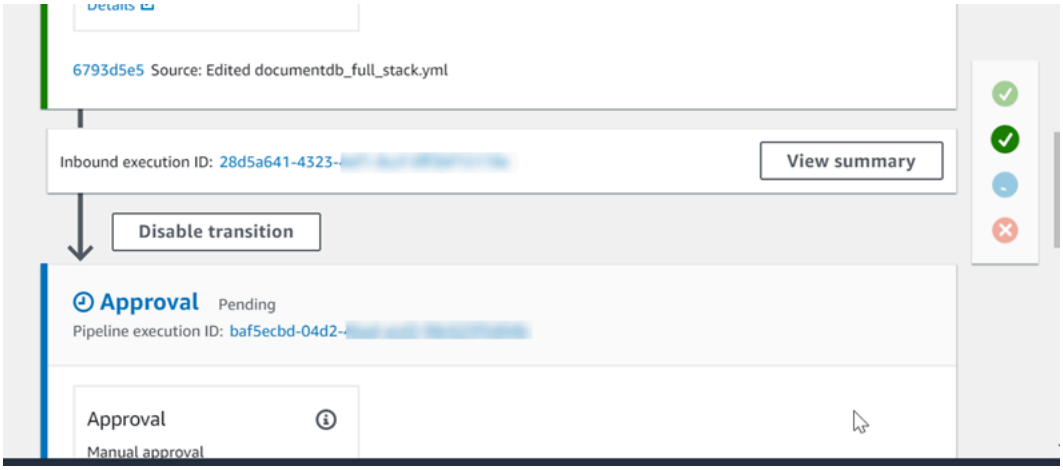
Wenn Sie keine eingehende Ausführung sehen, gibt es in einer deaktivierten Phase des Übergangs keine ausstehenden Ausführungen.

1. Melden Sie sich unter http://console.aws.amazon.com/codesuite/codepipeline/home bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole.

Die Namen aller Pipelines, die mit Ihrem AWS Konto verknüpft sind, werden angezeigt.

2. Wählen Sie den Namen der Pipeline, für die Sie die eingehende Ausführung anzeigen möchten, und führen Sie einen der folgenden Schritte aus:

   - Wählen Sie View (Anzeige). Im Pipeline-Diagramm können Sie im Feld Inbound Execution ID vor Ihrer deaktivierten Transition die Ausführungs-ID für den eingehenden Datenverkehr einsehen.



     Wählen Sie Zusammenfassung anzeigen, um Ausführungsdetails wie die Ausführungs-ID, den Quell-Trigger und den Namen der nächsten Phase anzuzeigen.

   - Wählen Sie die Pipeline aus und klicken Sie auf Verlauf anzeigen.

Anzeigen von Revisionen der Pipeline-Ausführungsquelle (Konsole)

Sie können Einzelheiten zu Quellartefakten (Ausgabeartefakten, die aus der ersten Phase einer Pipeline stammen) anzeigen, die bei der Ausführung einer Pipeline verwendet werden. Dazu gehören IDs wie z. B. Commit-IDs, Kommentare zu Zugängen und bei Verwendung der CLI die Versionsnummern der Pipeline-Build-Aktionen. Bei einigen Revisionstypen können Sie die URL des Commits anzeigen und öffnen. Quell-Revisionen setzen sich folgendermaßen zusammen:

- Zusammenfassung: Zusammengefasste Informationen über die letzte Revision des Artefakts. Für GitHub und CodeCommit Repositorys, die Commit-Nachricht. Für Amazon S3 S3-Buckets oder -Aktionen der vom Benutzer bereitgestellte Inhalt eines codepipeline-artifact-revision-summary Schlüssels, der in den Objektmetadaten angegeben ist.
- revisionUrl: Die Revisions-URL für die Artefakt-Revision (z. B. die externe Repository-URL).

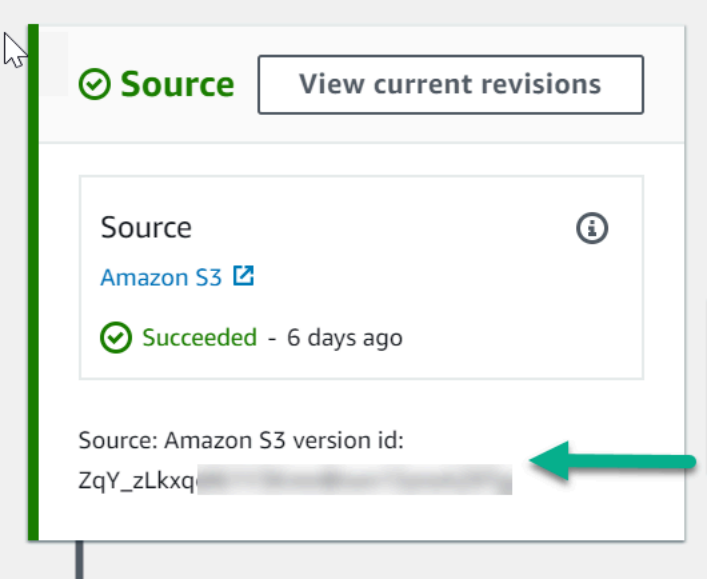
- revisionId: Die Versions-ID für die Artefakt-Revision. Für eine Quellenänderung in einem CodeCommit GitHub OR-Repository ist dies beispielsweise die Commit-ID. Bei Artefakten, die in GitHub oder CodeCommit Repositorien gespeichert sind, ist die Commit-ID mit einer Commit-Detailseite verknüpft.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

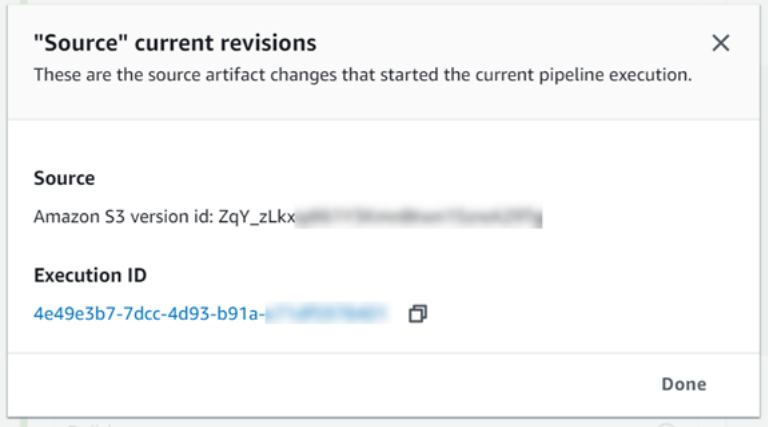
   Die Namen aller Pipelines, die mit Ihrer verknüpft sind AWS-Konto, werden angezeigt.

2. Wählen Sie den Namen der Pipeline aus, für die Sie die Quellrevision anzeigen möchten. Führen Sie eine der folgenden Aktionen aus:

   - Wählen Sie View history (Verlauf anzeigen). Unter Source revisions (Quell-Revisionen) wird die Quelländerung für jede Ausführung aufgeführt.
   - Suchen Sie eine Aktion, für die Sie Details anzeigen möchten, und suchen Sie dann die Quellrevisionsinformationen unten in ihrer Stufe:



   Wählen Sie View current revisions (Aktuelle Versionen anzeigen) aus, um Quellinformationen anzuzeigen. Mit Ausnahme von Artefakten, die in Amazon S3 S3-Buckets gespeichert sind, sind Identifikatoren wie Commit-IDs in dieser Informationsdetailansicht mit Quellinformationsseiten für die Artefakte verknüpft.



Anzeigen von Aktionsausführungen (Konsole)

Sie können Aktionsdetails für eine Pipeline anzeigen, wie etwa die Aktionsausführungs-ID, Eingabeartefakte, Ausgabeartefakte und den Status. Sie können Aktionsdetails anzeigen, indem Sie in der Konsole eine Pipeline und dann eine Ausführungs-ID auswählen.

1. Melden Sie sich unter http://console.aws.amazon.com/codesuite/codepipeline/home bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie den Namen der Pipeline aus, für die Sie die Aktionsdetails anzeigen möchten, und wählen Sie dann View history (Verlauf anzeigen).
3. Wählen Sie unter Execution ID (Ausführungs-ID) die Ausführungs-ID, für die Sie die Details der Aktionsausführung anzeigen möchten.
4. Auf der Registerkarte Timeline (Zeitleiste) sehen Sie die folgenden Informationen:

   a. Wählen Sie unter Action name (Aktionsname) den Link zum Öffnen der Aktion, wo Sie Status, Phasenname, Aktionsname, Konfigurationsdaten und Artefaktinformationen finden.

   b. Wählen Sie unter Provider (Anbieter) den link zur Anzeige der Details zum Aktionsanbieter. Wenn Sie beispielsweise in der vorherigen Beispielpipeline entweder die Staging- oder die Produktionsphase wählen CodeDeploy , wird die CodeDeploy Konsolenseite für die für diese Phase konfigurierte CodeDeploy Anwendung angezeigt.

Anzeigen von Informationen zu Aktionsartefakten und zum Artefaktspeicher (Konsole)

Sie können Eingabe- und Ausgabe-Artefaktdetails für eine Aktion anzeigen. Sie können auch einen Link auswählen, über den Sie die Artefakt-Informationen für diese Aktion anzeigen können. Da der Artefaktspeicher Versioning verwendet, hat jede Aktionsausführung einen eindeutigen Speicherort für Eingabe- und Ausgabe-Artefakte.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie den Namen der Pipeline aus, für die Sie die Aktionsdetails anzeigen möchten, und wählen Sie dann View history (Verlauf anzeigen).
3. Wählen Sie unter Execution ID (Ausführungs-ID) die Ausführungs-ID, für die Sie die Aktionsdetails anzeigen möchten.
4. Wählen Sie auf der Registerkarte Timeline unter Action name (Aktionsname) den Link zum Öffnen einer Detailseite für die Aktion.
5. Sehen Sie sich auf der Detailseite auf der Registerkarte Ausführung den Status und den Zeitpunkt der Ausführung der Aktion an.
6. Sehen Sie sich auf der Registerkarte Konfiguration die Ressourcenkonfiguration für die Aktion an (z. B. den Namen des CodeBuild Build-Projekts).
7. Sehen Sie sich auf der Registerkarte Artefakte die Artefaktdetails unter Artefakttyp und Artefaktanbieter an. Wählen Sie den Link unter Artifact name (Artefaktname) um die Artefakte im Artefaktspeicher anzuzeigen.
8. Sehen Sie sich auf der Registerkarte Ausgabevariablen die aufgelösten Variablen von Aktionen in der Pipeline für die Aktionsausführung an.

Anzeigen von der Pipeline-Details und des Verlaufs (CLI)

Sie können die folgenden Befehle verwenden, um weitere Details zu Ihren Pipelines und Pipeline-Ausführungen anzuzeigen:

- list-pipelinesBefehl, um eine Zusammenfassung aller Pipelines anzuzeigen, die mit Ihrem AWS-Konto verknüpft sind.

- get-pipeline-Befehle, um Details einer einzelnen Pipeline zu überprüfen.
- list-pipeline-executions, um Zusammenfassungen der letzten Ausführungen für eine Pipeline anzuzeigen
- get-pipeline-execution, um Informationen zu einer Ausführung einer Pipeline zurück, einschließlich Details zu Artefakten, Pipeline-Ausführungs-ID und Name, Version und Status der Pipeline anzuzeigen.
- get-pipeline-state-Befehl zum Anzeigen von Pipeline, Phase und Aktionsstatus.
- Aktion list-action-executions zum Anzeigen von Ausführungsdetails für eine Pipeline.

Themen

Ausführungsverlauf anzeigen mit **`list-pipeline-executions`** (CLI)

Sie können den Ausführungsverlauf einer Pipeline anzeigen.

- Um Details zu vorherigen Ausführungen einer Pipeline anzuzeigen, führen Sie den Befehl list-pipeline-executions unter Angabe des eindeutigen Namens der Pipeline aus. Um beispielsweise Details zum aktuellen Status einer Pipeline mit dem Namen anzuzeigen *`MyFirstPipeline`*, geben Sie Folgendes ein:

  ```
  aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
  ```

  Dieser Befehl gibt zusammenfassende Informationen zu allen Pipelineausführungen zurück, für die ein Verlauf aufgezeichnet wurde. Die Zusammenfassung enthält Start- und Endzeiten, Dauer und Status.

  Bei Pipeline-Ausführungen, für die ein Rollback vorgenommen wurde, wird der Ausführungstyp `Rollback` angezeigt. Für die fehlgeschlagene Ausführung, die das automatische Rollback ausgelöst hat, wird die ID der fehlgeschlagenen Ausführung angezeigt.

Das folgende Beispiel zeigt die zurückgegebenen Daten für eine Pipeline mit dem Namen *MyFirstPipeline*, die dreimal ausgeführt wurde:


```
{
    "pipelineExecutionSummaries": [
        {
            "pipelineExecutionId": "eb7ebd36-353a-4551-90dc-18ca5EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T09:00:28.185000+00:00",
            "lastUpdateTime": "2024-04-16T09:00:29.665000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
                    "revisionUrl": "console-URL"
                }
            ],
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "trigger_ARN"
            },
            "executionMode": "SUPERSEDED"
        },
        {
            "pipelineExecutionId": "fcd61d8b-4532-4384-9da1-2aca1EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T08:58:56.601000+00:00",
            "lastUpdateTime": "2024-04-16T08:59:04.274000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
                    "revisionUrl": "console_URL"
                }
            ],
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "trigger_ARN"
            },
            "executionMode": "SUPERSEDED"
```

```
        }
```

Um weitere Details über eine Pipeline-Ausführung anzuzeigen, führen Sie den Befehl get-pipeline-execution unter Angabe der eindeutigen ID der Pipeline-Ausführung aus. Um beispielsweise weitere Informationen über die erste Ausführung im vorherigen Beispiel anzuzeigen, geben Sie Folgendes ein:

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-execution-id 7cf7f7cb-3137-539g-j458-d7eu3EXAMPLE
```

Dieser Befehl gibt Informationen zu einer Ausführung einer Pipeline, einschließlich Details zu Artefakten, Pipeline-Ausführungs-ID und Name, Version und Status der Pipeline zurück.

Das folgende Beispiel zeigt die zurückgegebenen Daten für eine Pipeline mit dem Namen *MyFirstPipeline*:

```
{
    "pipelineExecution": {
        "pipelineExecutionId": "3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE",
        "pipelineVersion": 2,
        "pipelineName": "MyFirstPipeline",
        "status": "Succeeded",
        "artifactRevisions": [
            {
                "created": 1496380678.648,
                "revisionChangeIdentifier": "1496380258.243",
                "revisionId": "7636d59f3c461cEXAMPLE8417dbc6371",
                "name": "MyApp",
                "revisionSummary": "Updating the application for feature 12-4820"
            }
        ]
    }
}
```

Pipeline-Status anzeigen mit **`get-pipeline-state`** (CLI)

Sie können die CLI verwenden, um den Pipeline-, Phasen- und Aktionsstatus anzuzeigen.

- Um Informationen über den aktuellen Status einer Pipeline anzuzeigen, führen Sie den Befehl get-pipeline-state unter Angabe des eindeutigen Namens der Pipeline aus. Um beispielsweise

Details zum aktuellen Status einer Pipeline mit dem Namen anzuzeigen *MyFirstPipeline*, geben Sie Folgendes ein:

```
aws codepipeline get-pipeline-state --name MyFirstPipeline
```

Dieser Befehl gibt den aktuellen Status aller Stufen der Pipeline und den Status der Aktionen innerhalb dieser Stufen zurück.

Das folgende Beispiel zeigt die zurückgegebenen Daten für eine dreistufige Pipeline mit dem Namen *MyFirstPipeline*, wobei die ersten beiden Phasen und Aktionen Erfolg, die dritte einen Fehler anzeigen und der Übergang zwischen der zweiten und dritten Phase deaktiviert ist:

```
{
    "updated": 1427245911.525,
    "created": 1427245911.525,
    "pipelineVersion": 1,
    "pipelineName": "MyFirstPipeline",
    "stageStates": [
        {
            "actionStates": [
                {
                    "actionName": "Source",
                    "entityUrl": "https://console.aws.amazon.com/s3/home?#",
                    "latestExecution": {
                        "status": "Succeeded",
                        "lastStatusChange": 1427298837.768
                    }
                }
            ],
            "stageName": "Source"
        },
        {
            "actionStates": [
                {
                    "actionName": "Deploy-CodeDeploy-Application",
                    "entityUrl": "https://console.aws.amazon.com/codedeploy/home?
#",
                    "latestExecution": {
                        "status": "Succeeded",
                        "lastStatusChange": 1427298939.456,
                        "externalExecutionUrl": "https://console.aws.amazon.com/?
#",
```

```
                        "externalExecutionId": ""c53dbd42-This-Is-An-Example"",
                        "summary": "Deployment Succeeded"
                    }
                }
            ],
            "inboundTransitionState": {
                "enabled": true
            },
            "stageName": "Staging"
        },
        {
            "actionStates": [
                {
                    "actionName": "Deploy-Second-Deployment",
                    "entityUrl": "https://console.aws.amazon.com/codedeploy/home?
#",
                    "latestExecution": {
                        "status": "Failed",
                        "errorDetails": {
                            "message": "Deployment Group is already deploying
 deployment ...",
                            "code": "JobFailed"
                        },
                        "lastStatusChange": 1427246155.648
                    }
                }
            ],
            "inboundTransitionState": {
                "disabledReason": "Disabled while I investigate the failure",
                "enabled": false,
                "lastChangedAt": 1427246517.847,
                "lastChangedBy": "arn:aws:iam::80398EXAMPLE:user/CodePipelineUser"
            },
            "stageName": "Production"
        }
    ]
}
```

Ausführungsstatus eingehender Nachrichten mit **`get-pipeline-state`** (CLI) anzeigen

Sie können die CLI verwenden, um den Ausführungsstatus eingehender Nachrichten anzuzeigen. Wenn der Übergang aktiviert ist oder die Phase verfügbar wird, wird eine eingehende Ausführung `InProgress` fortgesetzt und tritt in die Phase ein. Eine eingehende Ausführung mit einem `Stopped` Status tritt nicht in die Phase ein. Der Status einer eingehenden Ausführung ändert sich in, `Failed` wenn die Pipeline bearbeitet wird. Wenn Sie eine Pipeline bearbeiten, werden alle laufenden Ausführungen nicht fortgesetzt, und der Ausführungsstatus ändert sich auf. `Failed`

- Um Informationen über den aktuellen Status einer Pipeline anzuzeigen, führen Sie den Befehl get-pipeline-state unter Angabe des eindeutigen Namens der Pipeline aus. Um beispielsweise Details zum aktuellen Status einer Pipeline mit dem Namen anzuzeigen *`MyFirstPipeline`*, geben Sie Folgendes ein:

  ```
  aws codepipeline get-pipeline-state --name MyFirstPipeline
  ```

  Dieser Befehl gibt den aktuellen Status aller Stufen der Pipeline und den Status der Aktionen innerhalb dieser Stufen zurück. Die Ausgabe zeigt auch die Pipeline-Ausführungs-ID in jeder Phase und ob es für eine Phase mit deaktiviertem Übergang eine eingehende Ausführungs-ID gibt.

  Das folgende Beispiel zeigt die zurückgegebenen Daten für eine zweistufige Pipeline mit dem Namen *`MyFirstPipeline`*, wobei die erste Phase einen aktivierten Übergang und eine erfolgreiche Pipelineausführung anzeigt und die zweite Phase, benanntBeta, einen deaktivierten Übergang und eine Ausführungs-ID für eingehenden Datenverkehr anzeigt. Die eingehende Ausführung kann den Status `InProgressStopped`, oder haben. `FAILED`

  ```
  {
      "pipelineName": "MyFirstPipeline",
      "pipelineVersion": 2,
      "stageStates": [
          {
              "stageName": "Source",
              "inboundTransitionState": {
                  "enabled": true
              },
              "actionStates": [
                  {
  ```

```
                    "actionName": "SourceAction",
                    "currentRevision": {
                        "revisionId": "PARcnxX_u0SMRBnKh83pHL09.zPRLLMu"
                    },
                    "latestExecution": {
                        "actionExecutionId": "14c8b311-0e34-4bda-EXAMPLE",
                        "status": "Succeeded",
                        "summary": "Amazon S3 version id: PARcnxX_u0EXAMPLE",
                        "lastStatusChange": 1586273484.137,
                        "externalExecutionId": "PARcnxX_u0EXAMPLE"
                    },
                    "entityUrl": "https://console.aws.amazon.com/s3/home?#"
                }
            ],
            "latestExecution": {
                "pipelineExecutionId": "27a47e06-6644-42aa-EXAMPLE",
                "status": "Succeeded"
            }
        },
        {
            "stageName": "Beta",
            "inboundExecution": {
                "pipelineExecutionId": "27a47e06-6644-42aa-958a-EXAMPLE",
                "status": "InProgress"
            },
            "inboundTransitionState": {
                "enabled": false,
                "lastChangedBy": "USER_ARN",
                "lastChangedAt": 1586273583.949,
                "disabledReason": "disabled"
            },
                    "currentRevision": {
            "actionStates": [
                {
                    "actionName": "BetaAction",
                    "latestExecution": {
                        "actionExecutionId": "a748f4bf-0b52-4024-98cf-EXAMPLE",
                        "status": "Succeeded",
                        "summary": "Deployment Succeeded",
                        "lastStatusChange": 1586272707.343,
                        "externalExecutionId": "d-KFGF3EXAMPLE",
                        "externalExecutionUrl": "https://us-
west-2.console.aws.amazon.com/codedeploy/home?#/deployments/d-KFGF3WTS2"
                    },
```

```
                    "entityUrl": "https://us-west-2.console.aws.amazon.com/
codedeploy/home?#/applications/my-application"
                }
            ],
            "latestExecution": {
                "pipelineExecutionId": "f6bf1671-d706-4b28-EXAMPLE",
                "status": "Succeeded"
            }
        }
    ],
    "created": 1585622700.512,
    "updated": 1586273472.662
}
```

Status- und Quellversionen mit **`get-pipeline-execution`** (CLI) anzeigen

Sie können Einzelheiten zu Quellartefakten (Ausgabeartefakten, die aus der ersten Phase einer Pipeline stammen) anzeigen, die bei der Ausführung einer Pipeline verwendet werden. Diese Informationen umfassen IDs wie z. B. Commit-IDs, Kommentare zu Zugängen, Zeit seit Erstellung oder Aktualisierung des Artefakts und bei Verwendung der CLI die Versionsnummern der Build-Aktionen. Bei einigen Revisionstypen können Sie die URL des Commits für die Artefaktversion ansehen und öffnen. Quell-Revisionen setzen sich folgendermaßen zusammen:

- Zusammenfassung: Zusammengefasste Informationen über die letzte Revision des Artefakts. Für GitHub und AWS CodeCommit Repositorys, die Commit-Nachricht. Für Amazon S3 S3-Buckets oder -Aktionen der vom Benutzer bereitgestellte Inhalt eines codepipeline-artifact-revision-summary Schlüssels, der in den Objektmetadaten angegeben ist.
- revisionUrl: Die Commit-ID für die Artefaktrevision. Bei Artefakten, die in GitHub oder AWS CodeCommit Repositorys gespeichert sind, ist die Commit-ID mit einer Commit-Detailseite verknüpft.

Sie können mithilfe des Befehls get-pipeline-execution Informationen über die letzten Quellrevisionen, die in die Ausführung einer Pipeline eingeschlossen wurden, anzeigen lassen. Nach der ersten Ausführung des Befehls get-pipeline-state zum Abruf von Details zu allen Phasen in einer Pipeline identifizieren Sie die Ausführungs-ID, die für eine Phase gilt, für die Sie Quell-Revisionsdetails abrufen möchten. Anschließend verwenden Sie die Ausführungs-ID im Befehl get-pipeline-execution. (Da Phasen in einer Pipeline möglicherweise während verschiedener Pipeline-Ausführungen zuletzt erfolgreich abgeschlossen wurden, können sie unterschiedliche Ausführungs-IDs aufweisen.)

Das heißt, wenn Sie Details über Artefakte in der aktuellen Staging-Phase einsehen möchten, führen Sie den Befehl get-pipeline-state aus, ermitteln Sie die aktuelle Ausführungs-ID der Staging-Phase und führen Sie dann den Befehl get-pipeline-execution anhand dieser Ausführungs-ID aus.

Um Status- und Quellversionen in einer Pipeline anzuzeigen

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie die AWS CLI , um den get-pipeline-state Befehl auszuführen. Für eine Pipeline mit dem Namen *MyFirstPipeline*würden Sie Folgendes eingeben:

   ```
   aws codepipeline get-pipeline-state --name MyFirstPipeline
   ```

   Dieser Befehl gibt den letzten Status einer Pipeline zurück (einschließlich der neuesten Pipeline-Ausführungs-ID für jede Stufe).

2. Um Details zur Ausführung einer Pipeline anzuzeigen, führen Sie den Befehl get-pipeline-execution unter Angabe des eindeutigen Namens der Pipeline und der Ausführungs-ID der Ausführung aus, für die Sie die Artefaktdetails anzeigen möchten. Um beispielsweise Details zur Ausführung einer Pipeline mit der Ausführungs-ID *MyFirstPipeline*3137f7cb-7cf7-039j-S83L-D7EU3Example anzuzeigen, würden Sie Folgendes eingeben:

   ```
   aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
   execution-id 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE
   ```

   Dieser Befehl gibt Informationen zu jeder Quellrevision zurück, die Teil der Pipeline-Ausführung ist, sowie ID-Informationen zur Pipeline. Es sind Informationen zu den in der Ausführung enthaltenen Pipeline-Stufen enthalten. Es gibt möglicherweise andere Stufen in der Pipeline, die nicht Teil dieser Pipeline-Ausführung waren.

   Das folgende Beispiel zeigt die zurückgegebenen Daten für einen Teil der Pipeline mit dem Namen, in dem ein Artefakt namens "" in einem Repository gespeichert ist: *MyFirstPipeline*MyApp GitHub

3. ```
   {
       "pipelineExecution": {
           "artifactRevisions": [
               {
                   "created": 1427298837.7689769,
                   "name": "MyApp",
                   "revisionChangeIdentifier": "1427298921.3976923",
   ```
   ```

```
                "revisionId": "7636d59f3c461cEXAMPLE8417dbc6371",
                "revisionSummary": "Updating the application for feature 12-4820",
                "revisionUrl": "https://api.github.com/repos/anycompany/MyApp/git/
commits/7636d59f3c461cEXAMPLE8417dbc6371"
            }
        ],
        "pipelineExecutionId": "3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE",
        "pipelineName": "MyFirstPipeline",
        "pipelineVersion": 2,
        "status": "Succeeded",
        "executionMode": "SUPERSEDED",
        "executionType": "ROLLBACK",
        "rollbackMetadata": {
            "rollbackTargetPipelineExecutionId": "4f47bed9-6998-476c-a49d-
e60beEXAMPLE"
         }
    }
}
```

Aktionsausführungen anzeigen mit **`list-action-executions`** (CLI)

Sie können Aktionsausführungsdetails für eine Pipeline anzeigen, wie etwa die Aktionsausführungs-ID, Eingabeartefakte, Ausgabeartefakte sowie das Ausführungsergebnis und den Status. Sie geben den Ausführungs-ID-Filter an, um eine Liste der Aktionen in einer Pipeline-Ausführung auszugeben:

> **Note**
>
> Der detaillierte Ausführungsverlauf ist für die Ausführungen verfügbar, die am oder nach dem 21. Februar 2019 erfolgten.

- Gehen Sie zum Anzeigen der Aktionsausführungen für ein Pipeline wie folgt vor:
  - Um Details zu allen Ausführungen in einer Pipeline anzuzeigen, führen Sie den Befehl list-action-executions unter Angabe des eindeutigen Namens der Pipeline aus. Um beispielsweise Aktionsausführungen in einer Pipeline mit dem Namen anzuzeigen *`MyFirstPipeline`*, geben Sie Folgendes ein:

    ```
    aws codepipeline list-action-executions --pipeline-name MyFirstPipeline
    ```

Das folgende Beispiel zeigt einen Teil der Beispielausgabe für diesen Befehl:

```
{
    "actionExecutionDetails": [
        {
            "actionExecutionId": "ID",
            "lastUpdateTime": 1552958312.034,
            "startTime": 1552958246.542,
            "pipelineExecutionId": "Execution_ID",
            "actionName": "Build",
            "status": "Failed",
            "output": {
                "executionResult": {
                    "externalExecutionUrl": "Project_ID",
                    "externalExecutionSummary": "Build terminated with state:
 FAILED",
                    "externalExecutionId": "ID"
                },
                "outputArtifacts": []
            },
            "stageName": "Beta",
            "pipelineVersion": 8,
            "input": {
                "configuration": {
                    "ProjectName": "java-project"
                },
                "region": "us-east-1",
                "inputArtifacts": [
                    {
                        "s3location": {
                            "bucket": "codepipeline-us-east-1-ID",
                            "key": "MyFirstPipeline/MyApp/Object.zip"
                        },
                        "name": "MyApp"
                    }
                ],
                "actionTypeId": {
                    "version": "1",
                    "category": "Build",
                    "owner": "AWS",
                    "provider": "CodeBuild"
                }
            }
```

```
        },

. . .
```

- Um Details zu allen Ausführungen in einer Pipelineausführung anzuzeigen, führen Sie den Befehl list-action-executions unter Angabe des eindeutigen Namens der Pipeline und der Ausführungs-ID aus. Wenn Sie beispielsweise die Aktionsausführungen für eine *Ausführungs-ID* anzeigen möchten, geben Sie Folgendes ein:

```
aws codepipeline list-action-executions --pipeline-name MyFirstPipeline --filter
 pipelineExecutionId=Execution_ID
```

- Das folgende Beispiel zeigt einen Teil der Beispielausgabe für diesen Befehl:

```
{
    "actionExecutionDetails": [
        {
            "stageName": "Beta",
            "pipelineVersion": 8,
            "actionName": "Build",
            "status": "Failed",
            "lastUpdateTime": 1552958312.034,
            "input": {
                "configuration": {
                    "ProjectName": "java-project"
                },
                "region": "us-east-1",
                "actionTypeId": {
                    "owner": "AWS",
                    "category": "Build",
                    "provider": "CodeBuild",
                    "version": "1"
                },
                "inputArtifacts": [
                    {
                        "s3location": {
                            "bucket": "codepipeline-us-east-1-ID",
                            "key": "MyFirstPipeline/MyApp/Object.zip"
                        },
                        "name": "MyApp"
                    }
                ]
            },
```

```
. . .
```

Legen Sie den Pipeline-Ausführungsmodus fest oder ändern Sie ihn

Sie können den Ausführungsmodus für Ihre Pipeline festlegen, um anzugeben, wie mehrere Ausführungen behandelt werden.

Weitere Informationen zu den Ausführungsmodi der Pipeline finden Sie unterSo funktionieren Pipeline-Ausführungen.

> **Important**
>
> Bei Pipelines im PARALLEL-Modus wird der aktualisierte Status nicht als PARALLEL angezeigt, wenn der Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED geändert wird. Weitere Informationen finden Sie unter Bei Pipelines, die vom PARALLEL-Modus aus geändert wurden, wird ein früherer Ausführungsmodus angezeigt.

> **Important**
>
> Bei Pipelines im Modus PARALLEL wird die Pipeline-Definition für die Pipeline in den einzelnen Modi nicht aktualisiert, wenn der Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED geändert wird. Weitere Informationen finden Sie unter Pipelines im PARALLEL-Modus weisen eine veraltete Pipeline-Definition auf, wenn sie beim Wechsel in den QUEUED- oder SUPERSEDED-Modus bearbeitet werden.

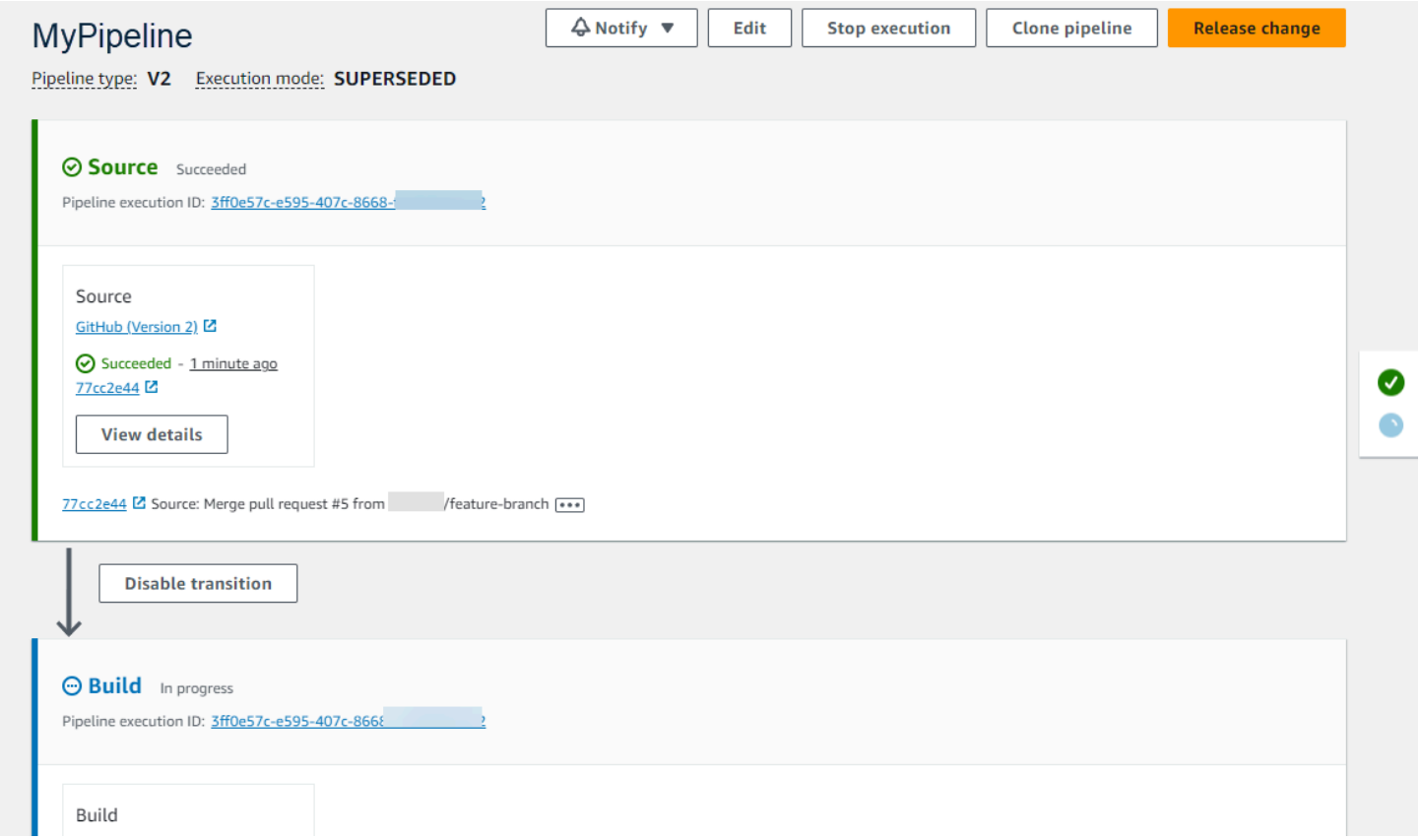
Überlegungen zur Anzeige der Ausführungsmodi

Es gibt Überlegungen zur Anzeige von Pipelines in bestimmten Ausführungsmodi.

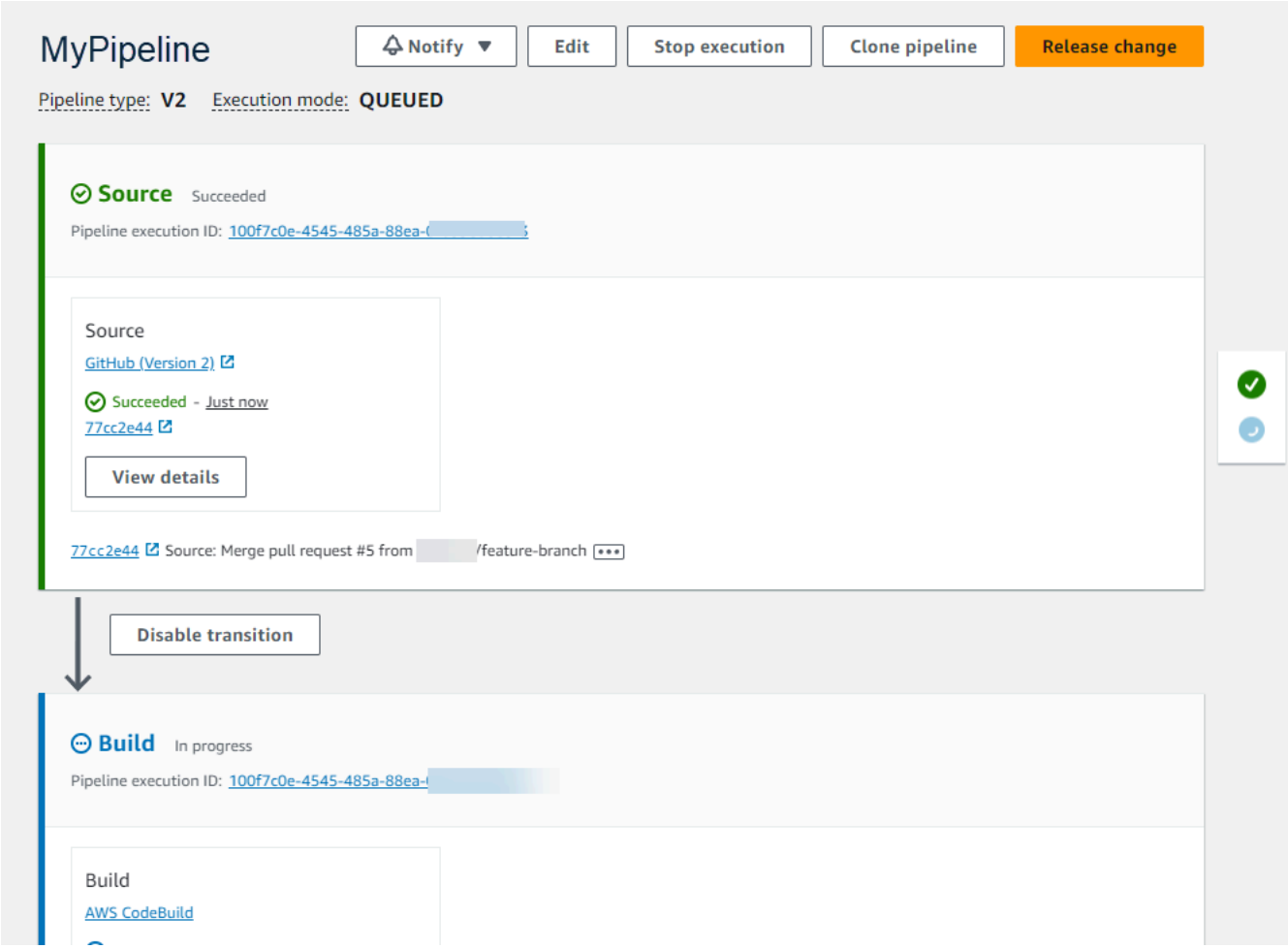
Verwenden Sie für die Modi SUPERSEDED und QUEUED die Pipeline-Ansicht, um die laufenden Ausführungen zu sehen, und klicken Sie auf die Ausführungs-ID, um Details und den Verlauf

anzuzeigen. Klicken Sie im Modus PARALLEL auf die Ausführungs-ID, um die laufende Ausführung auf der Registerkarte Visualisierung anzuzeigen.

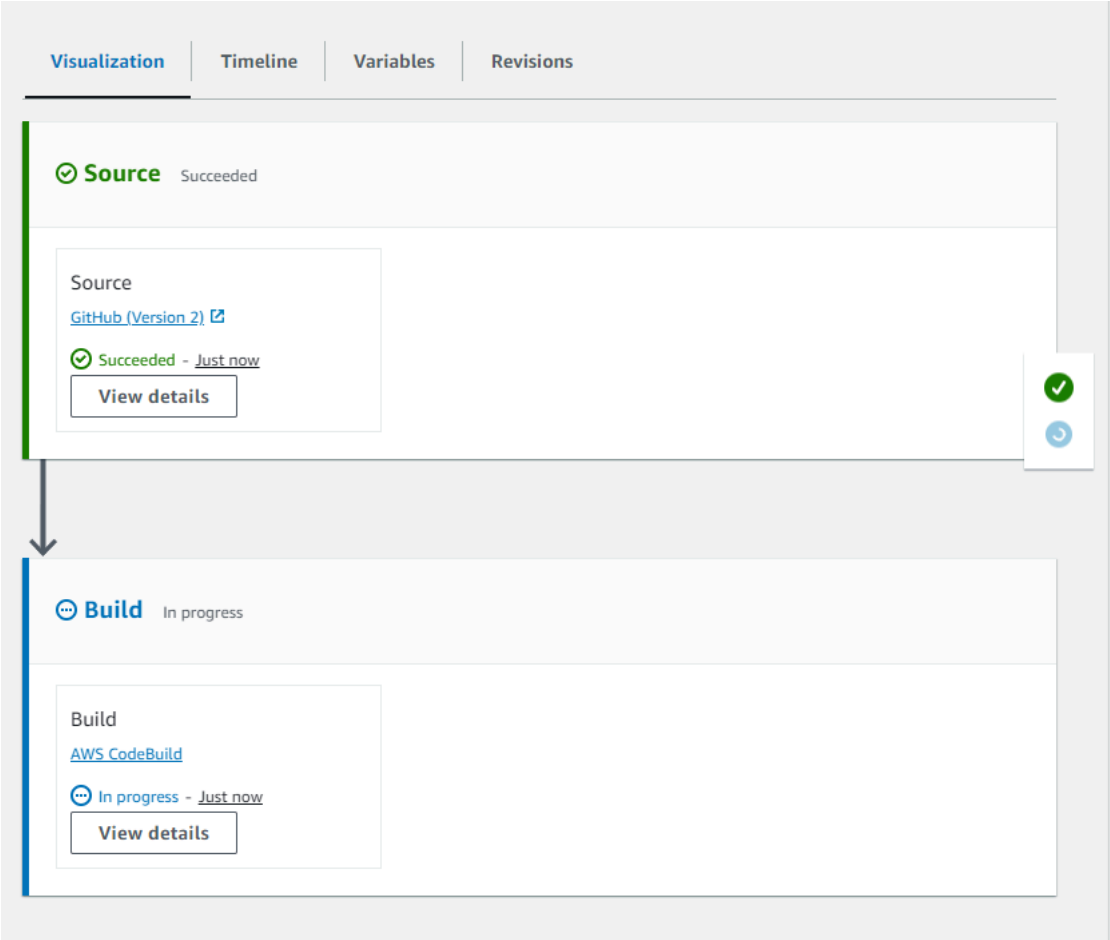
Die folgende Abbildung zeigt die Ansicht für den Modus SUPERSEDED in. CodePipeline



Die folgende Abbildung zeigt die Ansicht für den QUEUED-Modus in CodePipeline.



Die folgende Abbildung zeigt die Ansicht für den PARALLEL-Modus in CodePipeline.



Überlegungen zum Umschalten zwischen Ausführungsmodi

Im Folgenden finden Sie Überlegungen zu Pipelines, wenn Sie den Modus für die Pipeline ändern. Wenn Sie im Bearbeitungsmodus von einem Ausführungsmodus in einen anderen wechseln und die Änderung anschließend speichern, können sich bestimmte Ansichten oder Status anpassen.

Wenn Sie beispielsweise vom PARALLEL-Modus in den QUEUED- oder SUPERSEDED-Modus wechseln, wird die im PARALLEL-Modus gestartete Ausführung weiterhin ausgeführt. Diese können auf der Seite mit dem Ausführungsverlauf eingesehen werden. In der Pipeline-Ansicht wird die Ausführung angezeigt, die zuvor im QUEUED- oder SUPERSEDED-Modus oder andernfalls in einem leeren Zustand ausgeführt wurde.

Ein weiteres Beispiel: Wenn Sie vom QUEUED- oder SUPERSEDED-Modus in den PARALLEL-Modus wechseln, wird die Pipeline-Ansicht/Statusseite nicht mehr angezeigt. Um eine Ausführung im PARALLEL-Modus anzuzeigen, verwenden Sie die Registerkarte „Visualisierung“ auf der Seite

mit den Ausführungsdetails. Ausführungen, die im Modus SUPERSEDED oder QUEUED gestartet wurden, werden abgebrochen.

Die folgende Tabelle enthält weitere Einzelheiten.

Moduswechsel	Einzelheiten zur ausstehenden und aktiven Ausführung	Einzelheiten zum Pipeline-Status
ERSETZT DURCH ERSETZT//ERSETZT DURCH WARTESCHLANGE	• Aktive Ausführungen werden abgebrochen, nachdem die laufenden Aktionen abgeschlossen sind. • Ausstehende Ausführungen werden storniert.	Der Pipeline-Status, z. B. „Storniert“, wird zwischen der Version des ersten Modus und dem zweiten Modus beibehalten.
QUEUED bis QUEUED//QUEUED bis SUPERSEDED	• Aktive Ausführungen werden abgebrochen, nachdem die laufenden Aktionen abgeschlossen sind. • Ausstehende Ausführungen werden storniert.	Der Pipeline-Status, z. B. „Storniert“, wird zwischen der Version des ersten Modus und dem zweiten Modus beibehalten.
PARALLEL zu PARALLEL	Alle Ausführungen dürfen unabhängig von Aktualisierungen der Pipeline-Definitionen ausgeführt werden.	Leer. Der Parallelmodus hat keinen Pipeline-Status.
ERSETZT DURCH PARALLEL//QUEUED durch PARALLEL	• Aktive Ausführungen werden abgebrochen, nachdem die laufenden Aktionen abgeschlossen sind. • Ausstehende Ausführungen werden storniert.	Leer. Der Parallelmodus hat keinen Pipeline-Status.

Stellen Sie den Pipeline-Ausführungsmodus ein oder ändern Sie ihn (Konsole)

Sie können die Konsole verwenden, um den Pipeline-Ausführungsmodus festzulegen.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen und der Status aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie auf der Seite Bearbeiten die Option Bearbeiten: Pipeline-Eigenschaften aus.
5. Wählen Sie den Modus für Ihre Pipeline.

   - Ersetzt
   - In der Warteschlange (Pipeline-Typ V2 erforderlich)
   - Parallel (Pipeline-Typ V2 erforderlich)
6. Wählen Sie auf der Seite Bearbeiten die Option Fertig aus.

Legen Sie den Pipeline-Ausführungsmodus (CLI) fest

Verwenden Sie den `update-pipeline` Befehl `create-pipeline` oder, AWS CLI um den Pipeline-Ausführungsmodus festzulegen.

1. Öffnen Sie eine Terminalsitzung (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und führen Sie den get-pipeline Befehl aus, um die Pipeline-Struktur in eine JSON-Datei zu kopieren. Geben Sie für eine Pipeline mit dem Namen **`MyFirstPipeline`** den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.
2. Öffnen Sie die JSON-Datei in einem beliebigen Klartexteditor und ändern Sie die Struktur der Datei so, dass sie den Pipeline-Ausführungsmodus widerspiegelt, den Sie festlegen möchten, z. B. QUEUED.

```
"executionMode": "QUEUED"
```

Das folgende Beispiel zeigt, wie Sie den Ausführungsmodus in einer Beispielpipeline mit zwei Stufen auf QUEUED festlegen würden.

```
{
    "pipeline": {
        "name": "MyPipeline",
        "roleArn": "arn:aws:iam::111122223333:role/service-role/
AWSCodePipelineServiceRole-us-east-1-dkpippe",
        "artifactStore": {
            "type": "S3",
            "location": "bucket"
        },
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "provider": "CodeCommit",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "BranchName": "main",
                            "OutputArtifactFormat": "CODE_ZIP",
                            "PollForSourceChanges": "true",
                            "RepositoryName": "MyDemoRepo"
                        },
                        "outputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "inputArtifacts": [],
                        "region": "us-east-1",
                        "namespace": "SourceVariables"
                    }
```

```
                ]
            },
            {
                "name": "Build",
                "actions": [
                    {
                        "name": "Build",
                        "actionTypeId": {
                            "category": "Build",
                            "owner": "AWS",
                            "provider": "CodeBuild",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "ProjectName": "MyBuildProject"
                        },
                        "outputArtifacts": [
                            {
                                "name": "BuildArtifact"
                            }
                        ],
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "region": "us-east-1",
                        "namespace": "BuildVariables"
                    }
                ]
            }
        ],
        "version": 1,
        "executionMode": "QUEUED"
    }
}
```

3. Wenn Sie mit einer Pipeline-Struktur arbeiten, die Sie mit dem Befehl get-pipeline abgerufen haben, müssen Sie die Struktur in der JSON-Datei ändern. Sie müssen die `metadata`-Zeilen aus der Datei entfernen, damit der Befehl update-pipeline sie verwenden kann. Entfernen Sie den Abschnitt aus der Pipeline-Struktur in der JSON-Datei (die `"metadata": { }`-Zeilen und die Fehler `"created"`, `"pipelineARN"` und `"updated"`).

Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

```
"metadata": {
  "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
  "created": "date",
  "updated": "date"
  }
```

Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

> **Important**
>
> Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

```
aws codepipeline update-pipeline --cli-input-json file://pipeline.json
```

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen.

Wiederholen Sie eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase

Sie können eine Phase wiederholen, bei der ein Fehler aufgetreten ist, ohne eine Pipeline erneut von Anfang an ausführen zu müssen. Sie tun dies, indem Sie entweder die fehlgeschlagenen Aktionen in einer Phase erneut versuchen oder indem Sie alle Aktionen in der Phase wiederholen, beginnend mit

der ersten Aktion in der Phase. Wenn Sie die fehlgeschlagenen Aktionen in einer Phase wiederholen, funktionieren alle Aktionen, die noch ausgeführt werden, weiterhin, und fehlgeschlagene Aktionen werden erneut ausgelöst. Wenn Sie eine fehlgeschlagene Phase von der ersten Aktion in der Phase wiederholen, können in der Phase keine Aktionen ausgeführt werden. Bevor eine Phase erneut versucht werden kann, müssen entweder alle Aktionen fehlgeschlagen sein oder einige Aktionen sind fehlgeschlagen und andere erfolgreich.

Important

Beim erneuten Versuch einer fehlgeschlagenen Phase werden alle Aktionen der Phase von der ersten Aktion in der Phase wiederholt, und bei einem erneuten Versuch, fehlgeschlagene Aktionen durchzuführen, werden alle fehlgeschlagenen Aktionen in der Phase wiederholt. Dadurch werden Ausgabeartefakte zuvor erfolgreicher Aktionen in derselben Ausführung außer Kraft gesetzt.
Artefakte können zwar überschrieben werden, der Ausführungsverlauf zuvor erfolgreicher Aktionen wird jedoch beibehalten.

Wenn Sie die Konsole verwenden, um eine Pipeline anzuzeigen, erscheint entweder die Schaltfläche Phase wiederholen oder Fehlgeschlagene Aktionen wiederholen auf der Stufe, die erneut versucht werden kann.

Wenn Sie die AWS CLI verwenden, können Sie den get-pipeline-state Befehl verwenden, um festzustellen, ob Aktionen fehlgeschlagen sind.

Note

In den folgenden Fällen können Sie eine Phase möglicherweise nicht erneut versuchen:

- Alle Aktionen in der Phase waren erfolgreich, sodass die Phase nicht den Status Fehlgeschlagen hat.
- Die gesamte Pipeline-Struktur hat sich geändert, nachdem die Phase gescheitert war.
- Es wird bereits ein anderer Wiederholungsversuch in der Phase ausgeführt.

Themen

- Eine fehlgeschlagene Phase erneut versuchen (CLI)

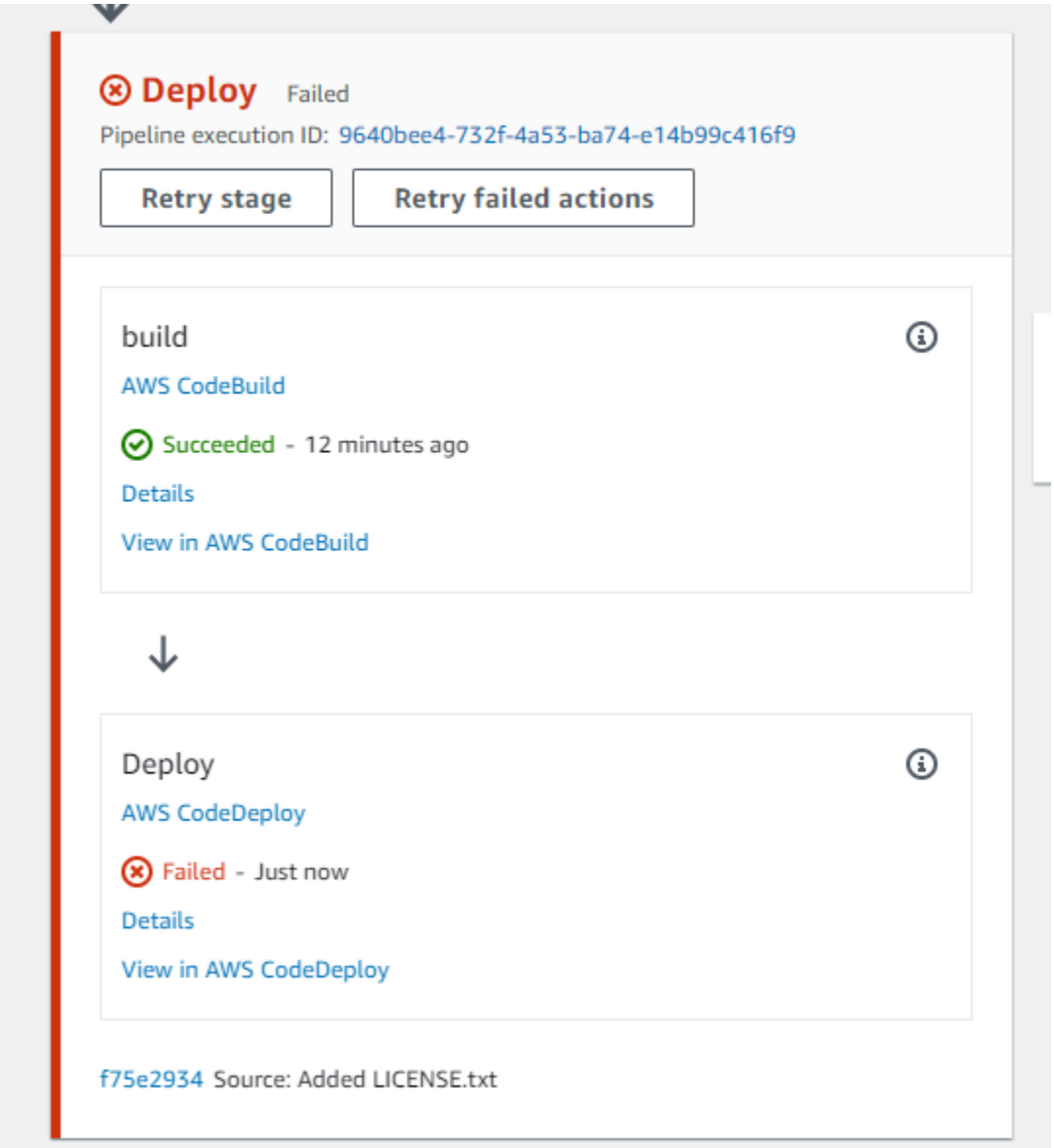
Versuchen Sie es erneut mit einer fehlgeschlagenen Phase (Konsole)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase erneut zu versuchen: Konsole

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline.
3. Suchen Sie die Phase mit der fehlgeschlagenen Aktion und wählen Sie dann eine der folgenden Optionen aus:

   - Um alle Aktionen in der Phase erneut zu versuchen, wählen Sie Phase wiederholen aus.
   - Um nur fehlgeschlagene Aktionen in der Phase erneut zu versuchen, wählen Sie Fehlgeschlagene Aktionen wiederholen.



Wenn alle wiederholten Aktionen in der Stufe erfolgreich abgeschlossen sind, wird die Pipeline weiter ausgeführt.

Eine fehlgeschlagene Phase erneut versuchen (CLI)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase erneut zu versuchen - CLI

Um alle Aktionen oder alle fehlgeschlagenen Aktionen erneut AWS CLI zu versuchen, führen Sie den retry-stage-execution Befehl mit den folgenden Parametern aus:

```
--pipeline-name <value>
--stage-name <value>
--pipeline-execution-id <value>
--retry-mode ALL_ACTIONS/FAILED_ACTIONS
```

> **Note**
>
> Die Werte, für die Sie verwenden können, `retry-mode` sind `FAILED_ACTIONS` und`ALL_ACTIONS`.

1. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer retry-stage-executionBefehlszeile (Windows) aus, wie im folgenden Beispiel für eine Pipeline mit dem Namen gezeigt`MyPipeline`.

   ```
   aws codepipeline retry-stage-execution --pipeline-name MyPipeline --stage-name
    Deploy --pipeline-execution-id b59babff-5f34-EXAMPLE --retry-mode FAILED_ACTIONS
   ```

   Die Ausgabe gibt die Ausführungs-ID zurück:

   ```
   {
       "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
   }
   ```

2. Sie können den Befehl auch mit einer JSON-Eingabedatei ausführen. Sie erstellen zunächst eine JSON-Datei, die die Pipeline, die Stufe mit den fehlgeschlagenen Aktionen und die letzte Pipeline-Ausführung in dieser Stufe ermittelt. Führen Sie den Befehl retry-stage-execution mit dem Parameter `--cli-input-json` aus. Am einfachsten können Sie die für die JSON-Datei benötigten Details über den Befehl get-pipeline-state abrufen.

   a. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer get-pipeline-stateBefehlszeile (Windows) in einer Pipeline aus. Für eine Pipeline mit dem Namen würden Sie MyFirstPipeline beispielsweise etwas Ähnliches wie das Folgende eingeben:

      ```
      aws codepipeline get-pipeline-state --name MyFirstPipeline
      ```

Die Ausgabe des Befehls umfasst die Pipeline-Statusinformationen für jede einzelne Stufe. Im folgenden Beispiel zeigt die Ausgabe, dass eine oder mehrere Aktionen in der Staging-Stufe fehlgeschlagen sind:

```
{
    "updated": 1427245911.525,
    "created": 1427245911.525,
    "pipelineVersion": 1,
    "pipelineName": "MyFirstPipeline",
    "stageStates": [
        {
            "actionStates": [...],
            "stageName": "Source",
            "latestExecution": {
                "pipelineExecutionId": "9811f7cb-7cf7-SUCCESS",
                "status": "Succeeded"
            }
        },
        {
            "actionStates": [...],
            "stageName": "Staging",
            "latestExecution": {
                "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
                "status": "Failed"
            }
        }
    ]
}
```

b. Erstellen Sie in einem Texteditor eine Datei im JSON-Format, in der Sie Folgendes festhalten:

- Der Name der Pipeline mit den fehlgeschlagenen Aktionen
- Der Name der Stufe mit den fehlgeschlagenen Aktionen
- Die ID der letzten Pipeline-Ausführung in der Stufe
- Der Retry-Modus.

Für das vorherige MyFirstPipeline Beispiel würde Ihre Datei etwa so aussehen:

```
{
```

```
    "pipelineName": "MyFirstPipeline",
    "stageName": "Staging",
    "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
    "retryMode": "FAILED_ACTIONS"
}
```

c. Speichern Sie die Datei mit einem Namen wie **`retry-failed-actions.json`**.

d. Rufen Sie die beim Ausführen des Befehls retry-stage-execution erstellte Datei auf. Beispielsweise:

> **Important**
>
> Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

```
aws codepipeline retry-stage-execution --cli-input-json file://retry-failed-
actions.json
```

e. Um die Ergebnisse des Wiederholungsversuchs anzuzeigen, öffnen Sie entweder die CodePipeline Konsole und wählen Sie die Pipeline aus, die die fehlgeschlagenen Aktionen enthält, oder verwenden Sie den get-pipeline-state Befehl erneut. Weitere Informationen finden Sie unter Pipelines und Details anzeigen in CodePipeline.

Stage Rollback konfigurieren

Sie können eine Phase auf eine Ausführung zurücksetzen, die in dieser Phase erfolgreich war. Sie können eine Phase für das Rollback bei einem Fehler vorkonfigurieren, oder Sie können eine Phase manuell rückgängig machen. Der Rollback-Vorgang führt zu einer neuen Ausführung. Die für das Rollback gewählte Ziel-Pipeline-Ausführung wird zum Abrufen von Quellrevisionen und Variablen verwendet.

Die Art der Ausführung, entweder Standard oder Rollback, wird im Pipeline-Verlauf, im Pipeline-Status und in den Details zur Pipeline-Ausführung angezeigt.

Themen

- Überlegungen zu Rollbacks
- Manuelles Rollback einer Phase

- Konfigurieren Sie eine Phase für das automatische Rollback
- Den Rollback-Status in der Ausführungsliste anzeigen
- Details zum Rollback-Status anzeigen

Überlegungen zu Rollbacks

Beim Stufen-Rollback sollten folgende Überlegungen angestellt werden:

- Sie können eine Quellstufe nicht rückgängig machen.
- Die Pipeline kann nur dann zu einer vorherigen Ausführung zurückkehren, wenn die vorherige Ausführung in der aktuellen Pipeline-Strukturversion gestartet wurde.
- Sie können nicht zu einer Ziel-Ausführungs-ID zurückkehren, bei der es sich um einen Rollback-Ausführungstyp handelt.

Manuelles Rollback einer Phase

Sie können eine Phase mithilfe der Konsole oder CLI manuell rückgängig machen. Die Pipeline kann nur dann zu einer vorherigen Ausführung zurückkehren, wenn die vorherige Ausführung in der aktuellen Version der Pipeline-Struktur gestartet wurde.

Sie können eine Phase auch so konfigurieren, dass bei einem Fehler ein automatisches Rollback ausgeführt wird, wie unter beschriebenKonfigurieren Sie eine Phase für das automatische Rollback.

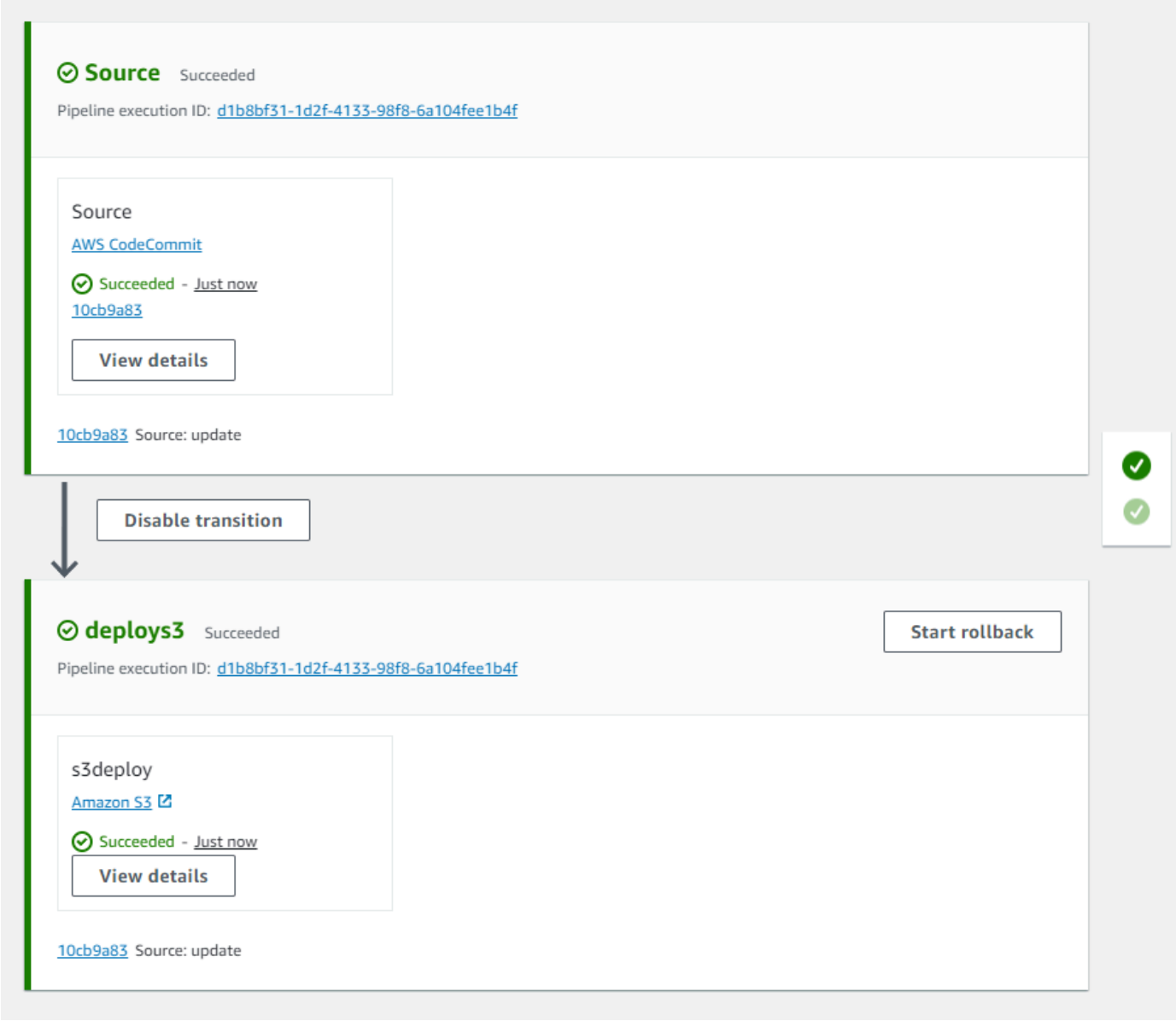
Manuelles Rollback einer Phase (Konsole)

Sie können die Konsole verwenden, um eine Phase manuell auf eine Ziel-Pipeline-Ausführung zurückzusetzen. Wenn eine Phase zurückgesetzt wird, wird in der Pipeline-Visualisierung in der Konsole ein Rollback-Label angezeigt.

Manuelles Rollback einer Phase (Konsole)

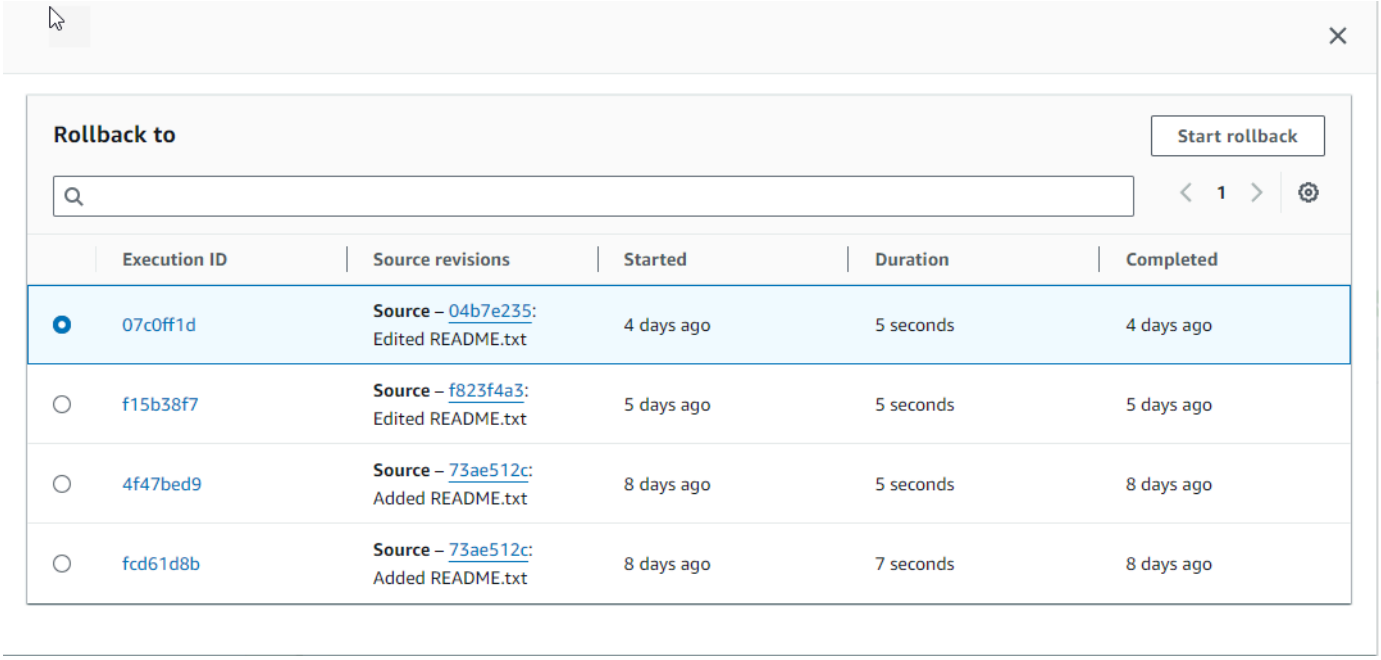
1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen und der Status aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie unter Name den Namen der Pipeline mit der Phase aus, für die ein Rollback ausgeführt werden soll.

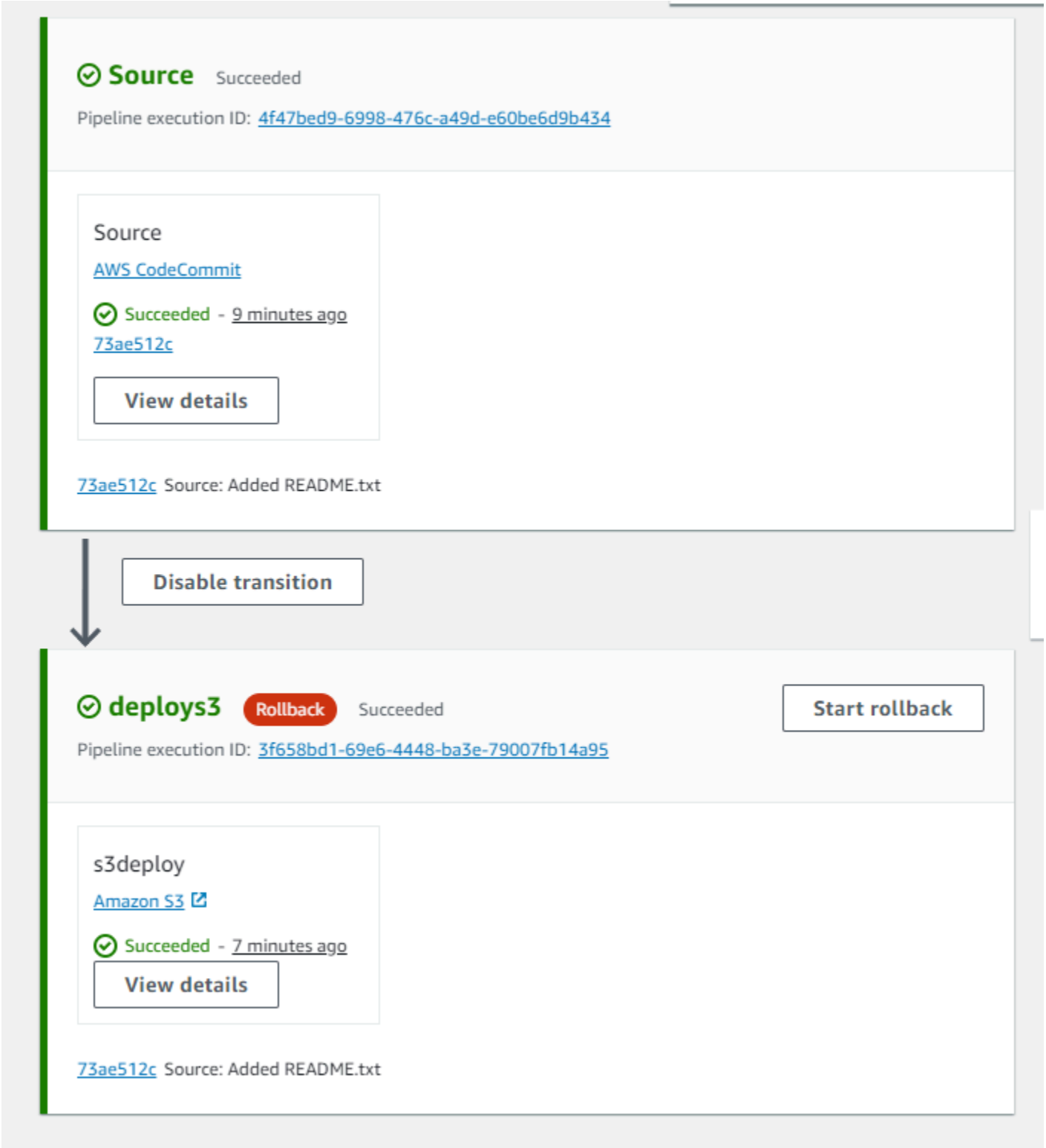


3. Wählen Sie auf der Bühne Rollback starten aus. Das Dialogfeld „Zurück zur Seite“ wird angezeigt.
4. Wählen Sie die Zielausführung aus, auf die Sie die Phase zurücksetzen möchten.

> **Note**
>
> Die Liste der verfügbaren Ziel-Pipeline-Ausführungen umfasst alle Ausführungen in der aktuellen Pipeline-Version, die am 1. Februar 2024 beginnt.



Das folgende Diagramm zeigt ein Beispiel für die Rollback-Stufe mit der neuen Ausführungs-ID.



Manuelles Rollback einer Phase (CLI)

Verwenden Sie AWS CLI den `rollback-stage` Befehl, um eine Phase manuell zurückzusetzen.

Sie können eine Phase auch manuell rückgängig machen, wie unter beschriebenManuelles Rollback einer Phase.

> **Note**
>
> Die Liste der verfügbaren Ziel-Pipeline-Ausführungen umfasst alle Ausführungen in der aktuellen Pipeline-Version, die am 1. Februar 2024 beginnt.

Manuelles Rollback einer Phase (CLI)

1. Der CLI-Befehl für das manuelle Rollback erfordert die Ausführungs-ID einer zuvor erfolgreichen Pipeline-Ausführung in der Phase. Um die von Ihnen angegebene Ziel-Pipeline-Ausführungs-ID abzurufen, verwenden Sie den list-pipeline-executions Befehl mit einem Filter, der erfolgreiche Ausführungen in der Phase zurückgibt. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie die, AWS CLI um den `list-pipeline-executions` Befehl auszuführen. Geben Sie dabei den Namen der Pipeline und den Filter für erfolgreiche Ausführungen in der Phase an. In diesem Beispiel werden in der Ausgabe Pipeline-Ausführungen für die angegebene Pipeline MyFirstPipeline und für erfolgreiche Ausführungen in der genannten Phase aufgeführt. `deploys3`

   ```
   aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline --filter
    succeededInStage={stageName=deploys3}
   ```

   Kopieren Sie in der Ausgabe die Ausführungs-ID der zuvor erfolgreichen Ausführung, die Sie für das Rollback angeben möchten. Sie werden diese im nächsten Schritt als Zielausführungs-ID verwenden.

2. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie die, AWS CLI um den `rollback-stage` Befehl auszuführen. Geben Sie dabei den Namen der Pipeline, den Namen der Phase und die Zielausführung an, zu der Sie zurückkehren möchten. Um beispielsweise eine Phase namens Deploy für eine Pipeline mit dem Namen „Bereitstellen“ rückgängig zu machen *`MyFirstPipeline`*:

   ```
   aws codepipeline rollback-stage --pipeline-name MyFirstPipeline --stage-name Deploy
    --target-pipeline-execution-id bc022580-4193-491b-8923-9728dEXAMPLE
   ```

Die Ausgabe gibt die Ausführungs-ID für die neue Ausführung zurück, für die ein Rollback ausgeführt wurde. Dies ist eine separate ID, die die Quellversionen und Parameter der angegebenen Zielausführung verwendet.

Konfigurieren Sie eine Phase für das automatische Rollback

Sie können Phasen in einer Pipeline so konfigurieren, dass sie bei einem Ausfall automatisch zurückgesetzt werden. Wenn die Phase fehlschlägt, wird die Phase auf die letzte erfolgreiche Ausführung zurückgesetzt. Die Pipeline kann nur dann zu einer vorherigen Ausführung zurückkehren, wenn die vorherige Ausführung in der aktuellen Version der Pipeline-Struktur gestartet wurde. Da die automatische Rollback-Konfiguration Teil der Pipeline-Definition ist, wird Ihre Pipeline-Phase erst nach einer erfolgreichen Pipeline-Ausführung in der Pipeline-Phase automatisch zurückgesetzt.

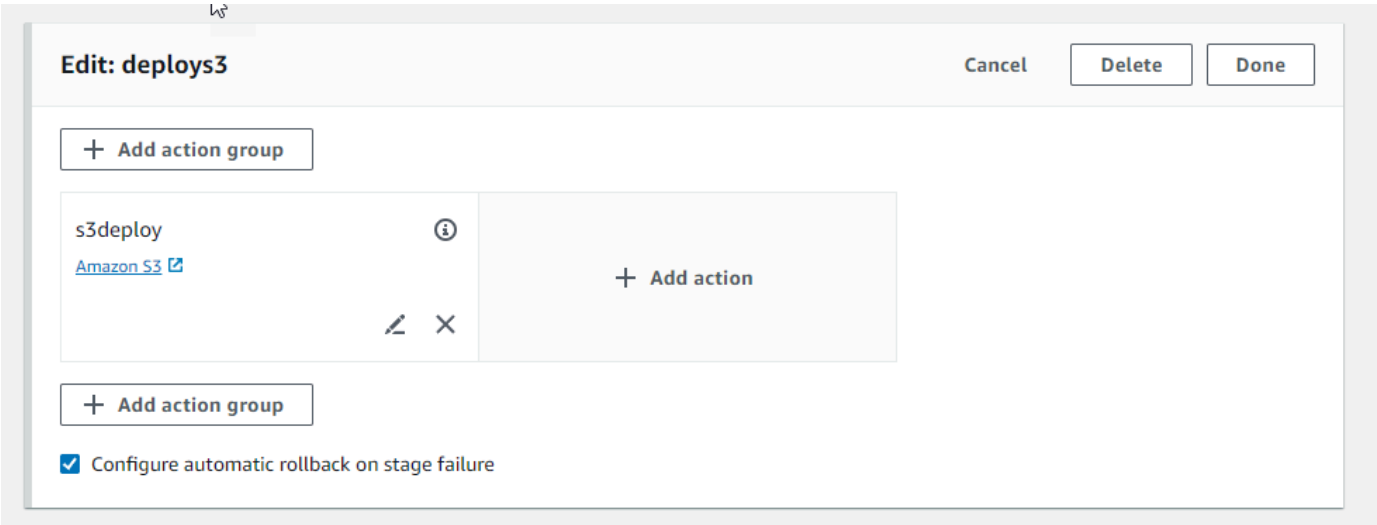
Konfigurieren Sie eine Phase für das automatische Rollback (Konsole)

Sie können eine Phase auf eine bestimmte vorherige erfolgreiche Ausführung zurücksetzen. Weitere Informationen finden Sie RollbackStageim CodePipeline API-Leitfaden.

Konfigurieren Sie eine Phase für das automatische Rollback (Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen und der Status aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie auf der Seite Bearbeiten für die Aktion, die Sie bearbeiten möchten, die Option Phase bearbeiten aus.
5. Wählen Sie Automatisches Rollback bei Stufenausfall konfigurieren. Speichern Sie die Änderungen an Ihrer Pipeline.



Konfigurieren Sie eine Phase für automatisches Rollback (CLI)

AWS CLI Um die Phase „Fehlgeschlagen“ so zu konfigurieren, dass sie automatisch zur letzten erfolgreichen Ausführung zurückkehrt, verwenden Sie die Befehle zum Erstellen oder Aktualisieren einer Pipeline, wie unter Erstellen Sie eine Pipeline in CodePipeline und Eine Pipeline bearbeiten in CodePipeline beschrieben.

- Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie die, AWS CLI um den `update-pipeline` Befehl auszuführen, wobei Sie die Fehlerbedingung in der Pipeline-Struktur angeben. Im folgenden Beispiel wird das automatische Rollback für ein Staging-Objekt mit dem Namen konfiguriert: `S3Deploy`

```
{
                "name": "S3Deploy",
                "actions": [
                    {
                        "name": "s3deployaction",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "provider": "S3",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "BucketName": "static-website-bucket",
                            "Extract": "false",
```

```
                        "ObjectKey": "SampleApp.zip"
                    },
                    "outputArtifacts": [],
                    "inputArtifacts": [
                        {
                            "name": "SourceArtifact"
                        }
                    ],
                    "region": "us-east-1"
                }
            ],
            "onFailure": {
                "result": "ROLLBACK"
            }
        }
```

Weitere Informationen zur Konfiguration von Fehlerbedingungen für das Stage-Rollback finden Sie FailureConditionsin der API-Referenz. CodePipeline

Konfigurieren Sie eine Phase für das automatische Rollback ()AWS CloudFormation

Verwenden Sie AWS CloudFormation den `OnFailure` Parameter, um eine Phase so zu konfigurieren, dass sie bei einem Fehler automatisch zurückgesetzt wird. Bei einem Fehler führt die Phase automatisch ein Rollback zur letzten erfolgreichen Ausführung durch.

```
OnFailure:
     Result: ROLLBACK
```

- Aktualisieren Sie die Vorlage wie im folgenden Codeausschnitt gezeigt. Im folgenden Beispiel wird das automatische Rollback für ein Staging-Objekt mit dem Namen konfiguriert: `Release`

```
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    RoleArn:
      Ref: CodePipelineServiceRole
    Stages:
      -
        Name: Source
        Actions:
          -
```

```
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              -
                Name: SourceOutput
            Configuration:
              S3Bucket:
                Ref: SourceS3Bucket
              S3ObjectKey:
                Ref: SourceS3ObjectKey
            RunOrder: 1
      -
        Name: Release
        Actions:
          -
            Name: ReleaseAction
            InputArtifacts:
              -
                Name: SourceOutput
            ActionTypeId:
              Category: Deploy
              Owner: AWS
              Version: 1
              Provider: CodeDeploy
            Configuration:
              ApplicationName:
                Ref: ApplicationName
              DeploymentGroupName:
                Ref: DeploymentGroupName
            RunOrder: 1
       OnFailure:
            Result: ROLLBACK
    ArtifactStore:
      Type: S3
      Location:
        Ref: ArtifactStoreS3Location
      EncryptionKey:
        Id: arn:aws:kms:useast-1:ACCOUNT-ID:key/KEY-ID
        Type: KMS
    DisableInboundStageTransitions:
```

```
      -
        StageName: Release
        Reason: "Disabling the transition until integration tests are completed"
    Tags:
      - Key: Project
        Value: ProjectA
      - Key: IsContainerBased
        Value: 'true'
```

Weitere Informationen zur Konfiguration der Fehlerbedingungen für das Stage-Rollback finden Sie weiter unten *`StageDeclaration`* im OnFailureBenutzerhandbuch.AWS CloudFormation

Den Rollback-Status in der Ausführungsliste anzeigen

Sie können den Status und die Zielausführungs-ID für eine Rollback-Ausführung anzeigen.

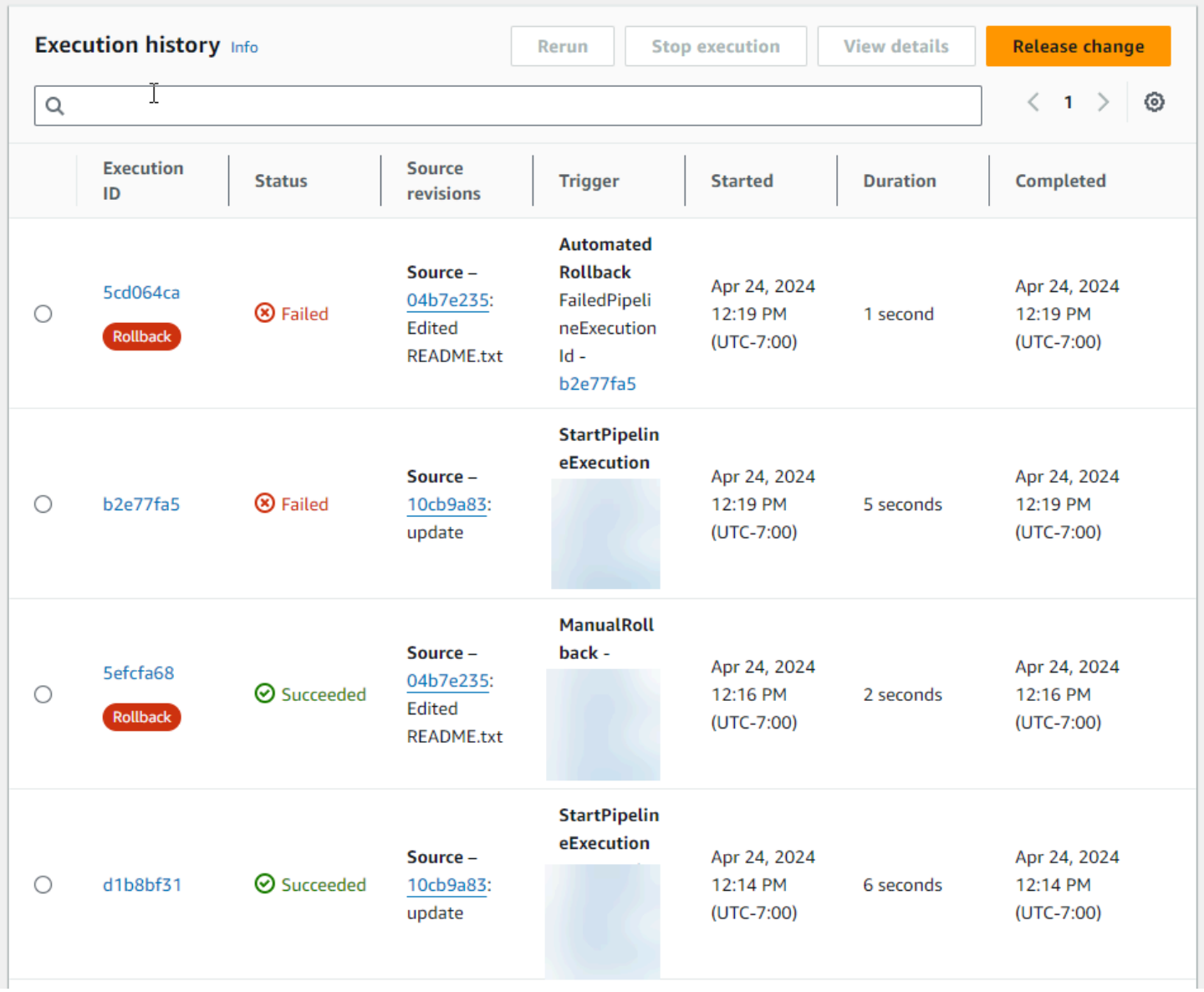
Den Rollback-Status in der Liste der Ausführungen anzeigen (Konsole)

Sie können die Konsole verwenden, um den Status und die Zielausführungs-ID für eine Rollback-Ausführung in der Ausführungsliste anzuzeigen.

Den Status der Rollback-Ausführung in der Liste der Ausführungen anzeigen (Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen und der Status aller Pipelines, die mit Ihrer verknüpft AWS-Konto sind, werden angezeigt.
2. Wählen Sie unter Name den Namen der Pipeline aus, die Sie anzeigen möchten.
3. Wählen Sie History (Verlauf) aus. In der Liste der Ausführungen wird die Bezeichnung Rollback angezeigt.



Wählen Sie die Ausführungs-ID, für die Sie Details anzeigen möchten.

Rollback-Status anzeigen mit **`list-pipeline-executions`** (CLI)

Sie können die CLI verwenden, um den Status und die Zielausführungs-ID für eine Rollback-Ausführung anzuzeigen.

- Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie den AWS CLI , um den list-pipeline-executions Befehl auszuführen:

```
aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
```

Dieser Befehl gibt eine Liste aller abgeschlossenen Ausführungen zurück, die mit der Pipeline verknüpft sind.

Das folgende Beispiel zeigt die zurückgegebenen Daten für eine Pipeline mit dem Namen, *MyFirstPipeline*wo die Pipeline durch eine Rollback-Ausführung gestartet wurde.

```
{
    "pipelineExecutionSummaries": [
        {
            "pipelineExecutionId": "eb7ebd36-353a-4551-90dc-18ca5EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T09:00:28.185000+00:00",
            "lastUpdateTime": "2024-04-16T09:00:29.665000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
                    "revisionUrl": "console-URL"
                }
            ],
            "trigger": {
                "triggerType": "ManualRollback",
                "triggerDetail": "{arn:aws:sts::<account_ID>:assumed-role/<role>"}"
            },
            "executionMode": "SUPERSEDED",
            "executionType": "ROLLBACK",
            "rollbackMetadata": {
                "rollbackTargetPipelineExecutionId":
 "f15b38f7-20bf-4c9e-94ed-2535eEXAMPLE"
            }
        },
        {
            "pipelineExecutionId": "fcd61d8b-4532-4384-9da1-2aca1EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T08:58:56.601000+00:00",
            "lastUpdateTime": "2024-04-16T08:59:04.274000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
```

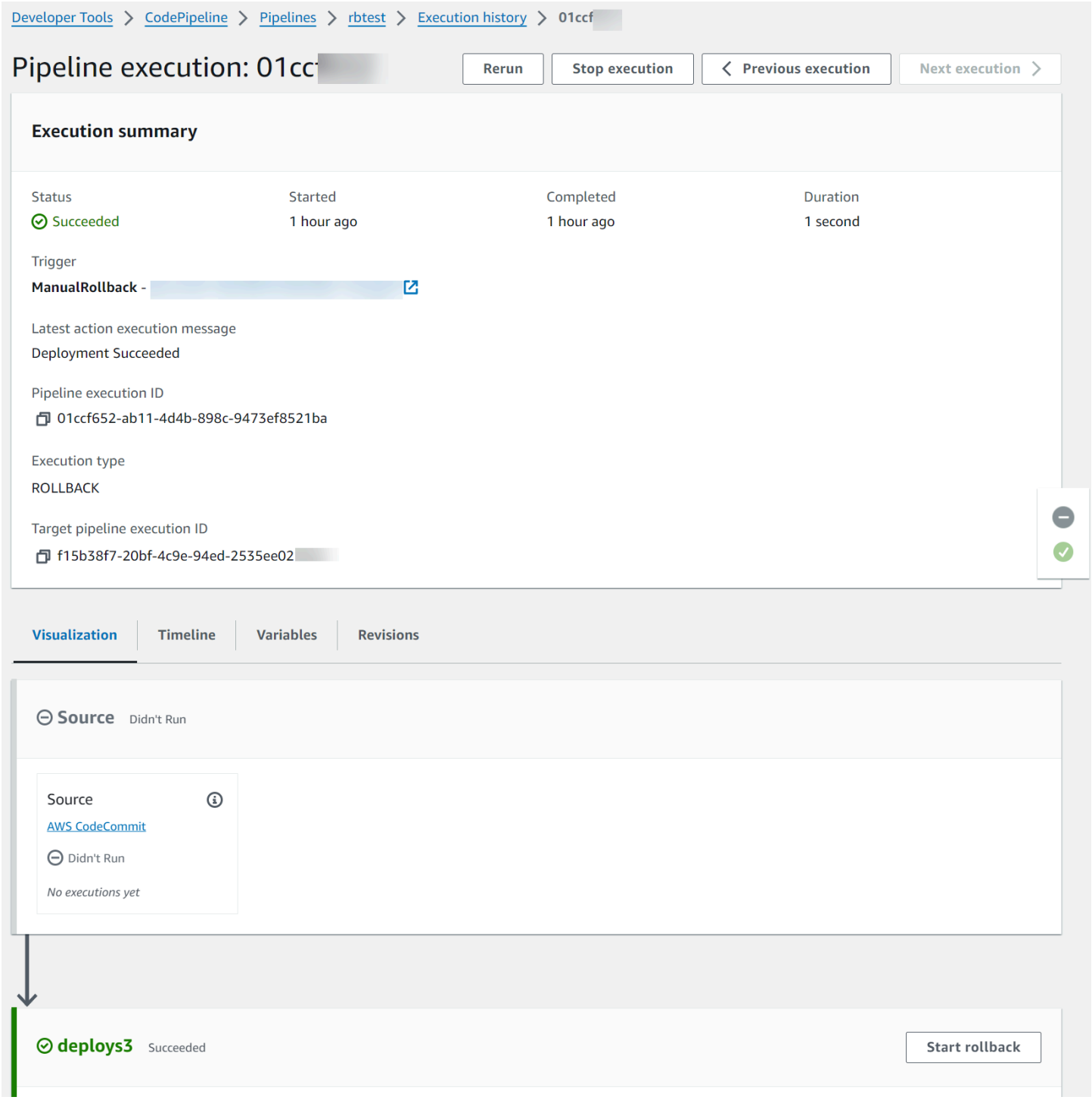
```
                    "revisionUrl": "console_URL"
                }
            ],
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
            },
            "executionMode": "SUPERSEDED"
        },
        {
            "pipelineExecutionId": "5cd064ca-bff7-425f-8653-f41d9EXAMPLE",
            "status": "Failed",
            "startTime": "2024-04-24T19:19:50.781000+00:00",
            "lastUpdateTime": "2024-04-24T19:19:52.119000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "<revision_ID>",
                    "revisionSummary": "Edited README.txt",
                    "revisionUrl": "<revision_URL>"
                }
            ],
            "trigger": {
                "triggerType": "AutomatedRollback",
                "triggerDetail": "{\"FailedPipelineExecutionId\":
\"b2e77fa5-9285-4dea-ae66-4389EXAMPLE\"}"
            },
            "executionMode": "SUPERSEDED",
            "executionType": "ROLLBACK",
            "rollbackMetadata": {
                "rollbackTargetPipelineExecutionId": "5efcfa68-d838-4ca7-
a63b-4a743EXAMPLE"
            }
         },
```

Details zum Rollback-Status anzeigen

Sie können den Status und die Zielausführungs-ID für eine Rollback-Ausführung anzeigen.

Den Rollback-Status auf der Detailseite (Konsole) anzeigen

Sie können die Konsole verwenden, um den Status und die Ziel-Pipeline-Ausführungs-ID für eine Rollback-Ausführung anzuzeigen.



Rollback-Details anzeigen mit **`get-pipeline-execution`** (CLI)

Pipeline-Ausführungen, für die ein Rollback vorgenommen wurde, werden in der Ausgabe zum Abrufen der Pipeline-Ausführung angezeigt.

- Um Informationen zu einer Pipeline anzuzeigen, führen Sie den Befehl get-pipeline-execution unter Angabe des eindeutigen Namens der Pipeline aus. Um beispielsweise Details zu einer Pipeline mit dem Namen anzuzeigen *`MyFirstPipeline`*, geben Sie Folgendes ein:

  ```
  aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-execution-id 3f658bd1-69e6-4448-ba3e-79007EXAMPLE
  ```

  Dieser Befehl gibt die Struktur der Pipeline zurück.

  Das folgende Beispiel zeigt die zurückgegebenen Daten für einen Teil einer Pipeline mit dem Namen *`MyFirstPipeline`*, in dem die Rollback-Ausführungs-ID und die Metadaten angezeigt werden.

  ```
  {
      "pipelineExecution": {
          "pipelineName": "MyFirstPipeline",
          "pipelineVersion": 6,
          "pipelineExecutionId": "2004a94e-8b46-4c34-a695-c8d20EXAMPLE",
          "status": "Succeeded",
          "artifactRevisions": [
              {
                  "name": "SourceArtifact",
                  "revisionId": "<ID>",
                  "revisionSummary": "Added README.txt",
                  "revisionUrl": "<console_URL>"
              }
          ],
          "trigger": {
              "triggerType": "ManualRollback",
              "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
          },
          "executionMode": "SUPERSEDED",
          "executionType": "ROLLBACK",
          "rollbackMetadata": {
              "rollbackTargetPipelineExecutionId": "4f47bed9-6998-476c-a49d-e60beEXAMPLE"
          }
  ```

```
    }
}
```

Rollback-Status anzeigen mit **`get-pipeline-state`** (CLI)

Pipeline-Ausführungen, für die ein Rollback vorgenommen wurde, werden in der Ausgabe zum Abrufen des Pipeline-Status angezeigt.

- Um Informationen zu einer Pipeline anzuzeigen, führen Sie den Befehl get-pipeline-state unter Angabe des eindeutigen Namens der Pipeline aus. Um beispielsweise Statusdetails zu einer Pipeline mit dem Namen anzuzeigen *`MyFirstPipeline`*, geben Sie Folgendes ein:

  ```
  aws codepipeline get-pipeline-state --name MyFirstPipeline
  ```

  Das folgende Beispiel zeigt die zurückgegebenen Daten mit dem Rollback-Ausführungstyp.

  ```
  {
      "pipelineName": "MyFirstPipeline",
      "pipelineVersion": 7,
      "stageStates": [
          {
              "stageName": "Source",
              "inboundExecutions": [],
              "inboundTransitionState": {
                  "enabled": true
              },
              "actionStates": [
                  {
                      "actionName": "Source",
                      "currentRevision": {
                          "revisionId": "<Revision_ID>"
                      },
                      "latestExecution": {
                          "actionExecutionId": "13bbd05d-
  b439-4e35-9c7e-887cb789b126",
                          "status": "Succeeded",
                          "summary": "update",
                          "lastStatusChange": "2024-04-24T20:13:45.799000+00:00",
                          "externalExecutionId": "10cbEXAMPLEID"
                      },
                      "entityUrl": "console-url",
  ```

```
                    "revisionUrl": "console-url"
                }
            ],
            "latestExecution": {
                "pipelineExecutionId": "cf95a8ca-0819-4279-ae31-03978EXAMPLE",
                "status": "Succeeded"
            }
        },
        {
            "stageName": "deploys3",
            "inboundExecutions": [],
            "inboundTransitionState": {
                "enabled": true
            },
            "actionStates": [
                {
                    "actionName": "s3deploy",
                    "latestExecution": {
                        "actionExecutionId":
 "3bc4e3eb-75eb-45b9-8574-8599aEXAMPLE",
                        "status": "Succeeded",
                        "summary": "Deployment Succeeded",
                        "lastStatusChange": "2024-04-24T20:14:07.577000+00:00",
                        "externalExecutionId": "mybucket/SampleApp.zip"
                    },
                    "entityUrl": "console-URL"
                }
            ],
            "latestExecution": {
                "pipelineExecutionId": "fdf6b2ae-1472-4b00-9a83-1624eEXAMPLE",
                "status": "Succeeded",
                "type": "ROLLBACK"
            }
        }
    ],
    "created": "2024-04-15T21:29:01.635000+00:00",
    "updated": "2024-04-24T20:12:24.480000+00:00"
}
```

Arbeiten mit Aktionen in CodePipeline

AWS CodePipeline In ist eine Aktion Teil der Sequenz in einer Phase einer Pipeline. Sie ist eine Aufgabe, die beim Artefakt in dieser Stufe ausgeführt wird. Pipeline-Aktionen treten in einer bestimmte Reihenfolge – nacheinander oder parallel – auf, die bei der Konfiguration der Phase festgelegt wird.

CodePipeline bietet Unterstützung für sechs Arten von Aktionen:

- Quelle
- Entwicklung
- Test
- Bereitstellen
- Genehmigung
- Aufrufen

Informationen zu den Produkten AWS-Service und Services von Partnern, die Sie je nach Aktionstyp in Ihre Pipeline integrieren können, finden Sie unterIntegrationen mit CodePipeline Aktionstypen.

Themen

Mit Aktionstypen arbeiten

Aktionstypen sind vorkonfigurierte Aktionen, die Sie als Anbieter für Kunden erstellen, indem Sie eines der unterstützten Integrationsmodelle in AWS CodePipeline verwenden.

Sie können Aktionstypen anfordern, anzeigen und aktualisieren. Wenn der Aktionstyp für Ihr Konto als Eigentümer erstellt wurde, können Sie ihn verwenden, AWS CLI um die Eigenschaften und die Struktur Ihres Aktionstyps anzuzeigen oder zu aktualisieren. Wenn Sie der Anbieter oder Eigentümer des Aktionstyps sind, können Ihre Kunden die Aktion auswählen und sie zu ihren Pipelines hinzufügen, sobald sie in CodePipeline verfügbar ist.

> **Note**
>
> Sie erstellen vor `owner` Ort Aktionen, die mit `custom` einem Job-Worker ausgeführt werden sollen. Sie erstellen sie nicht mit einem Integrationsmodell. Informationen zu benutzerdefinierten Aktionen finden Sie unterErstellen und fügen Sie eine benutzerdefinierte Aktion hinzu in CodePipeline.

Komponenten des Aktionstyps

Die folgenden Komponenten bilden einen Aktionstyp.

- Aktionstyp-ID — Die ID besteht aus der Kategorie, dem Besitzer, dem Anbieter und der Version. Das folgende Beispiel zeigt eine Aktionstyp-ID mit einem Besitzer von`ThirdParty`, einer Kategorie von`Test`, einem Anbieter namens `TestProvider` und einer Version von`1`.

  ```
  {
      "Category": "Test",
      "Owner": "ThirdParty",
      "Provider": "TestProvider",
      "Version": "1"
  },
  ```

- Executor-Konfiguration — Das Integrationsmodell oder die Action Engine, das bei der Erstellung der Aktion angegeben wurde. Wenn Sie den Executor für einen Aktionstyp angeben, wählen Sie einen von zwei Typen:
  - Lambda: Der Besitzer des Aktionstyps schreibt die Integration als Lambda-Funktion, die immer dann aufgerufen wird, CodePipeline wenn ein Job für die Aktion verfügbar ist.
  - JobWorker: Der Eigentümer des Aktionstyps schreibt die Integration als Job Worker, der in Kunden-Pipelines nach verfügbaren Jobs sucht. Der Job-Worker führt dann den Job aus und sendet das Job-Ergebnis mithilfe CodePipeline von APIs zurück an. CodePipeline

> **Note**
>
> Das Jobworker-Integrationsmodell ist nicht das bevorzugte Integrationsmodell.

- Eingabe- und Ausgabeartefakte: Grenzwerte für die Artefakte, die der Eigentümer des Aktionstyps für Kunden der Aktion festlegt.
- Berechtigungen: Die Berechtigungsstrategie, mit der Kunden bestimmt werden, die auf den Aktionstyp eines Drittanbieters zugreifen können. Die verfügbaren Berechtigungsstrategien hängen vom ausgewählten Integrationsmodell für den Aktionstyp ab.
- URLs: Deep-Links zu Ressourcen, mit denen der Kunde interagieren kann, z. B. zur Konfigurationsseite des Aktionstyp-Inhabers.

Themen

Fordern Sie einen Aktionstyp an

Wenn ein neuer CodePipeline Aktionstyp von einem Drittanbieter angefordert wird, wird der Aktionstyp für den Aktionstyp erstellt CodePipeline, und der Eigentümer kann den Aktionstyp verwalten und anzeigen.

Ein Aktionstyp kann entweder eine private oder eine öffentliche Aktion sein. Wenn Ihr Aktionstyp erstellt wird, ist er privat. Wenn Sie beantragen möchten, dass ein Aktionstyp in eine öffentliche Aktion geändert wird, wenden Sie sich an das CodePipeline Serviceteam.

Bevor Sie Ihre Aktionsdefinitionsdatei, die Ressourcen für den Ausführenden und die Aktionstypanforderung für das CodePipeline Team erstellen, müssen Sie ein Integrationsmodell auswählen.

Schritt 1: Wählen Sie Ihr Integrationsmodell

Wählen Sie Ihr Integrationsmodell und erstellen Sie dann die Konfiguration für dieses Modell. Nachdem Sie das Integrationsmodell ausgewählt haben, müssen Sie Ihre Integrationsressourcen konfigurieren.

- Für das Lambda-Integrationsmodell erstellen Sie eine Lambda-Funktion und fügen Berechtigungen hinzu. Fügen Sie Ihrer Integrator-Lambda-Funktion Berechtigungen hinzu, um dem CodePipeline Dienst Berechtigungen zum Aufrufen mithilfe des CodePipeline Dienstprinzipals zu gewähren:. `codepipeline.amazonaws.com` Die Berechtigungen können über die Befehlszeile AWS CloudFormation oder über die Befehlszeile hinzugefügt werden.
  - Beispiel für das Hinzufügen von Berechtigungen mit AWS CloudFormation:

```
  CodePipelineLambdaBasedActionPermission:
    Type: 'AWS::Lambda::Permission'
    Properties:
      Action: 'lambda:invokeFunction'
      FunctionName: {"Fn::Sub": "arn:${AWS::Partition}:lambda:${AWS::Region}:
${AWS::AccountId}:function:function-name"}
      Principal: codepipeline.amazonaws.com
```

  - Dokumentation für die Befehlszeile
- Für das Job Worker-Integrationsmodell erstellen Sie eine Integration mit einer Liste zulässiger Konten, über die der Job Worker über die CodePipeline APIs nach Jobs abfragt.

Schritt 2: Erstellen Sie eine Aktionstyp-Definitionsdatei

Sie definieren einen Aktionstyp in einer Aktionstyp-Definitionsdatei mithilfe von JSON. In der Datei geben Sie die Aktionskategorie, das Integrationsmodell, das zur Verwaltung des Aktionstyps verwendet wird, und die Konfigurationseigenschaften an.

> **Note**
>
> Nachdem Sie eine öffentliche Aktion erstellt haben, können Sie die Aktionstypeigenschaft unter `properties` von `optional` bis nicht ändern`required`. Sie können die auch nicht ändern`owner`.

Weitere Informationen zu den Parametern der Aktionstyp-Definitionsdatei finden Sie unter ActionTypeDeclarationund UpdateActionTypein der CodePipeline API-Referenz.

Die Aktionstyp-Definitionsdatei besteht aus acht Abschnitten:

- `description`: Die Beschreibung für den Aktionstyp, der aktualisiert werden soll.
- `executor`: Informationen über den Executor für einen Aktionstyp, der mit einem unterstützten Integrationsmodell (entweder `Lambda` oder) erstellt wurde. `job worker` Je nach Typ Ihres Executors können Sie nur entweder `jobWorkerExecutorConfiguration` oder `lambdaExecutorConfiguration` angeben.
  - `configuration`: Ressourcen für die Konfiguration des Aktionstyps, basierend auf dem ausgewählten Integrationsmodell. Verwenden Sie für das Lambda-Integrationsmodell die Lambda-Funktion ARN. Verwenden Sie für das Job Worker-Integrationsmodell das Konto oder die Liste der Konten, von denen aus der Job Worker ausgeführt wird.
  - `jobTimeout`: Das Timeout in Sekunden für den Job. Eine Aktionsausführung kann aus mehreren Aufträgen bestehen. Dies ist das Timeout für einen einzelnen Job und nicht für die gesamte Aktionsausführung.

    > **Note**
    >
    > Für das Lambda-Integrationsmodell beträgt das maximale Timeout 15 Minuten.

  - `policyStatementsTemplate`: Die Richtlinienerklärung, die die Berechtigungen im CodePipeline Kundenkonto festlegt, die für die erfolgreiche Ausführung einer Aktion erforderlich sind.
  - `type`: Das Integrationsmodell, das zur Erstellung und Aktualisierung des Aktionstyps verwendet wurde, entweder `Lambda` oder`JobWorker`.
- `id`: Die Kategorie, der Besitzer, der Anbieter und die Versions-ID für den Aktionstyp:
  - `category`: Die Art der Aktion kann in der Phase „Quelle“, „Build“, „Bereitstellen“, „Testen“, „Aufrufen“ oder „Genehmigung“ ausgeführt werden.
  - `provider`: Der Anbieter des aufgerufenen Aktionstyps, z. B. die Firma oder der Produktname des Anbieters. Der Anbietername wird bei der Erstellung des Aktionstyps angegeben.
  - `owner`: Der Ersteller des Aktionstyps, der aufgerufen wird: `AWS` oder`ThirdParty`.
  - `version`: Eine Zeichenfolge, die zur Versionierung des Aktionstyps verwendet wird. Setzen Sie für die erste Version die Versionsnummer auf 1.

- `inputArtifactDetails`: Die Anzahl der Artefakte, die in der vorherigen Phase der Pipeline zu erwarten sind.
- `outputArtifactDetails`: Die Anzahl der Artefakte, die vom Ergebnis der Aktionstypphase zu erwarten sind.
- `permissions`: Details zur Identifizierung der Konten mit Berechtigungen zur Verwendung des Aktionstyps.
- `properties`: Die Parameter, die für die Ausführung Ihrer Projektaufgaben erforderlich sind.
  - `description`: Die Beschreibung der Eigenschaft, die Benutzern angezeigt wird.
  - `optional`: Ob die Konfigurationseigenschaft optional ist.
  - `noEcho`: Ob der vom Kunden eingegebene Feldwert nicht im Protokoll enthalten ist. Wenn `true`, dann wird der Wert geschwärzt, wenn er mit einer GetPipeline API-Anfrage zurückgegeben wird.
  - `key`: Ob die Konfigurationseigenschaft ein Schlüssel ist.
  - `queryable`: Ob die Eigenschaft beim Polling verwendet wird. Ein Aktionstyp kann bis zu eine abfragbare Eigenschaft haben. Wenn dies der Fall ist, muss diese Eigenschaft erforderlich und nicht geheim sein.
  - `name`: Der Eigenschaftsname, der Benutzern angezeigt wird.
- `urls`: Ihren Benutzern wird eine Liste der URLs CodePipeline angezeigt.
  - `entityUrlTemplate`: URL zu den externen Ressourcen für den Aktionstyp, z. B. eine Konfigurationsseite.
  - `executionUrlTemplate`: URL zu den Details für die letzte Ausführung der Aktion.
  - `revisionUrlTemplate`: In der CodePipeline Konsole angezeigte URL zu der Seite, auf der Kunden die Konfiguration der externen Aktion aktualisieren oder ändern können.
  - `thirdPartyConfigurationUrl`: URL einer Seite, auf der sich Benutzer für einen externen Service registrieren und die Erstkonfiguration der von diesem Service bereitgestellten Aktion durchführen können.

Der folgende Code zeigt ein Beispiel für eine Definitionsdatei für einen Aktionstyp.

```
{
   "actionType": {
      "description": "string",
      "executor": {
         "configuration": {
            "jobWorkerExecutorConfiguration": {
```

```
            "pollingAccounts": [ "string" ],
            "pollingServicePrincipals": [ "string" ]
         },
         "lambdaExecutorConfiguration": {
            "lambdaFunctionArn": "string"
         }
      },
      "jobTimeout": number,
      "policyStatementsTemplate": "string",
      "type": "string"
   },
   "id": {
      "category": "string",
      "owner": "string",
      "provider": "string",
      "version": "string"
   },
   "inputArtifactDetails": {
      "maximumCount": number,
      "minimumCount": number
   },
   "outputArtifactDetails": {
      "maximumCount": number,
      "minimumCount": number
   },
   "permissions": {
      "allowedAccounts": [ "string" ]
   },
   "properties": [
      {
         "description": "string",
         "key": boolean,
         "name": "string",
         "noEcho": boolean,
         "optional": boolean,
         "queryable": boolean
      }
   ],
   "urls": {
      "configurationUrl": "string",
      "entityUrlTemplate": "string",
      "executionUrlTemplate": "string",
      "revisionUrlTemplate": "string"
   }
```

```
    }
}
```

Schritt 3: Registrieren Sie Ihre Integration bei CodePipeline

Um Ihren Aktionstyp bei zu registrieren CodePipeline, wenden Sie sich mit Ihrer Anfrage an das CodePipeline Serviceteam.

Das CodePipeline Serviceteam registriert die neue Aktionstyp-Integration, indem es Änderungen an der Service-Codebasis vornimmt. CodePipeline registriert zwei neue Aktionen: eine öffentliche Aktion und eine private Aktion. Sie verwenden die private Aktion zum Testen, und wenn Sie bereit sind, aktivieren Sie die öffentliche Aktion, um den Kundenverkehr zu bedienen.

Um eine Anfrage für eine Lambda-Integration zu registrieren

- Senden Sie mit dem folgenden Formular eine Anfrage an das CodePipeline Serviceteam.

```
This issue will track the onboarding of [Name] in CodePipeline.


[Contact engineer] will be the primary point of contact for this integration.

Name of the action type as you want it to appear to customers: Example.com Testing

Initial onboard checklist:

1. Attach an action type definition file in JSON format. This includes the schema
 for the action type

2. A list of test accounts for the allowlist which can access the new action type
 [{account, account_name}]

3. The Lambda function ARN

4. List of AWS-Regionen where your action will be available

5. Will this be available as a public action?
```

Um eine Anfrage für eine Jobarbeiterintegration zu registrieren

- Senden Sie mit dem folgenden Formular eine Anfrage an das CodePipeline Serviceteam.

```
This issue will track the onboarding of [Name] in CodePipeline.

[Contact engineer] will be the primary point of contact for this integration.


Name of the action type as you want it to appear to customers: Example.com Testing

Initial onboard checklist:

1. Attach an action type definition file in JSON format. This includes the schema
 for the action type.

2. A list of test accounts for the allowlist which can access the new action type
 [{account, account_name}]

3. URL information:
Website URL: https://www.example.com/%TestThirdPartyName%/%TestVersionNumber%

Example URL pattern where customers will be able to review their configuration
 information for the action: https://www.example.com/%TestThirdPartyName%/
%customer-ID%/%CustomerActionConfiguration%

Example runtime URL pattern: https://www.example.com/%TestThirdPartyName%/
%customer-ID%/%TestRunId%

4. List of AWS-Regionen where your action will be available

5. Will this be available as a public action?
```

Schritt 4: Aktivieren Sie Ihre neue Integration

Wenden Sie sich an das CodePipeline Serviceteam, wenn Sie bereit sind, die neue Integration öffentlich zu verwenden.

Fügen Sie einer Pipeline (Konsole) einen verfügbaren Aktionstyp hinzu

Sie fügen Ihren Aktionstyp zu einer Pipeline hinzu, damit Sie ihn testen können. Sie können dies tun, indem Sie eine neue Pipeline erstellen oder eine bestehende bearbeiten.

> **Note**
>
> Wenn es sich bei Ihrem Aktionstyp um eine Aktion der Kategorie „Quelle“, „Build“ oder „Bereitstellung“ handelt, können Sie ihn hinzufügen, indem Sie eine Pipeline erstellen. Wenn sich Ihr Aktionstyp in der Testkategorie befindet, müssen Sie ihn hinzufügen, indem Sie eine vorhandene Pipeline bearbeiten.

Um Ihren Aktionstyp über die CodePipeline Konsole zu einer vorhandenen Pipeline hinzuzufügen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie in der Liste der Pipelines die Pipeline aus, der Sie den Aktionstyp hinzufügen möchten.
3. Wählen Sie auf der Übersichtsseite der Pipeline die Option Bearbeiten aus.
4. Wählen Sie, ob Sie die Phase bearbeiten möchten. Wählen Sie in der Phase, in der Sie Ihren Aktionstyp hinzufügen möchten, die Option Aktionsgruppe hinzufügen aus. Die Seite Aktion bearbeiten wird angezeigt.
5. Geben Sie auf der Seite Aktion bearbeiten im Feld Aktionsname einen Namen für die Aktion ein. Dies ist der Name, der für die Phase in Ihrer Pipeline angezeigt wird.
6. Wählen Sie unter Aktionsanbieter Ihren Aktionstyp aus der Liste aus.

   Beachten Sie, dass der Wert in der Liste auf dem in der Aktionstyp-Definitionsdatei `provider` angegebenen Wert basiert.
7. Geben Sie im Feld Eingabeartefakte den Namen des Artefakts in diesem Format ein:

   `Artifactname``::`*`FileName`*

   Beachten Sie, dass die zulässigen Mindest- und Höchstmengen auf der Grundlage der in der Aktionstyp-Definitionsdatei `inputArtifactDetails` angegebenen Werte definiert werden.
8. Wählen Sie Connect<Action_Name>.

   Ein Browserfenster wird geöffnet und stellt eine Verbindung zu der Website her, die Sie für Ihren Aktionstyp erstellt haben.
9. Melden Sie sich als Kunde auf Ihrer Website an und führen Sie die Schritte aus, die ein Kunde zur Verwendung Ihres Aktionstyps unternimmt. Ihre Schritte variieren je nach Aktionskategorie,

Website und Konfiguration, beinhalten aber in der Regel eine Abschlussaktion, die den Kunden zur Seite Aktion bearbeiten zurückbringt.

10. Auf der Seite Aktion CodePipeline bearbeiten werden die zusätzlichen Konfigurationsfelder für die Aktion angezeigt. Bei den angezeigten Feldern handelt es sich um die Konfigurationseigenschaften, die Sie in der Aktionsdefinitionsdatei angegeben haben. Geben Sie die Informationen in die Felder ein, die für Ihren Aktionstyp angepasst sind.

    Wenn in der Aktionsdefinitionsdatei beispielsweise eine Eigenschaft mit dem Namen angegeben wurde`Host`, wird auf der Seite Aktion bearbeiten für Ihre Aktion ein Feld mit der Bezeichnung Host angezeigt.

11. Geben Sie im Feld Ausgabeartefakte den Namen des Artefakts in diesem Format ein:

    `Artifactname``::`*`FileName`*

    Beachten Sie, dass die zulässigen Mindest- und Höchstmengen auf der Grundlage der in der Aktionstyp-Definitionsdatei `outputArtifactDetails` angegebenen Werte definiert werden.

12. Wählen Sie Fertig, um zur Seite mit den Pipeline-Details zurückzukehren.

    > **Note**
    >
    > Ihre Kunden können optional die CLI verwenden, um den Aktionstyp zu ihrer Pipeline hinzuzufügen.

13. Um Ihre Aktion zu testen, übernehmen Sie eine Änderung an der Quelle, die in der Quellphase der Pipeline angegeben wurde, oder folgen Sie den Schritten unter Manuelles Starten einer Pipeline.

Um eine Pipeline mit Ihrem Aktionstyp zu erstellen, folgen Sie den Schritten unter Erstellen Sie eine Pipeline in CodePipeline und wählen Sie Ihren Aktionstyp aus so vielen Phasen aus, wie Sie testen möchten.

Zeigen Sie einen Aktionstyp an

Sie können die CLI verwenden, um Ihren Aktionstyp anzuzeigen. Verwenden Sie den get-action-type Befehl, um Aktionstypen anzuzeigen, die mithilfe eines Integrationsmodells erstellt wurden.

Um einen Aktionstyp anzuzeigen

1. Erstellen Sie eine JSON-Eingabedatei und geben Sie der Datei einen Namen`file.json`. Fügen Sie Ihre Aktionstyp-ID im JSON-Format hinzu, wie im folgenden Beispiel gezeigt.

```
{
    "category": "Test",
    "owner": "ThirdParty",
    "provider": "TestProvider",
    "version": "1"
}
```

2. Führen Sie den Befehl in einem Terminalfenster oder in der get-action-type Befehlszeile aus.

```
aws codepipeline get-action-type --cli-input-json file://file.json
```

Dieser Befehl gibt die Aktionsdefinitionsausgabe für einen Aktionstyp zurück. Dieses Beispiel zeigt einen Aktionstyp, der mit dem Lambda-Integrationsmodell erstellt wurde.

```
{
    "actionType": {
        "executor": {
            "configuration": {
                "lambdaExecutorConfiguration": {
                    "lambdaFunctionArn": "arn:aws:lambda:us-west-2:<account-
id>:function:my-function"
                }
            },
            "type": "Lambda"
        },
        "id": {
            "category": "Test",
            "owner": "ThirdParty",
            "provider": "TestProvider",
            "version": "1"
        },
        "inputArtifactDetails": {
            "minimumCount": 0,
            "maximumCount": 1
        },
        "outputArtifactDetails": {
            "minimumCount": 0,
```

```
            "maximumCount": 1
        },
        "permissions": {
            "allowedAccounts": [
                "<account-id>"
            ]
        },
        "properties": []
    }
}
```

Aktualisieren Sie einen Aktionstyp

Sie können die CLI verwenden, um Aktionstypen zu bearbeiten, die mit einem Integrationsmodell erstellt wurden.

Bei einem öffentlichen Aktionstyp können Sie den Besitzer nicht aktualisieren, Sie können optionale Eigenschaften nicht in Erforderlich ändern und Sie können nur neue optionale Eigenschaften hinzufügen.

1. Verwenden Sie den `get-action-type` Befehl, um die Struktur für Ihren Aktionstyp abzurufen. Kopieren Sie die Struktur.
2. Erstellen Sie eine JSON-Eingabedatei und geben Sie ihr einen Namen`action.json`. Fügen Sie die Aktionstypstruktur ein, die Sie im vorherigen Schritt kopiert haben. Aktualisieren Sie alle Parameter, die Sie ändern möchten. Sie können auch optionale Parameter hinzufügen.

   Weitere Informationen zu den Parametern für die Eingabedatei finden Sie in der Beschreibung der Aktionsdefinitionsdatei unterSchritt 2: Erstellen Sie eine Aktionstyp-Definitionsdatei.

   Das folgende Beispiel zeigt, wie ein mit dem Lambda-Integrationsmodell erstellter Beispielaktionstyp aktualisiert wird. In diesem Beispiel werden die folgenden Änderungen vorgenommen:

   - Ändert den `provider` Namen in`TestProvider1`.
   - Fügt ein Job-Timeout-Limit von 900 Sekunden hinzu.
   - Fügt eine Aktionskonfigurationseigenschaft mit dem Namen hinzu`Host`, die dem Kunden angezeigt wird, der die Aktion verwendet.

     ```
     {
     ```

```
    "actionType": {
        "executor": {
            "configuration": {
                "lambdaExecutorConfiguration": {
                    "lambdaFunctionArn": "arn:aws:lambda:us-west-2:<account-
id>:function:my-function"
                }
            },
            "type": "Lambda",
            "jobTimeout": 900
        },
        "id": {
            "category": "Test",
            "owner": "ThirdParty",
            "provider": "TestProvider1",
            "version": "1"
        },
        "inputArtifactDetails": {
            "minimumCount": 0,
            "maximumCount": 1
        },
        "outputArtifactDetails": {
            "minimumCount": 0,
            "maximumCount": 1
        },
        "permissions": {
            "allowedAccounts": [
                "account-id"
            ]
        },
        "properties": {
         "description": "Owned build action parameter description",
         "optional": true,
         "noEcho": false,
         "key": true,
         "queryable": false,
         "name": "Host"
         }
    }
}
```

3. Führen Sie im Terminal oder in der Befehlszeile den update-action-type Befehl aus

```
aws codepipeline update-action-type --cli-input-json file://action.json
```

Dieser Befehl gibt die Ausgabe des Aktionstyps zurück, die Ihren aktualisierten Parametern entspricht.

Erstellen und fügen Sie eine benutzerdefinierte Aktion hinzu in CodePipeline

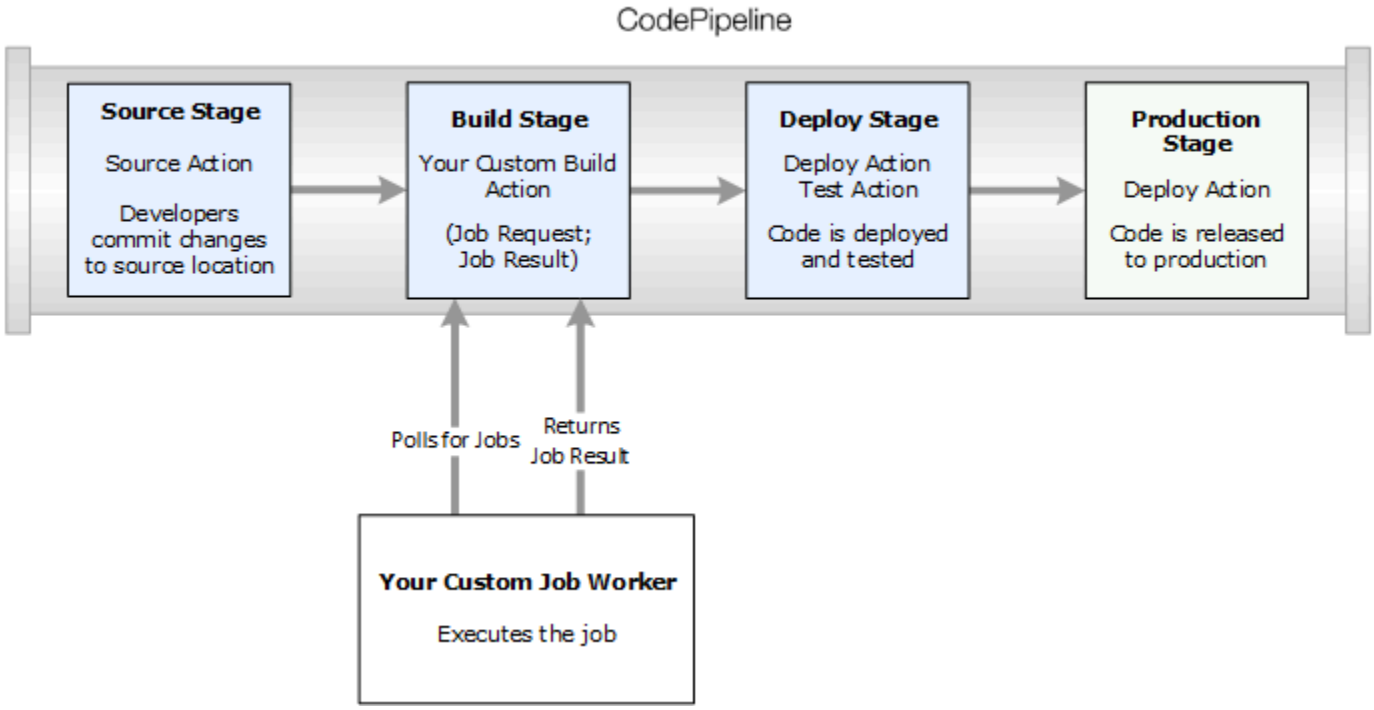
AWS CodePipeline umfasst eine Reihe von Aktionen, mit denen Sie Ressourcen für den automatisierten Release-Prozess konfigurieren, testen und bereitstellen können. Falls Ihr Freigabeverfahren Aktivitäten beinhaltet, die nicht durch die Standardaktionen abgedeckt sind (wie z. B. ein intern entwickelter Erstellungsprozess oder eine Testsuite), können Sie dafür eine benutzerdefinierte Aktion erstellen und diese Ihrer Pipeline hinzufügen. Sie können den verwenden AWS CLI , um benutzerdefinierte Aktionen in Pipelines zu erstellen, die mit Ihrem AWS Konto verknüpft sind.

Sie können benutzerdefinierte Aktionen für die folgenden AWS CodePipeline Aktionskategorien erstellen:

- Eine benutzerdefinierte Build-Aktionen für die Erstellung oder Umwandlung von Elementen
- Eine benutzerdefinierte Deploy-Aktion zur Bereitstellung der Elemente für eine(n) oder mehrere Server, Webseiten oder Repositorys
- Eine benutzerdefinierte Testaktion für die Konfiguration und Ausführung von automatisierten Tests
- Eine benutzerdefinierte Aufrufaktionen für das Ausführen von Funktionen

Wenn Sie eine benutzerdefinierte Aktion erstellen, müssen Sie auch einen Job-Worker erstellen, der CodePipeline Jobanfragen für diese benutzerdefinierte Aktion abfragt, den Job ausführt und das Statusergebnis zurückgibt CodePipeline. Dieser Jobworker kann sich auf jedem Computer oder jeder Ressource befinden, sofern er Zugriff auf den öffentlichen Endpunkt für hat CodePipeline. Um Zugriff und Sicherheit einfach zu verwalten, sollten Sie erwägen, Ihren Job Worker auf einer Amazon EC2 EC2-Instance zu hosten.

Das folgende Diagramm zeigt einen umfassenden Überblick über eine Pipeline mit einer benutzerdefinierten Build-Aktion:



Wenn eine Pipeline eine benutzerdefinierte Aktion als Teil einer Phase enthält, erstellt die Pipeline eine Jobanfrage. Ein benutzerdefinierter Auftragsworker erkennt diese Anforderung und führt den Auftrag aus (in diesem Beispiel ein benutzerdefinierter Vorgang mit Software von Drittanbietern). Sobald die Aktion abgeschlossen ist, wird vom Auftragsworker das Ergebnis über die erfolgreiche oder fehlgeschlagene Durchführung zurückgegeben. Wenn ein erfolgreiches Ergebnis eingeht, stellt die Pipeline die Revision und ihre Artefakte für die nächste Aktion bereit. Wenn ein Fehler zurückgegeben wird, stellt die Pipeline die Revision nicht für die nächste Aktion in der Pipeline bereit.

> **Note**
>
> Für diese Anleitungen müssen Sie die Schritte unter Erste Schritte mit CodePipeline ausgeführt haben.

Themen

- Erstellen einer benutzerdefinierten Aktion
- Erstellen eines Auftragsworkers für Ihre benutzerdefinierte Aktion
- Hinzufügen einer benutzerdefinierten Aktion zu einer Pipeline

Erstellen einer benutzerdefinierten Aktion

Um eine benutzerdefinierte Aktion mit dem zu erstellen AWS CLI

1. Öffnen Sie einen Texteditor und erstellen Sie eine JSON-Datei mit der Aktion, dem Aktionsanbieter und allen erforderlichen Einstellungen für Ihre benutzerdefinierte Aktion. Ihre JSON-Datei für eine benutzerdefinierte Aktion mit nur einer Eigenschaft könnte beispielsweise wie folgt aussehen:

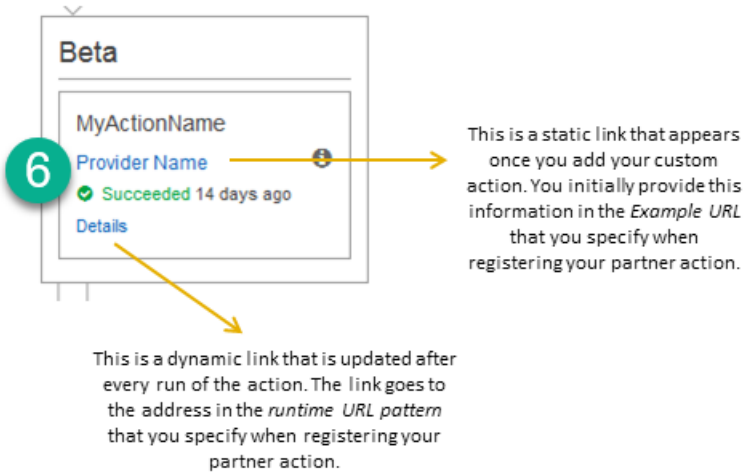
```
{
    "category": "Build",
    "provider": "My-Build-Provider-Name",
    "version": "1",
    "settings": {
        "entityUrlTemplate": "https://my-build-instance/job/{Config:ProjectName}/",
        "executionUrlTemplate": "https://my-build-instance/job/
{Config:ProjectName}/lastSuccessfulBuild/{ExternalExecutionId}/"
    },
    "configurationProperties": [{
        "name": "ProjectName",
        "required": true,
        "key": true,
        "secret": false,
        "queryable": false,
        "description": "The name of the build project must be provided when this
 action is added to the pipeline.",
        "type": "String"
    }],
    "inputArtifactDetails": {
        "maximumCount": integer,
        "minimumCount": integer
    },
    "outputArtifactDetails": {
        "maximumCount": integer,
        "minimumCount": integer
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
    }]
}
```

In diesem Beispiel wird der benutzerdefinierten Aktion Tagging hinzugefügt, indem der benutzerdefinierten Aktion der Tag-Schlüssel `Project` und der Wert `ProjectA` hinzugefügt werden. Weitere Informationen zum Markieren von Ressourcen in finden Sie CodePipeline unterMarkieren von Ressourcen.

Die JSON-Datei enthält zwei Eigenschaften, `entityUrlTemplate` und `executionUrlTemplate`. Solange die Konfigurationseigenschaft erforderlich und nicht geheim ist, können Sie über das Format `{Config:name}` in den URL-Vorlagen auf einen Namen in den Konfigurationseigenschaften der benutzerdefinierten Aktion verweisen. Im obigen Beispiel bezieht sich der `entityUrlTemplate` Wert beispielsweise auf die Konfigurationseigenschaft *`ProjectName`*.

- `entityUrlTemplate`: Der statische Link mit Informationen über die Dienstanbieter für die Aktion. Im Beispiel umfasst das Build-System einen statischen Link zu den einzelnen Build-Projekten. Das Linkformat variiert je nach Build-Anbieter oder Dienstanbieter (beispielsweise bei anderen Aktionstypen wie Test). Sie müssen dieses Linkformat so angeben, dass der Benutzer beim Hinzufügen der benutzerdefinierten Aktion mit dem Link eine Webseite im Browser öffnen kann, die die Einzelheiten für das Build-Projekt (oder die Testumgebung) enthält.
- `executionUrlTemplate`: Der dynamische Link, der mit Informationen zur aktuellen oder letzten Ausführung der Aktion aktualisiert wird. Wenn Ihr benutzerdefinierter Auftragsworker den Status eines Auftrags aktualisiert (z. B. bei Erfolg, Fehler oder während der Ausführung), stellt er zusätzlich eine `externalExecutionId` bereit, die zur Vervollständigung des Links verwendet wird. Dieser Link kann verwendet werden, um Details zum Ausführen einer Aktion bereitzustellen.

Wenn Sie die Aktion beispielsweise in der Pipeline anzeigen, sehen Sie die beiden folgenden Links:



1

Dieser statische Link wird angezeigt, nachdem Sie Ihre benutzerdefinierte Aktion hinzugefügt haben. Er verweist auf die Adresse in `entityUrlTemplate`, die Sie bei der Erstellung Ihrer benutzerdefinierten Aktion angegeben haben.

2

Dieser dynamische Link wird nach jeder Ausführung der Aktion aktualisiert und verweist auf die Adresse in `executionUrlTemplate`, die Sie bei der Erstellung Ihrer benutzerdefinierten Aktion angegeben haben.

Weitere Informationen zu diesen Linktypen sowie zu `RevisionURLTemplate` und `ThirdPartyURL` finden Sie unter ActionTypeSettingsund CreateCustomActionTypein der CodePipeline API-Referenz. Weitere Informationen über die Anforderungen zur Aktionsstruktur und zur Erstellung einer Aktion finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

2. Speichern Sie die JSON-Datei und geben Sie ihr einen Namen, den Sie sich leicht merken können (z. B. „*`MyCustomAction`*.json“).
3. Öffnen Sie eine Terminalsitzung (Linux, OS X, Unix) oder eine Eingabeaufforderung (Windows) auf einem Computer, auf dem die AWS CLI installiert ist.
4. Verwenden Sie den, AWS CLI um den aws codepipeline create-custom-action-type Befehl auszuführen, und geben Sie dabei den Namen der JSON-Datei an, die Sie gerade erstellt haben.

Um beispielsweise eine benutzerdefinierte Build-Aktion zu erstellen:

> **Important**
>
> Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

```
aws codepipeline create-custom-action-type --cli-input-json
 file://MyCustomAction.json
```

5. Dieser Befehl gibt die gesamte Struktur der erstellten, benutzerdefinierten Aktion sowie die für Sie hinzugefügte `JobList`-Aktionskonfigurationseigenschaft zurück. Wenn Sie die benutzerdefinierte Aktion zu einer Pipeline hinzufügen, können Sie `JobList` nutzen, um die Projekte vom Anbieter anzugeben, für die Sie Aufträge abfragen können. Ohne entsprechende Konfiguration werden bei der Abfrage von Aufträgen durch Ihren benutzerdefinierten Auftragsworker alle verfügbaren Aufträge zurückgegeben.

   Der vorherige Befehl könnte beispielsweise eine Struktur ähnlich der Folgenden zurückgeben:

```
{
    "actionType": {
        "inputArtifactDetails": {
            "maximumCount": 1,
            "minimumCount": 1
       },
       "actionConfigurationProperties": [
            {
                "secret": false,
                "required": true,
                "name": "ProjectName",
                "key": true,
                "description": "The name of the build project must be provided when
 this action is added to the pipeline."
            }
        ],
        "outputArtifactDetails": {
            "maximumCount": 0,
            "minimumCount": 0
        },
```

```
        "id": {
            "category": "Build",
            "owner": "Custom",
            "version": "1",
            "provider": "My-Build-Provider-Name"
        },
        "settings": {
            "entityUrlTemplate": "https://my-build-instance/job/
{Config:ProjectName}/",
            "executionUrlTemplate": "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild/{ExternalExecutionId}/"
        }
    }
}
```

> **Note**
>
> Als Teil der Ausgabe des create-custom-action-type Befehls umfasst der `id` Abschnitt`"owner": "Custom"`. CodePipeline weist benutzerdefinierte Aktionstypen automatisch `Custom` als Besitzer zu. Dieser Wert kann zugewiesen oder geändert werden, wenn Sie den Befehl create-custom-action-type oder den Befehl update-pipeline verwenden.

Erstellen eines Auftragsworkers für Ihre benutzerdefinierte Aktion

Für benutzerdefinierte Aktionen ist ein Job-Worker erforderlich, der Jobanfragen für die benutzerdefinierte Aktion abfragt, den Job ausführt und das Statusergebnis CodePipeline zurückgibt. CodePipeline Der Job Worker kann sich auf jedem Computer oder jeder Ressource befinden, sofern er Zugriff auf den öffentlichen Endpunkt für hat CodePipeline.

Sie können Ihren Auftragsworker auf vielerlei Arten gestalten. Die folgenden Abschnitte enthalten einige praktische Anleitungen zur Entwicklung Ihres individuellen Job Workers für CodePipeline.

Themen

- Wählen und Konfigurieren einer Strategie für die Berechtigungsverwaltung Ihres Auftragsworkers
- Entwickeln eines Auftragsworkers für Ihre benutzerdefinierte Aktion
- Benutzerdefinierte Auftragsworkerarchitektur und Beispiele

Wählen und Konfigurieren einer Strategie für die Berechtigungsverwaltung Ihres Auftragsworkers

Um einen benutzerdefinierten Job Worker für Ihre benutzerdefinierte Aktion in zu entwickeln CodePipeline, benötigen Sie eine Strategie für die Integration von Benutzer- und Rechteverwaltung.

Die einfachste Strategie besteht darin, die Infrastruktur, die Sie für Ihren benutzerdefinierten Job Worker benötigen, hinzuzufügen, indem Sie Amazon EC2 EC2-Instances mit einer IAM-Instance-Rolle erstellen, sodass Sie die Ressourcen, die Sie für Ihre Integration benötigen, einfach skalieren können. Sie können die integrierte Integration mit verwenden AWS , um die Interaktion zwischen Ihrem benutzerdefinierten Job Worker und zu vereinfachen. CodePipeline

So richten Sie Amazon EC2 EC2-Instances ein

1. Erfahren Sie mehr über Amazon EC2 und finden Sie heraus, ob es die richtige Wahl für Ihre Integration ist. Weitere Informationen finden Sie unter Amazon EC2 — Virtual Server Hosting.
2. Beginnen Sie mit der Erstellung Ihrer Amazon EC2 EC2-Instances. Weitere Informationen finden Sie unter Erste Schritte mit Amazon EC2 EC2-Linux-Instances.

Eine weitere Strategie, die Sie in Betracht ziehen sollten, ist die Verwendung eines Identitätsverbunds mit IAM zur Integration Ihres vorhandenen Identitätsanbietersystems und Ihrer Ressourcen. Diese Strategie ist besonders nützlich, wenn Sie bereits einen Unternehmensidentitätsanbieter haben oder eine Konfiguration vorgenommen wurde, um Benutzer mit Web-Identitätsanbietern zu unterstützen. Mit dem Identitätsverbund können Sie sicheren Zugriff auf AWS Ressourcen gewähren CodePipeline, auch ohne IAM-Benutzer erstellen oder verwalten zu müssen. Sie können Funktionen und Richtlinien für die Sicherheitsanforderungen bezüglich Passwörtern und die Rotation von Anmeldeinformationen nutzen. Sie können Beispielanwendungen als Vorlage für Ihre selbsterstellten Anwendungen verwenden.

So richten Sie einen Identitätsverbund ein

1. Erfahren Sie mehr über den IAM-Identitätsverbund. Weitere Informationen finden Sie unter Manage Federation.
2. Sehen Sie sich die Beispiele in Scenarios for Granting Temporary Access an, um das Szenario für den temporären Zugriff zu ermitteln, das am besten für Ihre benutzerdefinierte Aktion geeignet ist.

3. Sehen Sie sich die für Ihre Infrastruktur relevanten Codebeispiele zum Identitätsverbund an. Diese sind beispielsweise:

   - Identity Federation Sample Application for an Active Directory Use Case

4. Erste Schritte bei der Konfiguration eines Identitätsverbunds. Weitere Informationen finden Sie unter Identity Providers and Federation im IAM-Benutzerhandbuch.

Erstellen Sie eine der folgenden Optionen, die Sie unter Ihrem verwenden können, AWS-Konto wenn Sie Ihre benutzerdefinierte Aktion und Ihren Job Worker ausführen.

Benutzer benötigen programmgesteuerten Zugriff, wenn sie mit AWS anderen interagieren möchten. AWS Management Console Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt vom Benutzertyp ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zu AWS IAM Identity Center verwendenden im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs, Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch für AWS SDKs und Tools.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Folgen Sie den Anweisungen unter Verwenden temporäre r Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu finden Sie unter Authentifizierung mithilfe von IAM-Benut zeranmeldedaten im Benutzerhandbuch. AWS CLIAWS Command Line Interface • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldein formationen im Referenzh andbuch für AWS SDKs und Tools. • Informationen zu AWS APIs finden Sie unter Verwaltun g von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benut zerhandbuch.

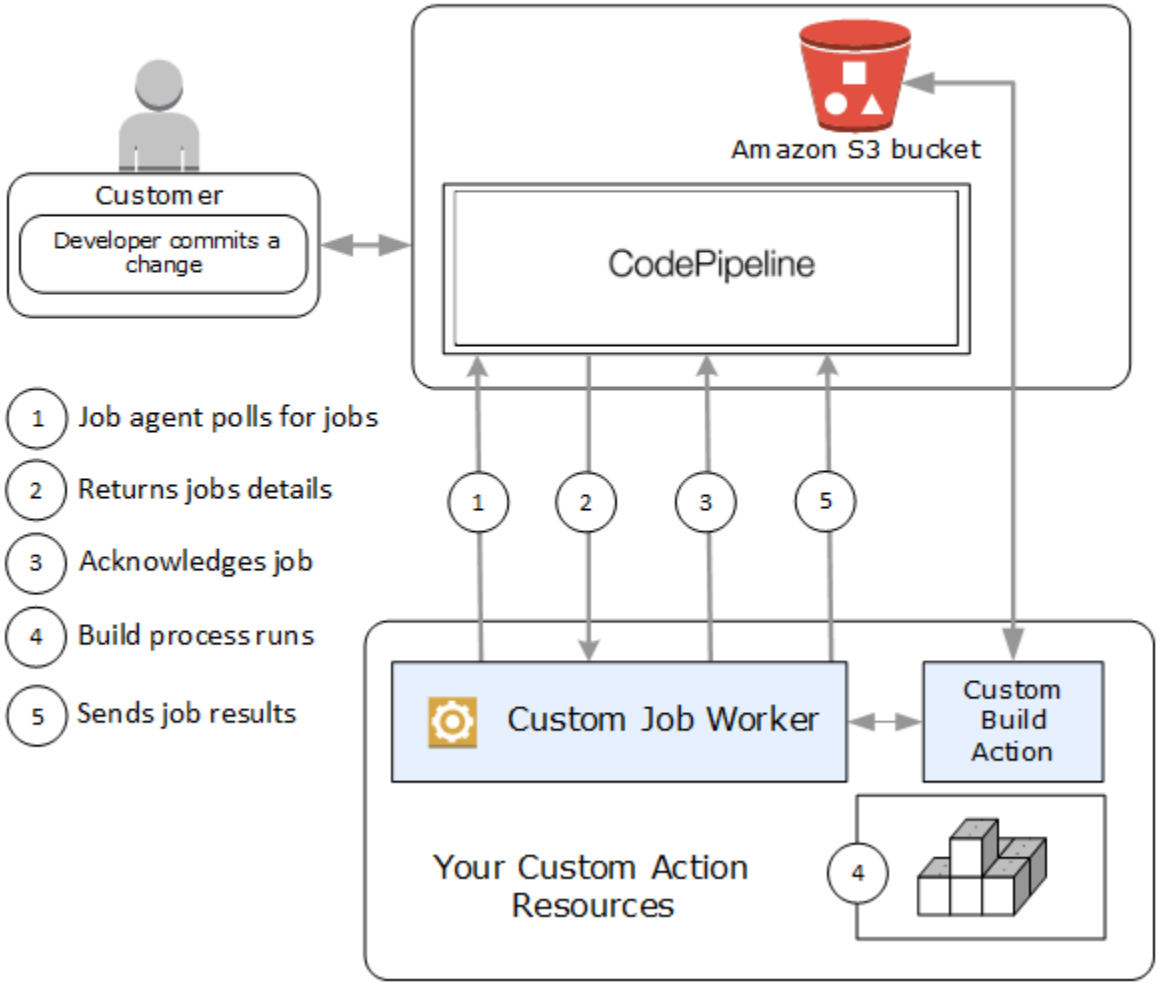
Es folgt ein Beispiel für eine Richtlinie, die Sie für die Verwendung mit Ihrem benutzerdefinierten Auftragsworker erstellen können. Diese Richtlinie dient lediglich als Beispiel und wird unverändert bereitgestellt.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:PollForJobs",
        "codepipeline:AcknowledgeJob",
        "codepipeline:GetJobDetails",
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
      ],
      "Resource": [
        "arn:aws:codepipeline:us-east-2::actionType:custom/Build/MyBuildProject/1/"
      ]
    }
  ]
}
```

> **Note**
>
> Erwägen Sie die Verwendung der `AWSCodePipelineCustomActionAccess` verwalteten Richtlinie.

Entwickeln eines Auftragsworkers für Ihre benutzerdefinierte Aktion

Nachdem Sie Ihre Strategie für die Verwaltung von Berechtigungen ausgewählt haben, sollten Sie sich überlegen, wie Ihr Mitarbeiter mit Ihrem Job umgehen wird CodePipeline. Das folgende allgemeine Diagramm zeigt den Arbeitsablauf einer benutzerdefinierten Aktion und eines Job Workers für einen Build-Prozess.



1. Ihr Jobworker fragt CodePipeline nach Jobs ab mit`PollForJobs`.
2. Wenn eine Pipeline durch eine Veränderung in der Quellphase ausgelöst wird (wenn beispielsweise ein Entwickler eine Änderung vornimmt), wird das automatisierte Freigabeverfahren gestartet. Der Vorgang wird fortgesetzt, bis Ihre benutzerdefinierte Aktion konfiguriert wurde. Wenn in dieser Phase Ihre Aktion erreicht ist, wird ein Job in CodePipeline die Warteschlange gestellt. Dieser Auftrag wird angezeigt, wenn Ihr Auftragsworker erneut `PollForJobs` aufruft, um den Status abzurufen. Entnehmen Sie `PollForJobs` die Auftragsdetails und geben Sie diese an Ihren Auftragsworker zurück.
3. Der Arbeitsarbeiter ruft an`AcknowledgeJob`, um CodePipeline eine Auftragsbestätigung zu senden. CodePipeline gibt eine Bestätigung zurück, die besagt, dass der Arbeitsnehmer den Job fortsetzen soll (`InProgress`), oder, wenn mehr als ein Jobarbeiter nach Stellen sucht und ein anderer Jobarbeiter den Job bereits beansprucht hat, wird eine `InvalidNonceException`

Fehlerantwort zurückgegeben. CodePipeline Wartet nach der `InProgress` Bestätigung auf die Rückgabe der Ergebnisse.

4. Der Job-Worker initiiert Ihre benutzerdefinierte Aktion für die Revision, und dann wird Ihre Aktion ausgeführt. Zusammen mit allen anderen Aktionen gibt Ihre benutzerdefinierte Aktion ein Ergebnis an den Job-Worker zurück. Im Beispiel einer benutzerdefinierten Build-Aktion ruft die Aktion Artefakte aus dem Amazon S3 S3-Bucket ab, erstellt sie und überträgt erfolgreich erstellte Artefakte zurück in den Amazon S3 S3-Bucket.
5. Während die Aktion ausgeführt wird, kann der Job-Worker `PutJobSuccessResult` mit einem Fortsetzungstoken (der Serialisierung des Status des vom Job-Worker generierten Jobs, z. B. einer Build-ID im JSON-Format oder einem Amazon S3 S3-Objektschlüssel) sowie mit den `ExternalExecutionId` Informationen, die zum Auffüllen des Links verwendet werden, aufrufen. `executionUrlTemplate` Dadurch wird die Konsolenansicht der Pipeline während des Vorgangs mit einem funktionierenden Link zu spezifischen Aktionsdetails versehen. Dieses Vorgehen ist zwar nicht erforderlich, jedoch eine bewährte Methode, da sie Benutzern das Abrufen des Status der benutzerdefinierten Aktion ermöglicht, während diese ausgeführt wird.

   Sobald `PutJobSuccessResult` aufgerufen wird, gilt der Auftrag als abgeschlossen. Es wird ein neuer Job erstellt CodePipeline , der das Fortsetzungstoken enthält. Dieser Auftrag wird angezeigt, wenn Ihr Auftragsworker erneut `PollForJobs` aufruft. Dieser neue Auftrag kann für die Überprüfung des Aktionsstatus verwendet werden und wird entweder mit einem Fortsetzungstoken oder nach Abschluss der Aktion ohne ein Fortsetzungstoken zurückgegeben.

   > **Note**
   >
   > Wenn Ihr Auftragsworker alle Aufgaben für eine benutzerdefinierte Aktion übernimmt, sollten Sie erwägen, die Verarbeitung Ihres Auftragsworkers in mindestens zwei Schritte aufzuteilen. Im ersten Schritt wird die Detailseite für Ihre Aktion aufgebaut. Sobald Sie die Detailseite erstellt haben, kann der Status des Auftragsworkers je nach Größenbeschränkungen serialisiert und als Fortsetzungstoken zurückgesendet werden (siehe Kontingente in AWS CodePipeline). Sie können beispielsweise den Status der Aktion in die Zeichenfolge schreiben, die Sie als Fortsetzungstoken verwenden. Im zweiten Verarbeitungsschritt (und allen nachfolgenden Schritten) Ihres Auftragsworkers wird die eigentliche Aufgabe ausgeführt. Der letzte Schritt gibt Erfolg oder Misserfolg zurück CodePipeline, ohne dass im letzten Schritt ein Fortsetzungstoken vorhanden ist.

Weitere Informationen zur Verwendung des Fortsetzungstokens finden Sie in den Spezifikationen für `PutJobSuccessResult` in der CodePipeline API-Referenz.

6. Sobald die benutzerdefinierte Aktion abgeschlossen ist, gibt der Job Worker das Ergebnis der benutzerdefinierten Aktion zurück, CodePipeline indem er eine von zwei APIs aufruft:
   - `PutJobSuccessResult`ohne ein Fortsetzungstoken, was darauf hinweist, dass die benutzerdefinierte Aktion erfolgreich ausgeführt wurde
   - `PutJobFailureResult`, was darauf hinweist, dass die benutzerdefinierte Aktion nicht erfolgreich ausgeführt wurde

   Je nach Ergebnis fährt die Pipeline entweder mit der nächsten Aktion fort (bei Erfolg) oder wird unterbrochen (bei Fehlern).

Benutzerdefinierte Auftragsworkerarchitektur und Beispiele

Nachdem Sie den komplexen Workflow entworfen haben, können Sie Ihren Auftragsworker erstellen. Durch die Einzelheiten Ihrer benutzerdefinierten Aktion wird zwar letztendlich festgelegt, was für Ihren Auftragsworker erforderlich ist, aber die meisten Auftragsworker für benutzerdefinierte Aktionen verfügen über die folgenden Funktionen:

- Abfragen nach Aufträgen aus der CodePipeline Verwendung `PollForJobs` von.
- Bestätigen von Aufträgen und Rückgabe von Ergebnissen bei CodePipeline Verwendung von `AcknowledgeJobPutJobSuccessResult`, und. `PutJobFailureResult`
- Artefakte werden aus dem Amazon S3-Bucket abgerufen und/oder in den Amazon S3 S3-Bucket für die Pipeline gelegt. Um Artefakte aus dem Amazon S3 S3-Bucket herunterzuladen, müssen Sie einen Amazon S3 S3-Client erstellen, der Signature Version 4-Signatur (Sig V4) verwendet. Sig V4 ist erforderlich für AWS KMS.

  Um Artefakte in den Amazon S3 S3-Bucket hochzuladen, müssen Sie die Amazon S3 `PutObject` S3-Anfrage zusätzlich für die Verschlüsselung konfigurieren. Derzeit wird nur der AWS Key Management Service (AWS KMS) für die Verschlüsselung unterstützt. AWS KMS verwendet AWS KMS keys. Um zu wissen, ob ein Von AWS verwalteter Schlüssel oder ein vom Kunden verwalteter Schlüssel zum Hochladen von Artefakten verwendet werden soll, muss sich Ihr benutzerdefinierter Job Worker die Auftragsdaten ansehen und die Eigenschaft des Verschlüsselungsschlüssels überprüfen. Wenn die Eigenschaft festgelegt ist, sollten Sie bei der Konfiguration diese vom Kunden verwaltete Schlüssel-ID verwenden AWS KMS. Wenn die Schlüsseleigenschaft Null

ist, verwenden Sie die Von AWS verwalteter Schlüssel. CodePipeline verwendet die, Von AWS verwalteter Schlüssel sofern nicht anders konfiguriert.

Ein Beispiel, das zeigt, wie die AWS KMS Parameter in Java oder .NET erstellt werden, finden Sie unter Spezifying the AWS Key Management Service in Amazon S3 Using the AWS SDKs. Weitere Informationen zum Amazon S3 S3-Bucket für CodePipeline finden Sie unterCodePipeline Konzepte .

Ein komplexeres Beispiel für einen benutzerdefinierten Job Worker finden Sie unter GitHub. Dieses Beispiel ist ein Open Source-Beispiel und wird unverändert bereitgestellt.

- Beispiel Job Worker für CodePipeline: Laden Sie das Beispiel aus dem GitHub Repository herunter.

Hinzufügen einer benutzerdefinierten Aktion zu einer Pipeline

Sobald Sie einen Job Worker eingerichtet haben, können Sie Ihre benutzerdefinierte Aktion zu einer Pipeline hinzufügen, indem Sie eine neue erstellen und diese auswählen, wenn Sie den Assistenten „Pipeline erstellen“ verwenden, indem Sie eine bestehende Pipeline bearbeiten und die benutzerdefinierte Aktion hinzufügen oder indem Sie die SDKs oder die APIs verwenden. AWS CLI

> **Note**
>
> Sie können eine Pipeline über den "Create Pipeline" (Pipeline erstellen)-Assistenten anlegen, die eine benutzerdefinierte Aktion enthält, wenn sie eine Build- oder Deploy-Aktion ist. Wenn sich Ihre benutzerdefinierte Aktion in der Testkategorie befindet, müssen Sie sie durch Bearbeiten einer vorhandenen Pipeline hinzufügen.

Themen

- Hinzufügen einer benutzerdefinierten Aktion zu einer bestehenden Pipeline (CLI)

Hinzufügen einer benutzerdefinierten Aktion zu einer bestehenden Pipeline (CLI)

Sie können den verwenden AWS CLI , um einer vorhandenen Pipeline eine benutzerdefinierte Aktion hinzuzufügen.

1. Öffnen Sie eine Terminalsitzung (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und führen Sie den get-pipeline Befehl aus, um die Pipeline-Struktur, die Sie bearbeiten möchten, in eine JSON-Datei zu kopieren. Für eine Pipeline mit dem Namen **`MyFirstPipeline`** würden Sie beispielsweise den folgenden Befehl verwenden:

```
aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem Texteditor und bearbeiten Sie die Struktur der Datei, um Ihre benutzerdefinierte Aktion zu einer vorhandenen Stufe hinzuzufügen.

> **Note**
>
> Wenn Sie möchten, dass Ihre Aktion mit einer anderen Aktion in dieser Stufe parallel ausgeführt wird, weisen Sie ihr den gleichen `runOrder`-Wert wie der anderen Aktion zu.

Um beispielsweise die Struktur einer Pipeline zu bearbeiten und ihr eine Stufe mit dem Namen Build hinzufügen und gleichzeitig eine benutzerdefinierte Aktion zu dieser Stufe hinzuzufügen, können Sie die JSON-Datei so ändern, dass die Build-Stufe vor einer Bereitstellungsstufe eingefügt ist. Beispiel:

```
,
    {
      "name": "MyBuildStage",
      "actions":  [
              {
                "inputArtifacts": [
                {
                   "name": "MyApp"
                 }
                   ],
                    "name": "MyBuildCustomAction",
                    "actionTypeId": {
                        "category": "Build",
                        "owner": "Custom",
                        "version": "1",
                        "provider": "My-Build-Provider-Name"
```

```
                    },
                    "outputArtifacts": [
                        {
                          "name": "MyBuiltApp"
                        }
                    ],
                    "configuration": {
                        "ProjectName": "MyBuildProject"
                    },
                    "runOrder": 1
                }
            ]
        },
        {
            "name": "Staging",
            "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "MyBuiltApp"
                            }
                        ],
                        "name": "Deploy-CodeDeploy-Application",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "CodePipelineDemoApplication",
                            "DeploymentGroupName": "CodePipelineDemoFleet"
                        },
                        "runOrder": 1
                    }
                ]
             }
    ]
}
```

3. Führen Sie den Befehl update-pipeline aus, um die Änderungen zu übernehmen. Geben Sie die Pipeline-JSON-Datei dabei folgendermaßen an:

> **Important**
>
> Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

```
aws codepipeline update-pipeline --cli-input-json file://pipeline.json
```

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

4. Öffnen Sie die CodePipeline Konsole und wählen Sie den Namen der Pipeline, die Sie gerade bearbeitet haben.

   Die Pipeline zeigt die Änderungen an. Wenn Sie das nächste Mal eine Änderung am Quellspeicherort durchführen, führt die Pipeline die Revision über die überarbeitete Struktur der Pipeline aus.

Kennzeichnen Sie eine benutzerdefinierte Aktion in CodePipeline

Tags sind Schlüssel-Wert-Paare, die Ressourcen zugeordnet AWS sind. Sie können die Konsole oder die CLI verwenden, um Tags auf Ihre benutzerdefinierten Aktionen in anzuwenden CodePipeline. Informationen zu CodePipeline Ressourcen-Tagging, Anwendungsfällen, Einschränkungen für Tag-Schlüssel und -Werte und unterstützte Ressourcentypen finden Sie unterMarkieren von Ressourcen.

Sie können die Werte der Tags in einer benutzerdefinierten Aktionen hinzufügen, entfernen und aktualisieren. Sie können jeder benutzerdefinierten Aktion bis zu 50 Tags hinzufügen.

Themen

- Hinzufügen von Tags zu benutzerdefinierten Aktionen
- Anzeigen von Tags für eine benutzerdefinierte Aktion
- Bearbeiten von Tags für eine benutzerdefinierte Aktion
- Entfernen von Tags aus einer benutzerdefinierten Aktion

Hinzufügen von Tags zu benutzerdefinierten Aktionen

Gehen Sie wie folgt vor AWS CLI , um einer benutzerdefinierten Aktion ein Tag hinzuzufügen. Informationen zum Hinzufügen eines Tags zu einer benutzerdefinierten Aktion während der Erstellung finden Sie unter Erstellen und fügen Sie eine benutzerdefinierte Aktion hinzu in CodePipeline.

Bei diesen Schritten wird davon ausgegangen, dass Sie bereits eine aktuelle Version der AWS CLI installiert oder eine Aktualisierung auf die aktuelle Version vorgenommen haben. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface.

Führen Sie im Terminal oder in der Befehlszeile den Befehl tag-resource aus und geben Sie dabei den Amazon-Ressourcennamen (ARN) der benutzerdefinierten Aktion, der Sie Tags hinzufügen möchten, sowie den Schlüssel und den Wert des Tags an, das Sie hinzufügen möchten. Sie können einer benutzerdefinierten Aktion mehr als ein Tag hinzufügen. Um beispielsweise eine benutzerdefinierte Aktion mit zwei Tags zu taggen, einem Tag-Schlüssel *TestActionType*mit dem Tag-Wert von *UnitTest*und einem Tag-Schlüssel *ApplicationName*mit dem Tag-Wert von *MyApplication*:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-
id:actiontype:Owner/Category/Provider/Version --tags key=TestActionType,value=UnitTest
 key=ApplicationName,value=MyApplication
```

Bei erfolgreicher Ausführung gibt dieser Befehl nichts zurück.

Anzeigen von Tags für eine benutzerdefinierte Aktion

Gehen Sie wie folgt vor AWS CLI , um die AWS Tags für eine benutzerdefinierte Aktion anzuzeigen. Wenn keine Tags hinzugefügt wurden, ist die zurückgegebene Liste leer.

Führen Sie am Terminal oder über die Befehlszeile den Befehl list-tags-for-resource aus. Um beispielsweise eine Liste der Tag-Schlüssel und Tag-Werte für eine benutzerdefinierte Aktion mit dem ARN `arn:aws:codepipeline:`*us-west-2*`:`*account-id*`:actiontype:`*Owner*`/`*Category*`/`*Provider*`/`*Version* anzuzeigen:

```
aws codepipeline list-tags-for-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version
```

Bei erfolgreicher Ausführung gibt dieser Befehl etwa wie folgt aussehende Informationen zurück:

```
{
    "tags": {
        "TestActionType": "UnitTest",
        "ApplicationName": "MyApplication"
    }
}
```

Bearbeiten von Tags für eine benutzerdefinierte Aktion

Gehen Sie wie folgt vor, AWS CLI um mit dem ein Tag für eine benutzerdefinierte Aktion zu bearbeiten. Sie können den Wert für einen vorhandenen Schlüssel ändern oder einen anderen Schlüssel hinzufügen. Sie können Tags auch aus einer benutzerdefinierten Aktion entfernen wie im nächsten Abschnitt gezeigt.

Führen Sie im Terminal oder in der Befehlszeile den Befehl tag-resource aus und geben Sie dabei den Amazon-Ressourcennamen (ARN) der benutzerdefinierten Aktion, für die Sie ein Tag aktualisieren möchten, sowie den Schlüssel und den Wert des Tags an, das Sie aktualisieren möchten:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version --tags
 key=TestActionType,value=IntegrationTest
```

Entfernen von Tags aus einer benutzerdefinierten Aktion

Gehen Sie wie folgt vor AWS CLI , um ein Tag aus einer benutzerdefinierten Aktion zu entfernen. Wenn Sie Tags aus der zugehörigen Ressource entfernen, werden die Tags gelöscht.

> **Note**
>
> Wenn Sie eine benutzerdefinierte Aktion löschen, werden alle Tag-Zuordnungen aus der gelöschten benutzerdefinierten Aktion entfernt. Sie müssen Tags nicht entfernen, bevor Sie eine benutzerdefinierte Aktion löschen.

Führen Sie im Terminal oder in der Befehlszeile den Befehl untag-resource aus und geben Sie dabei den Amazon-Ressourcennamen (ARN) der benutzerdefinierten Aktion, aus der Sie ein Tag entfernen möchten, sowie den Schlüssel und den Wert des Tags an, das Sie entfernen möchten:

Um beispielsweise ein Tag aus einer benutzerdefinierten Aktion mit dem Tag-Schlüssel zu entfernen *TestActionType*:

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:actiontype:Owner/Category/Provider/Version --tag-keys TestActionType
```

Bei erfolgreicher Ausführung gibt dieser Befehl nichts zurück. Um die der benutzerdefinierten Aktion zugeordneten Tags zu überprüfen, führen Sie den Befehl list-tags-for-resource aus.

Rufen Sie eine AWS Lambda Funktion in einer Pipeline auf in CodePipeline

AWS Lambda ist ein Datenverarbeitungsservice, mit dem Sie Code ausführen können, ohne Server bereitstellen oder verwalten zu müssen. Sie können Lambda-Funktionen erstellen und sie als Aktionen zu Ihren Pipelines hinzufügen. Da Sie mit Lambda Funktionen für fast jede Aufgabe schreiben können, können Sie die Funktionsweise Ihrer Pipeline anpassen.

> **Important**
>
> Protokollieren Sie nicht das JSON-Ereignis, das CodePipeline an Lambda gesendet wird, da dies dazu führen kann, dass Benutzeranmeldeinformationen in CloudWatch Logs protokolliert werden. Die CodePipeline Rolle verwendet ein JSON-Ereignis, um temporäre Anmeldeinformationen im `artifactCredentials` Feld an Lambda zu übergeben. Ein Beispiel für ein Ereignis finden Sie unter JSON-Beispielereignis.

Hier sind einige Möglichkeiten, wie Lambda-Funktionen in Pipelines verwendet werden können:

- Um Ressourcen nach Bedarf in einer Phase einer Pipeline zu erstellen AWS CloudFormation und sie in einer anderen Phase zu löschen.
- Um Anwendungsversionen ohne Ausfallzeiten AWS Elastic Beanstalk mit einer Lambda-Funktion bereitzustellen, die CNAME-Werte tauscht.
- Zur Bereitstellung auf Amazon ECS-Docker-Instances.
- Sichern von Ressourcen mithilfe eines AMI-Snapshots vor dem Erstellen oder Bereitstellen
- Integrieren von Drittanbieterprodukten in Ihre Pipeline, wie beispielsweise das Versenden von Nachrichten an einen IRC-Client

> **Note**
>
> Das Erstellen und Ausführen von Lambda-Funktionen kann zu Gebühren für Ihr AWS Konto führen. Weitere Informationen finden Sie unter -Preisgestaltung.

In diesem Thema wird vorausgesetzt, dass Sie mit Pipelines AWS CodePipeline AWS Lambda und Funktionen sowie den IAM-Richtlinien und -Rollen, von denen sie abhängen, vertraut sind und wissen, wie diese erstellt werden. In diesem Thema wird Folgendes veranschaulicht:

- Erstellen Sie eine Lambda-Funktion, die testet, ob eine Webseite erfolgreich bereitgestellt wurde.
- Konfigurieren Sie die Ausführungsrollen CodePipeline und die Lambda-Ausführungsrollen sowie die Berechtigungen, die für die Ausführung der Funktion als Teil der Pipeline erforderlich sind.
- Bearbeiten Sie eine Pipeline, um die Lambda-Funktion als Aktion hinzuzufügen.
- Testen der Aktion durch manuelles Veröffentlichen einer Änderung

> **Note**
>
> Wenn Sie die regionsübergreifende Lambda-Aufrufaktion in verwenden CodePipeline, PutJobFailureResultsollte der Status der Lambda-Ausführung mithilfe von PutJobSuccessResultund an die Region gesendet werden, in der die Lambda-Funktion vorhanden ist, und nicht an die AWS Region, in der sie existiert. CodePipeline

Dieses Thema enthält Beispielfunktionen, um die Flexibilität der Arbeit mit Lambda-Funktionen in CodePipeline folgenden Bereichen zu demonstrieren:

- Basic Lambda function
  - Erstellen einer grundlegenden Lambda-Funktion zur Verwendung mit CodePipeline.
  - Die Rückgabe von Erfolgs- oder Fehlschlagsergebnissen CodePipeline erfolgt über den Link „Details“ für die Aktion.
- Beispiel für eine Python-Funktion, die eine AWS CloudFormation Vorlage verwendet
  - Verwenden von JSON-verschlüsselten Benutzerparametern für die Weiterleitung mehrerer Konfigurationswerte an die Funktion (`get_user_params`)
  - Interaktion mit ZIP-Artefakten in einem Artefakt-Bucket (`get_template`)

- Verwenden eines Fortsetzungstokens, um einen lange andauernden asynchronen Prozess zu überwachen (`continue_job_later`). Dadurch kann die Aktion fortgesetzt und die Funktion erfolgreich ausgeführt werden, auch wenn sie eine Laufzeit von fünfzehn Minuten überschreitet (ein Limit in Lambda).

Jede Beispielfunktion beinhaltet Informationen zu den Berechtigungen, die Sie der Rolle hinzufügen müssen. Informationen zu Grenzwerten in finden Sie unter Grenzwerte im AWS Lambda Entwicklerhandbuch.AWS Lambda

> **Important**
>
> Beispiel-Code, Rollen und Richtlinien, die in diesem Thema enthalten sind, dienen lediglich als Beispiele und werden unverändert bereitgestellt.

Themen

Schritt 1: Erstellen einer Pipeline

In diesem Schritt erstellen Sie eine Pipeline, zu der Sie später die Lambda-Funktion hinzufügen. Das ist dieselbe Pipeline, die Sie unter CodePipeline Anleitungen erstellt haben. Wenn diese Pipeline noch für Ihr Konto konfiguriert ist und sich in derselben Region befindet, in der Sie die Lambda-Funktion erstellen möchten, können Sie diesen Schritt überspringen.

So erstellen Sie die Pipeline

1. Folgen Sie den ersten drei Schritten unterTutorial: Erstellen einer einfachen Pipeline (S3-Bucket), um einen Amazon S3 S3-Bucket, CodeDeploy Ressourcen und eine zweistufige

Pipeline zu erstellen. Wählen Sie die Amazon Linux-Option für Ihre Instance-Typen. Sie können einen beliebigen Namen für die Pipeline verwenden, aber die Schritte in diesem Thema verwenden MyLambdaTestPipeline.

2. Wählen Sie auf der Statusseite für Ihre Pipeline in der CodeDeploy Aktion die Option Details aus. Wählen Sie auf der Seite zu den Details der Bereitstellungsgruppe eine Instance-ID aus der Liste aus.
3. Kopieren Sie in der Amazon EC2 EC2-Konsole auf der Registerkarte Details für die Instance die IP-Adresse in Öffentliche IPv4-Adresse (z. B.**`192.0.2.4`**). Sie verwenden diese Adresse als Ziel der Funktion in AWS Lambda.

> **Note**
>
> Die standardmäßige Servicerollenrichtlinie für CodePipeline beinhaltet die Lambda-Berechtigungen, die zum Aufrufen der Funktion erforderlich sind. Sollten Sie jedoch die standardmäßige Servicerolle geändert oder eine andere Rolle ausgewählt haben, stellen Sie sicher, dass die Richtlinie für die Rolle die Berechtigungen für `lambda:InvokeFunction` und `lambda:ListFunctions` zulässt. Andernfalls schlagen Pipelines fehl, die Lambda-Aktionen enthalten.

Schritt 2: Lambda-Funktion erstellen

In diesem Schritt erstellen Sie eine Lambda-Funktion, die eine HTTP-Anfrage stellt und nach einer Textzeile auf einer Webseite sucht. Im Rahmen dieses Schritts müssen Sie auch eine IAM-Richtlinie und eine Lambda-Ausführungsrolle erstellen. Weitere Informationen finden Sie im Berechtigungsmodell im AWS Lambda -Entwicklerhandbuch.

So erstellen Sie eine Ausführungsrolle

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies aus und wählen Sie dann Create Policy aus. Wählen Sie die Registerkarte JSON aus und kopieren Sie dann die folgende JSON-Richtlinie in das Feld.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Action": [
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Action": [
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
        ],
        "Effect": "Allow",
        "Resource": "*"
     }
  ]
}
```

3. Wählen Sie Richtlinie prüfen.
4. Geben Sie auf der Seite Review policy (Richtlinie prüfen) unter Name einen Namen für die Richtlinie ein (z. B. **`CodePipelineLambdaExecPolicy`**). Geben Sie unter Description (Beschreibung) Folgendes ein: **`Enables Lambda to execute code`**.

   Wählen Sie Richtlinie erstellen aus.

   > **Note**
   >
   > Dies sind die Mindestberechtigungen, die für die Zusammenarbeit einer Lambda-Funktion mit CodePipeline Amazon erforderlich sind. CloudWatch Wenn Sie diese Richtlinie erweitern möchten, um Funktionen zuzulassen, die mit anderen AWS Ressourcen interagieren, sollten Sie diese Richtlinie ändern, um die für diese Lambda-Funktionen erforderlichen Aktionen zuzulassen.

5. Wählen Sie auf der Seite mit dem Richtlinien-Dashboard Roles (Rollen) aus und wählen Sie dann Create role (Rolle erstellen) aus.
6. Wählen Sie auf der Seite Rolle erstellen die Option AWS-Service. Wählen Sie Lambda und klicken Sie dann auf Next: Permissions (Weiter: Berechtigungen).
7. Aktivieren Sie auf der Seite „Berechtigungsrichtlinien anhängen“ das Kontrollkästchen neben CodePipelineLambdaExecPolicyund wählen Sie dann Weiter: Tags aus. Wählen Sie Weiter: Prüfen aus.

8. Geben Sie auf der Seite Review (Überprüfen) unter Role name (Rollenname) den Namen ein. Klicken Sie dann auf Create role (Rolle erstellen).

Um die Lambda-Beispielfunktion zu erstellen, die mit verwendet werden soll CodePipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
2. Klicken Sie auf der Seite Functions (Funktionen) auf Create function (Funktion erstellen).

   > **Note**
   >
   > Wenn Sie statt der Lambda-Seite eine Willkommensseite sehen, wählen Sie Jetzt loslegen.

3. Wählen Sie auf der Seite Create function die Option Author from scratch. Geben Sie im Feld Funktionsname einen Namen für Ihre Lambda-Funktion ein (z. B.**`MyLambdaFunctionForAWSCodePipeline`**). Wählen Sie in Runtime Node.js 20.x aus.
4. Wählen Sie für Role (Rolle) die Option Choose an existing role (Eine vorhandene Rolle wählen) aus. Wählen Sie Ihre Rolle unter Existing role (Vorhandene Rolle) aus und klicken Sie dann auf Create function (Funktion erstellen).

   Die Detailseite für Ihre erstellte Funktion wird geöffnet.
5. Fügen Sie den folgenden Code in das Feld Function code (Funktionscode) ein:

   > **Note**
   >
   > Das Ereignisobjekt unter dem Schlüssel CodePipeline .job enthält die Auftragsdetails. Ein vollständiges Beispiel für die CodePipeline Rückkehr des JSON-Ereignisses zu Lambda finden Sie unterJSON-Beispielereignis.

```
import { CodePipelineClient, PutJobSuccessResultCommand,
 PutJobFailureResultCommand } from "@aws-sdk/client-codepipeline";
import http from 'http';
import assert from 'assert';

export const handler = (event, context) => {
```

```
   const codepipeline = new CodePipelineClient();

   // Retrieve the Job ID from the Lambda action
   const jobId = event["CodePipeline.job"].id;

   // Retrieve the value of UserParameters from the Lambda action configuration in
CodePipeline, in this case a URL which will be
   // health checked by this function.
   const url =
event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;

   // Notify CodePipeline of a successful job
   const putJobSuccess = async function(message) {
       const command = new PutJobSuccessResultCommand({
           jobId: jobId
       });
        try {
            await codepipeline.send(command);
            context.succeed(message);
        } catch (err) {
            context.fail(err);
        }
   };

   // Notify CodePipeline of a failed job
   const putJobFailure = async function(message) {
       const command = new PutJobFailureResultCommand({
           jobId: jobId,
           failureDetails: {
               message: JSON.stringify(message),
               type: 'JobFailed',
               externalExecutionId: context.awsRequestId
           }
       });
       await codepipeline.send(command);
       context.fail(message);
   };

   // Validate the URL passed in UserParameters
   if(!url || url.indexOf('http://') === -1) {
       putJobFailure('The UserParameters field must contain a valid URL address to
test, including http:// or https://');
       return;
```

```
    }

    // Helper function to make a HTTP GET request to the page.
    // The helper will test the response and succeed or fail the job accordingly
    const getPage = function(url, callback) {
        var pageObject = {
            body: '',
            statusCode: 0,
            contains: function(search) {
                return this.body.indexOf(search) > -1;
            }
        };
        http.get(url, function(response) {
            pageObject.body = '';
            pageObject.statusCode = response.statusCode;

            response.on('data', function (chunk) {
                pageObject.body += chunk;
            });

            response.on('end', function () {
                callback(pageObject);
            });

            response.resume();
        }).on('error', function(error) {
            // Fail the job if our request failed
            putJobFailure(error);
        });
    };

    getPage(url, function(returnedPage) {
        try {
            // Check if the HTTP response has a 200 status
            assert(returnedPage.statusCode === 200);
            // Check if the page contains the text "Congratulations"
            // You can change this to check for different text, or add other tests
as required
            assert(returnedPage.contains('Congratulations'));

            // Succeed the job
            putJobSuccess("Tests passed.");
        } catch (ex) {
            // If any of the assertions failed then fail the job
```

```
            putJobFailure(ex);
        }
    });
};
```

6. Behalten Sie für Handler den Standardwert und für Role (Rolle) die Standardrolle **`CodePipelineLambdaExecRole`** bei.
7. Geben Sie unter Basic settings (Grundlegende Einstellungen) für Timeout den Wert **`20`** Sekunden ein.
8. Wählen Sie Speichern.

Schritt 3: Fügen Sie die Lambda-Funktion zu einer Pipeline in der CodePipeline Konsole hinzu

In diesem Schritt fügen Sie Ihrer Pipeline eine neue Phase hinzu und fügen dann eine Lambda-Aktion hinzu, die Ihre Funktion zu dieser Phase aufruft.

So fügen Sie eine Stufe hinzu

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Welcome (Willkommen) die von Ihnen erstellte Pipeline aus.
3. Wählen Sie auf der Pipeline-Seite Edit aus.
4. Wählen Sie auf der Seite Bearbeiten die Option + Phase hinzufügen aus, um nach der Bereitstellungsphase mit der CodeDeploy Aktion eine Phase hinzuzufügen. Geben Sie einen Namen für die Stufe ein (z. B. **`LambdaStage`**) und wählen Sie Add stage (Stufe hinzufügen).

   > **Note**
   >
   > Sie können sich auch dafür entscheiden, Ihre Lambda-Aktion zu einer vorhandenen Phase hinzuzufügen. Zu Demonstrationszwecken fügen wir die Lambda-Funktion als einzige Aktion in einer Phase hinzu, damit Sie ihren Fortschritt leicht verfolgen können, während Artefakte eine Pipeline durchlaufen.

5. Wählen Sie + Add action group (Aktionsgruppe hinzufügen). Geben Sie unter Aktion bearbeiten unter Aktionsname einen Namen für Ihre Lambda-Aktion ein (z. B.**`MyLambdaAction`**). Wählen Sie in Provider (Anbieter) AWS Lambda aus. Wählen Sie

unter Funktionsname den Namen Ihrer Lambda-Funktion aus, oder geben Sie ihn ein (z. B.**`MyLambdaFunctionForAWSCodePipeline`**). Geben Sie unter Benutzerparameter die IP-Adresse für die Amazon EC2 EC2-Instance an, die Sie zuvor kopiert haben (z. B.**`http://192.0.2.4`**), und wählen Sie dann Fertig.

> **Note**
>
> Dieses Thema verwendet eine IP-Adresse. In echten Szenarien sollten Sie jedoch Ihren registrierten Website-Namen verwenden (z. B. **`http://www.example.com`**). Weitere Informationen zu Ereignisdaten und Handlern finden Sie unter Programmiermodell im AWS Lambda Entwicklerhandbuch. AWS Lambda

6. Wählen Sie auf der Seite Edit action (Aktion bearbeiten) die Option Save (Speichern).

Schritt 4: Testen Sie die Pipeline mit der Lambda-Funktion

Um die Funktion zu testen, führen Sie die aktuelleste Änderung über die Pipeline aus.

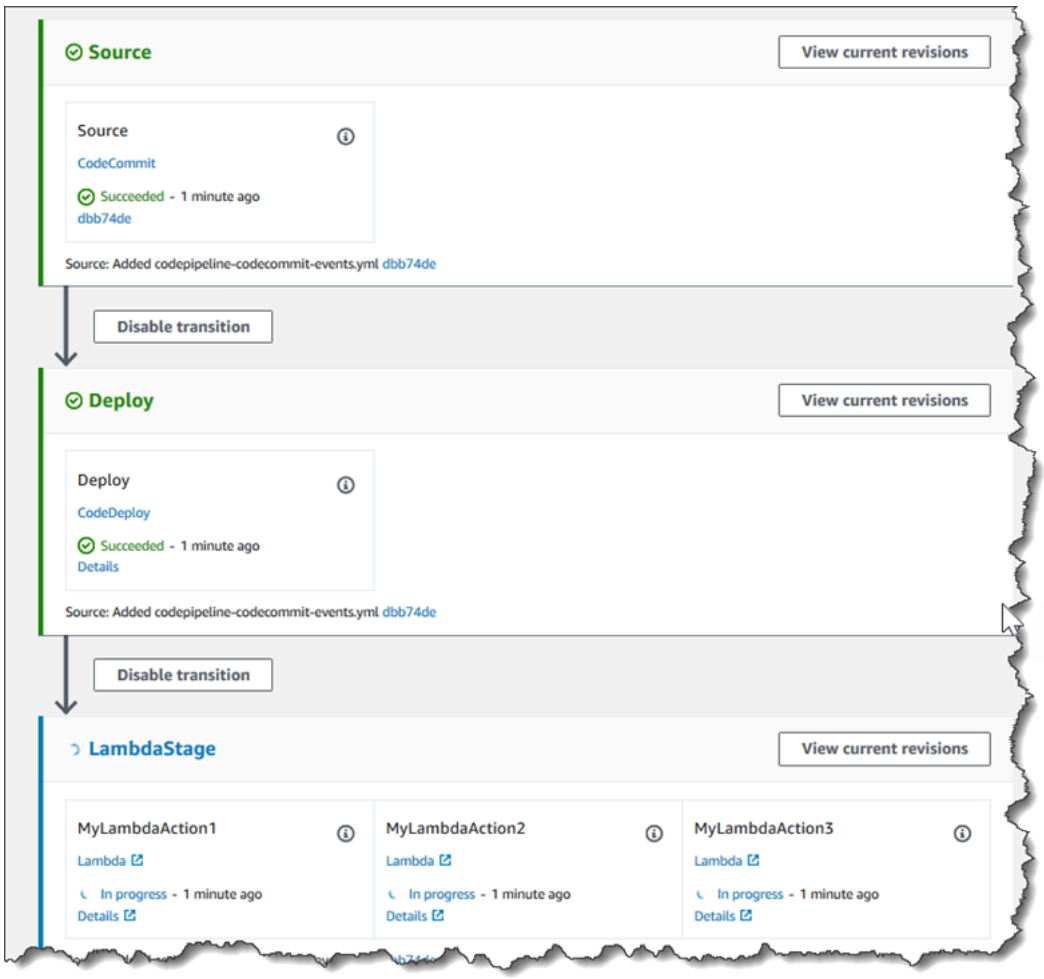
So verwenden Sie die Konsole zum Ausführen der letzten Version eines Artefaktes in einer Pipeline

1. Wählen Sie auf der Seite mit den Pipeline-Details die Option Änderung veröffentlichen aus. Dadurch wird die letzte Revision gestartet, die in jedem in einer Quellaktion der Pipeline angegebenen Quellspeicherort vorhanden ist.
2. Wenn die Lambda-Aktion abgeschlossen ist, wählen Sie den Link Details, um den Protokollstream für die Funktion in Amazon anzuzeigen CloudWatch, einschließlich der abgerechneten Dauer des Ereignisses. Wenn die Funktion fehlgeschlagen ist, enthält das CloudWatch Protokoll Informationen über die Ursache.

Schritt 5: Nächste Schritte

Nachdem Sie erfolgreich eine Lambda-Funktion erstellt und als Aktion in einer Pipeline hinzugefügt haben, können Sie Folgendes versuchen:

- Fügen Sie Ihrer Phase weitere Lambda-Aktionen hinzu, um andere Webseiten zu überprüfen.
- Ändern Sie die Lambda-Funktion, um nach einer anderen Textzeichenfolge zu suchen.
- Erkunden Sie Lambda-Funktionen und erstellen Sie Ihre eigenen Lambda-Funktionen und fügen Sie sie zu Pipelines hinzu.



Nachdem Sie mit dem Experimentieren mit der Lambda-Funktion fertig sind, sollten Sie erwägen, sie aus Ihrer Pipeline zu entfernen, sie aus IAM zu löschen und die Rolle aus IAM zu löschen AWS Lambda, um mögliche Gebühren zu vermeiden. Weitere Informationen finden Sie unter Eine Pipeline bearbeiten in CodePipeline, Löschen Sie eine Pipeline in CodePipeline und Löschen von Rollen oder Instance-Profilen.

JSON-Beispielereignis

Das folgende Beispiel zeigt ein JSON-Beispielereignis, das von CodePipeline an Lambda gesendet wurde. Die Struktur dieses Ereignisses ähnelt der Antwort unter `GetJobDetails API`, aber ohne den `actionTypeId`- und `pipelineContext`-Datentyp. Zwei Aktionskonfigurationsdetails, `FunctionName` und `UserParameters`, sind sowohl im JSON-Ereignis als auch in der Antwort auf die `GetJobDetails`-API enthalten. Die Werte in *`roter Kursivschrift`* sind Beispiele oder Erklärungen, keine realen Werte.

```
{
```

```
    "CodePipeline.job": {
        "id": "11111111-abcd-1111-abcd-111111abcdef",
        "accountId": "111111111111",
        "data": {
            "actionConfiguration": {
                "configuration": {
                    "FunctionName": "MyLambdaFunctionForAWSCodePipeline",
                    "UserParameters": "some-input-such-as-a-URL"
                }
            },
            "inputArtifacts": [
                {
                    "location": {
                        "s3Location": {
                            "bucketName": "the name of the bucket configured as the
pipeline artifact store in Amazon S3, for example codepipeline-us-east-2-1234567890",
                            "objectKey": "the name of the application, for example
CodePipelineDemoApplication.zip"
                        },
                        "type": "S3"
                    },
                    "revision": null,
                    "name": "ArtifactName"
                }
            ],
            "outputArtifacts": [],
            "artifactCredentials": {
                "secretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
                "sessionToken": "MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
3rrszlaEXAMPLE=",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
```

```
            },
            "continuationToken": "A continuation token if continuing job",
            "encryptionKey": {
              "id": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
              "type": "KMS"
            }
        }
    }
}
```

Weitere Beispielfunktionen

Die folgenden Lambda-Beispielfunktionen demonstrieren zusätzliche Funktionen, die Sie für Ihre Pipelines in verwenden können. CodePipeline Um diese Funktionen verwenden zu können, müssen Sie möglicherweise die Richtlinie für die Lambda-Ausführungsrolle ändern, wie in der Einführung für jedes Beispiel angegeben.

Themen

- Beispiel für eine Python-Funktion, die eine AWS CloudFormation Vorlage verwendet

Beispiel für eine Python-Funktion, die eine AWS CloudFormation Vorlage verwendet

Das folgende Beispiel zeigt eine Funktion, die einen Stack auf der Grundlage einer bereitgestellten AWS CloudFormation Vorlage erstellt oder aktualisiert. Die Vorlage erstellt einen Amazon S3 S3-Bucket. Dies erfolgt nur zu Demonstrationszwecken, um Kosten zu sparen. Im Idealfall sollten Sie den Stack löschen, bevor Sie Inhalte in den Bucket hochladen. Falls Sie Dateien in den Bucket hochladen, können Sie den Bucket nicht mehr löschen, wenn Sie den Stack löschen. In diesem Fall müssen Sie alle Inhalte des Buckets manuell löschen, bevor Sie den Bucket selbst löschen können.

In diesem Python-Beispiel wird davon ausgegangen, dass Sie über eine Pipeline verfügen, die einen Amazon S3 S3-Bucket als Quellaktion verwendet, oder dass Sie Zugriff auf einen versionierten Amazon S3 S3-Bucket haben, den Sie mit der Pipeline verwenden können. Sie erstellen die AWS CloudFormation Vorlage, komprimieren sie und laden sie als ZIP-Datei in diesen Bucket hoch. Anschließend müssen Sie Ihrer Pipeline eine Quellaktion hinzufügen, welche die ZIP-Datei aus dem Bucket abruft.

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, können Sie Ihre Quelldatei (en) in eine einzige ZIP-Datei komprimieren und die ZIP-Datei in Ihren Quell-Bucket hochladen. Sie können auch eine einzelne Datei ungezippt hochladen, aber nachgelagerte Aktionen, die eine ZIP-Datei erwarten, schlagen dann fehl.

In diesem Beispiel wird Folgendes gezeigt:

- Die Verwendung von JSON-verschlüsselten Benutzerparametern für die Weiterleitung mehrerer Konfigurationswerte an die Funktion (`get_user_params`)
- Die Interaktion mit ZIP-Artefakten in einem Artefakt-Bucket (`get_template`)
- Die Verwendung eines Fortsetzungstokens, um einen lange andauernden asynchronen Prozess zu überwachen (`continue_job_later`). Dadurch kann die Aktion fortgesetzt und die Funktion erfolgreich ausgeführt werden, auch wenn sie eine Laufzeit von fünfzehn Minuten überschreitet (ein Limit in Lambda).

Um diese Lambda-Beispielfunktion verwenden zu können, muss die Richtlinie für die Lambda-Ausführungsrolle über `Allow` Berechtigungen in Amazon S3 verfügen und AWS CloudFormation, wie in dieser Beispielrichtlinie gezeigt CodePipeline, folgende Berechtigungen haben:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Action": [
                "codepipeline:PutJobSuccessResult",
                "codepipeline:PutJobFailureResult"
            ],
            "Effect": "Allow",
            "Resource": "*"
```

```
        },
        {
            "Action": [
                "cloudformation:DescribeStacks",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Um die AWS CloudFormation Vorlage zu erstellen, öffnen Sie einen beliebigen Klartext-Editor und kopieren Sie den folgenden Code und fügen Sie ihn ein:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "CloudFormation template which creates an S3 bucket",
  "Resources" : {
    "MySampleBucket" : {
      "Type" : "AWS::S3::Bucket",
      "Properties" : {
      }
    }
  },
  "Outputs" : {
    "BucketName" : {
      "Value" : { "Ref" : "MySampleBucket" },
      "Description" : "The name of the S3 bucket"
    }
  }
}
```

Speichern Sie dies als JSON-Datei mit dem Namen **`template.json`** in einem Verzeichnis mit dem Namen **`template-package`**. Erstellen Sie eine komprimierte Datei (.zip) mit diesem Verzeichnis

und der Datei mit dem Namen **`template-package.zip`** und laden Sie die komprimierte Datei in einen versionierten Amazon S3 S3-Bucket hoch. Wenn Sie bereits einen Bucket für Ihre Pipeline konfiguriert haben, können Sie diesen verwenden. Bearbeiten Sie als Nächstes Ihre Pipeline und fügen Sie eine Quellaktion hinzu, welche die ZIP-Datei abruft. Benennen Sie die Ausgabe für diese Aktion. *`MyTemplate`* Weitere Informationen finden Sie unter Eine Pipeline bearbeiten in CodePipeline.

> **Note**
>
> Die Lambda-Beispielfunktion erwartet diese Dateinamen und die komprimierte Struktur. Sie können dieses Beispiel jedoch durch Ihre eigene AWS CloudFormation Vorlage ersetzen. Wenn Sie Ihre eigene Vorlage verwenden, stellen Sie sicher, dass Sie die Richtlinie für die Lambda-Ausführungsrolle ändern, um alle zusätzlichen Funktionen zuzulassen, die für Ihre AWS CloudFormation Vorlage erforderlich sind.

Um den folgenden Code als Funktion in Lambda hinzuzufügen

1. Öffnen Sie die Lambda-Konsole und wählen Sie Create function.
2. Wählen Sie auf der Seite Create function die Option Author from scratch. Geben Sie im Feld Funktionsname einen Namen für Ihre Lambda-Funktion ein.
3. Wählen Sie unter Runtime (Laufzeit) die Option Python 2.7 aus.
4. Wählen Sie unter Ausführungsrolle auswählen oder erstellen die Option Vorhandene Rolle verwenden aus. Wählen Sie Ihre Rolle unter Existing role (Vorhandene Rolle) aus und klicken Sie dann auf Create function (Funktion erstellen).

   Die Detailseite für Ihre erstellte Funktion wird geöffnet.
5. Fügen Sie den folgenden Code in das Feld Function code (Funktionscode) ein:

```
from __future__ import print_function
from boto3.session import Session

import json
import urllib
import boto3
import zipfile
import tempfile
import botocore
import traceback
```

```
print('Loading function')

cf = boto3.client('cloudformation')
code_pipeline = boto3.client('codepipeline')

def find_artifact(artifacts, name):
    """Finds the artifact 'name' among the 'artifacts'

    Args:
        artifacts: The list of artifacts available to the function
        name: The artifact we wish to use
    Returns:
        The artifact dictionary found
    Raises:
        Exception: If no matching artifact is found

    """
    for artifact in artifacts:
        if artifact['name'] == name:
            return artifact

    raise Exception('Input artifact named "{0}" not found in event'.format(name))

def get_template(s3, artifact, file_in_zip):
    """Gets the template artifact

    Downloads the artifact from the S3 artifact store to a temporary file
    then extracts the zip and returns the file containing the CloudFormation
    template.

    Args:
        artifact: The artifact to download
        file_in_zip: The path to the file within the zip containing the template

    Returns:
        The CloudFormation template as a string

    Raises:
        Exception: Any exception thrown while downloading the artifact or unzipping
 it

    """
    tmp_file = tempfile.NamedTemporaryFile()
```

```
    bucket = artifact['location']['s3Location']['bucketName']
    key = artifact['location']['s3Location']['objectKey']

    with tempfile.NamedTemporaryFile() as tmp_file:
        s3.download_file(bucket, key, tmp_file.name)
        with zipfile.ZipFile(tmp_file.name, 'r') as zip:
            return zip.read(file_in_zip)

def update_stack(stack, template):
    """Start a CloudFormation stack update

    Args:
        stack: The stack to update
        template: The template to apply

    Returns:
        True if an update was started, false if there were no changes
        to the template since the last update.

    Raises:
        Exception: Any exception besides "No updates are to be performed."

    """
    try:
        cf.update_stack(StackName=stack, TemplateBody=template)
        return True

    except botocore.exceptions.ClientError as e:
        if e.response['Error']['Message'] == 'No updates are to be performed.':
            return False
        else:
            raise Exception('Error updating CloudFormation stack
 "{0}"'.format(stack), e)

def stack_exists(stack):
    """Check if a stack exists or not

    Args:
        stack: The stack to check

    Returns:
        True or False depending on whether the stack exists

    Raises:
```

```
        Any exceptions raised .describe_stacks() besides that
        the stack doesn't exist.

    """
    try:
        cf.describe_stacks(StackName=stack)
        return True
    except botocore.exceptions.ClientError as e:
        if "does not exist" in e.response['Error']['Message']:
            return False
        else:
            raise e

def create_stack(stack, template):
    """Starts a new CloudFormation stack creation

    Args:
        stack: The stack to be created
        template: The template for the stack to be created with

    Throws:
        Exception: Any exception thrown by .create_stack()
    """
    cf.create_stack(StackName=stack, TemplateBody=template)

def get_stack_status(stack):
    """Get the status of an existing CloudFormation stack

    Args:
        stack: The name of the stack to check

    Returns:
        The CloudFormation status string of the stack such as CREATE_COMPLETE

    Raises:
        Exception: Any exception thrown by .describe_stacks()

    """
    stack_description = cf.describe_stacks(StackName=stack)
    return stack_description['Stacks'][0]['StackStatus']

def put_job_success(job, message):
    """Notify CodePipeline of a successful job
```

```
    Args:
        job: The CodePipeline job ID
        message: A message to be logged relating to the job status

    Raises:
        Exception: Any exception thrown by .put_job_success_result()

    """
    print('Putting job success')
    print(message)
    code_pipeline.put_job_success_result(jobId=job)

def put_job_failure(job, message):
    """Notify CodePipeline of a failed job

    Args:
        job: The CodePipeline job ID
        message: A message to be logged relating to the job status

    Raises:
        Exception: Any exception thrown by .put_job_failure_result()

    """
    print('Putting job failure')
    print(message)
    code_pipeline.put_job_failure_result(jobId=job, failureDetails={'message':
 message, 'type': 'JobFailed'})

def continue_job_later(job, message):
    """Notify CodePipeline of a continuing job

    This will cause CodePipeline to invoke the function again with the
    supplied continuation token.

    Args:
        job: The JobID
        message: A message to be logged relating to the job status
        continuation_token: The continuation token

    Raises:
        Exception: Any exception thrown by .put_job_success_result()

    """
```

```
    # Use the continuation token to keep track of any job execution state
    # This data will be available when a new job is scheduled to continue the
 current execution
    continuation_token = json.dumps({'previous_job_id': job})

    print('Putting job continuation')
    print(message)
    code_pipeline.put_job_success_result(jobId=job,
 continuationToken=continuation_token)

def start_update_or_create(job_id, stack, template):
    """Starts the stack update or create process

    If the stack exists then update, otherwise create.

    Args:
        job_id: The ID of the CodePipeline job
        stack: The stack to create or update
        template: The template to create/update the stack with

    """
    if stack_exists(stack):
        status = get_stack_status(stack)
        if status not in ['CREATE_COMPLETE', 'ROLLBACK_COMPLETE',
 'UPDATE_COMPLETE']:
            # If the CloudFormation stack is not in a state where
            # it can be updated again then fail the job right away.
            put_job_failure(job_id, 'Stack cannot be updated when status is: ' +
 status)
            return

        were_updates = update_stack(stack, template)

        if were_updates:
            # If there were updates then continue the job so it can monitor
            # the progress of the update.
            continue_job_later(job_id, 'Stack update started')

        else:
            # If there were no updates then succeed the job immediately
            put_job_success(job_id, 'There were no stack updates')
    else:
        # If the stack doesn't already exist then create it instead
        # of updating it.
```

```
        create_stack(stack, template)
        # Continue the job so the pipeline will wait for the CloudFormation
        # stack to be created.
        continue_job_later(job_id, 'Stack create started')

def check_stack_update_status(job_id, stack):
    """Monitor an already-running CloudFormation update/create

    Succeeds, fails or continues the job depending on the stack status.

    Args:
        job_id: The CodePipeline job ID
        stack: The stack to monitor

    """
    status = get_stack_status(stack)
    if status in ['UPDATE_COMPLETE', 'CREATE_COMPLETE']:
        # If the update/create finished successfully then
        # succeed the job and don't continue.
        put_job_success(job_id, 'Stack update complete')

    elif status in ['UPDATE_IN_PROGRESS', 'UPDATE_ROLLBACK_IN_PROGRESS',
    'UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS', 'CREATE_IN_PROGRESS',
    'ROLLBACK_IN_PROGRESS', 'UPDATE_COMPLETE_CLEANUP_IN_PROGRESS']:
        # If the job isn't finished yet then continue it
        continue_job_later(job_id, 'Stack update still in progress')

    else:
        # If the Stack is a state which isn't "in progress" or "complete"
        # then the stack update/create has failed so end the job with
        # a failed result.
        put_job_failure(job_id, 'Update failed: ' + status)

def get_user_params(job_data):
    """Decodes the JSON user parameters and validates the required properties.

    Args:
        job_data: The job data structure containing the UserParameters string which
 should be a valid JSON structure

    Returns:
        The JSON parameters decoded as a dictionary.

    Raises:
```

```
        Exception: The JSON can't be decoded or a property is missing.

    """
    try:
        # Get the user parameters which contain the stack, artifact and file
 settings
        user_parameters = job_data['actionConfiguration']['configuration']
['UserParameters']
        decoded_parameters = json.loads(user_parameters)

    except Exception as e:
        # We're expecting the user parameters to be encoded as JSON
        # so we can pass multiple values. If the JSON can't be decoded
        # then fail the job with a helpful message.
        raise Exception('UserParameters could not be decoded as JSON')

    if 'stack' not in decoded_parameters:
        # Validate that the stack is provided, otherwise fail the job
        # with a helpful message.
        raise Exception('Your UserParameters JSON must include the stack name')

    if 'artifact' not in decoded_parameters:
        # Validate that the artifact name is provided, otherwise fail the job
        # with a helpful message.
        raise Exception('Your UserParameters JSON must include the artifact name')

    if 'file' not in decoded_parameters:
        # Validate that the template file is provided, otherwise fail the job
        # with a helpful message.
        raise Exception('Your UserParameters JSON must include the template file
 name')

    return decoded_parameters

def setup_s3_client(job_data):
    """Creates an S3 client

    Uses the credentials passed in the event by CodePipeline. These
    credentials can be used to access the artifact bucket.

    Args:
        job_data: The job data structure

    Returns:
```

```
        An S3 client with the appropriate credentials

    """
    key_id = job_data['artifactCredentials']['accessKeyId']
    key_secret = job_data['artifactCredentials']['secretAccessKey']
    session_token = job_data['artifactCredentials']['sessionToken']

    session = Session(aws_access_key_id=key_id,
        aws_secret_access_key=key_secret,
        aws_session_token=session_token)
    return session.client('s3',
 config=botocore.client.Config(signature_version='s3v4'))

def lambda_handler(event, context):
    """The Lambda function handler

    If a continuing job then checks the CloudFormation stack status
    and updates the job accordingly.

    If a new job then kick of an update or creation of the target
    CloudFormation stack.

    Args:
        event: The event passed by Lambda
        context: The context passed by Lambda

    """
    try:
        # Extract the Job ID
        job_id = event['CodePipeline.job']['id']

        # Extract the Job Data
        job_data = event['CodePipeline.job']['data']

        # Extract the params
        params = get_user_params(job_data)

        # Get the list of artifacts passed to the function
        artifacts = job_data['inputArtifacts']

        stack = params['stack']
        artifact = params['artifact']
        template_file = params['file']
```

```
        if 'continuationToken' in job_data:
            # If we're continuing then the create/update has already been triggered
            # we just need to check if it has finished.
            check_stack_update_status(job_id, stack)
        else:
            # Get the artifact details
            artifact_data = find_artifact(artifacts, artifact)
            # Get S3 client to access artifact with
            s3 = setup_s3_client(job_data)
            # Get the JSON template file out of the artifact
            template = get_template(s3, artifact_data, template_file)
            # Kick off a stack update or create
            start_update_or_create(job_id, stack, template)

    except Exception as e:
        # If any other exceptions which we didn't expect are raised
        # then fail the job and log the exception message.
        print('Function failed due to exception.')
        print(e)
        traceback.print_exc()
        put_job_failure(job_id, 'Function exception: ' + str(e))

    print('Function complete.')
    return "Complete."
```

6. Belassen Sie Handler auf dem Standardwert und belassen Sie Role bei dem Namen, den Sie zuvor ausgewählt oder erstellt haben**`CodePipelineLambdaExecRole`**.
7. Ersetzen Sie unter Basic settings (Grundlegende Einstellungen) im Feld Timeout den Standardwert von 3 Sekunden durch **`20`**.
8. Wählen Sie Speichern.
9. Bearbeiten Sie in der CodePipeline Konsole die Pipeline, um die Funktion als Aktion in einer Phase Ihrer Pipeline hinzuzufügen. Wählen Sie Bearbeiten für die Pipeline-Phase, die Sie ändern möchten, und wählen Sie Aktionsgruppe hinzufügen aus. Geben Sie auf der Seite Aktion bearbeiten im Feld Aktionsname einen Namen für Ihre Aktion ein. Wählen Sie unter Aktionsanbieter die Option Lambda aus.

   Wählen Sie unter Eingabeartefakte die Option aus`MyTemplate`. UserParametersIn müssen Sie eine JSON-Zeichenfolge mit drei Parametern angeben:

   - Stack name
   - AWS CloudFormation Vorlagenname und Pfad zur Datei

- Eingabeartefakt

Verwenden Sie geschweifte Klammern ({}) und trennen Sie die Parameter durch Kommas. Um beispielsweise einen Stack mit dem Namen *MyTestStack*für eine Pipeline mit dem Eingabeartefakt in zu erstellen *MyTemplate*, geben Sie Folgendes ein: {"stack“:“ *MyTestStack*„UserParameters, "file“ :"template-package/template.json“, "artifact“:“ „}. *MyTemplate*

> **Note**
>
> Obwohl Sie das Eingabeartefakt in angegeben haben, müssen Sie dieses Eingabeartefakt auch für die Aktion unter UserParametersEingabeartefakte angeben.

10. Speichern Sie Ihre Änderungen an der Pipeline und geben Sie dann manuell eine Änderung frei, um die Aktion und die Lambda-Funktion zu testen.

Eine fehlgeschlagene Aktion in einer Phase wiederholen

Sie können eine fehlgeschlagene Phase erneut versuchen, ohne eine Pipeline erneut von Anfang an ausführen zu müssen. Sie tun dies, indem Sie entweder die fehlgeschlagenen Aktionen in einer Phase erneut versuchen oder indem Sie alle Aktionen in der Phase wiederholen, beginnend mit der ersten Aktion in der Phase. Wenn Sie die fehlgeschlagenen Aktionen in einer Phase wiederholen, funktionieren alle Aktionen, die noch ausgeführt werden, weiterhin, und fehlgeschlagene Aktionen werden erneut ausgelöst. Wenn Sie eine fehlgeschlagene Phase von der ersten Aktion in der Phase wiederholen, können in der Phase keine Aktionen ausgeführt werden. Bevor eine Phase erneut versucht werden kann, müssen entweder alle Aktionen fehlgeschlagen sein oder einige Aktionen sind fehlgeschlagen und andere erfolgreich.

> **Important**
>
> Beim erneuten Versuch einer fehlgeschlagenen Phase werden alle Aktionen der Phase von der ersten Aktion in der Phase wiederholt, und bei einem erneuten Versuch, fehlgeschlagene Aktionen durchzuführen, werden alle fehlgeschlagenen Aktionen in der Phase wiederholt. Dadurch werden Ausgabeartefakte zuvor erfolgreicher Aktionen in derselben Ausführung außer Kraft gesetzt.

Artefakte können zwar überschrieben werden, der Ausführungsverlauf zuvor erfolgreicher Aktionen wird jedoch beibehalten.

Wenn Sie die Konsole verwenden, um eine Pipeline anzuzeigen, erscheint entweder die Schaltfläche Phase wiederholen oder Fehlgeschlagene Aktionen wiederholen auf der Stufe, die erneut versucht werden kann.

Wenn Sie die AWS CLI verwenden, können Sie den get-pipeline-state Befehl verwenden, um festzustellen, ob Aktionen fehlgeschlagen sind.

Note

In den folgenden Fällen können Sie eine Phase möglicherweise nicht erneut versuchen:

- Alle Aktionen in der Phase waren erfolgreich, sodass die Phase nicht den Status Fehlgeschlagen hat.
- Die gesamte Pipeline-Struktur hat sich geändert, nachdem die Phase gescheitert war.
- Es wird bereits ein anderer Wiederholungsversuch in der Phase ausgeführt.

Themen

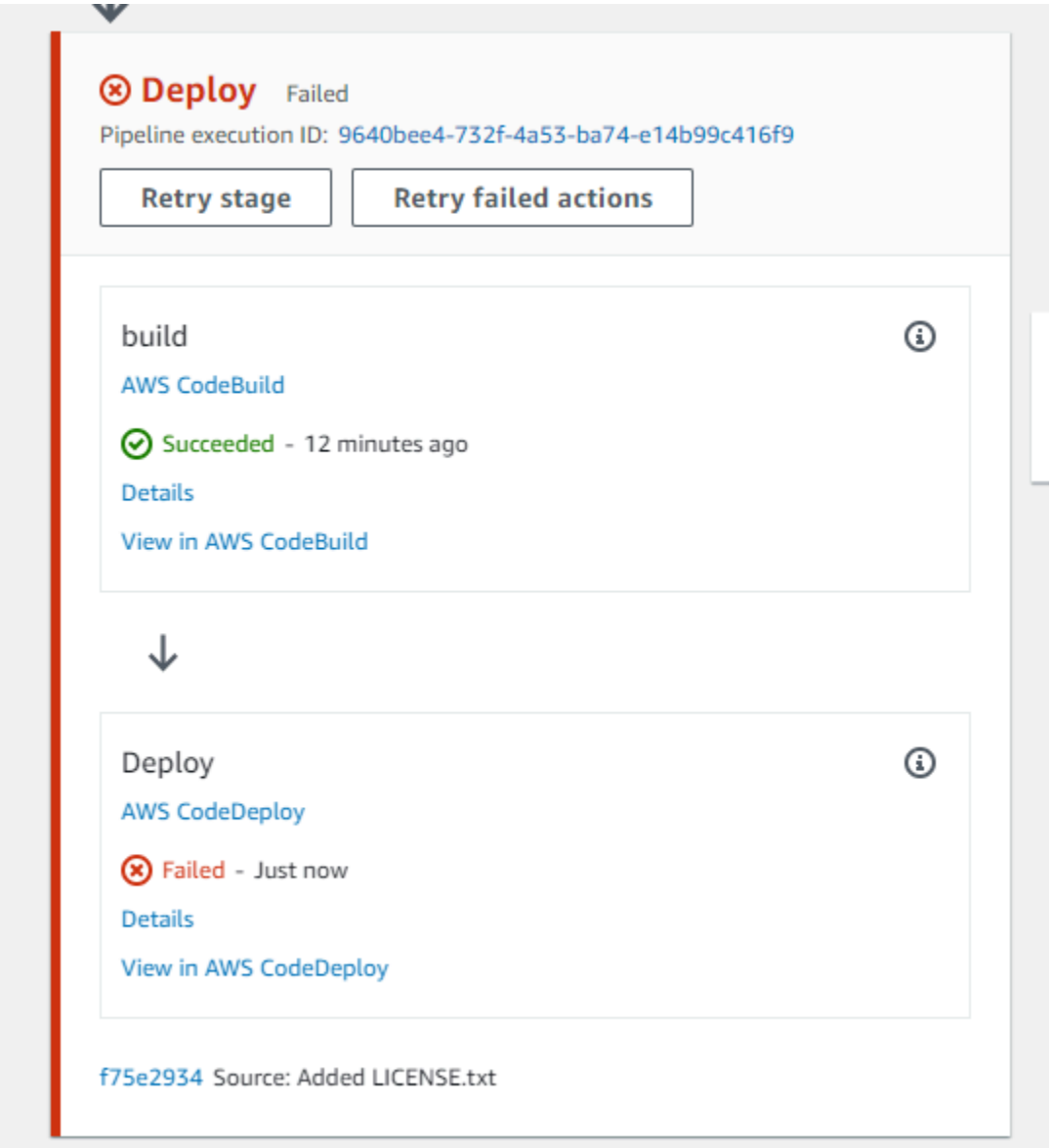
Wiederholen fehlgeschlagener Aktionen (Konsole)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase erneut zu versuchen: Konsole

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline.
3. Suchen Sie die Phase mit der fehlgeschlagenen Aktion und wählen Sie dann eine der folgenden Optionen aus:

- Um alle Aktionen in der Phase erneut zu versuchen, wählen Sie Phase wiederholen aus.
- Um nur fehlgeschlagene Aktionen in der Phase erneut zu versuchen, wählen Sie Fehlgeschlagene Aktionen wiederholen.



Wenn alle wiederholten Aktionen in der Stufe erfolgreich abgeschlossen sind, wird die Pipeline weiter ausgeführt.

Wiederholen fehlgeschlagener Aktionen (CLI)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase erneut zu versuchen - CLI

Um alle Aktionen oder alle fehlgeschlagenen Aktionen erneut AWS CLI zu versuchen, führen Sie den retry-stage-execution Befehl mit den folgenden Parametern aus:

```
--pipeline-name <value>
--stage-name <value>
--pipeline-execution-id <value>
--retry-mode ALL_ACTIONS/FAILED_ACTIONS
```

> **Note**
>
> Die Werte, für die Sie verwenden können, `retry-mode` sind `FAILED_ACTIONS` und`ALL_ACTIONS`.

1. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer [retry-stage-execution](#)Befehlszeile (Windows) aus, wie im folgenden Beispiel für eine Pipeline mit dem Namen gezeigt`MyPipeline`.

   ```
   aws codepipeline retry-stage-execution --pipeline-name MyPipeline --stage-name
    Deploy --pipeline-execution-id b59babff-5f34-EXAMPLE --retry-mode FAILED_ACTIONS
   ```

   Die Ausgabe gibt die Ausführungs-ID zurück:

   ```
   {
       "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
   }
   ```

2. Sie können den Befehl auch mit einer JSON-Eingabedatei ausführen. Sie erstellen zunächst eine JSON-Datei, die die Pipeline, die Stufe mit den fehlgeschlagenen Aktionen und die letzte Pipeline-Ausführung in dieser Stufe ermittelt. Führen Sie den Befehl retry-stage-execution mit dem Parameter `--cli-input-json` aus. Am einfachsten können Sie die für die JSON-Datei benötigten Details über den Befehl get-pipeline-state abrufen.

a. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer get-pipeline-stateBefehlszeile (Windows) in einer Pipeline aus. Für eine Pipeline mit dem Namen würden Sie MyFirstPipeline beispielsweise etwas Ähnliches wie das Folgende eingeben:

```
aws codepipeline get-pipeline-state --name MyFirstPipeline
```

Die Ausgabe des Befehls umfasst die Pipeline-Statusinformationen für jede einzelne Stufe. Im folgenden Beispiel zeigt die Ausgabe, dass eine oder mehrere Aktionen in der Staging-Stufe fehlgeschlagen sind:

```
{
    "updated": 1427245911.525,
    "created": 1427245911.525,
    "pipelineVersion": 1,
    "pipelineName": "MyFirstPipeline",
    "stageStates": [
        {
            "actionStates": [...],
            "stageName": "Source",
            "latestExecution": {
                "pipelineExecutionId": "9811f7cb-7cf7-SUCCESS",
                "status": "Succeeded"
            }
        },
        {
            "actionStates": [...],
            "stageName": "Staging",
            "latestExecution": {
                "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
                "status": "Failed"
            }
        }
    ]
}
```

b. Erstellen Sie in einem Texteditor eine Datei im JSON-Format, in der Sie Folgendes festhalten:

- Der Name der Pipeline mit den fehlgeschlagenen Aktionen
- Der Name der Stufe mit den fehlgeschlagenen Aktionen

- Die ID der letzten Pipeline-Ausführung in der Stufe
- Der Retry-Modus.

Für das vorherige MyFirstPipeline Beispiel würde Ihre Datei etwa so aussehen:

```
{
    "pipelineName": "MyFirstPipeline",
    "stageName": "Staging",
    "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
    "retryMode": "FAILED_ACTIONS"
}
```

c. Speichern Sie die Datei mit einem Namen wie **`retry-failed-actions.json`**.

d. Rufen Sie die beim Ausführen des Befehls retry-stage-execution erstellte Datei auf. Beispielsweise:

> **Important**
>
> Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

```
aws codepipeline retry-stage-execution --cli-input-json file://retry-failed-
actions.json
```

e. Um die Ergebnisse des Wiederholungsversuchs anzuzeigen, öffnen Sie entweder die CodePipeline Konsole und wählen Sie die Pipeline aus, die die fehlgeschlagenen Aktionen enthält, oder verwenden Sie den get-pipeline-state Befehl erneut. Weitere Informationen finden Sie unter Pipelines und Details anzeigen in CodePipeline.

Genehmigungsaktionen verwalten in CodePipeline

In können Sie einer Phase in einer Pipeline an dem Punkt AWS CodePipeline, an dem die Pipeline-Ausführung beendet werden soll, eine Genehmigungsaktion hinzufügen, sodass eine Person mit den erforderlichen AWS Identity and Access Management Berechtigungen die Aktion genehmigen oder ablehnen kann.

Wird die Aktion genehmigt, wird die Pipeline-Ausführung fortgesetzt. Wenn die Aktion abgelehnt wird — oder wenn innerhalb von sieben Tagen, nachdem die Pipeline die Aktion erreicht und gestoppt hat, niemand die Aktion genehmigt oder ablehnt — ist das Ergebnis dasselbe wie eine fehlgeschlagene Aktion, und die Pipeline-Ausführung wird nicht fortgesetzt.

Es gibt die folgenden Gründe für die Verwendung manueller Genehmigungen:

- Sie möchten, dass jemand eine Code- oder Change-Management-Prüfung durchführt, bevor eine Überarbeitung für die nächste Stufe einer Pipeline erlaubt wird.
- Sie möchten, dass jemand einen manuellen Qualitätssicherungstest bei der letzten Version einer Anwendung durchführt oder die Integrität eines Build-Artefakts bestätigt, bevor eine Veröffentlichung stattfindet.
- Sie möchten, dass jemand einen neuen oder aktualisierten Text prüft, bevor er auf einer Firmenwebsite veröffentlicht wird.

Konfigurationsoptionen für manuelle Genehmigungsaktionen in CodePipeline

CodePipeline bietet drei Konfigurationsoptionen, mit denen Sie Genehmiger über die Genehmigungsaktion informieren können.

Genehmigungsbenachrichtigungen veröffentlichen Sie können eine Genehmigungsaktion so konfigurieren, dass eine Nachricht in einem Amazon Simple Notification Service-Thema veröffentlicht wird, wenn die Pipeline bei der Aktion stoppt. Amazon SNS übermittelt die Nachricht an jeden Endpunkt, der das Thema abonniert hat. Sie müssen ein Thema verwenden, das in derselben AWS Region wie die Pipeline erstellt wurde und die Genehmigungsaktion enthalten wird. Wenn Sie ein Thema erstellen, sollten Sie ihm einen Namen geben, der seinen Zweck erkennen lässt, wie z. B. `MyFirstPipeline-us-east-2-approval`.

Wenn Sie Genehmigungsbenachrichtigungen zu Amazon SNS-Themen veröffentlichen, können Sie aus Formaten wie E-Mail- oder SMS-Empfängern, SQS-Warteschlangen, HTTP/HTTPS-Endpunkten oder AWS Lambda Funktionen wählen, die Sie mit Amazon SNS aufrufen. Informationen zu Amazon SNS SNS-Themenbenachrichtigungen finden Sie in den folgenden Themen:

- Was ist Amazon Simple Notification Service?
- Ein Thema in Amazon SNS erstellen
- Senden von Amazon SNS-Nachrichten an Amazon SQS-Warteschlagen

- Abonnieren einer Warteschlange für ein Amazon SNS-Thema
- Senden von Amazon SNS-Nachrichten an HTTP/HTTPS-Endpunkte
- Aufrufen von Lambda-Funktionen mit Amazon SNS-Benachrichtigungen

Unter JSON-Datenformat für manuelle Genehmigungsbenachrichtigungen in CodePipeline finden Sie die Struktur der JSON-Daten, die für eine Genehmigungsaktionsbenachrichtigung generiert werden.

Festlegen einer URL zur Prüfung – Als Teil der Konfiguration der Genehmigungsaktion können Sie eine URL angeben, die geprüft werden soll. Bei der URL kann es sich um einen Link zu einer Webanwendung handeln, die Genehmiger testen sollen, oder um eine Seite mit weiteren Informationen zu Ihrer Genehmigungsanfrage. Die URL ist in der Benachrichtigung enthalten, die im Amazon SNS SNS-Thema veröffentlicht wird. Genehmiger können die Konsole oder CLI zum Anzeigen verwenden.

Eingeben von Kommentaren für Genehmiger Beim Erstellen einer Genehmigungsaktion können Sie ebenfalls Kommentare hinzufügen, die für die Empfänger der Benachrichtigungen oder für die Personen, die die Aktion in der Konsole oder CLI-Reaktion sehen, angezeigt werden.

Keine Konfigurationsoptionen – Sie können auch keine dieser drei Optionen konfigurieren. Sie brauchen sie möglicherweise nicht, wenn Sie beispielsweise jemanden direkt darüber benachrichtigen, dass die Aktion von ihm geprüft werden kann, oder wenn Sie einfach nur möchten, dass die Pipeline so lange anhält, bis Sie sich dazu entschieden haben, selbst die Aktion zu genehmigen.

Überblick über die Einrichtung und den Arbeitsablauf für Genehmigungsaktionen in CodePipeline

Nachstehend finden Sie eine Übersicht zur Einrichtung und Verwendung manueller Genehmigungen.

1. Sie gewähren einer oder mehreren IAM-Rollen in Ihrer Organisation die IAM-Berechtigungen, die zum Genehmigen oder Ablehnen von Genehmigungsaktionen erforderlich sind.
2. (Optional) Wenn Sie Amazon SNS SNS-Benachrichtigungen verwenden, stellen Sie sicher, dass die Servicerolle, die Sie in Ihrem CodePipeline Betrieb verwenden, berechtigt ist, auf Amazon SNS SNS-Ressourcen zuzugreifen.
3. (Optional) Wenn Sie Amazon SNS SNS-Benachrichtigungen verwenden, erstellen Sie ein Amazon SNS SNS-Thema und fügen ihm einen oder mehrere Abonnenten oder Endpunkte hinzu.

4. Wenn Sie die AWS CLI verwenden, um die Pipeline zu erstellen, oder nachdem Sie die CLI oder Konsole zum Erstellen der Pipeline verwendet haben, fügen Sie einer Phase in der Pipeline eine Genehmigungsaktion hinzu.

   Wenn Sie Benachrichtigungen verwenden, nehmen Sie den Amazon-Ressourcennamen (ARN) des Amazon SNS-Themas in die Konfiguration der Aktion auf. (Ein ARN ist eine eindeutige Kennung für eine Amazon-Ressource. ARNs für Amazon SNS SNS-Themen sind wie *arn:aws:sns:us-east* - 2:80398 EXAMPLE: strukturiert. MyApprovalTopic Weitere Informationen finden Sie unter Amazon-Ressourcennamen (ARNs) und AWS-Service Namespaces in der.) Allgemeine Amazon Web Services-Referenz
5. Die Pipeline stoppt, wenn sie die Genehmigungsaktion erreicht. Wenn ein Amazon SNS SNS-Themen-ARN in der Konfiguration der Aktion enthalten war, wird eine Benachrichtigung unter dem Amazon SNS SNS-Thema veröffentlicht, und allen Abonnenten des Themas oder abonnierten Endpunkten wird eine Nachricht mit einem Link zur Überprüfung der Genehmigungsaktion in der Konsole zugestellt.
6. Ein Genehmiger untersucht die Ziel-URL und prüft Kommentare, sofern vorhanden.
7. Mithilfe der Konsole, der CLI oder dem SDK stellt der Genehmiger einen Übersichtskommentar bereit und sendet eine Antwort:
   - Genehmigt: Die Pipeline-Ausführung wird fortgesetzt.
   - Abgelehnt: Der Stufenstatus wird in "Fehlgeschlagen" geändert und die Pipeline-Ausführung wird nicht fortgesetzt.

   Wenn innerhalb von sieben Tagen keine Antwort übermittelt wird, wird die Aktion als "fehlgeschlagen" markiert.

Erteilen Sie einem IAM-Benutzer Genehmigungsberechtigungen in CodePipeline

Bevor IAM-Benutzer in Ihrer Organisation Genehmigungsaktionen genehmigen oder ablehnen können, müssen ihnen Berechtigungen für den Zugriff auf Pipelines und die Aktualisierung des Status von Genehmigungsaktionen erteilt werden. Sie können Berechtigungen für den Zugriff auf alle Pipelines und Genehmigungsaktionen in Ihrem Konto erteilen, indem Sie die `AWSCodePipelineApproverAccess` verwaltete Richtlinie einem IAM-Benutzer, einer Rolle oder einer Gruppe zuordnen. Oder Sie können eingeschränkte Berechtigungen gewähren, indem Sie die einzelnen Ressourcen angeben, auf die ein IAM-Benutzer, eine IAM-Rolle oder eine Gruppe zugreifen kann.

> **Note**
>
> Die in diesem Thema beschriebenen Berechtigungen gewähren einen äußerst begrenzten Zugriff. Um einem Benutzer, einer Rolle oder einer Gruppe mehr zu erlauben, als Genehmigungsaktionen anzunehmen oder abzulehnen, können Sie weitere verwaltete Richtlinien anfügen. Informationen zu den verfügbaren verwalteten Richtlinien finden Sie unter. CodePipeline AWS verwaltete Richtlinien für AWS CodePipeline

Gewähren von Genehmigungsberechtigungen für alle Pipelines und Genehmigungsaktionen

Verwenden Sie für Benutzer, die Genehmigungsaktionen in ausführen müssen CodePipeline, die `AWSCodePipelineApproverAccess` verwaltete Richtlinie.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Festlegen von Genehmigungsberechtigungen für bestimmte Pipelines und Genehmigungsaktionen

Verwenden Sie für Benutzer, die Genehmigungsaktionen in ausführen müssen CodePipeline, die folgende benutzerdefinierte Richtlinie. Geben Sie in der folgenden Richtlinie die einzelnen

Ressourcen an, auf die ein Benutzer zugreifen kann. Die folgende Richtlinie gewährt Benutzern beispielsweise die Befugnis, nur die in der `MyFirstPipeline` Pipeline angegebene Aktion `MyApprovalAction` in der Region USA Ost (Ohio) (us-east-2) zu genehmigen oder abzulehnen:

> **Note**
>
> Die `codepipeline:ListPipelines` Berechtigung ist nur erforderlich, wenn IAM-Benutzer auf das CodePipeline Dashboard zugreifen müssen, um diese Pipeline-Liste anzuzeigen. Wenn kein Konsolenzugriff erforderlich ist, können Sie `codepipeline:ListPipelines` weglassen.

So verwenden Sie den JSON-Richtlinieneditor zum Erstellen einer Richtlinie

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).

   Wenn Sie zum ersten Mal Policies (Richtlinien) auswählen, erscheint die Seite Welcome to Managed Policies (Willkommen bei verwalteten Richtlinien). Wählen Sie Get Started.
3. Wählen Sie oben auf der Seite Create policy (Richtlinie erstellen) aus.
4. Wählen Sie im Bereich Policy editor (Richtlinien-Editor) die Option JSON aus.
5. Geben Sie folgendes JSON-Richtliniendokument ein:

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "codepipeline:ListPipelines"
           ],
           "Resource": [
               "*"
           ]
       },
       {
           "Effect": "Allow",
           "Action": [
```

```
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:GetPipelineExecution"
            ],
            "Resource": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline"
        },
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PutApprovalResult"
            ],
            "Resource": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline/MyApprovalStage/MyApprovalAction"
        }
    ]
}
```

6. Wählen Sie Weiter aus.

> **Note**
>
> Sie können jederzeit zwischen den Editoroptionen Visual und JSON wechseln. Wenn Sie jedoch Änderungen vornehmen oder im Visual-Editor Weiter wählen, strukturiert IAM Ihre Richtlinie möglicherweise um, um sie für den visuellen Editor zu optimieren. Weitere Informationen finden Sie unter Richtlinienrestrukturierung im IAM-Benutzerhandbuch.

7. Geben Sie auf der Seite Prüfen und erstellen unter Richtlinienname einen Namen und unter Beschreibung (optional) eine Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie Permissions defined in this policy (In dieser Richtlinie definierte Berechtigungen), um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden.
8. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.

Gewähren Sie Amazon SNS SNS-Berechtigungen für eine CodePipeline Servicerolle

Wenn Sie Amazon SNS verwenden möchten, um Benachrichtigungen zu Themen zu veröffentlichen, bei denen Genehmigungsaktionen überprüft werden müssen, muss der Servicerolle, die Sie in Ihrem CodePipeline Betrieb verwenden, die Berechtigung zum Zugriff auf die Amazon SNS SNS-

Ressourcen erteilt werden. Sie können die IAM-Konsole verwenden, um diese Berechtigung zu Ihrer Servicerolle hinzuzufügen.

Geben Sie in der folgenden Richtlinie die Richtlinie für die Veröffentlichung mit SNS an. Für die folgende Richtlinie können Sie ihr einen Namen geben`SNSPublish`. Verwenden Sie die folgende Richtlinie, indem Sie sie Ihrer Servicerolle zuordnen.

> **Important**
>
> Stellen Sie sicher, dass Sie AWS Management Console mit denselben Kontoinformationen, die Sie verwendet haben, bei Erste Schritte mit CodePipeline dem angemeldet sind.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "*"
        }
    ]
}
```

So verwenden Sie den JSON-Richtlinieneditor zum Erstellen einer Richtlinie

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).

   Wenn Sie zum ersten Mal Policies (Richtlinien) auswählen, erscheint die Seite Welcome to Managed Policies (Willkommen bei verwalteten Richtlinien). Wählen Sie Get Started.
3. Wählen Sie oben auf der Seite Create policy (Richtlinie erstellen) aus.
4. Wählen Sie im Bereich Policy editor (Richtlinien-Editor) die Option JSON aus.
5. Geben oder fügen Sie ein JSON-Richtliniendokument ein. Weitere Informationen zur IAM-Richtliniensprache finden Sie in der IAM-JSON-Richtlinienreferenz.
6. Beheben Sie alle Sicherheitswarnungen, Fehler oder allgemeinen Warnungen, die während der Richtlinien-Validierung erzeugt wurden, und wählen Sie dann Weiter.

> **Note**
>
> Sie können jederzeit zwischen den Editoroptionen Visual und JSON wechseln. Wenn Sie jedoch Änderungen vornehmen oder im Visual-Editor Weiter wählen, strukturiert IAM Ihre Richtlinie möglicherweise um, um sie für den visuellen Editor zu optimieren. Weitere Informationen finden Sie unter Richtlinienrestrukturierung im IAM-Benutzerhandbuch.

7. (Optional) Wenn Sie eine Richtlinie in der erstellen oder bearbeiten AWS Management Console, können Sie eine JSON- oder YAML-Richtlinienvorlage generieren, die Sie in AWS CloudFormation Vorlagen verwenden können.

   Wählen Sie dazu im Richtlinien-Editor Aktionen und anschließend CloudFormationVorlage generieren aus. Weitere Informationen AWS CloudFormation dazu finden Sie in der Referenz zum AWS Identity and Access Management Ressourcentyp im AWS CloudFormation Benutzerhandbuch.
8. Wenn Sie mit dem Hinzufügen von Berechtigungen zur Richtlinie fertig sind, wählen Sie Next (Weiter) aus.
9. Geben Sie auf der Seite Prüfen und erstellen unter Richtlinienname einen Namen und unter Beschreibung (optional) eine Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie Permissions defined in this policy (In dieser Richtlinie definierte Berechtigungen), um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden.
10. (Optional) Fügen Sie der Richtlinie Metadaten hinzu, indem Sie Tags als Schlüssel-Wert-Paare anfügen. Weitere Informationen zur Verwendung von Tags in IAM finden Sie unter Markieren von IAM-Ressourcen im IAM-Benutzerhandbuch.
11. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.

Fügen Sie einer Pipeline eine manuelle Genehmigungsaktion hinzu in CodePipeline

Sie können einer Phase in einer CodePipeline Pipeline an dem Punkt, an dem die Pipeline gestoppt werden soll, eine Genehmigungsaktion hinzufügen, sodass jemand die Aktion manuell genehmigen oder ablehnen kann.

> **Note**
>
> Genehmigungsaktionen können nicht zu Quellenstufen hinzugefügt werden. Quellenstufen können nur Quellenaktionen enthalten.

Wenn Sie Amazon SNS verwenden möchten, um Benachrichtigungen zu versenden, wenn eine Genehmigungsaktion zur Überprüfung bereit ist, müssen Sie zunächst die folgenden Voraussetzungen erfüllen:

- Erteilen Sie Ihrer CodePipeline Servicerolle die Erlaubnis, auf Amazon SNS SNS-Ressourcen zuzugreifen. Weitere Informationen finden Sie unter Gewähren Sie Amazon SNS SNS-Berechtigungen für eine CodePipeline Servicerolle.
- Erteilen Sie einer oder mehreren IAM-Identitäten in Ihrer Organisation die Erlaubnis, den Status einer Genehmigungsaktion zu aktualisieren. Weitere Informationen finden Sie unter Erteilen Sie einem IAM-Benutzer Genehmigungsberechtigungen in CodePipeline.

In diesem Beispiel erstellen Sie eine neue Genehmigungsphase und fügen der Phase eine manuelle Genehmigungsaktion hinzu. Sie können einer vorhandenen Phase, die andere Aktionen enthält, auch eine manuelle Genehmigungsaktion hinzufügen.

Fügen Sie einer CodePipeline Pipeline (Konsole) eine manuelle Genehmigungsaktion hinzu

Sie können die CodePipeline Konsole verwenden, um einer vorhandenen CodePipeline Pipeline eine Genehmigungsaktion hinzuzufügen. Sie müssen die AWS CLI verwenden, wenn Sie beim Erstellen einer neuen Pipeline Genehmigungsaktionen hinzufügen möchten.

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie in Name die Pipeline aus.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wenn Sie eine Genehmigungsaktion zu einer neuen Stufe hinzufügen möchten, wählen Sie an dem Punkt in der Pipeline, an dem Sie eine Genehmigungsanforderung hinzufügen möchten, +Add stage (+Stufe hinzufügen) und geben Sie einen Namen für die Stufe ein. Geben Sie auf der Seite Add stage (Schritt hinzufügen) unter Stage name (Stagenname) Ihren neuen Stagennamen ein. Fügen Sie beispielsweise eine neue Stufe hinzu und nennen Sie sie `Manual_Approval`.

Wenn Sie eine Genehmigungsaktion zu einer vorhandenen Stufe hinzufügen möchten, klicken Sie auf Edit stage (Stufe bearbeiten).

5. Wählen Sie in der Phase, in der Sie die Genehmigungsaktion hinzufügen möchten, + Add action group (+ Aktionsgruppe hinzufügen).
6. Führen Sie auf der Seite Edit action (Aktion bearbeiten) die folgenden Schritte aus:

    1. Geben Sie unter Action name (Aktionsname) einen Namen für die Aktion ein, um sie zu identifizieren.
    2. Wähen Sie unter Action provider (Aktionsanbieter) unter Approval (Genehmigung) die Option Manual approval (Manuelle Genehmigung).
    3. (Optional) Wählen Sie unter SNS topic ARN (ARN für SNS-Thema) den Namen des Themas aus, das zum Senden von Benachrichtigungen für die Genehmigungsaktion verwendet werden soll.
    4. (Optional) Geben Sie in URL for review die URL der Seite oder Anwendung ein, die der Genehmiger prüfen soll. Genehmiger können über einen Link in der Konsolenansicht der Pipeline auf diese URL zugreifen.
    5. (Optional) Geben Sie unter Comments (Kommentare) zusätzliche Informationen für den Genehmiger ein.
    6. Wählen Sie Speichern.

Hinzufügen einer manuellen Genehmigungsaktion zu einer CodePipeline Pipeline (CLI)

Sie können die CLI zum Hinzufügen einer Genehmigungsaktion zu einer vorhandenen Pipeline verwenden oder wenn Sie eine Pipeline erstellen. Hierzu schließen Sie eine Genehmigungsaktion mit dem Typ „manuelle Genehmigung“ in einer Stufe ein, die Sie erstellen oder bearbeiten.

Weitere Informationen zum Erstellen und Bearbeiten von Pipelines finden Sie unter Erstellen Sie eine Pipeline in CodePipeline und Eine Pipeline bearbeiten in CodePipeline.

Um eine Stufe einer Pipeline hinzuzufügen, die nur eine Genehmigungsaktion enthält, sollten Sie etwas Ähnliches wie im folgenden Beispiel einschließen, wenn Sie die Pipeline erstellen oder aktualisieren.

> **Note**
>
> Der Abschnitt `configuration` ist optional. Dies ist nur ein Teil, nicht die komplette Struktur der Datei. Weitere Informationen finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

```
{
    "name": "MyApprovalStage",
    "actions": [
        {
            "name": "MyApprovalAction",
            "actionTypeId": {
                "category": "Approval",
                "owner": "AWS",
                "version": "1",
                "provider": "Manual"
            },
            "inputArtifacts": [],
            "outputArtifacts": [],
            "configuration": {
                "NotificationArn": "arn:aws:sns:us-
east-2:80398EXAMPLE:MyApprovalTopic",
                "ExternalEntityLink": "http://example.com",
                "CustomData": "The latest changes include feedback from Bob."},
            "runOrder": 1
        }
    ]
}
```

Wenn sich die Genehmigungsaktion in einer Stufe mit weiteren Aktionen befindet, sieht der Abschnitt Ihrer JSON-Datei, die die Stufe enthält, stattdessen ungefähr aus wie im folgenden Beispiel.

> **Note**
>
> Der Abschnitt `configuration` ist optional. Dies ist nur ein Teil, nicht die komplette Struktur der Datei. Weitere Informationen finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

```
,
{
    "name": "Production",
    "actions": [
        {
            "inputArtifacts": [],
            "name": "MyApprovalAction",
            "actionTypeId": {
                "category": "Approval",
                "owner": "AWS",
                "version": "1",
                "provider": "Manual"
            },
            "outputArtifacts": [],
            "configuration": {
                "NotificationArn": "arn:aws:sns:us-
east-2:80398EXAMPLE:MyApprovalTopic",
                "ExternalEntityLink": "http://example.com",
                "CustomData": "The latest changes include feedback from Bob."
            },
            "runOrder": 1
        },
        {
            "inputArtifacts": [
                {
                    "name": "MyApp"
                }
            ],
            "name": "MyDeploymentAction",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "version": "1",
                "provider": "CodeDeploy"
            },
            "outputArtifacts": [],
            "configuration": {
                "ApplicationName": "MyDemoApplication",
                "DeploymentGroupName": "MyProductionFleet"
            },
            "runOrder": 2
        }
    ]
```

```
}
```

Genehmigen oder lehnen Sie eine Genehmigungsaktion ab in CodePipeline

Wenn eine Pipeline eine Genehmigungsaktion enthält, wird die Pipeline-Ausführung an dem Punkt angehalten, an dem die Aktion hinzugefügt wurde. Die Pipeline wird nur fortgesetzt, wenn jemand manuell die Aktion genehmigt. Wenn ein Genehmiger die Aktion abgelehnt oder wenn keine Genehmigungsantwort innerhalb von sieben Tagen nach Anhalten der Pipeline für die Genehmigungsaktion empfangen wird, ändert sich der Pipeline-Status in "Fehlgeschlagen".

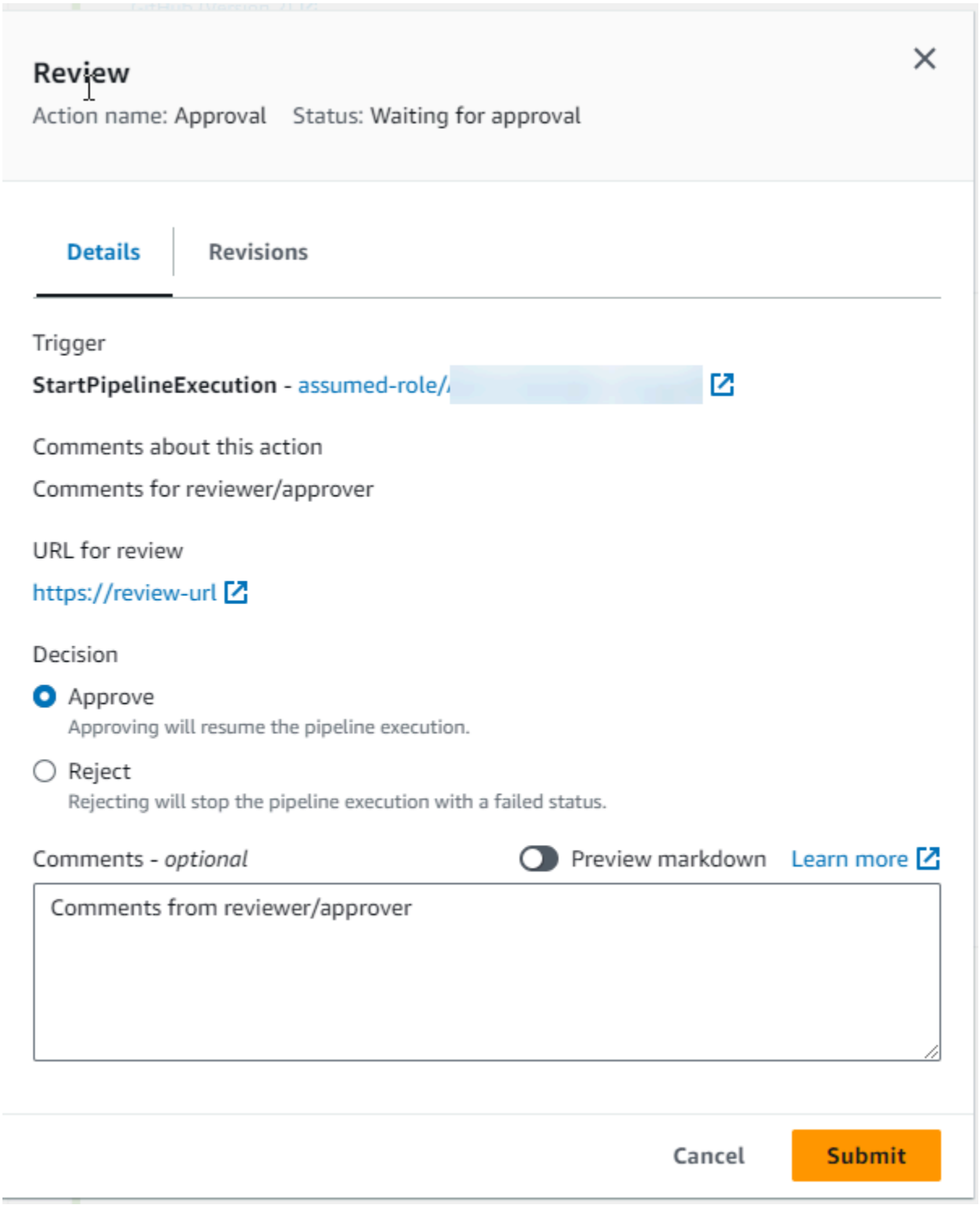
Wenn die Person, die die Genehmigungsaktion zur Pipeline hinzugefügt hat, Benachrichtigungen konfiguriert hat, erhalten Sie möglicherweise eine E-Mail mit den Pipeline-Informationen und dem Status der Genehmigung.

Erlauben oder Ablehnen einer Genehmigungsaktion (Konsole)

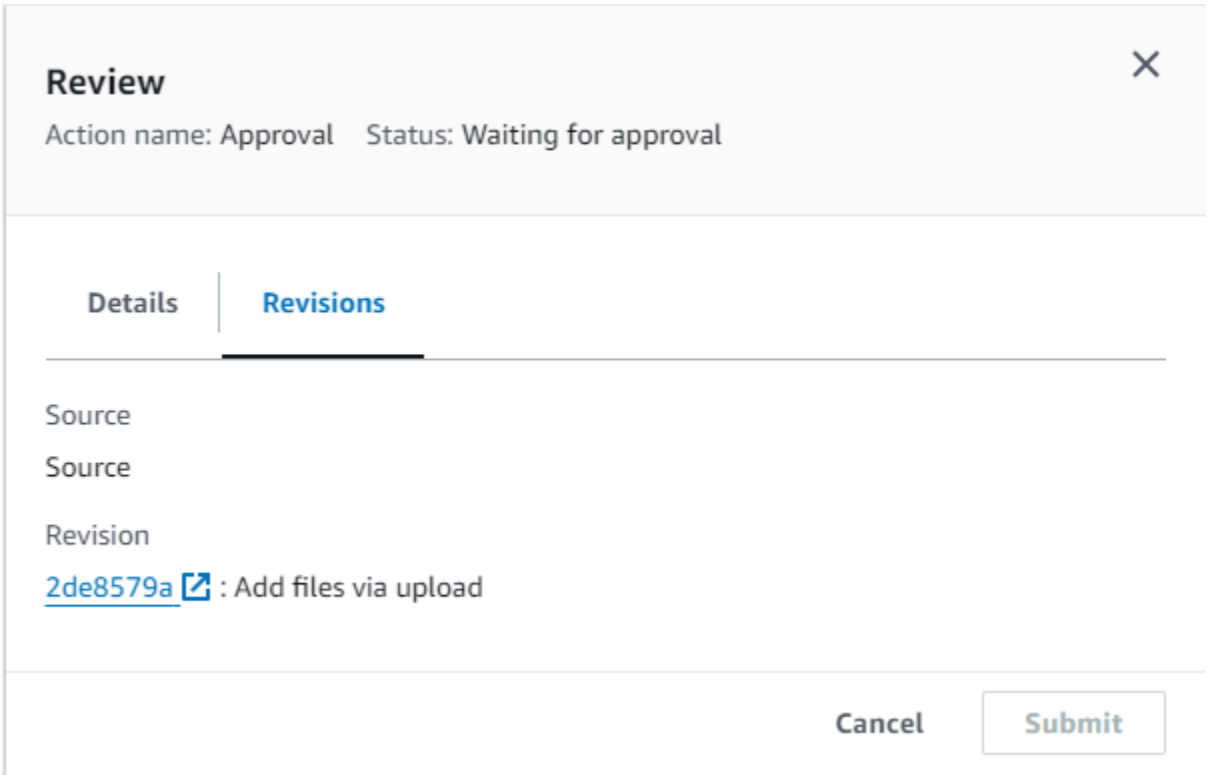
Wenn Sie eine Benachrichtigung erhalten, die einen direkten Link zu einer Genehmigungsaktion enthält, klicken Sie auf den Link Approve or reject (Genehmigen oder ablehnen), melden Sie sich bei der Konsole an und fahren Sie dann mit Schritt 7 hier fort. Führen Sie anderenfalls alle der folgenden Schritte aus.

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie auf der Seite All Pipelines den Namen der Pipeline aus.
3. Suchen Sie die Stufe mit der Genehmigungsaktion. Wählen Sie Überprüfen aus.

   Das Dialogfeld „Überprüfen“ wird angezeigt. Auf der Registerkarte „Details“ werden der Inhalt und die Kommentare der Bewertung angezeigt.



Auf der Registerkarte „Revisionen“ werden die Quellversionen für die Ausführung angezeigt.



4. Sehen Sie sich auf der Registerkarte Details die Kommentare und die URL an, falls vorhanden. Die Nachricht zeigt auch die URL des von Ihnen zu prüfenden Inhalts an, sofern eingeschlossen.
5. Wenn eine URL angegeben wurde, wählen Sie in der Aktion zum Öffnen der Zielwebseite den Link zur URL für die Überprüfung aus, und überprüfen Sie dann den Inhalt.
6. Geben Sie im Überprüfungsfenster Kommentare zur Bewertung ein, z. B. warum Sie die Aktion genehmigen oder ablehnen, und wählen Sie dann Genehmigen oder Ablehnen aus.
7. Wählen Sie Absenden aus.

Erlauben oder Ablehnen einer Genehmigungsanfrage (CLI)

Um mit der CLI auf eine Genehmigungsaktion zu reagieren, müssen Sie zuerst den Befehl get-pipeline-stateverwenden, um das Token abzurufen, das mit der letzten Ausführung der Genehmigungsaktion verknüpft ist.

1. Führen Sie an einem Terminal (Linux, macOS oder Unix) oder einer Befehlszeile (Windows) den get-pipeline-stateBefehl in der Pipeline aus, die die Genehmigungsaktion enthält. Geben Sie beispielsweise für eine Pipeline mit dem Namen *MyFirstPipeline*Folgendes ein:

```
aws codepipeline get-pipeline-state --name MyFirstPipeline
```

2. Suchen Sie in der Antwort auf den Befehl nach dem `token`-Wert, der unter `latestExecution` im Bereich `actionStates` für die Genehmigungsaktion erscheint, wie hier veranschaulicht:

```
{
    "created": 1467929497.204,
    "pipelineName": "MyFirstPipeline",
    "pipelineVersion": 1,
    "stageStates": [
        {
            "actionStates": [
                {
                    "actionName": "MyApprovalAction",
                    "currentRevision": {
                        "created": 1467929497.204,
                        "revisionChangeId": "CEM7d6Tp7zfelUSLCPPwo234xEXAMPLE",
                        "revisionId": "HYGp7zmwbCPPwo23xCMdTeqIlEXAMPLE"
                    },
                    "latestExecution": {
                        "lastUpdatedBy": "identity",
                        "summary": "The new design needs to be reviewed before
 release.",
                        "token": "1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN"
                    }
                }
//More content might appear here
}
```

3. Erstellen Sie in einem Texteditor eine Datei im JSON-Format, in der Sie Folgendes hinzufügen:

   - Den Namen der Pipeline, die die Genehmigungsaktion enthält.
   - Der Name der Stufe, die die Genehmigungsaktion enthält.
   - Den Namen der Genehmigungsaktion.
   - Der Token-Wert, den Sie im vorherigen Schritt gesammelt haben.
   - Ihre Antwort auf die Aktion (Genehmigt oder Abgelehnt). Die Antwort muss großgeschrieben werden.
   - Ihre zusammenfassenden Kommentare.

   Für das vorherige *MyFirstPipeline*Beispiel sollte Ihre Datei wie folgt aussehen:

```
{
```

```
  "pipelineName": "MyFirstPipeline",
  "stageName": "MyApprovalStage",
  "actionName": "MyApprovalAction",
  "token": "1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN",
  "result": {
    "status": "Approved",
    "summary": "The new design looks good. Ready to release to customers."
  }
}
```

4. Speichern Sie die Datei mit einem Namen wie **`approvalstage-approved.json`**.
5. Führen Sie den put-approval-resultBefehl aus und geben Sie den Namen der Genehmigungs-JSON-Datei an, ähnlich wie im Folgenden:

> **Important**
>
> Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

```
aws codepipeline put-approval-result --cli-input-json file://approvalstage-approved.json
```

JSON-Datenformat für manuelle Genehmigungsbenachrichtigungen in CodePipeline

Für Genehmigungsaktionen, die Amazon SNS-Benachrichtigungen verwenden, werden JSON-Daten über die Aktion erstellt und in Amazon SNS veröffentlicht, wenn die Pipeline stoppt. Sie können die JSON-Ausgabe verwenden, um Nachrichten an Amazon SQS SQS-Warteschlangen zu senden oder Funktionen in aufzurufen. AWS Lambda

> **Note**
>
> In dieser Anleitung wird nicht behandelt, wie Benachrichtigungen mit JSON konfiguriert werden. Weitere Informationen finden Sie unter Senden von Amazon SNS SNS-Nachrichten an Amazon SQS SQS-Warteschlangen und Aufrufen von Lambda-Funktionen mithilfe von Amazon SNS SNS-Benachrichtigungen im Amazon SNS SNS-Entwicklerhandbuch.

Das folgende Beispiel zeigt die Struktur der JSON-Ausgabe, die mit Genehmigungen verfügbar ist. CodePipeline

```
{
    "region": "us-east-2",
    "consoleLink": "https://console.aws.amazon.com/codepipeline/home?region=us-east-2#/
view/MyFirstPipeline",
    "approval": {
        "pipelineName": "MyFirstPipeline",
        "stageName": "MyApprovalStage",
        "actionName": "MyApprovalAction",
        "token": "1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN",
        "expires": "2016-07-07T20:22Z",
        "externalEntityLink": "http://example.com",
        "approvalReviewLink": "https://console.aws.amazon.com/codepipeline/
home?region=us-east-2#/view/MyFirstPipeline/MyApprovalStage/MyApprovalAction/
approve/1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN",
        "customData": "Review the latest changes and approve or reject within seven
 days."
    }
}
```

Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline

AWS CodePipeline umfasst eine Reihe von Aktionen, mit denen Sie Ressourcen für Ihren automatisierten Release-Prozess konfigurieren, testen und bereitstellen können. Sie können Ihrer Pipeline Aktionen hinzufügen, die sich in einer anderen AWS Region als Ihrer Pipeline befinden. Wenn an der Anbieter für eine Aktion AWS-Service ist und sich dieser Aktionstyp/dieser Anbietertyp in einer anderen AWS Region als Ihre Pipeline befindet, handelt es sich um eine regionsübergreifende Aktion.

> **Note**
>
> Regionalübergreifende Aktionen werden unterstützt und können nur in den AWS Regionen erstellt werden, in denen sie unterstützt werden. CodePipeline Eine Liste der unterstützten AWS Regionen für finden Sie CodePipeline unter Kontingente in AWS CodePipeline.

Sie können die Konsole oder verwenden AWS CLI, AWS CloudFormation um regionsübergreifende Aktionen in Pipelines hinzuzufügen.

> **Note**
>
> Bestimmte Aktionstypen in sind CodePipeline möglicherweise nur in bestimmten AWS Regionen verfügbar. Beachten Sie auch, dass es AWS Regionen geben kann, in denen ein Aktionstyp verfügbar ist, ein bestimmter AWS Anbieter für diesen Aktionstyp jedoch nicht verfügbar ist.

Wenn Sie eine Pipeline erstellen oder bearbeiten, müssen Sie einen Artefakt-Bucket in der Pipelineregion haben, sowie einen Artefakt-Bucket für jede Region, in der Sie eine Aktion ausführen möchten. Weitere Informationen zum Parameter `ArtifactStores` erhalten Sie unter CodePipeline Referenz zur Pipeline-Struktur.

> **Note**
>
> CodePipeline verarbeitet das Kopieren von Artefakten von einer AWS Region in die anderen Regionen, wenn regionsübergreifende Aktionen ausgeführt werden.

Wenn Sie die Konsole verwenden, um eine Pipeline oder regionsübergreifende Aktionen zu erstellen, werden Standardartefakt-Buckets CodePipeline in den Regionen konfiguriert, in denen Sie Aktionen haben. Wenn Sie das AWS CLI, oder ein SDK verwenden AWS CloudFormation, um eine Pipeline oder regionsübergreifende Aktionen zu erstellen, geben Sie den Artefakt-Bucket für jede Region an, in der Sie Aktionen haben.

> **Note**
>
> Sie müssen den Artefakt-Bucket und den Verschlüsselungsschlüssel in derselben AWS Region wie die regionsübergreifende Aktion und in demselben Konto wie Ihre Pipeline erstellen.

Für die folgenden Aktionstypen können Sie keine regionsübergreifenden Aktionen erstellen:

- Quellaktionen
- Drittanbieteraktionen
- Benutzerdefinierte Aktionen

> **Note**
>
> Wenn Sie die regionsübergreifende Lambda-Aufrufaktion in verwenden CodePipeline, PutJobFailureResultsollte der Status der Lambda-Ausführung mithilfe von PutJobSuccessResultund an die Region gesendet werden, in der die Lambda-Funktion vorhanden ist, und nicht an die AWS Region, in der sie existiert. CodePipeline

Wenn eine Pipeline eine regionsübergreifende Aktion als Teil einer Phase enthält, werden nur die Eingabeartefakte der regionsübergreifenden Aktion von der Pipeline-Region in die Region der Aktion CodePipeline repliziert.

> **Note**
>
> Die Pipeline-Region und die Region, in der Ihre Ressourcen zur Erkennung von CloudWatch Events-Änderungen verwaltet werden, bleiben unverändert. Die Region, in der Ihre Pipeline gehostet wird, ändert sich nicht.

Verwalten von regionsübergreifenden Aktionen in einer Pipeline (Konsole)

Sie können die CodePipeline Konsole verwenden, um einer vorhandenen Pipeline eine regionsübergreifende Aktion hinzuzufügen. Zur Erstellung einer neuen Pipeline mit regionsübergreifenden Aktionen mithilfe des Assistenten zum Erstellen einer Pipeline vgl. Erstellen einer Pipeline (Konsole).

In der Konsole erstellen Sie eine regionsübergreifende Aktion in einer Pipeline-Phase, indem Sie den Anbieter der Aktion und das Feld Region auswählen, das die von Ihnen in der jeweiligen Region für diesen Anbieter erstellten Ressourcen auflistet. Wenn Sie eine regionsübergreifende Aktion hinzufügen, CodePipeline verwendet einen separaten Artefakt-Bucket in der Region der Aktion. Weitere Informationen zu regionsübergreifenden Artefakt-Buckets finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline-Phase (Konsole)

Verwenden Sie die Konsole zum Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline.

> **Note**
>
> Wenn die Pipeline ausgeführt wird, während Änderungen gespeichert werden, wird diese Ausführung nicht abgeschlossen.

So fügen Sie eine regionsübergreifende Aktion hinzu

1. Melden Sie sich bei der Konsole unter http://console.aws.amazon.com/codesuite/codepipeline an.
2. Wählen Sie Ihre Pipeline und dann Edit (Bearbeiten).
3. Wählen Sie unten im Diagramm + Add stage (Phase hinzufügen), wenn Sie eine neue Phase erstellen, oder Edit stage (Phase bearbeiten), wenn Sie die Aktion einer bestehenden Phase hinzufügen möchten.
4. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) die Option + Add action group (Aktionsgruppe hinzufügen), um eine serielle Aktion hinzuzufügen. Oder wählen Sie +Add action (Aktion hinzufügen), um eine parallele Aktion hinzuzufügen.
5. Führen Sie auf der Seite Edit action (Aktion bearbeiten) Folgendes durch:
   a. Geben Sie unter Action name (Aktionsname) einen Namen für die regionsübergreifende Aktion ein.
   b. Wählen Sie unter Action provider (Aktionsanbieter) den Anbieter der Aktion.
   c. Wählen Sie unter Region die AWS Region aus, in der Sie die Ressource für die Aktion erstellt haben oder erstellen möchten. Wenn die Region ausgewählt ist, werden die verfügbaren Ressourcen für diese Region zur Auswahl aufgelistet. Das Feld Region gibt an, wo die AWS Ressourcen für diesen Aktionstyp und Anbietertyp erstellt werden. Dieses Feld wird nur für Aktionen angezeigt, bei denen es sich bei dem Aktionsanbieter um einen handelt. AWS-Service Das Feld Region ist standardmäßig auf dasselbe AWS-Region wie Ihre Pipeline eingestellt.
   d. Wählen Sie unter Input artifacts (Eingabeartefakte) die korrekte Eingabe aus der vorherigen Phase. Wenn es sich bei der vorherigen Phase beispielsweise um eine Quellstufe handelt, wählen Sie SourceArtifact.
   e. Füllen Sie alle erforderlichen Felder für den Aktionsanbieter aus, den Sie konfigurieren.

f. Wählen Sie unter Output artifacts (Ausgabeartefakte) die korrekte Ausgabe zur nächsten Phase. Wenn es sich bei der nächsten Phase beispielsweise um eine Bereitstellungsphase handelt, wählen Sie BuildArtifact.

g. Wählen Sie Speichern.

6. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) Done (Fertig).
7. Wählen Sie Speichern.

Bearbeiten einer regionsübergreifenden Aktion in einer Pipeline-Phase (Konsole)

Verwenden Sie die Konsole zur Bearbeitung einer vorhandenen regionsübergreifenden Aktion in einer Pipeline.

> **Note**
>
> Wenn die Pipeline ausgeführt wird, während Änderungen gespeichert werden, wird diese Ausführung nicht abgeschlossen.

So bearbeiten Sie eine regionsübergreifende Aktion

1. Melden Sie sich unter https://console.aws.amazon.com/codesuite/codepipeline/home. bei der Konsole an.
2. Wählen Sie Ihre Pipeline und dann Edit (Bearbeiten).
3. Wählen Sie Edit stage (Phase bearbeiten).
4. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) das Symbol zur Bearbeitung einer vorhandenen Aktion.
5. Nehmen Sie auf der Seite Edit action (Aktion bearbeiten) die erforderlichen Änderungen an den Feldern vor.
6. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) Done (Fertig).
7. Wählen Sie Speichern.

Löschen einer regionsübergreifenden Aktion aus einer Pipeline-Phase (Konsole)

Verwenden Sie die Konsole zum Löschen einer vorhandenen regionsübergreifenden Aktion aus einer Pipeline.

> **Note**
>
> Wenn die Pipeline ausgeführt wird, während Änderungen gespeichert werden, wird diese Ausführung nicht abgeschlossen.

So löschen Sie eine regionsübergreifende Aktion

1. Melden Sie sich bei der Konsole unter http://console.aws.amazon.com/codesuite/codepipeline an.
2. Wählen Sie Ihre Pipeline und dann Edit (Bearbeiten).
3. Wählen Sie Edit stage (Phase bearbeiten).
4. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) das Symbol zum Löschen einer vorhandenen Aktion.
5. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) Done (Fertig).
6. Wählen Sie Speichern.

Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline (CLI)

Sie können den verwenden AWS CLI , um einer vorhandenen Pipeline eine regionsübergreifende Aktion hinzuzufügen.

Um eine regionsübergreifende Aktion in einer Pipeline-Phase mit dem zu erstellen AWS CLI, fügen Sie die Konfigurationsaktion zusammen mit einem optionalen `region` Feld hinzu. Außerdem müssen Sie bereits einen Artefakt-Bucket in der Region der Aktion erstellt haben. Anstatt den Parameter `artifactStore` der Einzelregionpipeline anzugeben, verwenden Sie den Parameter `artifactStores`, um eine Liste der Artefakt-Buckets jeder Region einzufügen.

> **Note**
>
> In dieser Anleitung und den entsprechenden Beispielen ist *`RegionA`* die Region, in der die Pipeline erstellt wird. Es hat Zugriff auf den Amazon S3 S3-Bucket *`RegionA`*, der zum Speichern von Pipeline-Artefakten verwendet wird, und auf die Servicerolle, die von CodePipeline verwendet wird. *`RegionB`* ist die Region, in der die CodeDeploy Anwendung, die Bereitstellungsgruppe und die Servicerolle erstellt werden, die von verwendet CodeDeploy werden.

Voraussetzungen

Sie müssen Folgendes erstellt haben:

- Eine Pipeline in *RegionA*
- Ein Amazon S3 S3-Artefakt-Bucket in *RegionB*
- Die Ressourcen für Ihre Aktion, z. B. Ihre CodeDeploy Anwendung und Bereitstellungsgruppe für eine *regionsübergreifende* Bereitstellungsaktion, in Region B.

Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline (CLI)

Verwenden Sie die AWS CLI , um einer Pipeline eine regionsübergreifende Aktion hinzuzufügen.

So fügen Sie eine regionsübergreifende Aktion hinzu

1. Führen Sie für eine Pipeline in *RegionA* den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.
2. Fügen Sie das Feld `region` zum Hinzufügen einer neuen Phase mit Ihrer regionsübergreifenden Aktion hinzu, die die Region und Ressourcen für Ihre Aktion enthält. Das folgende JSON-Beispiel fügt eine Bereitstellungsphase mit einer regionsübergreifenden Bereitstellungsaktion hinzu CodeDeploy, in der sich der Anbieter in einer neuen Region befindet. `us-east-1`

   ```
   {
                   "name": "Deploy",
                   "actions": [
                       {
                           "inputArtifacts": [
                               {
                                   "name": "SourceArtifact"
                               }
                           ],
                           "name": "Deploy",
                           "region": "RegionB",
   ```

```
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "name",
                            "DeploymentGroupName": "name"
                        },
                        "runOrder": 1
                    }
```

3. Entfernen Sie in der Pipeline-Struktur das `artifactStore`-Feld und fügen Sie die `artifactStores`-Zuweisung für Ihre neue regionsübergreifende Aktion hinzu. Die Zuordnung muss einen Eintrag für jede AWS Region enthalten, in der Sie Aktionen haben. Für jeden Eintrag in der Zuordnung müssen sich die Ressourcen in der jeweiligen AWS Region befinden. In dem Beispiel unten ist `ID-A` die Verschlüsselungsschlüssel-ID für *`RegionA`* und `ID-B` die Verschlüsselungsschlüssel-ID für *`RegionB`*.

```
"artifactStores":{
   "RegionA":{
      "encryptionKey":{
         "id":"ID-A",
         "type":"KMS"
      },
      "location":"Location1",
      "type":"S3"
   },
   "RegionB":{
      "encryptionKey":{
         "id":"ID-B",
         "type":"KMS"
      },
      "location":"Location2",
      "type":"S3"
   }
}
```

Das folgende JSON-Beispiel zeigt den us-west-2-Bucket als `my-storage-bucket` und fügt den neuen us-east-1-Bucket mit dem Namen `my-storage-bucket-us-east-1` hinzu.

```
        "artifactStores": {
            "us-west-2": {
                "type": "S3",
                "location": "my-storage-bucket"
            },
            "us-east-1": {
                "type": "S3",
                "location": "my-storage-bucket-us-east-1"
            }
        },
```

4. Wenn Sie mit einer Pipeline-Struktur arbeiten, die Sie mit dem Befehl get-pipeline abgerufen haben, müssen Sie die `metadata`-Zeilen aus der JSON-Datei entfernen. Andernfalls kann der update-pipeline-Befehl sie nicht nutzen. Entfernen Sie die `"metadata": { }`-Zeilen und die Felder `"created"`, `"pipelineARN"` und `"updated"`.

   Entfernen Sie z. B. die folgenden Zeilen aus der Struktur:

```
"metadata": {
  "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
  "created": "date",
  "updated": "date"
  }
```

   Speichern Sie die Datei.

5. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

   > **Important**
   >
   > Achten Sie darauf, dass `file://` vor dem Dateinamen steht. Dies ist bei diesem Befehl erforderlich.

```
aws codepipeline update-pipeline --cli-input-json file://pipeline.json
```

   Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück. Die Ausgabe sieht folgendermaßen oder ähnlich aus.

```
{
    "pipeline": {
        "version": 4,
        "roleArn": "ARN",
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "inputArtifacts": [],
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeCommit"
                        },
                        "outputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "configuration": {
                            "PollForSourceChanges": "false",
                            "BranchName": "main",
                            "RepositoryName": "MyTestRepo"
                        },
                        "runOrder": 1
                    }
                ]
            },
            {
                "name": "Deploy",
                "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "name": "Deploy",
                        "region": "us-east-1",
                        "actionTypeId": {
```

```
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "name",
                            "DeploymentGroupName": "name"
                        },
                        "runOrder": 1
                    }
                ]
            }
        ],
        "name": "AnyCompanyPipeline",
        "artifactStores": {
            "us-west-2": {
                "type": "S3",
                "location": "my-storage-bucket"
            },
            "us-east-1": {
                "type": "S3",
                "location": "my-storage-bucket-us-east-1"
            }
        }
    }
}
```

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den **`start-pipeline-execution`**-Befehl, um Ihre Pipeline manuell zu starten.

6. Nachdem Sie Ihre Pipeline aktualisiert haben, wird die Aktion „Regionsübergreifend“ in der Konsole angezeigt.

Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline (AWS CloudFormation)

Sie können sie verwenden AWS CloudFormation , um einer vorhandenen Pipeline eine regionsübergreifende Aktion hinzuzufügen.

Um eine regionsübergreifende Aktion hinzuzufügen mit AWS CloudFormation

1. Fügen Sie den `Region`-Parameter der `ActionDeclaration`-Ressource in Ihrer Vorlage hinzu, wie in diesem Beispiel gezeigt:

```
ActionDeclaration:
              Type: Object
              Properties:
                ActionTypeId:
                  Type: ActionTypeId
                  Required: true
                Configuration:
                  Type: Map
                InputArtifacts:
                  Type: Array
                  ItemType:
                    Type: InputArtifact
                Name:
                  Type: String
                  Required: true
                OutputArtifacts:
                  Type: Array
```

```
                ItemType:
                  Type: OutputArtifact
              RoleArn:
                Type: String
              RunOrder:
                Type: Integer
              Region:
                Type: String
```

2. Fügen Sie unter `Mappings` die Regionskarte, wie in diesem Beispiel gezeigt, für eine Zuordnung mit dem Namen `SecondRegionMap` hinzu, die Werte für die Schlüssel `RegionA` und `RegionB` zuordnet. Fügen Sie unter der `Pipeline`-Ressource dem `artifactStore`-Feld die `artifactStores`-Zuweisung für Ihre neue regionsübergreifende Aktion wie folgt hinzu:

```
Mappings:
  SecondRegionMap:
    RegionA:
      SecondRegion: "RegionB"
    RegionB:
      SecondRegion: "RegionA"

...

          Properties:
            ArtifactStores:
              -
                Region: RegionB
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-RegionB
              -
                Region: RegionA
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-RegionA
```

Das folgende YAML-Beispiel zeigt den *RegionA*-Bucket `us-west-2` und fügt den neuen *RegionB*-Bucket `eu-central-1` hinzu:

```
Mappings:
  SecondRegionMap:
```

```
    us-west-2:
      SecondRegion: "eu-central-1"
    eu-central-1:
      SecondRegion: "us-west-2"

...

          Properties:
            ArtifactStores:
              -
                Region: eu-central-1
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-eu-central-1
              -
                Region: us-west-2
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-us-west-2
```

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation -Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Arbeiten mit Variablen

Einige Aktionen unter Variablen CodePipeline generieren. So verwenden Sie Variablen:

- Sie weisen einer Aktion einen Namespace zu, um die von der Aktion erzeugten Variablen für eine nachgeschaltete Aktionskonfiguration verfügbar zu machen.
- Sie konfigurieren die nachgeschaltete Aktion für den Verbrauch der von der Aktion generierten Variablen.

Sie können die Details für die einzelnen Aktionsausführungen anzeigen, um die Werte für jede Ausgabevariable anzuzeigen, die von der Aktion zur Ausführungszeit generiert wurde.

Um step-by-step Beispiele für die Verwendung von Variablen zu sehen:

- Ein Tutorial mit einer Lambda-Aktion, die Variablen aus einer Upstream-Aktion (CodeCommit) verwendet und Ausgabevariablen generiert, finden Sie unterTutorial: Variablen mit Lambda-Aufrufaktionen verwenden.
- Ein Tutorial mit einer AWS CloudFormation Aktion, die auf Stack-Ausgabevariablen aus einer CloudFormation Upstream-Aktion verweist, finden Sie unterTutorial: Eine Pipeline erstellen, die Variablen aus AWS CloudFormation Bereitstellungsaktionen verwendet.
- Ein Beispiel für eine manuelle Genehmigungsaktion mit Nachrichtentext, der auf Ausgabevariablen verweist, die in die CodeCommit Commit-ID und die Commit-Nachricht aufgelöst werden, finden Sie unterBeispiel: Variablen in manuellen Genehmigungen verwenden.
- Ein Beispiel für eine CodeBuild Aktion mit einer Umgebungsvariablen, die in den GitHub Branch-Namen aufgelöst wird, finden Sie unterBeispiel: Verwenden Sie eine BranchName Variable mit CodeBuild Umgebungsvariablen.
- CodeBuild Aktionen erzeugen alle Umgebungsvariablen, die als Teil des Builds exportiert wurden, als Variablen. Weitere Informationen finden Sie unter CodeBuild Ausgabevariablen für Aktionen. Eine Liste der Umgebungsvariablen, in denen Sie verwenden können CodeBuild, finden Sie im AWS CodeBuild Benutzerhandbuch unter Umgebungsvariablen in Build-Umgebungen.

Themen

Konfigurieren von Aktionen für Variablen

Wenn Sie einer Pipeline eine Aktion hinzufügen, können Sie ihr einen Namespace zuweisen und für den Verbrauch von Variablen aus vorherigen Aktionen konfigurieren.

Konfigurieren von Aktionen mit Variablen (Konsole)

In diesem Beispiel wird eine Pipeline mit einer CodeCommit Quellaktion und einer CodeBuild Build-Aktion erstellt. Die CodeBuild Aktion ist so konfiguriert, dass sie die von der CodeCommit Aktion erzeugten Variablen verwendet.

Wenn der Namespace nicht angegeben ist, kann in der Aktionskonfiguration nicht auf die Variablen verwiesen werden. Wenn Sie die Konsole zum Erstellen einer Pipeline verwenden, wird der Namespace für jede Aktion automatisch generiert.

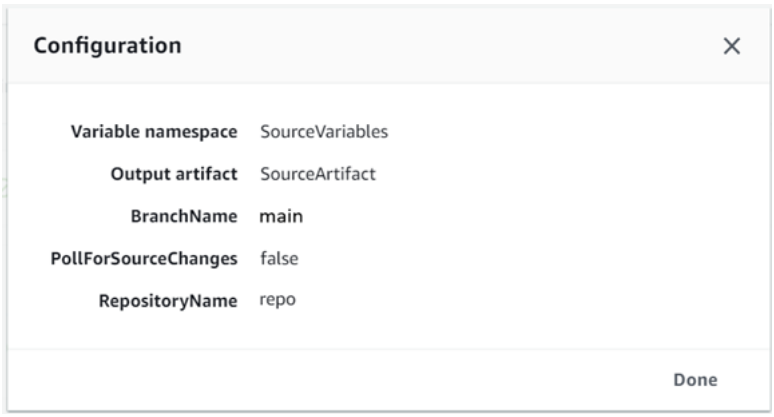
So erstellen Sie eine Pipeline mit Variablen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie Create pipeline (Pipeline erstellen) aus. Geben Sie einen Namen für die Pipeline ein und wählen Sie anschließend Next (Weiter) aus.
3. Wählen Sie unter Quelle unter Anbieter die Option aus CodeCommit. Wählen Sie das CodeCommit Repository und den Branch für die Quellaktion aus, und klicken Sie dann auf Weiter.
4. Wählen Sie unter Build unter Provider die Option aus CodeBuild. Wählen Sie den Namen eines vorhandenen CodeBuild Build-Projekts oder wählen Sie Projekt erstellen aus. Erstellen Sie unter Build-Projekt erstellen ein Build-Projekt und wählen Sie dann Zurück zu CodePipeline.

   Wählen Sie in Environment variables (Umgebungsvariablen) die Option Add environment variables (Umgebungsvariablen hinzufügen) aus. Geben Sie beispielsweise die Ausführungs-ID mit der Variablensyntax `#{codepipeline.PipelineExecutionId}` und die Commit-ID mit der Variablensyntax ein`#{SourceVariables.CommitId}`.

   > **Note**
   >
   > Sie können in jedes Aktionskonfigurationsfeld des Assistenten Variablensyntax eingeben.

5. Wählen Sie Erstellen.
6. Nach der Erstellung der Pipeline können Sie den Namespace anzeigen, der vom Assistenten erstellt wurde. Wählen Sie in der Pipeline das Symbol für die Phase aus, für die Sie den Namespace anzeigen möchten. In diesem Beispiel wird der automatisch generierte Namespace der Quellaktion angezeigt, `SourceVariables`.



So bearbeiten Sie den Namespace für eine vorhandene Aktion

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie die Pipeline aus, die Sie bearbeiten möchten, und anschließend Edit (Bearbeiten). Wählen Sie für die Quellphase Edit stage (Phase bearbeiten) aus. Fügen Sie die CodeCommit Aktion hinzu.
3. Zeigen Sie in Edit action (Aktion bearbeiten) das Feld Variable namespace (Namespace der Variable) an. Wenn die vorhandene Aktion zuvor oder ohne Verwendung des Assistenten erstellt wurde, müssen Sie einen Namespace hinzufügen. Geben Sie in Variable namespace (Namespace der Variable) einen Namespace-Namen ein. Wählen Sie anschließend Save (Speichern) aus.

So zeigen Sie Ausgabevariablen an

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wenn die Pipeline erstellt wurde und erfolgreich ausgeführt wird, können Sie die Variablen auf der Seite Action execution details (Details der Aktionsausführung) anzeigen. Weitere Informationen finden Sie unter Anzeigen von Variablen (Konsole).

Konfigurieren von Aktionen für Variablen (CLI)

Wenn Sie den Befehl create-pipeline zum Erstellen einer Pipeline oder den Befehl update-pipeline zum Bearbeiten einer Pipeline verwenden, können Sie Variablen in der Konfiguration einer Aktion referenzieren/verwenden.

Wenn kein Namespace angegeben ist, können die von dieser Aktion erzeugten Variablen in Konfigurationen für nachgeschaltete Aktionen nicht referenziert werden.

So konfigurieren Sie eine Aktion mit einem Namespace

1. Befolgen Sie die Schritte unter Erstellen Sie eine Pipeline in CodePipeline zum Erstellen einer Pipeline über die CLI. Starten Sie eine Eingabedatei, um den Befehl create-pipeline mit dem Parameter `--cli-input-json` bereitzustellen. Fügen Sie in der Pipelinestruktur den Parameter `namespace` hinzu und geben Sie einen Namen an, z. B. `SourceVariables`.

```
. . .
{
          "inputArtifacts": [],
          "name": "Source",
          "region": "us-west-2",
          "namespace": "SourceVariables",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeCommit"
          },
          "outputArtifacts": [

. . .
```

2. Speichern Sie die Datei mit einem Namen wie **`MyPipeline.json`**.
3. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer Befehlszeile (Windows) aus und erstellen Sie die Pipeline. create-pipeline

   Rufen Sie die beim Ausführen des Befehls create-pipeline erstellte Datei auf. Beispielsweise:

```
aws codepipeline create-pipeline --cli-input-json file://MyPipeline.json
```

So konfigurieren Sie nachgeschaltete Aktionen für den Verbrauch von Variablen

1. Bearbeiten Sie eine Eingabedatei, um dem Befehl update-pipeline den Parameter `--cli-input-json` bereitzustellen. Fügen Sie in der nachgeschalteten Aktion die Variable der Konfiguration für diese Aktion hinzu. Eine Variable besteht aus einem Namespace und einem Schlüssel, getrennt durch einen Punkt. Um beispielsweise Variablen für die

Pipeline-Ausführungs-ID und die Quell-Commit-ID hinzuzufügen, geben Sie den Namespace `codepipeline` für die Variable `#{codepipeline.PipelineExecutionId}` an. Geben Sie den Namespace `SourceVariables` für die Variable `#{SourceVariables.CommitId}` an.

```
{
    "name": "Build",
    "actions": [
        {
            "outputArtifacts": [
                {
                    "name": "BuildArtifacts"
                }
            ],
            "name": "Build",
            "configuration": {
                "EnvironmentVariables": "[{\"name\":\"Execution_ID\",\"value
\":\"#{codepipeline.PipelineExecutionId}\",\"type\":\"PLAINTEXT\"},{\"name\":
\"Commit_ID\",\"value\":\"#{SourceVariables.CommitId}\",\"type\":\"PLAINTEXT\"}]",
                "ProjectName": "env-var-test"
            },
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-west-2",
            "actionTypeId": {
                "provider": "CodeBuild",
                "category": "Build",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

2. Speichern Sie die Datei mit einem Namen wie **`MyPipeline.json`**.
3. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer Befehlszeile (Windows) aus und erstellen Sie die Pipeline. create-pipeline

   Rufen Sie die beim Ausführen des Befehls create-pipeline erstellte Datei auf. Beispielsweise:

```
aws codepipeline create-pipeline --cli-input-json file://MyPipeline.json
```

Anzeigen von Ausgabevariablen

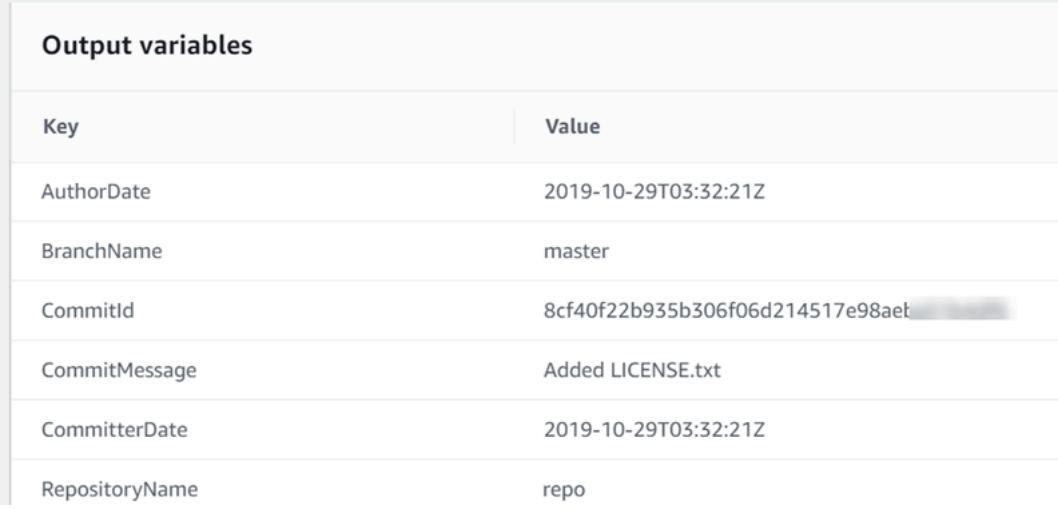
Sie können die Details der Aktionsausführung anzeigen, um die Variablen für diese Aktion anzuzeigen, spezifisch für jede Ausführung.

Anzeigen von Variablen (Konsole)

Sie können die Konsole verwenden, um Variablen für eine Aktion anzuzeigen.

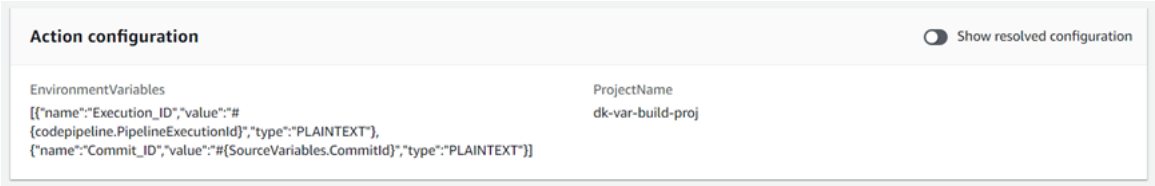
1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline.
3. Wählen Sie View history (Verlauf anzeigen).
4. Nach erfolgreicher Ausführung der Pipeline können Sie die von der Quellaktion erzeugten Variablen anzeigen. Wählen Sie View history (Verlauf anzeigen). Wählen Sie in der Aktionsliste für die Pipeline-Ausführung die Option Quelle aus, um die Details der Aktionsausführung für die CodeCommit Aktion anzuzeigen. Sie können die Variablen im Bildschirm „Action detail (Aktionsdetails)“ unter Output variables (Ausgabevariablen) anzeigen.

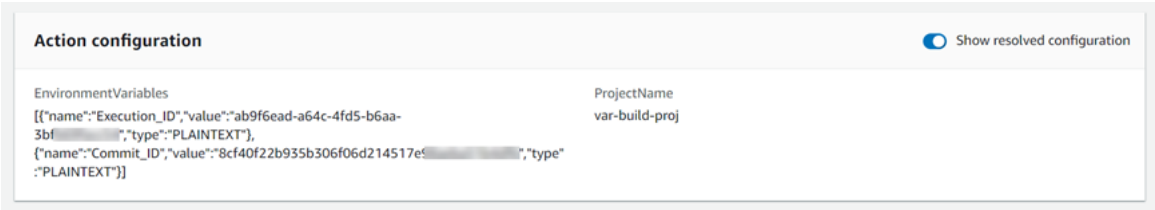


5. Nach erfolgreicher Ausführung der Pipeline können Sie die von der Build-Aktion erzeugten Variablen anzeigen. Wählen Sie View history (Verlauf anzeigen). Wählen Sie in der Aktionsliste

für die Pipeline-Ausführung die Option Build aus, um die Details zur Aktionsausführung für die CodeBuild Aktion anzuzeigen. Sie können die Variablen auf der Seite „Action detail (Aktionsdetails)“ unter Action configuration (Aktionskonfiguration) anzeigen. Der automatisch generierte Namespace wird angezeigt.



Standardmäßig wird in Action configuration (Aktionskonfiguration) die Variablensyntax angezeigt. Sie können Show resolved configuration (Aufgelöste Konfiguration anzeigen) auswählen, um zur Anzeige der Werte zu wechseln, die während der Aktionsausführung erzeugt wurden.



Anzeigen von Variablen (CLI)

Sie können den Befehl list-action-executions verwenden, um Variablen für eine Aktion anzuzeigen.

1. Verwenden Sie den folgenden Befehl:

```
aws codepipeline list-action-executions
```

Die Ausgabe zeigt den Parameter `outputVariables` wie hier gezeigt an.

```
"outputVariables": {
                    "BranchName": "main",
                    "CommitMessage": "Updated files for test",
                    "AuthorDate": "2019-11-08T22:24:34Z",
                    "CommitId": "d99b0083cc10EXAMPLE",
                    "CommitterDate": "2019-11-08T22:24:34Z",
                    "RepositoryName": "variables-repo"
                },
```

2. Verwenden Sie den folgenden Befehl:

```
aws codepipeline get-pipeline --name <pipeline-name>
```

In der Aktionskonfiguration für die CodeBuild Aktion können Sie die Variablen anzeigen:

```
{
    "name": "Build",
    "actions": [
        {
            "outputArtifacts": [
                {
                    "name": "BuildArtifact"
                }
            ],
            "name": "Build",
            "configuration": {
                "EnvironmentVariables": "[{\"name\":\"Execution_ID\",\"value
\":\"#{codepipeline.PipelineExecutionId}\",\"type\":\"PLAINTEXT\"},{\"name\":
\"Commit_ID\",\"value\":\"#{SourceVariables.CommitId}\",\"type\":\"PLAINTEXT\"}]",
                "ProjectName": "env-var-test"
            },
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-west-2",
            "actionTypeId": {
                "provider": "CodeBuild",
                "category": "Build",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

Beispiel: Variablen in manuellen Genehmigungen verwenden

Wenn Sie einen Namespace für eine Aktion angeben und diese Aktion Ausgabevariablen erzeugt, können Sie eine manuelle Genehmigung hinzufügen, die Variablen in der Genehmigungsmeldung anzeigt. In diesem Beispiel wird gezeigt, wie eine Variablensyntax zu einer manuellen Genehmigungsmeldung hinzugefügt wird.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt. Wählen Sie die Pipeline aus, der Sie die Genehmigung hinzufügen möchten.
2. Um die Pipeline zu bearbeiten, wählen Sie Edit (Bearbeiten). Fügen Sie nach der Quellaktion eine manuelle Genehmigung hinzu. Geben Sie unter Action name (Aktionsname) den Namen der Genehmigungsaktion ein.
3. Wählen Sie unter Action provider (Aktionsanbieter) die Option Manual approval (Manuelle Genehmigung).
4. Fügen Sie unter URL zur Überprüfung Ihrer CodeCommit URL die Variablensyntax für `CommitId` hinzu. Stellen Sie sicher, dass Sie den Namespace verwenden, der Ihrer Quellaktion zugewiesen ist. Die Variablensyntax für eine CodeCommit Aktion mit dem Standard-Namespace `SourceVariables` lautet `#{SourceVariables.CommitId}` beispielsweise.

   Geben Sie im Feld Kommentare in `CommitMessage` die Commit-Nachricht ein:

   ```
   Please approve this change. Commit message: #{SourceVariables.CommitMessage}
   ```
5. Nachdem die Pipeline erfolgreich läuft, können Sie die Variablenwerte in der Genehmigungsmeldung anzeigen.

Beispiel: Verwenden Sie eine BranchName Variable mit CodeBuild Umgebungsvariablen

Wenn Sie Ihrer Pipeline eine CodeBuild Aktion hinzufügen, können Sie CodeBuild Umgebungsvariablen verwenden, um auf eine `BranchName` Ausgabevariable aus einer Upstream-Quellaktion zu verweisen. Mit einer Ausgabevariablen aus einer Aktion in CodePipeline können Sie Ihre eigenen CodeBuild Umgebungsvariablen zur Verwendung in Ihren Build-Befehlen erstellen.

Dieses Beispiel zeigt Ihnen, wie Sie einer Umgebungsvariablen die Syntax einer CodeBuild Ausgabevariablen aus einer GitHub Quellaktion hinzufügen. Die Syntax der Ausgabevariablen in diesem Beispiel stellt die Ausgangsvariable für die GitHub Quellaktion dar`BranchName`. Nachdem die Aktion erfolgreich ausgeführt wurde, wird die Variable aufgelöst und zeigt den GitHub Zweignamen an.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Namen aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt. Wählen Sie die Pipeline aus, der Sie die Genehmigung hinzufügen möchten.
2. Um die Pipeline zu bearbeiten, wählen Sie Edit (Bearbeiten). Wählen Sie auf der Phase, die Ihre CodeBuild Aktion enthält, Phase bearbeiten aus.
3. Wählen Sie das Symbol, um Ihre CodeBuild Aktion zu bearbeiten.
4. Geben Sie auf der Seite Aktion bearbeiten unter Umgebungsvariablen Folgendes ein:

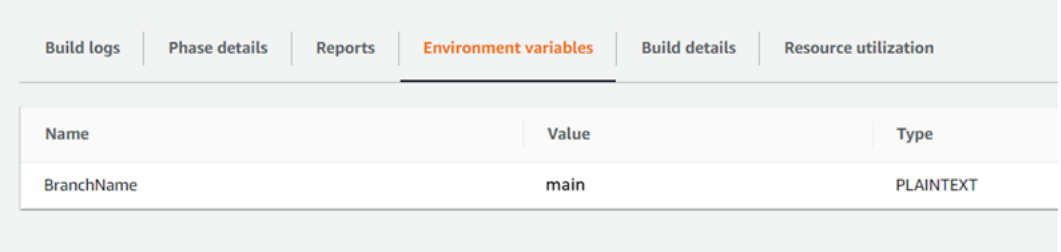
   - Geben Sie im Feld Name einen Namen für Ihre Umgebungsvariable ein.
   - Geben Sie im Feld Wert die Variablensyntax für Ihre Pipeline-Ausgabevariable ein, die den Namespace einschließt, der Ihrer Quellaktion zugewiesen ist. Die Syntax der Ausgabevariablen für eine GitHub Aktion mit dem Standard-Namespace `SourceVariables` lautet beispielsweise. `#{SourceVariables.BranchName}`
   - Wählen Sie unter Typ die Option Plaintext aus.
5. Nachdem die Pipeline erfolgreich ausgeführt wurde, können Sie sehen, wie die aufgelöste Ausgabevariable dem Wert in der Umgebungsvariablen entspricht. Wählen Sie eine der folgenden Optionen aus:

   - CodePipeline Konsole: Wählen Sie Ihre Pipeline und dann Verlauf aus. Wählen Sie die letzte Pipeline-Ausführung aus.
     - Wählen Sie unter Timeline den Selektor für Source aus. Dies ist die Quellaktion, die GitHub Ausgabevariablen generiert. Wählen Sie Ausführungsdetails anzeigen aus. Sehen Sie sich unter Ausgabevariablen die Liste der durch diese Aktion generierten Ausgabevariablen an.
     - Wählen Sie unter Timeline den Selektor für Build aus. Dies ist die Build-Aktion, die die CodeBuild Umgebungsvariablen für Ihr Build-Projekt angibt. Wählen Sie Ausführungsdetails anzeigen aus. Sehen Sie sich unter Aktionskonfiguration Ihre CodeBuild Umgebungsvariablen an. Wählen Sie Gelöste Konfiguration anzeigen aus. Ihr

Umgebungsvariablenwert ist die aufgelöste `BranchName` Ausgabevariable aus der GitHub Quellaktion. In diesem Beispiel lautet der aufgelöste Wert`main`.

Weitere Informationen finden Sie unter Anzeigen von Variablen (Konsole).

- CodeBuild Konsole: Wählen Sie Ihr Build-Projekt und wählen Sie den Link für Ihren Build-Run. Unter Umgebungsvariablen ist Ihre aufgelöste Ausgabevariable der Wert für die CodeBuild Umgebungsvariable. In diesem Beispiel ist die Umgebungsvariable Name `BranchName` und der Wert ist die aufgelöste `BranchName` Ausgabevariable aus der GitHub Quellaktion. In diesem Beispiel lautet der aufgelöste Wert`main`.



Arbeiten mit Stufenübergängen in CodePipeline

Übergänge sind Links zwischen Pipeline-Phasen, die deaktiviert oder aktiviert werden können. Standardmäßig sind sie aktiviert. Wenn Sie einen deaktivierten Übergang erneut aktivieren, wird die letzte Revision durch die verbleibenden Phasen der Pipeline geführt, es sei denn, es sind mehr als 30 Tage vergangen. Die Pipeline-Ausführung wird für Übergänge, die vor mehr als 30 Tagen deaktiviert wurden, nicht fortgesetzt, es sei denn, es wird eine neue Änderung entdeckt oder Sie führen die Pipeline manuell erneut aus.

Sie können die AWS CodePipeline Konsole oder die verwenden AWS CLI , um Übergänge zwischen Phasen in einer Pipeline zu deaktivieren oder zu aktivieren.

> **Note**
>
> Sie können eine Genehmigungsaktion verwenden, um die Ausführung einer Pipeline anzuhalten, bis die Fortsetzung manuell genehmigt wird. Weitere Informationen finden Sie unter Genehmigungsaktionen verwalten in CodePipeline.

Themen

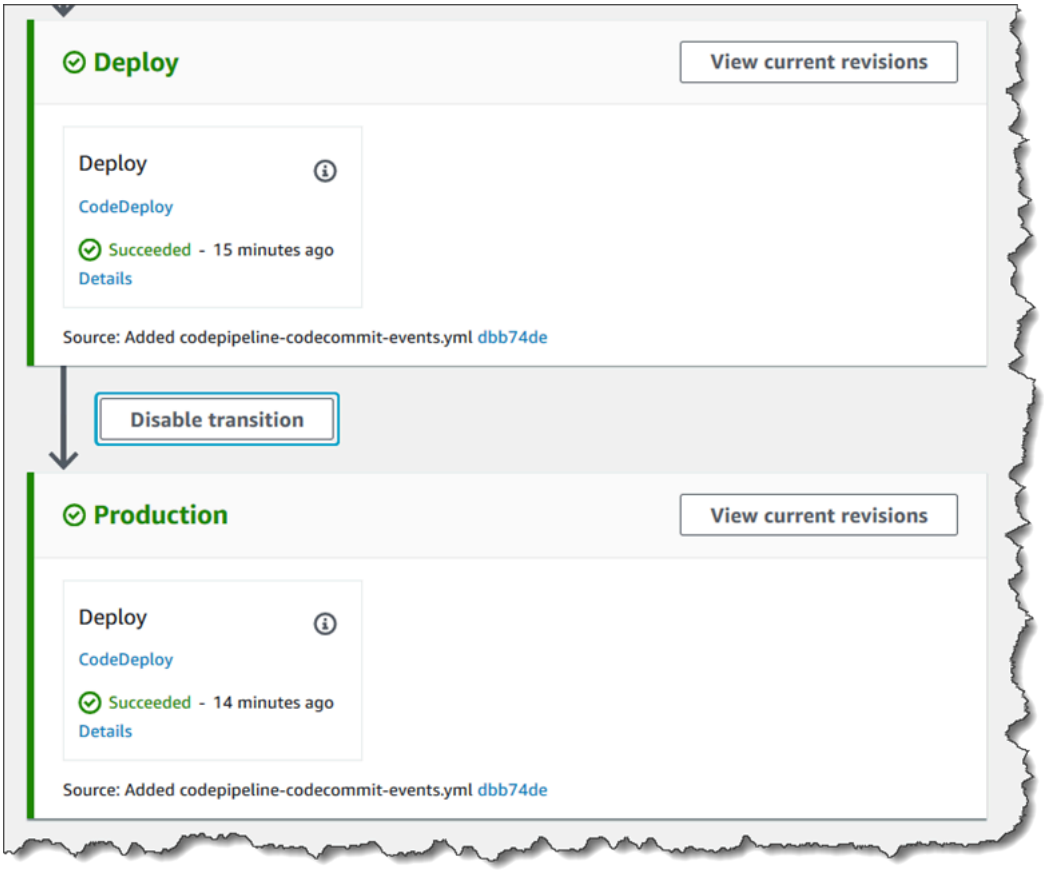
Deaktivieren oder Aktivieren von Übergängen (Konsole)

So deaktivieren oder aktivieren Sie Übergänge in einer Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

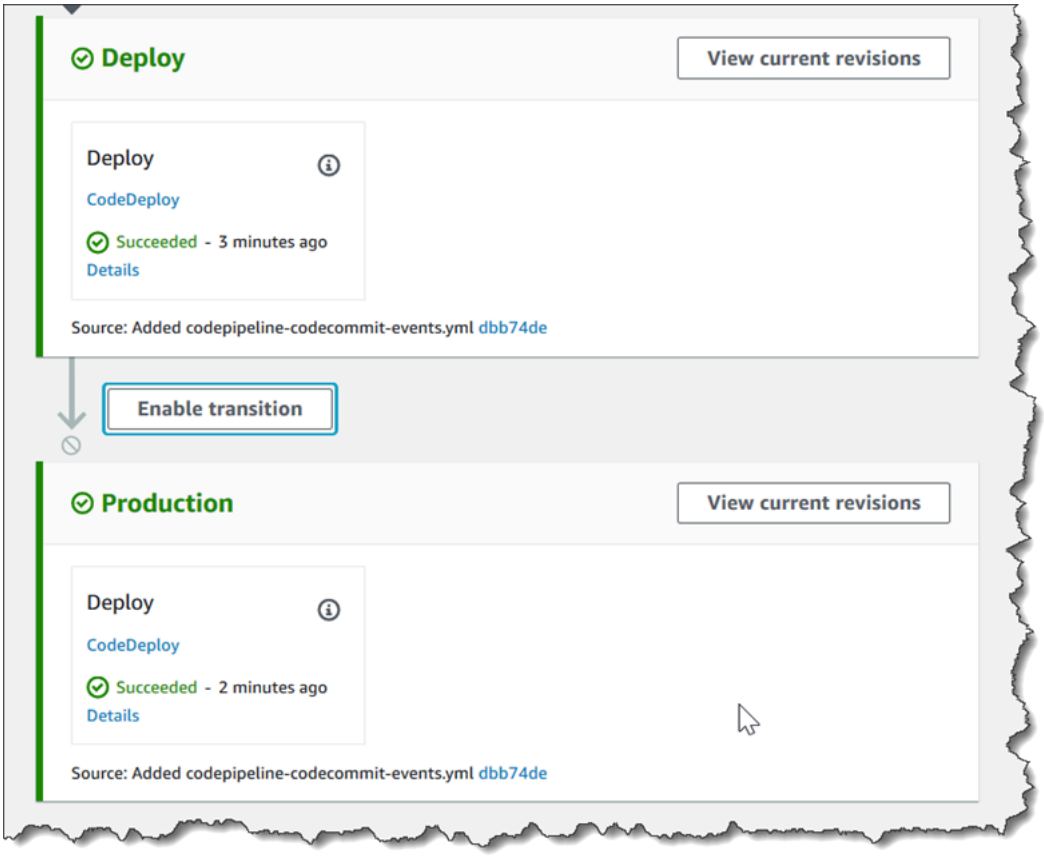
   Es werden die Namen aller Ihrem AWS -Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, für die Sie Übergänge aktivieren oder deaktivieren möchten. Eine detaillierte Ansicht der Pipeline wird geöffnet (einschließlich der Übergänge zwischen den Stufen der Pipeline).
3. Suchen Sie den Pfeil nach der letzten Stufe, die ausgeführt werden soll, und wählen Sie dann die Schaltfläche neben ihr aus. Wenn Sie beispielsweise bei der folgenden Pipeline möchten,

dass die Aktionen in der Stufe Staging ausgeführt werden, jedoch nicht die Aktionen in der Stufe mit dem Namen Production, dann klicken Sie zwischen den beiden Stufen auf die Schaltfläche Disable transition (Übergang deaktivieren):



4. Geben Sie im Dialogfeld Disable transition (Übergang deaktivieren) einen Grund für das Deaktivieren des Übergangs ein und klicken Sie dann auf Disable (Deaktivieren).

   Die Schaltfläche verwandelt sich, um anzuzeigen, dass Übergänge zwischen den Stufen vor und nach dem Pfeil deaktiviert sind. Alle Revisionen, die nach der Deaktivierung des Übergangs bereits in den Stufen ausgeführt werden, durchlaufen die Pipeline weiterhin. Alle folgenden Revisionen werden jedoch nur bis zum deaktivierten Übergang ausgeführt.



5. Klicken Sie auf die Schaltfläche Enable transition (Übergang aktivieren) neben dem Pfeil. Wählen Sie im Dialogfeld Enable transition die Option Enable aus. Die Pipeline aktiviert sofort den Übergang zwischen den beiden Stufen. Wenn nach dem Deaktivieren des Übergangs Revisionen die früheren Stufen durchlaufen haben, lässt die Pipeline nach einigen Momenten die neueste Revision die Stufen nach dem zuvor deaktivierten Übergang durchlaufen. Die Pipeline lässt die Revision durch alle verbleibenden Stufen in der Pipeline laufen.

> **Note**
>
> Es kann einige Sekunden dauern, bis die Änderungen in der CodePipeline Konsole angezeigt werden, nachdem Sie den Übergang aktiviert haben.

Deaktivieren oder Aktivieren von Übergängen (CLI)

Um einen Übergang zwischen Stufen mithilfe von zu deaktivieren AWS CLI, führen Sie den disable-stage-transition Befehl aus. Um einen deaktivierten Übergang zu aktivieren, führen Sie den Befehl enable-stage-transition aus.

So deaktivieren Sie einen Übergang

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und führen Sie den AWS CLI disable-stage-transitionBefehl mit dem aus. Geben Sie dabei den Namen der Pipeline, den Namen der Phase, zu der Sie Übergänge deaktivieren möchten, den Übergangstyp und den Grund, warum Sie Übergänge zu dieser Phase deaktivieren, an. Anders als bei Verwendung der Konsole müssen Sie auch angeben, ob Sie Übergänge in die Stufe (eingehend) oder Übergänge aus der Stufe nach Abschluss aller Aktionen (ausgehend) deaktivieren möchten.

   Um beispielsweise den Übergang zu einer Phase namens *Staging* in einer Pipeline mit dem Namen zu deaktivieren *MyFirstPipeline*, geben Sie einen Befehl ein, der dem folgenden ähnelt:

   ```
   aws codepipeline disable-stage-transition --pipeline-name MyFirstPipeline --stage-
   name Staging --transition-type Inbound --reason "My Reason"
   ```

   Der Befehl gibt nichts zurück.

2. Um zu überprüfen, ob der Übergang deaktiviert wurde, zeigen Sie entweder die Pipeline in der CodePipeline Konsole an oder führen Sie den get-pipeline-state Befehl aus. Weitere Informationen finden Sie unter Pipelines anzeigen (Konsole) und Anzeigen von der Pipeline-Details und des Verlaufs (CLI).

So aktivieren Sie einen Übergang

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und führen Sie den AWS CLI enable-stage-transitionBefehl mit dem aus. Geben Sie dabei den Namen der Pipeline, den Namen der Phase, zu der Sie Übergänge aktivieren möchten, und den Übergangstyp an.

   Um beispielsweise den Übergang zu einer Phase namens *Staging* in einer Pipeline mit dem Namen zu ermöglichen *MyFirstPipeline*, geben Sie einen Befehl ein, der dem folgenden ähnelt:

   ```
   aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stage-
   name Staging  --transition-type Inbound
   ```

   Der Befehl gibt nichts zurück.

2. Um zu überprüfen, ob der Übergang deaktiviert wurde, zeigen Sie entweder die Pipeline in der CodePipeline Konsole an oder führen Sie den get-pipeline-state Befehl aus. Weitere Informationen finden Sie unter Pipelines anzeigen (Konsole) und Anzeigen von der Pipeline-Details und des Verlaufs (CLI).

Überwachung von Pipelines

Überwachung ist wichtig, um die Zuverlässigkeit, Verfügbarkeit und Performance von AWS CodePipeline aufrechtzuerhalten. Sie sollten Überwachungsdaten aus allen Teilen Ihrer AWS Lösung sammeln, damit Sie einen etwaigen Ausfall an mehreren Stellen leichter debuggen können. Bevor Sie mit der Überwachung beginnen, sollten Sie einen Überwachungsplan erstellen, der die folgenden Fragen beantwortet:

- Was sind Ihre Überwachungsziele?
- Welche Ressourcen möchten Sie überwachen?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungstools stehen Ihnen zur Verfügung?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer sollte benachrichtigt werden, wenn etwas schiefgeht?

Sie können die folgenden Tools verwenden, um Ihre CodePipeline Pipelines und deren Ressourcen zu überwachen:

- EventBridge Event-Bus-Ereignisse — Sie können CodePipeline Ereignisse in überwachen EventBridge, wodurch Änderungen am Ausführungsstatus Ihrer Pipeline, Phase oder Aktion erkannt werden. EventBridge leitet diese Daten an Ziele wie AWS Lambda Amazon Simple Notification Service weiter. EventBridge Ereignisse sind dieselben wie die, die in Amazon CloudWatch Events erscheinen.
- Benachrichtigungen für Pipeline-Ereignisse in der Developer Tools-Konsole — Sie können CodePipeline Ereignisse mit Benachrichtigungen überwachen, die Sie in der Konsole eingerichtet haben, und dann ein Amazon Simple Notification Service-Thema und ein Abonnement für Amazon Simple Notification Service erstellen. Weitere Informationen finden Sie unter Was sind Benachrichtigungen im Developer Tools Console-Benutzerhandbuch.
- AWS CloudTrail— Wird verwendet CloudTrail , um API-Aufrufe, die von oder im Namen Ihres AWS Kontos getätigt wurden, zu erfassen und die Protokolldateien an einen Amazon S3 S3-Bucket zu übermitteln. CodePipeline Sie können festlegen, dass Amazon SNS SNS-Benachrichtigungen CloudWatch veröffentlicht werden, wenn neue Protokolldateien geliefert werden, damit Sie schnell handeln können.
- Konsole und CLI — Sie können die CodePipeline Konsole und die CLI verwenden, um Details zum Status einer Pipeline oder einer bestimmten Pipeline-Ausführung anzuzeigen.

Themen

CodePipeline Ereignisse überwachen

Sie können CodePipeline Ereignisse in überwachen EventBridge, wodurch ein Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, software-as-a-service (SaaS-) Anwendungen und bereitgestellt wird AWS-Services. EventBridge leitet diese Daten an Ziele wie AWS Lambda Amazon Simple Notification Service weiter. Diese Ereignisse sind identisch mit denen, die in Amazon CloudWatch Events erscheinen. Amazon Events liefert einen Stream von Systemereignissen, die Änderungen an AWS Ressourcen beschreiben, nahezu in Echtzeit. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? im EventBridge Amazon-Benutzerhandbuch.

> **Note**
>
> Amazon EventBridge ist die bevorzugte Methode, um Ihre Veranstaltungen zu verwalten. Amazon CloudWatch Events und EventBridge sind derselbe zugrunde liegende Service und dieselbe API, EventBridge bieten aber mehr Funktionen. Änderungen, die Sie entweder in CloudWatch Events oder vornehmen EventBridge , werden in jeder Konsole angezeigt.

Ereignisse bestehen aus Regeln. Eine Regel wird konfiguriert, indem Sie Folgendes wählen:

- Ereignismuster. Jede Regel wird als ein Ereignismuster mit der Quelle und Art der zu überwachenden Ereignisse sowie den Ereigniszielen ausgedrückt. Um Ereignisse zu überwachen, erstellen Sie eine Regel mit dem Dienst, den Sie überwachen, als Ereignisquelle, z. CodePipeline B. Sie können beispielsweise eine Regel mit einem Ereignismuster erstellen, das CodePipeline als Ereignisquelle verwendet wird, um die Regel auszulösen, wenn sich der Status einer Pipeline, Phase oder Aktion ändert.
- Ziele. Die neue Regel erhält einen ausgewählten Service als Ereignisziel. Möglicherweise möchten Sie einen Zieldienst einrichten, um Benachrichtigungen zu senden, Statusinformationen zu erfassen, Korrekturmaßnahmen zu ergreifen, Ereignisse einzuleiten oder andere Aktionen zu ergreifen. Wenn Sie Ihr Ziel hinzufügen, müssen Sie ihm auch Berechtigungen erteilen, EventBridge damit es den ausgewählten Zieldienst aufrufen kann.

Alle Arten von Änderungsereignissen in Bezug auf den Ausführungszustand geben Benachrichtigungen mit spezifischen Nachrichteninhalten aus. Dabei gilt Folgendes:

- Der erste `version` Eintrag zeigt die Versionsnummer für das Ereignis.
- Der `version`-Eintrag unter Pipeline-`detail` zeigt die Versionsnummer der Pipelinestruktur.
- Der `execution-id`-Eintrag unter Pipeline-`detail` zeigt die Ausführungs-ID für die Pipelineausführung, die die Statusänderung verursacht hat. Weitere Informationen zum GetPipelineExecution API-Aufruf finden Sie in der AWS CodePipeline API-Referenz.
- Der `pipeline-execution-attempt` Eintrag zeigt die Anzahl der Versuche oder Wiederholungen für die spezifische Ausführungs-ID.

CodePipeline meldet EventBridge jedes Mal, wenn sich der Status einer Ressource in Ihrem System AWS-Konto ändert, ein Ereignis. Ereignisse werden auf garantierter at-least-once Basis für die folgenden Ressourcen ausgelöst:

- Pipeline-Ausführungen
- Hinrichtungen in Etappen
- Aktionsausführungen

Ereignisse werden EventBridge mit dem oben beschriebenen Ereignismuster und Schema ausgegeben. Für verarbeitete Ereignisse, z. B. Ereignisse, die Sie über Benachrichtigungen erhalten, die Sie in der Developer Tools-Konsole konfiguriert haben, enthält die Ereignisnachricht Felder für das Ereignismuster mit einigen Variationen. Das `detail-type` Feld wird beispielsweise konvertiert in`detailType`. Weitere Informationen finden Sie unter dem PutEvents API-Aufruf in der Amazon EventBridge API-Referenz.

Die folgenden Beispiele zeigen Ereignisse für CodePipeline. Wo möglich, zeigt jedes Beispiel das Schema für ein ausgelöstes Ereignis zusammen mit dem Schema für ein verarbeitetes Ereignis.

Themen

Detailtypen

Wenn Sie Ereignisse für die Überwachung einrichten, können Sie den Detailtyp für das Ereignis auswählen.

Sie können konfigurieren, dass Benachrichtigungen gesendet werden, wenn sich der Status für Folgendes ändert:

- Angegebene Pipelines oder alle Pipelines. Dies wird über `"detail-type": "CodePipeline Pipeline Execution State Change"` gesteuert.
- Angegebene Stufen oder alle Stufen innerhalb einer bestimmten Pipeline oder all Ihrer Pipelines. Dies wird über `"detail-type": "CodePipeline Stage Execution State Change"` gesteuert.
- Angegebene Aktionen oder alle Aktionen in einer bestimmten Stufe oder allen Stufen innerhalb einer bestimmten Pipeline oder allen Ihren Pipelines. Dies wird über `"detail-type": "CodePipeline Action Execution State Change"` gesteuert.

> **Note**
>
> Ereignisse, die von ausgegeben werden, EventBridge enthalten den `detail-type` Parameter, der `detailType` bei der Verarbeitung von Ereignissen in diesen Wert umgewandelt wird.

Typ des Details	Status	Beschreibung
CodePipeline Änderung des Status der Pipeline-Ausführung	CANCELED	Die Pipelineausführung wurde abgebrochen, da die Pipelinestruktur aktualisiert wurde.
	FEHLGESCHLAGEN	Die Pipelineausführung wurde nicht erfolgreich abgeschlossen.
	RESUMED	Eine fehlgeschlagene Pipelineausführung wurde als Reaktion auf den API-Aufruf `RetryStageExecution` erneut ausgeführt.
	STARTED	Die Pipelineausführung wird derzeit ausgeführt.

Typ des Details	Status	Beschreibung
	STOPPED	Der Anhaltevorgang ist abgeschlossen, und die Pipeline-Ausführung wird angehalten.
	STOPPING	Die Pipeline-Ausführung wird aufgrund einer Anforderung mit "Anhalten und beenden" oder "Anhalten und warten" angehalten.
	SUCCEEDED	Die Pipelineausführung wurde erfolgreich abgeschlossen.
	SUPERSEDED	Während auf den Abschluss der nächsten Stufe dieser Pipelineausführung gewartet wurde, wurde stattdess en eine neuere Pipelineausführung über die Pipeline fortgesetzt.
CodePipel ine Änderung des Ausführun gsstatus der Phase	CANCELED	Die Stufe wurde abgebrochen, da die Pipelinestruktur aktualisiert wurde.
	FEHLGESCH LAGEN	Die Stufe wurde nicht erfolgreich abgeschlossen.
	RESUMED	Eine fehlgeschlagene Stufe wurde als Reaktion auf den API-Aufruf `RetryStageExecution` erneut ausgeführt.
	STARTED	Die Stufe wird derzeit ausgeführt.
	STOPPED	Der Anhaltevorgang ist abgeschlossen, und die Phasenausführung wird angehalten.
	STOPPING	Die Ausführung der Phase wird aufgrund einer Anforderu ng zum Anhalten und Warten oder zum Anhalten und Beenden der Pipeline-Ausführung gestoppt.
	SUCCEEDED	Die Stufe wurde erfolgreich abgeschlossen.

Typ des Details	Status	Beschreibung
CodePipeline Änderung des Ausführungsstatus der Aktion	VERLASSEN	Die Aktion wird aufgrund einer Anforderung mit "Anhalten und Beenden" für die Pipeline-Ausführung beendet.
	CANCELED	Die Aktion wurde abgebrochen, da die Pipelinestruktur aktualisiert wurde.
	FEHLGESCHLAGEN	Bei Genehmigungsaktionen bedeutet der Status FAILED (FEHLGESCHLAGEN), dass die Aktion entweder vom Prüfer abgelehnt wurde oder aufgrund einer falschen Aktionskonfiguration fehlgeschlagen ist.
	STARTED	Die Aktion wird derzeit ausgeführt.
	SUCCEEDED	Die Aktion wurde erfolgreich abgeschlossen.

Ereignisse auf Pipeline-Ebene

Ereignisse auf Pipeline-Ebene werden ausgelöst, wenn sich der Status einer Pipeline-Ausführung ändert.

Themen

Ereignis „Pipeline STARTED“

Wenn eine Pipeline-Ausführung gestartet wird, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline, die `"myPipeline"` in der `us-east-1` Region benannt ist. Das `id` Feld steht für die Ereignis-ID und das `account` Feld für die Konto-ID, unter der die Pipeline erstellt wurde.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:03:07Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "trigger-type": "StartPipelineExecution",
            "trigger-detail": "arn:aws:sts::123456789012:assumed-role/Admin/my-user"
        },
        "state": "STARTED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:44:50Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
east-1:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "trigger-type": "StartPipelineExecution",
```

```
            "trigger-detail": "arn:aws:sts::123456789012:assumed-role/Admin/my-user"
        },
        "state": "STARTED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

Ereignis zum STOPPEN DER Pipeline

Wenn die Ausführung einer Pipeline angehalten wird, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline, die `myPipeline` in der `us-west-2` Region benannt ist.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:02:20Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "STOPPING",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
        "stop-execution-comments": "Stopping the pipeline for an update"
    }
}
```

Ereignis Pipeline SUCCEED

Wenn eine Pipeline-Ausführung erfolgreich ist, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline, die `myPipeline` in der `us-east-1` Region benannt ist.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:03:44Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "SUCCEEDED",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-30T22:13:51Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
```

```
        "state": "SUCCEEDED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

Pipeline ERFOLGREICH (Beispiel mit Git-Tags)

Wenn eine Pipeline-Ausführung in einer Phase wiederholt und erfolgreich versucht wurde, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline, die `myPipeline` in der `eu-central-1` Region benannt `execution-trigger` ist, in der die für Git-Tags konfiguriert ist.

> **Note**
>
> Das `execution-trigger` Feld enthält entweder `tag-name` oder`branch-name`, je nachdem, welche Art von Ereignis die Pipeline ausgelöst hat.

```
{
    "version": "0",
    "id": "b128b002-09fd-4574-4eba-27152726c777",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2023-10-26T13:50:53Z",
    "region": "eu-central-1",
    "resources": [
        "arn:aws:codepipeline:eu-central-1:123456789012:BuildFromTag"
    ],
    "detail": {
        "pipeline": "BuildFromTag",
        "execution-id": "e17b5773-cc0d-4db2-9ad7-594c73888de8",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "author-display-name": "Mary Major",
            "full-repository-name": "mmajor/sample-project",
```

```
            "provider-type": "GitLab",
            "author-email": "email_address",
            "commit-message": "Update file README.md",
            "author-date": "2023-08-16T21:08:08Z",
            "tag-name": "gitlab-v4.2.1",
            "commit-id": "commit_ID",
            "connection-arn": "arn:aws:codestar-connections:eu-
central-1:123456789012:connection/0f5b706a-1a1d-46c5-86b6-f177321bcfb2",
            "author-id": "Mary Major"
        },
        "state": "SUCCEEDED",
        "version": 32.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

Ereignis Pipeline FAILED

Wenn die Ausführung einer Pipeline fehlschlägt, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline, die "`myPipeline`" in der `us-west-2` Region benannt ist.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "FAILED",
        "version": 4.0,
        "pipeline-execution-attempt": 1.0
    }
```

```
}
```

Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:46:16Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "FAILED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "failedActionCount": 1,
        "failedActions": [
            {
                "action": "Deploy",
                "additionalInformation": "Deployment <ID> failed"
            }
        ],
        "failedStage": "Deploy"
    }
```

Pipeline FEHLGESCHLAGEN (Beispiel mit Git-Tags)

Sofern sie nicht in der Quellphase fehlschlägt, gibt sie bei einer Pipeline, die mit Triggern konfiguriert ist, ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline, die `myPipeline` in der `eu-central-1` Region benannt `execution-trigger` ist, in der die für Git-Tags konfiguriert ist.

> **Note**
>
> Das `execution-trigger` Feld enthält entweder `tag-name` oder`branch-name`, je nachdem, welche Art von Ereignis die Pipeline ausgelöst hat.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "author-display-name": "Mary Major",
            "full-repository-name": "mmajor/sample-project",
            "provider-type": "GitLab",
            "author-email": "email_address",
            "commit-message": "Update file README.md",
            "author-date": "2023-08-16T21:08:08Z",
            "tag-name": "gitlab-v4.2.1",
            "commit-id": "commit_ID",
            "connection-arn": "arn:aws:codestar-connections:eu-
central-1:123456789012:connection/0f5b706a-1a1d-46c5-86b6-f177321bcfb2",
            "author-id": "Mary Major"
        },
        "state": "FAILED",
        "version": 4.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

Processed event


```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:46:16Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "author-display-name": "Mary Major",
            "full-repository-name": "mmajor/sample-project",
            "provider-type": "GitLab",
            "author-email": "email_address",
            "commit-message": "Update file README.md",
            "author-date": "2023-08-16T21:08:08Z",
            "tag-name": "gitlab-v4.2.1",
            "commit-id": "commit_ID",
            "connection-arn": "arn:aws:codestar-connections:eu-
central-1:123456789012:connection/0f5b706a-1a1d-46c5-86b6-f177321bcfb2",
            "author-id": "Mary Major"
        },
        "state": "FAILED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "failedActionCount": 1,
        "failedActions": [
            {
                "action": "Deploy",
                "additionalInformation": "Deployment <ID> failed"
            }
        ],
        "failedStage": "Deploy"
```

```
    }
```

Ereignisse auf Phasenebene

Ereignisse auf Phasenebene werden ausgelöst, wenn sich der Status einer Stufenausführung ändert.

Themen

- Ereignis „Phase STARTED“
- Ereignis STAGE STOPPING
- Ereignis „Phase STOPPED“
- Die Phase wurde nach dem Wiederholungsereignis der Phase FORTGESETZT

Ereignis „Phase STARTED“

Wenn die Ausführung einer Phase beginnt, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline, die `"myPipeline"` in der `us-east-1` Region benannt ist, für die Phase`Prod`.

Emitted event

```
{
    "version": "0",
    "id": 01234567-EXAMPLE,
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": 123456789012,
    "time": "2020-01-24T22:03:07Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "version": 1.0,
        "execution-id": 12345678-1234-5678-abcd-12345678abcd,
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Prod",
        "state": "STARTED",
        "pipeline-execution-attempt": 1.0
```

```
    }
}
```

Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Stage Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:45:40Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Source",
        "state": "STARTED",
        "version": 1.0,
        "pipeline-execution-attempt": 0.0
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "sourceActions": [
            {
                "sourceActionName": "Source",
                "sourceActionProvider": "CodeCommit",
                "sourceActionVariables": {
                    "BranchName": "main",
                    "CommitId": "<ID>",
                    "RepositoryName": "my-repo"
                }
            }
        ]
    }
}
```

Ereignis STAGE STOPPING

Wenn die Ausführung einer Phase angehalten wird, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline, die `myPipeline` in der `us-west-2` Region benannt ist, für die Phase`Deploy`.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:02:20Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Deploy",
        "state": "STOPPING",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

Ereignis „Phase STOPPED“

Wenn die Ausführung einer Phase gestoppt wird, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline, die `myPipeline` in der `us-west-2` Region benannt ist, für die Phase`Deploy`.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:21:39Z",
    "region": "us-west-2",
```

```
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Deploy",
        "state": "STOPPED",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```


Die Phase wurde nach dem Wiederholungsereignis der Phase FORTGESETZT

Wenn die Ausführung einer Phase wieder aufgenommen wird und eine Phase erneut versucht wurde, wird ein Ereignis ausgelöst, das Benachrichtigungen mit folgendem Inhalt sendet.

Wenn eine Phase erneut versucht wurde, wird das `stage-last-retry-attempt-time` Feld angezeigt, wie im Beispiel gezeigt. Das Feld wird bei allen Etappenereignissen angezeigt, ob ein Wiederholungsversuch durchgeführt wurde.

> **Note**
>
> Das `stage-last-retry-attempt-time` Feld wird in allen nachfolgenden Etappenereignissen vorhanden sein, nachdem eine Phase erneut versucht wurde.


```
{
    "version": "0",
    "id": "38656bcd-a798-5f92-c738-02a71be484e1",
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2023-10-26T14:14:56Z",
    "region": "eu-central-1",
    "resources": [
        "arn:aws:codepipeline:eu-central-1:123456789012:BuildFromTag"
    ],
    "detail": {
```

```
        "pipeline": "BuildFromTag",
        "execution-id": "05dafb6a-5a56-4951-a858-968795364846",
        "stage-last-retry-attempt-time": "2023-10-26T14:14:56.305Z",
        "stage": "Build",
        "state": "RESUMED",
        "version": 32.0,
        "pipeline-execution-attempt": 2.0
    }
}
```

Ereignisse auf Aktionsebene

Ereignisse auf Aktionsebene werden ausgelöst, wenn sich der Status einer Aktionsausführung ändert.

Themen

- Ereignis „Aktion GESTARTET“
- Aktion war erfolgreich (Ereignis)
- Ereignis Aktion FEHLGESCHLAGEN
- Aktion ABGEBROCHENES Ereignis

Ereignis „Aktion GESTARTET“

Wenn die Ausführung einer Aktion gestartet wird, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die `myPipeline` in der `us-east-1` Region angegebene Pipeline für die Bereitstellungsaktion`myAction`.

Emitted event

```
{
    "version": "0",
    "id": 01234567-EXAMPLE,
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": 123456789012,
    "time": "2020-01-24T22:03:07Z",
    "region": "us-east-1",
    "resources": [
```

```
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": 12345678-1234-5678-abcd-12345678abcd,
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Prod",
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "myAction",
        "state": "STARTED",
        "type": {
            "owner": "AWS",
            "category": "Deploy",
            "provider": "CodeDeploy",
            "version": "1"
        },
        "version": 2.0
        "pipeline-execution-attempt": 1.0
        "input-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-east-1-BUCKETEXAMPLE",
                    "key": "myPipeline/SourceArti/KEYEXAMPLE"
                }
            }
        ]
    }
}
```

Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Action Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:45:44Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
```

```
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Deploy",
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "input-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-east-1-EXAMPLE",
                    "key": "myPipeline/SourceArti/EXAMPLE"
                }
            }
        ],
        "state": "STARTED",
        "region": "us-east-1",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

Aktion war erfolgreich (Ereignis)

Wenn eine Aktion erfolgreich ausgeführt wird, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die `"myPipeline"` in der `us-west-2` Region angegebene Pipeline und auf die Quellaktion. `"Source"` Für diesen Ereignistyp gibt es zwei verschiedene `region` Felder. Das `region` Ereignisfeld gibt die Region für das Pipeline-Ereignis an. Das `region` Feld unter dem `detail` Abschnitt gibt die Region für die Aktion an.

Emitted event

```
{
    "version": "0",
```

```
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:03:11Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Source",
        "execution-result": {
            "external-execution-url": "https://us-west-2.console.aws.amazon.com/
codecommit/home#/repository/my-repo/commit/8cf40f2EXAMPLE",
            "external-execution-summary": "Added LICENSE.txt",
            "external-execution-id": "8cf40fEXAMPLE"
        },
        "output-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-west-2-BUCKETEXAMPLE",
                    "key": "myPipeline/SourceArti/KEYEXAMPLE"
                }
            }
        ],
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Source",
        "state": "SUCCEEDED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeCommit",
            "category": "Source",
            "version": "1"
        },
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

Processed event


```
{
    "account": "123456789012",
    "detailType": "CodePipeline Action Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:45:44Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:ACCOUNT:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "arn:aws:codepipeline:us-west-2:123456789012:myPipeline",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Source",
        "execution-result": {
            "external-execution-url": "https://us-west-2.console.aws.amazon.com/
codecommit/home#/repository/my-repo/commit/8cf40f2EXAMPLE",
            "external-execution-summary": "Edited index.html",
            "external-execution-id": "36ab3ab7EXAMPLE"
        },
        "output-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-west-2-EXAMPLE",
                    "key": "myPipeline/SourceArti/EXAMPLE"
                }
            }
        ],
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Source",
        "state": "SUCCEEDED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeCommit",
            "category": "Source",
            "version": "1"
        },
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
```

```
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

Ereignis Aktion FEHLGESCHLAGEN

Wenn die Ausführung einer Aktion fehlschlägt, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die `"myPipeline"` in der `us-west-2` Region angegebene Pipeline für die Aktion`"Deploy"`.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Deploy",
        "execution-result": {
            "external-execution-url": "https://us-west-2.console.aws.amazon.com/
codedeploy/home?#/deployments/<ID>",
            "external-execution-summary": "Deployment <ID> failed",
            "external-execution-id": "<ID>",
            "error-code": "JobFailed"
        },
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "state": "FAILED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
```

```
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 4.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Action Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:46:16Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "stage": "Deploy",
        "execution-result": {
            "external-execution-url": "https://console.aws.amazon.com/codedeploy/
home?region=us-west-2#/deployments/<ID>",
            "external-execution-summary": "Deployment <ID> failed",
            "external-execution-id": "<ID>",
            "error-code": "JobFailed"
        },
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "state": "FAILED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 13.0,
        "pipeline-execution-attempt": 1.0
    },
```

```
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "additionalInformation": "Deployment <ID> failed"
    }
}
```

Aktion ABGEBROCHENES Ereignis

Wenn die Ausführung einer Aktion abgebrochen wird, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die `"myPipeline"` in der `us-west-2` Region angegebene Pipeline für die Aktion`"Deploy"`.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:21:39Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "stage": "Deploy",
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "state": "ABANDONED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
```

```
}
```

Erstellen Sie eine Regel, die bei einem Pipeline-Ereignis eine Benachrichtigung sendet

Eine Regel überwacht bestimmte Ereignisse und leitet sie dann an von Ihnen gewählte AWS Ziele weiter. Sie können eine Regel erstellen, die eine AWS Aktion automatisch ausführt, wenn eine andere AWS Aktion stattfindet, oder eine Regel, die eine AWS Aktion regelmäßig nach einem festgelegten Zeitplan ausführt.

Themen

Eine Benachrichtigung senden, wenn sich der Pipeline-Status ändert (Konsole)

Diese Schritte zeigen, wie Sie mit der EventBridge Konsole eine Regel zum Senden von Benachrichtigungen über Änderungen in erstellen CodePipeline.

Um eine EventBridge Regel zu erstellen, die auf Ihre Pipeline mit einer Amazon S3 S3-Quelle abzielt

1. Öffnen Sie die EventBridge Amazon-Konsole unter https://console.aws.amazon.com/events/.
2. Wählen Sie im Navigationsbereich Regeln aus. Lassen Sie den Standardbus ausgewählt oder wählen Sie einen Event-Bus. Wählen Sie Regel erstellen aus.
3. Geben Sie im Feld Name einen Namen für Ihre Regel ein.
4. Wählen Sie unter Regeltyp die Option Regel mit einem Ereignismuster aus. Wählen Sie Weiter aus.
5. Wählen Sie unter Ereignismuster die Option AWS Dienste aus.
6. Wählen Sie in der Dropdown-Liste Event Type die Ebene der Statusänderung für die Benachrichtigung aus.

   - Wählen Sie für eine Regel, die für Ereignisse auf Pipeline-Ebene gilt, die Option CodePipelinePipeline Execution State Change aus.
   - Wählen Sie für eine Regel, die für Ereignisse auf Phasenebene gilt, die Option CodePipelineStage Execution State Change aus.

- Wählen Sie für eine Regel, die für Ereignisse auf Aktionsebene gilt, die Option Änderung des Ausführungsstatus der CodePipelineAktion aus.

7. Geben Sie die Änderungen an, für die die Regel gilt:

   - Wählen Sie für eine Regel, die für alle Statusänderungen gilt, Any state aus.
   - Wählen Sie für eine Regel, die nur für einige Statusänderungen gilt, Specific state(s) und dann einen oder mehrere Statuswerte aus der Liste aus.

8. Bei Ereignismustern, die detaillierter sind, als es die Selektoren zulassen, können Sie auch die Option Muster bearbeiten im Fenster Ereignismuster verwenden, um ein Ereignismuster im JSON-Format festzulegen.

   > **Note**
   >
   > Falls nicht anders angegeben, wird das Ereignis für alle Pipelines/Stufen/Aktionen und Status erstellt.

   Für detailliertere Ereignismuster können Sie die folgenden Beispiel-Eventmuster kopieren und in das Fenster Ereignismuster einfügen.

   - Example

     Verwenden Sie dieses Beispiel-Ereignismuster, um fehlgeschlagene Bereitstellungs- und Erstellungsaktionen in allen Pipelines zu erfassen.

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Deploy", "Build"]
    }
```

```
  }
}
```

- Example

  Verwenden Sie dieses Beispiel-Ereignismuster, um alle abgelehnten oder fehlgeschlagenen Genehmigungsaktionen in allen Pipelines zu erfassen.

```
{
 "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Approval"]
    }
  }
}
```

- Example

  Verwenden Sie diese Beispiel-Ereignismuster, um alle Ereignisse in den angegebenen Pipelines zu erfassen.

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Pipeline Execution State Change",
    "CodePipeline Action Execution State Change",
    "CodePipeline Stage Execution State Change"
  ],
  "detail": {
    "pipeline": ["myPipeline", "my2ndPipeline"]
  }
```

```
}
```

9. Wählen Sie Weiter aus.
10. Wählen Sie unter Zieltypen die Option AWS Service aus.
11. Wählen Sie unter Ziel auswählen die Option aus CodePipeline. Geben Sie im Feld Pipeline-ARN den Pipeline-ARN für die Pipeline ein, die mit dieser Regel gestartet werden soll.

> **Note**
>
> Um den Pipeline-ARN zu erhalten, führen Sie den Befehl get-pipeline aus. Der Pipeline-ARN wird in der Ausgabe angezeigt. Er wird in folgendem Format erstellt: *arn:aws:codepipeline: region: konto: Pipeline-Name*
> Pipeline-Beispiel-ARN:
> arn:aws:codepipeline:us-east- 2:80398 BEISPIEL: MyFirstPipeline

12. So erstellen oder spezifizieren Sie eine IAM-Dienstrolle, die EventBridge Berechtigungen zum Aufrufen des mit Ihrer Regel verknüpften Ziels gewährt (in diesem Fall lautet das Ziel): EventBridge CodePipeline

    - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen aus, um eine Servicerolle zu erstellen, die Ihnen EventBridge Berechtigungen für den Start Ihrer Pipeline-Ausführung erteilt.
    - Wählen Sie Bestehende Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihnen EventBridge Berechtigungen für den Start Ihrer Pipeline-Ausführungen erteilt.

13. Wählen Sie Weiter aus.
14. Wählen Sie auf der Seite „Tags“ die Option Weiter aus.
15. Überprüfen Sie auf der Seite Überprüfen und erstellen die Regelkonfiguration. Wenn Sie mit der Regel zufrieden sind, klicken Sie auf Create rule (Regel erstellen).

Eine Benachrichtigung senden, wenn sich der Pipeline-Status ändert (CLI)

Diese Schritte zeigen, wie Sie mit der CLI eine CloudWatch Ereignisregel erstellen, um Benachrichtigungen über Änderungen zu senden CodePipeline.

Um die zum Erstellen einer Regel AWS CLI zu verwenden, rufen Sie den put-rule Befehl auf und geben Sie Folgendes an:

- Einen Namen, der die von Ihnen erstellte Regel eindeutig bezeichnet. Dieser Name muss für alle Pipelines, die Sie erstellen und die mit Ihrem AWS Konto CodePipeline verknüpft sind, eindeutig sein.
- Das Ereignismuster für die Quelle und von der Regel verwendete Detailfelder. Weitere Informationen finden Sie unter Amazon EventBridge und Event Patterns.

Um eine EventBridge Regel mit CodePipeline als Ereignisquelle zu erstellen

1. Rufen Sie den Befehl put-rule auf, um eine Regel zu erstellen, die das Ereignismuster angibt. (Gültige Status finden Sie in den vorherigen Tabellen.)

   Mit dem folgenden Beispielbefehl wird eine Regel namens erstellt"`MyPipelineStateChanges`", die das CloudWatch Ereignis ausgibt, wenn eine Pipelineausführung für die Pipeline mit dem Namen „MyPipeline“ fehlschlägt. --event-pattern

   ```
   aws events put-rule --name "MyPipelineStateChanges" --event-pattern "{\"source\":
   [\"aws.codepipeline\"],\"detail-type\":[\"CodePipeline Pipeline Execution State
    Change\"],\"detail\":{\"pipeline\":[\"myPipeline\"],\"state\":[\"FAILED\"]}}"
   ```

2. Rufen Sie den put-targets Befehl auf und geben Sie die folgenden Parameter an:

   - Der Parameter `--rule` wird für den `rule_name` verwendet, den Sie mit put-rule erstellt haben.
   - Der `--targets` Parameter wird mit der Liste `Id` des Ziels in der Liste der Ziele und der Liste `ARN` des Amazon SNS SNS-Themas verwendet.

   Der folgende Beispielbefehl legt fest, dass für die Regel mit dem Namen `MyPipelineStateChanges` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dieses Ziel 1 ist. Der Beispielbefehl gibt auch ein Beispiel `ARN` für das Amazon SNS SNS-Thema an.

   ```
   aws events put-targets --rule MyPipelineStateChanges --targets
    Id=1,Arn=arn:aws:sns:us-west-2:11111EXAMPLE:MyNotificationTopic
   ```

3. Fügen Sie Berechtigungen hinzu EventBridge , um den angegebenen Zieldienst zum Aufrufen der Benachrichtigung zu verwenden. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

Platzhalter-Bucket-Referenz für Ereignisse

Dieser Abschnitt dient nur als Referenz. Hinweise zum Erstellen einer Pipeline mit Ereigniserkennungsressourcen finden Sie unter Quellaktionen und Methoden zur Änderungserkennung.

Quellaktionen, die von Amazon S3 bereitgestellt werden, und CodeCommit verwenden ereignisbasierte Ressourcen zur Änderungserkennung, um Ihre Pipeline auszulösen, wenn eine Änderung im Quell-Bucket oder Repository vorgenommen wird. Bei diesen Ressourcen handelt es sich um die Regeln für CloudWatch Ereignisse, die so konfiguriert sind, dass sie auf Ereignisse in der Pipeline-Quelle reagieren, wie z. B. eine Codeänderung im CodeCommit Repository. Wenn Sie CloudWatch Ereignisse für eine Amazon S3 S3-Quelle verwenden, müssen Sie die Option aktivieren, CloudTrail damit die Ereignisse protokolliert werden. CloudTrail benötigt einen S3-Bucket, in den er seine Digests senden kann. Sie können über den benutzerdefinierten Bucket auf die Protokolldateien für Ihre CloudWatch Events-Ressourcen zugreifen, aber Sie können nicht auf die Daten aus dem Platzhalter-Bucket zugreifen.

- Wenn Sie die CLI oder AWS CloudFormation zum Einrichten der CloudWatch Event-Ressourcen verwendet haben, finden Sie Ihre CloudTrail Dateien in dem Bucket, den Sie bei der Einrichtung Ihrer Pipeline angegeben haben.
- Wenn Sie die Konsole verwendet haben, um Ihre Pipeline mit einer S3-Quelle einzurichten, verwendet die Konsole einen CloudTrail Platzhalter-Bucket, wenn sie Ihre CloudWatch Events-Ressourcen für Sie erstellt. CloudTrail Digests werden im Platzhalter-Bucket gespeichert, in AWS-Region dem die Pipeline erstellt wurde.

Sie können die Konfiguration ändern, wenn Sie einen anderen Bucket als den Platzhalter-Bucket verwenden möchten.

> **Note**
>
> In CloudTrail Platzhalter-Buckets geschriebene Daten laufen automatisch nach einem Tag ab und werden nicht aufbewahrt.

Weitere Informationen zum Suchen und Verwalten Ihrer CloudTrail Protokolldateien finden Sie unter Abrufen und Anzeigen Ihrer CloudTrail Protokolldateien.

Themen

Platzhalter-Bucket-Namen für Ereignisse nach Region

In dieser Tabelle sind die Namen der S3-Platzhalter-Buckets aufgeführt, die Protokolldateien enthalten, die Änderungserkennungsereignisse für Pipelines mit Amazon S3 S3-Quellaktionen verfolgen.

Name der Region	Platzhalter-Bucket-Name	Region-ID
USA Ost (Ohio)	codepipeline-cloudtrail-placeholder-bucket-us-ost-2	us-east-2
USA Ost (Nord-Virginia)	codepipeline-cloudtrail-placeholder-bucket-us-ost-1	us-east-1
USA West (Nordkalifornien)	codepipeline-cloudtrail-placeholder-bucket-us-west-1	us-west-1
USA West (Oregon)	codepipeline-cloudtrail-placeholder-bucket-uns-West-2	us-west-2
Kanada (Zentral)	codepipeline-cloudtrail-placeholder-bucket-ca-zentral-1	ca-central-1
Europa (Frankfurt)	codepipeline-cloudtrail-placeholder-bucket-eu-central-1	eu-central-1
Europa (Irland)	codepipeline-cloudtrail-placeholder-bucket-eu-west-1	eu-west-1
Europa (London)	codepipeline-cloudtrail-placeholder-bucket-eu-west-2	eu-west-2
Europa (Paris)	codepipeline-cloudtrail-placeholder-bucket-eu-west-3	eu-west-3
Europa (Stockholm)	codepipeline-cloudtrail-placeholder-bucket-eu-nord-1	eu-north-1

Name der Region	Platzhalter-Bucket-Name	Region-ID
Asien-Pazifik (Hongkong)	codepipeline-cloudtrail-placeholder-bucket-ap-ost-1	ap-east-1
Asien-Pazifik (Hyderabad)	codepipeline-cloudtrail-placeholder-bucket-ap-Süd-2	ap-south-2
Asien-Pazifik (Jakarta)	codepipeline-cloudtrail-placeholder-bucket-ap-Südost-3	ap-southeast-3
Asien-Pazifik (Melbourne)	codepipeline-cloudtrail-placeholder-bucket-ap-Südost-4	ap-southeast-4
Asien-Pazifik (Mumbai)	codepipeline-cloudtrail-placeholder-bucket-ap-Süd-1	ap-south-1
Asien-Pazifik (Osaka)	codepipeline-cloudtrail-placeholder-bucket-ap-nordoast-3-prod	ap-northeast-3
Asien-Pazifik (Tokio)	codepipeline-cloudtrail-placeholder-bucket-ap-nordost-1	ap-northeast-1
Asien-Pazifik (Seoul)	codepipeline-cloudtrail-placeholder-bucket-ap-nordost-2	ap-northeast-2
Asien-Pazifik (Singapur)	codepipeline-cloudtrail-placeholder-bucket-ap-Südosten-1	ap-southeast-1
Asien-Pazifik (Sydney)	codepipeline-cloudtrail-placeholder-bucket-ap-Südost-2	ap-southeast-2
Asien-Pazifik (Tokio)	codepipeline-cloudtrail-placeholder-bucket-ap-nordost-1	ap-northeast-1
Kanada (Zentral)	codepipeline-cloudtrail-placeholder-bucket-ca-zentral-1	ca-central-1

Name der Region	Platzhalter-Bucket-Name	Region-ID
Europa (Frankfurt)	codepipeline-cloudtrail-placeholder-bucket-eu-central-1	eu-central-1
Europa (Irland)	codepipeline-cloudtrail-placeholder-bucket-eu-west-1	eu-west-1
Europa (London)	codepipeline-cloudtrail-placeholder-bucket-eu-west-2	eu-west-2
Europa (Milan)	codepipeline-cloudtrail-placeholder-bucket-eu-Süd-1	eu-south-1
Europa (Paris)	codepipeline-cloudtrail-placeholder-bucket-eu-west-3	eu-west-3
Europa (Spain)	codepipeline-cloudtrail-placeholder-bucket-eu-Süd-2	eu-south-2
Europa (Stockholm)	codepipeline-cloudtrail-placeholder-bucket-eu-nord-1	eu-north-1
Europa (Zürich) *	codepipeline-cloudtrail-placeholder-bucket-eu-central-2	eu-central-2
Israel (Tel Aviv)	codepipeline-cloudtrail-placeholder-bucket-il-central-1	il-central-1
Naher Osten (Bahrain) *	codepipeline-cloudtrail-placeholder-bucket-ich-Süd-1	me-south-1
Naher Osten (VAE)	codepipeline-cloudtrail-placeholder-bucket-mich-zentral-1	me-central-1
Südamerika (São Paulo)	codepipeline-cloudtrail-placeholder-bucket-sa-ost-1	sa-east-1

CodePipeline API-Aufrufe protokollieren mit AWS CloudTrail

AWS CodePipeline ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder AWS-Service einem CodePipeline Benutzer ausgeführt wurden. CloudTrail erfasst alle API-Aufrufe CodePipeline als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der CodePipeline Konsole und Codeaufrufen für die CodePipeline API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für CodePipeline. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage ermitteln CloudTrail, an die die Anfrage gestellt wurde CodePipeline, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

CodePipeline Informationen in CloudTrail

CloudTrail ist auf Ihrem aktiviert AWS-Konto , wenn Sie das Konto erstellen. Wenn eine Aktivität in stattfindet CodePipeline, wird diese Aktivität zusammen mit anderen AWS-Service Ereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS Konto ansehen, suchen und herunterladen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto , einschließlich der Ereignisse für CodePipeline, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole erstellen, gilt der Trail standardmäßig für alle AWS Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere konfigurieren, AWS-Services um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle CodePipeline Aktionen werden von der CodePipeline API-Referenz protokolliert CloudTrail und sind in dieser dokumentiert. Beispielsweise generieren Aufrufe von `GetPipelineExecution` und `UpdatePipeline` Aktionen Einträge in den CloudTrail Protokolldateien. `CreatePipeline`

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Anmeldeinformationen gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

Weitere Informationen finden Sie unter dem CloudTrail UserIdentity-Element.

Grundlegendes zu Einträgen CodePipeline in Protokolldateien

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für ein Aktualisierungspipeline-Ereignis, bei dem eine Pipeline mit dem Namen von dem Benutzer mit dem Namen JaneDoe - CodePipeline mit der Konto-ID 80398EXAMPLE - bearbeitet MyFirstPipeline wurde. Der Benutzer hat den Namen der Quellphase einer Pipeline von `Source` in `MySourceStage` geändert. Da `requestParameters` sowohl die als auch die `responseElements` Elemente im CloudTrail Protokoll die gesamte Struktur der bearbeiteten Pipeline enthalten, wurden diese Elemente im folgenden Beispiel abgekürzt. Hervorgehoben wurde der Abschnitt `requestParameters` der Pipeline, der die Änderung enthält, die Nummer der vorherigen Version der Pipeline und der Abschnitt `responseElements`, der die Versionsnummer erhöht durch 1 zeigt. Bearbeitete Abschnitte sind mit Auslassungspunkten (...) gekennzeichnet, um darzustellen, an welchen Stellen in einem realen Protokolleintrag weitere Daten stehen.

```
{
 "eventVersion":"1.03",
```

```
  "userIdentity": {
   "type":"IAMUser",
   "principalId":"AKIAI44QH8DHBEXAMPLE",
   "arn":"arn:aws:iam::80398EXAMPLE:user/JaneDoe-CodePipeline",
   "accountId":"80398EXAMPLE",
   "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
   "userName":"JaneDoe-CodePipeline",
   "sessionContext": {
   "attributes":{
    "mfaAuthenticated":"false",
    "creationDate":"2015-06-17T14:44:03Z"
    }
   },
"invokedBy":"signin.amazonaws.com"},
"eventTime":"2015-06-17T19:12:20Z",
"eventSource":"codepipeline.amazonaws.com",
"eventName":"UpdatePipeline",
"awsRegion":"us-east-2",
"sourceIPAddress":"192.0.2.64",
"userAgent":"signin.amazonaws.com",
"requestParameters":{
  "pipeline":{
  "version":1,
  "roleArn":"arn:aws:iam::80398EXAMPLE:role/CodePipeline_Service_Role",
  "name":"MyFirstPipeline",
  "stages":[
    {
    "actions":[
     {
         "name":"MySourceStage",
       "actionType":{
       "owner":"AWS",
       "version":"1",
       "category":"Source",
       "provider":"S3"
    },
   "inputArtifacts":[],
   "outputArtifacts":[
    {"name":"MyApp"}
    ],
   "runOrder":1,
   "configuration":{
    "S3Bucket":"awscodepipeline-demobucket-example-date",
    "S3ObjectKey":"sampleapp_linux.zip"
```

```
      }
      }
     ],
       "name":"Source"
     },
     (...)
               },
 "responseElements":{
   "pipeline":{
     "version":2,
     (...)
          },
     "requestID":"2c4af5c9-7ce8-EXAMPLE",
     "eventID":""c53dbd42-This-Is-An-Example"",
     "eventType":"AwsApiCall",
     "recipientAccountId":"80398EXAMPLE"
     }
      ]
}
```

Problembehebung CodePipeline

Die folgenden Informationen helfen Ihnen möglicherweise bei der Lösung häufiger Probleme in AWS CodePipeline.

Themen

- Pipelines mit Amazon S3, Amazon ECR oder CodeCommit Quelle werden nicht mehr automatisch gestartet
- Verbindungsfehler beim Herstellen einer Verbindung zu GitHub: „Es ist ein Problem aufgetreten, stellen Sie sicher, dass Cookies in Ihrem Browser aktiviert sind“ oder „Ein Organisationsinhaber muss die GitHub App installieren“
- Pipelines, deren Ausführungsmodus in den QUEUED- oder PARALLEL-Modus geändert wurde, schlagen fehl, wenn das Ausführungslimit erreicht ist
- Pipelines im PARALLEL-Modus weisen eine veraltete Pipeline-Definition auf, wenn sie beim Wechsel in den QUEUED- oder SUPERSEDED-Modus bearbeitet werden
- Bei Pipelines, die vom PARALLEL-Modus aus geändert wurden, wird ein früherer Ausführungsmodus angezeigt
- Pipelines mit Verbindungen, die Triggerfilterung nach Dateipfaden verwenden, beginnen möglicherweise nicht bei der Erstellung von Zweigen
- Pipelines mit Verbindungen, die Triggerfilterung nach Dateipfaden verwenden, werden möglicherweise nicht gestartet, wenn das Dateilimit erreicht ist
- CodeCommit oder S3-Quellversionen im PARALLEL-Modus stimmen möglicherweise nicht mit dem Ereignis überein EventBridge
- Benötigen Sie Hilfe für ein anderes Problem?

Pipeline-Fehler: Eine mit AWS Elastic Beanstalk konfigurierte Pipeline gibt eine Fehlermeldung zurück: „Bereitstellung fehlgeschlagen. Die angegebene Rolle verfügt nicht über ausreichende Berechtigungen: Service:AmazonElasticLoadBalancing“

Problem: Die Servicerolle für CodePipeline verfügt nicht über ausreichende Berechtigungen für AWS Elastic Beanstalk, einschließlich, aber nicht beschränkt auf, einige Operationen in Elastic Load Balancing. Die Servicerolle für CodePipeline wurde am 6. August 2015 aktualisiert, um dieses Problem zu beheben. Kunden, die ihre Service-Rolle vor diesem Datum erstellt haben, müssen Sie Richtlinienanweisung für ihre Service-Rolle ändern, um die erforderlichen Berechtigungen hinzuzufügen.

Mögliche Fehlerbehebungen: Die einfachste Lösung besteht darin, die Richtlinienanweisung für Ihre Servicerolle zu bearbeiten, wie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle beschrieben.

Nachdem Sie die bearbeitete Richtlinie angewendet haben, folgen Sie den Schritten unter, Manuelles Starten einer Pipeline um alle Pipelines, die Elastic Beanstalk verwenden, manuell erneut auszuführen.

Abhängig von Ihren Sicherheitsanforderungen können Sie die Berechtigungen auch auf andere Arten ändern.

Bereitstellungsfehler: Eine mit einer AWS Elastic Beanstalk Bereitstellungsaktion konfigurierte Pipeline bleibt hängen und schlägt nicht fehl, wenn die "" DescribeEvents -Berechtigung fehlt

Problem: Die Servicerolle für CodePipeline muss die `"elasticbeanstalk:DescribeEvents"` Aktion für alle Pipelines enthalten, die verwenden AWS Elastic Beanstalk. Ohne diese Berechtigung bleiben AWS Elastic Beanstalk Bereitstellungsaktionen hängen, ohne dass sie fehlschlagen oder auf einen Fehler hinweisen. Wenn diese Aktion in Ihrer Servicerolle fehlt, CodePipeline verfügt sie nicht über die erforderlichen Berechtigungen, um die Pipeline-Bereitstellungsphase in AWS Elastic Beanstalk Ihrem Namen auszuführen.

Mögliche Lösungen: Überprüfen Sie Ihre CodePipeline Servicerolle. Wenn die `"elasticbeanstalk:DescribeEvents"`-Aktion fehlt, folgen Sie den Schritten in Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle, um sie unter Verwendung der Funktion Edit Policy (Richtlinie bearbeiten) in der IAM-Konsole hinzuzufügen.

Nachdem Sie die bearbeitete Richtlinie angewendet haben, folgen Sie den Schritten unter, Manuelles Starten einer Pipeline um alle Pipelines, die Elastic Beanstalk verwenden, manuell erneut auszuführen.

Pipeline-Fehler: Eine Quellaktion gibt die Meldung mit unzureichenden Berechtigungen zurück: „Auf das Repository konnte nicht zugegriffen werden. CodeCommit **`repository-name`** Überprüfen Sie, ob die Pipeline-IAM-Rolle über ausreichende Berechtigungen für den Zugriff auf das Repository verfügt.“

Problem: Die Servicerolle für CodePipeline verfügt nicht über ausreichende Berechtigungen für CodeCommit und wurde wahrscheinlich erstellt, bevor die Unterstützung für die Verwendung von CodeCommit Repositorys am 18. April 2016 hinzugefügt wurde. Kunden, die ihre Service-Rolle vor diesem Datum erstellt haben, müssen Sie Richtlinienanweisung für ihre Service-Rolle ändern, um die erforderlichen Berechtigungen hinzuzufügen.

Mögliche Lösungen: Fügen Sie der Richtlinie für Ihre CodeCommit CodePipeline Servicerolle die erforderlichen Berechtigungen für hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.

Pipeline-Fehler: Ein Jenkins-Build oder eine Testaktion werden über eine lange Zeit ausgeführt und schlagen dann aufgrund fehlender Anmeldeinformationen oder Berechtigungen fehl.

Problem: Wenn der Jenkins-Server auf einer Amazon EC2 EC2-Instance installiert ist, wurde die Instance möglicherweise nicht mit einer Instance-Rolle erstellt, die über die erforderlichen Berechtigungen verfügt. CodePipeline Wenn Sie einen IAM-Benutzer auf einem Jenkins-Server, einer lokalen Instance oder einer Amazon EC2 EC2-Instance verwenden, die ohne die erforderliche IAM-Rolle erstellt wurde, verfügt der IAM-Benutzer entweder nicht über die erforderlichen Berechtigungen, oder der Jenkins-Server kann nicht über das auf dem Server konfigurierte Profil auf diese Anmeldeinformationen zugreifen.

Mögliche Lösungen: Stellen Sie sicher, dass die Amazon EC2 EC2-Instance-Rolle oder der IAM-Benutzer mit der `AWSCodePipelineCustomActionAccess` verwalteten Richtlinie oder mit den entsprechenden Berechtigungen konfiguriert ist. Weitere Informationen finden Sie unter AWS verwaltete Richtlinien für AWS CodePipeline.

Wenn Sie einen IAM-Benutzer verwenden, stellen Sie sicher, dass das auf der Instance konfigurierte AWS Profil den mit den richtigen Berechtigungen konfigurierten IAM-Benutzer verwendet.

Möglicherweise müssen Sie die IAM-Benutzeranmeldeinformationen, die Sie für die Integration zwischen Jenkins konfiguriert haben, CodePipeline direkt in die Jenkins-Benutzeroberfläche eingeben. Dies ist keine empfohlene bewährte Methode. Wenn Sie dies tun müssen, müssen Sie sicherstellen, dass der Jenkins-Server geschützt ist und HTTPS statt HTTP verwendet.

Pipeline-Fehler: Eine Pipeline, die in einer AWS Region mithilfe eines in einer anderen AWS Region erstellten Buckets erstellt wurde, gibt ein "" mit dem Code InternalError "“ zurück JobFailed

Problem: Der Download eines in einem Amazon S3 S3-Bucket gespeicherten Artefakts schlägt fehl, wenn die Pipeline und der Bucket in verschiedenen AWS Regionen erstellt werden.

Mögliche Lösungen: Stellen Sie sicher, dass sich der Amazon S3 S3-Bucket, in dem Ihr Artefakt gespeichert ist, in derselben AWS Region befindet wie die Pipeline, die Sie erstellt haben.

Bereitstellungsfehler: Eine ZIP-Datei, die eine WAR-Datei enthält, wurde erfolgreich bereitgestellt AWS Elastic Beanstalk, aber die Anwendungs-URL meldet den Fehler 404 Not Found

Problem: Eine WAR-Datei wird erfolgreich für eine AWS Elastic Beanstalk -Umgebung bereitgestellt, die Anwendungs-URL gibt jedoch den Fehler "404 Not Found" (404 Nicht gefunden) zurück.

Mögliche Korrekturen: AWS Elastic Beanstalk kann eine ZIP-Datei entpacken, aber keine WAR-Datei, die in einer ZIP-Datei enthalten ist. Geben Sie statt einer WAR-Datei in Ihrer Datei `buildspec.yml` einen Ordner an, der die Inhalte enthält, die bereitgestellt werden sollen. Beispielsweise:

```
version: 0.2

phases:
  post_build:
    commands:
      - mvn package
      - mv target/my-web-app ./
artifacts:
  files:
    - my-web-app/**/*
 discard-paths: yes
```

Ein Beispiel hierfür finden Sie unter AWS Elastic Beanstalk -Beispiel für CodeBuild.

Pipeline-Artefakt-Ordnernamen werden gekürzt

Problem: Wenn Sie sich die Namen von Pipeline-Artefakten in ansehen CodePipeline, scheinen die Namen gekürzt zu sein. Dadurch sehen manche Namen gleich aus oder enthalten augenscheinlich nicht mehr den gesamten Pipelinenamen.

Erklärung: CodePipeline Kürzt Artefaktnamen, um sicherzustellen, dass der vollständige Amazon S3-Pfad die Richtliniengrößenbeschränkungen nicht überschreitet, wenn temporäre Anmeldeinformationen für CodePipeline Jobarbeiter generiert werden.

Auch wenn der Artefaktname gekürzt zu sein scheint, wird er dem CodePipeline Artefakt-Bucket auf eine Weise zugeordnet, die nicht durch Artefakte mit gekürzten Namen beeinträchtigt wird. Die Pipeline kann normal ausgeführt werden. Dies ist kein Problem mit dem Ordner oder den Artefakten. Für Pipelinenamen besteht eine Längenbegrenzung auf 100 Zeichen. Auch wenn der Name des Artefaktordners gekürzt angezeigt wird, ist er für Ihre Pipeline eindeutig.

Fügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, Enterprise Server oder .com GitHub hinzu GitHub GitLab

Wenn du eine AWS CodeStar Verbindung in einer Quellaktion und einer CodeBuild Aktion verwendest, gibt es zwei Möglichkeiten, wie das Eingabeartefakt an den Build übergeben werden kann:

- Standard: Die Quellaktion erzeugt eine ZIP-Datei, die den CodeBuild heruntergeladenen Code enthält.
- Git-Klon: Der Quellcode kann direkt in die Build-Umgebung heruntergeladen werden.

  Der Git-Klon-Modus ermöglicht es Ihnen, mit dem Quellcode als funktionierendes Git-Repository zu interagieren. Um diesen Modus verwenden zu können, müssen Sie Ihrer CodeBuild Umgebung Berechtigungen zur Verwendung der Verbindung erteilen.

Um Ihrer CodeBuild Servicerollenrichtlinie Berechtigungen hinzuzufügen, erstellen Sie eine vom Kunden verwaltete Richtlinie, die Sie Ihrer CodeBuild Servicerolle zuordnen. Mit den folgenden

Schritten wird eine Richtlinie erstellt, bei der die `UseConnection`-Berechtigung im `action`-Feld und der Verbindungs-ARN im `Resource`-Feld angegeben wird.

So verwenden Sie die Konsole, um die UseConnection Berechtigungen hinzuzufügen

1. Um den Verbindungs-ARN für Ihre Pipeline zu finden, öffnen Sie die Pipeline und klicken Sie auf das (i)-Symbol in der Quellaktion. Sie fügen den Verbindungs-ARN zu Ihrer CodeBuild Servicerollenrichtlinie hinzu.

   Ein Beispiel für eine Verbindungs-ARN ist:

   ```
   arn:aws:codeconnections:eu-central-1:123456789123:connection/
   sample-1908-4932-9ecc-2ddacee15095
   ```

2. Um Ihre CodeBuild Servicerolle zu finden, öffnen Sie das in Ihrer Pipeline verwendete Build-Projekt und navigieren Sie zur Registerkarte Build-Details.
3. Wählen Sie den Link Service role (Servicerolle) . Dadurch wird die IAM-Konsole geöffnet, in der Sie eine neue Richtlinie hinzufügen können, die Zugriff auf Ihre Verbindung gewährt.
4. Wählen Sie in der IAM-Konsole Attach policies (Richtlinien anhängen) und dann Create policy (Richtlinie erstellen).

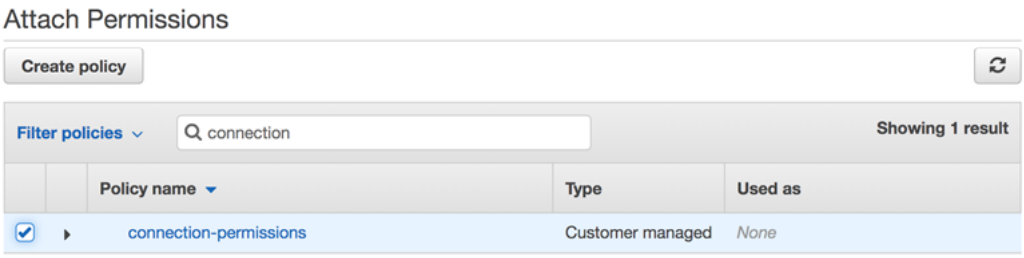
   Verwenden Sie die folgende Beispielrichtlinienvorlage. Fügen Sie Ihren Verbindungs-ARN in das `Resource`-Feld ein, wie in diesem Beispiel gezeigt:

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": "codestar-connections:UseConnection",
               "Resource": "insert connection ARN here"
           }
       ]
   }
   ```

   Fügen Sie auf der Registerkarte JSON Ihre Richtlinie ein.

5. Wählen Sie Richtlinie prüfen. Geben Sie einen Namen für die Richtlinie ein (beispielsweise **`connection-permissions`**) und wählen Sie dann Create policy (Richtlinie erstellen) aus.

6. Kehren Sie zu der Seite zurück, auf der Sie Berechtigungen angehängt haben, aktualisieren Sie die Richtlinienliste und wählen Sie die gerade erstellte Richtlinie aus. Wählen Sie Richtlinien anfügen.



Fügen Sie CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen hinzu

Wenn Ihre Pipeline über eine CodeCommit Quellaktion verfügt, gibt es zwei Möglichkeiten, das Eingabeartefakt an den Build zu übergeben:

- Standard — Die Quellaktion erzeugt eine ZIP-Datei, die den CodeBuild heruntergeladenen Code enthält.
- Vollständiger Klon — Der Quellcode kann direkt in die Build-Umgebung heruntergeladen werden.

  Die Option Vollständiges Klonen ermöglicht es Ihnen, mit dem Quellcode als funktionierendes Git-Repository zu interagieren. Um diesen Modus zu verwenden, müssen Sie Ihrer CodeBuild Umgebung Berechtigungen zum Abrufen von Inhalten aus Ihrem Repository hinzufügen.

Um Ihrer CodeBuild Servicerollenrichtlinie Berechtigungen hinzuzufügen, erstellen Sie eine vom Kunden verwaltete Richtlinie, die Sie Ihrer CodeBuild Servicerolle zuordnen. Mit den folgenden Schritten wird eine Richtlinie erstellt, die die `codecommit:GitPull` Berechtigungen in dem `action` Feld festlegt.

Um die Konsole zum Hinzufügen der GitPull Berechtigungen zu verwenden

1. Um Ihre CodeBuild Servicerolle zu finden, öffnen Sie das in Ihrer Pipeline verwendete Build-Projekt und navigieren Sie zur Registerkarte Build-Details.
2. Wählen Sie den Link Service role (Servicerolle) . Dadurch wird die IAM-Konsole geöffnet, in der Sie eine neue Richtlinie hinzufügen können, die Zugriff auf Ihr Repository gewährt.

3. Wählen Sie in der IAM-Konsole Attach policies (Richtlinien anhängen) und dann Create policy (Richtlinie erstellen).
4. Fügen Sie auf der Registerkarte JSON die folgende Beispielrichtlinie ein.

```
{
    "Action": [
        "codecommit:GitPull"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
```

5. Wählen Sie Richtlinie prüfen. Geben Sie einen Namen für die Richtlinie ein (beispielsweise **`codecommit-gitpull`**) und wählen Sie dann Create policy (Richtlinie erstellen) aus.
6. Kehren Sie zu der Seite zurück, auf der Sie Berechtigungen angehängt haben, aktualisieren Sie die Richtlinienliste und wählen Sie die gerade erstellte Richtlinie aus. Wählen Sie Richtlinien anfügen.

<source artifact name>Pipeline-Fehler: Bei einer Bereitstellung mit der CodeDeployTo ECS-Aktion wird die folgende Fehlermeldung zurückgegeben: „Ausnahme beim Versuch, die Artefaktdatei der Aufgabendefinition zu lesen aus:“

Problem:

Die Aufgabendefinitionsdatei ist ein erforderliches Artefakt für die CodePipeline Bereitstellungsaktion in Amazon ECS über CodeDeploy (die `CodeDeployToECS` Aktion). Die maximale ZIP-Größe des Artefakts in der Aktion „`CodeDeployToECS`Bereitstellen“ beträgt 3 MB. Die folgende Fehlermeldung wird zurückgegeben, wenn die Datei nicht gefunden wird oder die Größe des Artefakts 3 MB überschreitet:

Ausnahme beim Versuch, die Taskdefinitions-Artefaktdatei aus folgender Quelle zu lesen von: <source artifact name>

Mögliche Korrekturen: Stellen Sie sicher, dass die Aufgabendefinitionsdatei als Artefakt enthalten ist. Wenn die Datei bereits existiert, stellen Sie sicher, dass die komprimierte Größe weniger als 3 MB beträgt.

GitHub Quellaktion für Version 1: Die Repository-Liste zeigt verschiedene Repositorys

Problem:

Nach erfolgreicher Autorisierung für eine Aktion für GitHub Version 1 in der CodePipeline Konsole können Sie aus einer Liste Ihrer GitHub Repositorys wählen. Wenn die Liste nicht die Repositorys enthält, die Sie erwartet haben, können Sie Probleme mit dem Konto beheben, das für die Autorisierung verwendet wurde.

Mögliche Korrekturen: Die Liste der Repositorys in der CodePipeline Konsole basiert auf der GitHub Organisation, zu der das autorisierte Konto gehört. Vergewissern Sie sich, dass es sich bei dem Konto, das Sie für die Autorisierung verwenden, um das Konto GitHub handelt, das der GitHub Organisation zugeordnet ist, in der Ihr Repository erstellt wurde.

GitHub Quellaktion für Version 2: Die Verbindung für ein Repository konnte nicht abgeschlossen werden

Problem:

Da eine Verbindung zu einem GitHub Repository den AWS Connector für verwendet GitHub, benötigen Sie zum Herstellen der Verbindung die Rechte des Organisationsinhabers oder Administratorberechtigungen für das Repository.

Mögliche Korrekturen: Informationen zu den Berechtigungsstufen für ein GitHub Repository finden Sie unter https://docs.github.com/en/ free-pro-team @latest /github/ setting-up-and-managing - organizations-and-teams/permission-levels-for-an-organization.

Amazon S3 S3-Fehler: Für die CodePipeline <ARN>Servicerolle wurde der S3-Zugriff für den S3-Bucket verweigert < BucketName >

Problem:

Während der Bearbeitung CodePipeline überprüft die CodeCommit Aktion in, ob der Pipeline-Artefakt-Bucket vorhanden ist. Wenn die Aktion nicht über die Berechtigung zur Überprüfung verfügt, tritt in Amazon S3 ein `AccessDenied` Fehler auf und die folgende Fehlermeldung wird angezeigt in CodePipeline:

CodePipeline *Der Servicerolle „arn:aws:iam:: accountId:role/service-role/ roleID" wurde der S3-Zugriff für den S3-Bucket verweigert "“ BucketName*

CloudTrail In den Protokollen für die Aktion wird auch `AccessDenied` der Fehler protokolliert.

Mögliche Korrekturen: Gehen Sie wie folgt vor:

- Fügen `s3:ListBucket` Sie die Richtlinie, die Ihrer CodePipeline Servicerolle zugeordnet ist, der Liste der Aktionen in Ihrer Richtlinie hinzu. Anweisungen zum Anzeigen Ihrer Richtlinie für Servicerollen finden Sie unterDen Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen. Bearbeiten Sie die Richtlinienerklärung für Ihre Servicerolle, wie unter beschriebenHinzufügen von Berechtigungen zur CodePipeline-Servicerolle.
- Fügen Sie für die ressourcenbasierte Richtlinie, die dem Amazon S3 S3-Artefakt-Bucket für Ihre Pipeline beigefügt ist, auch Artifact-Bucket-Richtlinie genannt, eine Erklärung hinzu, damit die `s3:ListBucket` Berechtigung von Ihrer Servicerolle verwendet werden kann. CodePipeline

  Um Ihre Richtlinie zum Artifact-Bucket hinzuzufügen

  1. Folgen Sie den Schritten unterDen Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen, um Ihren Artefakt-Bucket auf der Seite mit den Pipeline-Einstellungen auszuwählen und ihn dann in der Amazon S3 S3-Konsole anzuzeigen.
  2. Wählen Sie Permissions (Berechtigungen).
  3. Wählen Sie unter Bucket-Richtlinie Bearbeiten aus.
  4. Geben Sie im Textfeld Richtlinie eine neue Bucket-Richtlinie ein oder bearbeiten Sie die bestehende Richtlinie, wie im folgenden Beispiel gezeigt. Die Bucket-Richtlinie ist eine JSON-Datei, daher müssen Sie eine gültige JSON-Datei eingeben.

     Das folgende Beispiel zeigt eine Bucket-Richtlinienanweisung für einen Artefakt-Bucket, wobei die Beispielrollen-ID für die Servicerolle *AROAEXAMPLEID* lautet.

```
{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::BucketName",
    "Condition": {
        "StringLike": {
            "aws:userid": "AROAEXAMPLEID:*"
        }
    }
```

```
}
```

Das folgende Beispiel zeigt dieselbe Bucket-Policy-Anweisung, nachdem die Berechtigung hinzugefügt wurde.

```
{
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890",
            "Condition": {
                "StringLike": {
                    "aws:userid": "AROAEXAMPLEID:*"
                }
            }
        },
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": false
                }
            }
```

```
        }
    ]
}
```

Weitere Informationen finden Sie in den Schritten unter https://aws.amazon.com/blogs/security/writing-iam-policies-how-to-grant-access-to-an-amazon-s3-bucket/.

5. Wählen Sie Speichern.

Nachdem Sie die bearbeitete Richtlinie angewendet haben, folgen Sie den Schritten unter, Manuelles Starten einer Pipeline um Ihre Pipeline manuell erneut auszuführen.

Pipelines mit Amazon S3, Amazon ECR oder CodeCommit Quelle werden nicht mehr automatisch gestartet

Problem:

Nach einer Änderung der Konfigurationseinstellungen für eine Aktion, die Ereignisregeln (EventBridgeoder CloudWatch Ereignisse) zur Änderungserkennung verwendet, erkennt die Konsole möglicherweise keine Änderung, wenn die Quell-Trigger-IDs ähnlich sind und identische Anfangszeichen haben. Da die neue Ereignisregel nicht von der Konsole erstellt wird, wird die Pipeline nicht mehr automatisch gestartet.

Ein Beispiel für eine geringfügige Änderung am Ende des Parameternamens für CodeCommit wäre die Änderung Ihres CodeCommit Zweignamens `MyTestBranch-1` in`MyTestBranch-2`. Da sich die Änderung am Ende des Zweignamens befindet, aktualisiert oder erstellt die Ereignisregel für die Quellaktion möglicherweise keine Regel für die neuen Quelleinstellungen.

Dies gilt wie folgt für Quellaktionen, die CWE-Ereignisse zur Änderungserkennung verwenden:

Quellaktion	Parameter/Trigger-Identifikatoren (Konsole)
Amazon ECR	Repository-Name Bild-Tag
Amazon S3	Bucket S3-Objektschlüssel

Quellaktion	Parameter/Trigger-Identifikatoren (Konsole)
CodeCommit	Repository-Name Name der Filiale

Mögliche Lösungen:

Führen Sie eine der folgenden Aktionen aus:

- Ändern Sie die CodeCommit /S3/ECR-Konfigurationseinstellungen so, dass Änderungen am Startteil des Parameterwerts vorgenommen werden.

  Beispiel: Ändern Sie Ihren Filialnamen in. `release-branch` `2nd-release-branch` Vermeiden Sie eine Änderung am Ende des Namens, wie `release-branch-2` z.
- Ändern Sie die CodeCommit /S3/ECR-Konfigurationseinstellungen für jede Pipeline.

  Beispiel: Ändern Sie Ihren Filialnamen in. `myRepo/myBranch` `myDeployRepo/myDeployBranch` Vermeiden Sie eine Änderung am Ende des Namens, wie `myRepo/myBranch2` z.
- Verwenden Sie statt der Konsole die CLI oder AWS CloudFormation um Ihre Regeln für Änderungserkennungsereignisse zu erstellen und zu aktualisieren. Anweisungen zum Erstellen von Ereignisregeln für eine S3-Quellaktion finden Sie unter. Amazon S3 S3-Quellaktionen und EventBridge mit AWS CloudTrail Anweisungen zum Erstellen von Ereignisregeln für eine Amazon ECR-Aktion finden Sie unter Amazon ECR-Quellaktionen und Ressourcen EventBridge . Anweisungen zum Erstellen von Ereignisregeln für eine CodeCommit Aktion finden Sie unter CodeCommit Quellaktionen und EventBridge.

Nachdem Sie Ihre Aktionskonfiguration in der Konsole bearbeitet haben, akzeptieren Sie die aktualisierten Ressourcen zur Änderungserkennung, die von der Konsole erstellt wurden.

Verbindungsfehler beim Herstellen einer Verbindung zu GitHub: „Es ist ein Problem aufgetreten, stellen Sie sicher, dass Cookies in Ihrem Browser aktiviert sind“ oder „Ein Organisationsinhaber muss die GitHub App installieren“

Problem:

Um die Verbindung für eine GitHub Quellaktion in herzustellen CodePipeline, müssen Sie der Eigentümer der GitHub Organisation sein. Bei Repositorys, die keiner Organisation angehören, müssen Sie der Repository-Besitzer sein. Erstellt eine andere Person als der Organisationsbesitzer eine Verbindung, wird eine Anfrage für den Organisationsbesitzer erstellt, und einer der folgenden Fehler wird angezeigt:

Es ist ein Problem aufgetreten. Stellen Sie sicher, dass Cookies in Ihrem Browser aktiviert sind

ODER

Ein Organisationsinhaber muss die GitHub App installieren

Mögliche Lösungen: Für Repositorys in einer GitHub Organisation muss der Organisationsinhaber die Verbindung zum GitHub Repository herstellen. Bei Repositorys, die keiner Organisation angehören, müssen Sie der Repository-Besitzer sein.

Pipelines, deren Ausführungsmodus in den QUEUED- oder PARALLEL-Modus geändert wurde, schlagen fehl, wenn das Ausführungslimit erreicht ist

Problem: Die maximale Anzahl gleichzeitiger Ausführungen für eine Pipeline im QUEUED-Modus beträgt 50 Ausführungen. Wenn dieses Limit erreicht ist, schlägt die Pipeline ohne Statusmeldung fehl.

Mögliche Lösungen: Wenn Sie die Pipeline-Definition für den Ausführungsmodus bearbeiten, führen Sie die Bearbeitung getrennt von anderen Bearbeitungsaktionen durch.

Weitere Hinweise zum Ausführungsmodus QUEUED oder PARALLEL finden Sie unterCodePipeline Konzepte .

Pipelines im PARALLEL-Modus weisen eine veraltete Pipeline-Definition auf, wenn sie beim Wechsel in den QUEUED- oder SUPERSEDED-Modus bearbeitet werden

Problem: Wenn Sie für Pipelines im Parallelmodus den Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED ändern, wird die Pipeline-Definition für den PARALLEL-Modus nicht aktualisiert. Die beim Aktualisieren des PARALLEL-Modus aktualisierte Pipeline-Definition wird im Modus SUPERSEDED oder QUEUED nicht verwendet

Mögliche Lösungen: Wenn Sie bei Pipelines im Parallelmodus den Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED ändern, vermeiden Sie es, die Pipeline-Definition gleichzeitig zu aktualisieren.

Weitere Hinweise zum Ausführungsmodus QUEUED oder PARALLEL finden Sie unter. CodePipeline Konzepte

Bei Pipelines, die vom PARALLEL-Modus aus geändert wurden, wird ein früherer Ausführungsmodus angezeigt

Problem: Bei Pipelines im PARALLEL-Modus wird beim Ändern des Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED der aktualisierte Status nicht als PARALLEL angezeigt. Wenn die Pipeline von PARALLEL in QUEUED oder SUPERSEDED geändert wurde, ist der Status der Pipeline im Modus ABGELÖST oder WARTESCHLANGE der letzte bekannte Status in einem dieser Modi. Wenn die Pipeline noch nie in diesem Modus ausgeführt wurde, ist der Status leer.

Mögliche Korrekturen: Beachten Sie bei Pipelines im Parallelmodus, wenn Sie den Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED ändern, dass in der Anzeige des Ausführungsmodus nicht der Status PARALLEL angezeigt wird.

Weitere Hinweise zum Ausführungsmodus QUEUED oder PARALLEL finden Sie unter. CodePipeline Konzepte

Pipelines mit Verbindungen, die Triggerfilterung nach Dateipfaden verwenden, beginnen möglicherweise nicht bei der Erstellung von Zweigen

Beschreibung: Für Pipelines mit Quellaktionen, die Verbindungen verwenden, wie z. B. eine BitBucket Quellaktion, können Sie einen Trigger mit einer Git-Konfiguration einrichten, mit der Sie nach Dateipfaden filtern können, um Ihre Pipeline zu starten. In bestimmten Fällen wird bei Pipelines mit Triggern, die nach Dateipfaden gefiltert werden, die Pipeline möglicherweise nicht gestartet, wenn ein Zweig mit einem Dateipfadfilter zum ersten Mal erstellt wird, da die CodeConnections Verbindung dadurch nicht in der Lage ist, die geänderten Dateien aufzulösen. Wenn die Git-Konfiguration für den Trigger so eingerichtet ist, dass nach Dateipfaden gefiltert wird, startet die Pipeline nicht, wenn der Branch mit dem Filter gerade im Quell-Repository erstellt wurde. Weitere Informationen zum Filtern nach Dateipfaden finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern.

Ergebnis: Beispielsweise werden Pipelines, CodePipeline die einen Dateipfadfilter für einen Zweig „B“ haben, nicht ausgelöst, wenn der Zweig „B“ erstellt wird. Wenn es keine Dateipfadfilter gibt, wird die Pipeline trotzdem gestartet.

Pipelines mit Verbindungen, die Triggerfilterung nach Dateipfaden verwenden, werden möglicherweise nicht gestartet, wenn das Dateilimit erreicht ist

Beschreibung: Für Pipelines mit Quellaktionen, die Verbindungen verwenden, wie z. B. eine BitBucket Quellaktion, können Sie einen Trigger mit einer Git-Konfiguration einrichten, mit der Sie nach Dateipfaden filtern können, um Ihre Pipeline zu starten. CodePipeline ruft bis zu die ersten 100 Dateien ab. Wenn die Git-Konfiguration für den Trigger so eingerichtet ist, dass nach Dateipfaden gefiltert wird, startet die Pipeline daher möglicherweise nicht, wenn mehr als 100 Dateien vorhanden sind. Weitere Informationen zum Filtern nach Dateipfaden finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern.

Ergebnis: Wenn ein Diff beispielsweise 150 Dateien enthält, CodePipeline werden die ersten 100 Dateien (in keiner bestimmten Reihenfolge) anhand des angegebenen Dateipfadfilters geprüft. Wenn die Datei, die dem Dateipfadfilter entspricht, nicht zu den 100 abgerufenen Dateien gehört CodePipeline, wird die Pipeline nicht aufgerufen.

CodeCommit oder S3-Quellversionen im PARALLEL-Modus stimmen möglicherweise nicht mit dem Ereignis überein EventBridge

Beschreibung: Bei Pipeline-Ausführungen im PARALLEL-Modus kann eine Ausführung mit der letzten Änderung beginnen, z. B. dem CodeCommit Repository-Commit, die möglicherweise nicht mit der Änderung für das EventBridge Ereignis identisch ist. In einigen Fällen, in denen ein Sekundenbruchteil zwischen Commits oder Image-Tags liegt, die die Pipeline starten, wird, wenn das Ereignis CodePipeline empfangen und die Ausführung gestartet wird, ein weiterer Commit oder Image-Tag übertragen CodePipeline (z. B. die CodeCommit Aktion), der HEAD-Commit in diesem Moment geklont.

Ergebnis: Bei Pipelines im PARALLEL-Modus mit einer CodeCommit oder S3-Quelle klont die Quellaktion unabhängig von der Änderung, die die Pipeline-Ausführung ausgelöst hat, den HEAD immer zu dem Zeitpunkt, zu dem er gestartet wird. Bei einer Pipeline im PARALLEL-Modus wird beispielsweise ein Commit per Push übertragen, wodurch die Pipeline für Ausführung 1 gestartet wird, und die zweite Pipeline-Ausführung verwendet den zweiten Commit.

Benötigen Sie Hilfe für ein anderes Problem?

Sie haben folgende weitere Möglichkeiten:

- AWS Support kontaktieren.
- Stellen Sie eine Frage im CodePipelineForum.
- Anfordern einer Kontingenterhöhung. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline.

> **Note**
>
> Die Verarbeitung von Anträgen zur Kontingenterhöhung kann bis zu zwei Wochen dauern.

Sicherheit in AWS CodePipeline

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die so konzipiert ist, dass sie die Anforderungen der sicherheitssensibelsten Unternehmen erfüllt.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS-Services in der läuft AWS Cloud. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Auditoren von Drittanbietern testen und überprüfen die Effektivität unserer Sicherheitsmaßnahmen im Rahmen der AWS -Compliance-Programme regelmäßig. Weitere Informationen zu den Compliance-Programmen, die für gelten AWS CodePipeline, finden Sie AWS-Service unter Umfang nach Compliance-Programmen.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS-Service , was Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, die Anforderungen Ihres Unternehmens und die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung anwenden können CodePipeline. In den folgenden Themen erfahren Sie, wie Sie die Konfiguration vornehmen CodePipeline , um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere verwenden können AWS-Services , die Ihnen bei der Überwachung und Sicherung Ihrer CodePipeline Ressourcen helfen.

Themen

Datenschutz in AWS CodePipeline

Das Modell der AWS gemeinsamen Verantwortung und geteilter Verantwortung gilt für den Datenschutz in AWS CodePipeline. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit der Konsole, der API CodePipeline oder den SDKs arbeiten oder diese anderweitig AWS-Services verwenden. AWS CLI AWS Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine

Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Die folgenden bewährten Sicherheitsmethoden beziehen sich auch auf den Datenschutz in CodePipeline folgenden Bereichen:

- Konfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte für CodePipeline
- Wird verwendet AWS Secrets Manager , um Datenbankkennwörter oder API-Schlüssel von Drittanbietern zu verfolgen

Richtlinie für den Datenverkehr zwischen Netzwerken

Amazon VPC ist eine AWS-Service , mit der Sie AWS Ressourcen in einem von Ihnen definierten virtuellen Netzwerk (virtuelle private Cloud) starten können. CodePipelineunterstützt Amazon VPC-Endpoints powered by AWS PrivateLink, eine AWS Technologie, die die private Kommunikation zwischen AWS-Services über eine elastic network interface mit privaten IP-Adressen ermöglicht. Das bedeutet, dass Sie CodePipeline über einen privaten Endpunkt in Ihrer VPC eine direkte Verbindung herstellen können, sodass der gesamte Datenverkehr innerhalb Ihrer VPC und des AWS Netzwerks bleibt. Bisher benötigten Anwendungen, die in einer VPC ausgeführt wurden, Internetzugang, um eine Verbindung herzustellen. CodePipeline Mit einer VPC haben Sie die Kontrolle über Ihre Netzwerkeinstellungen, z. B.:

- IP-Adressbereich,
- Subnetze,
- Routing-Tabellen und
- Netzwerk-Gateways.

Um Ihre VPC zu verbinden CodePipeline, definieren Sie einen VPC-Schnittstellen-Endpunkt für. CodePipeline Dieser Endpunkttyp ermöglicht es Ihnen, eine Verbindung zu Ihrer VPC herzustellen AWS-Services. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität, CodePipeline ohne dass ein Internet-Gateway, eine NAT-Instanz (Network Address Translation) oder eine VPN-Verbindung erforderlich ist. Weitere Informationen zur Einrichtung einer VPC finden Sie im VPC-Benutzerhandbuch.

Verschlüsselung im Ruhezustand

Eingeschaltete CodePipeline Daten werden im Ruhezustand mit verschlüsselt AWS KMS keys. Code-Artefakte werden in einem kundeneigenen S3-Bucket gespeichert und entweder mit dem Von AWS verwalteter Schlüssel oder einem vom Kunden verwalteten Schlüssel verschlüsselt. Weitere Informationen finden Sie unter Konfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte für CodePipeline.

Verschlüsselung während der Übertragung

service-to-service Die gesamte Kommunikation wird während der Übertragung mit SSL/TLS verschlüsselt.

Verwaltung von Verschlüsselungsschlüsseln

Wenn Sie die Standardoption für die Verschlüsselung von Codeartefakten wählen, CodePipeline verwendet die. Von AWS verwalteter Schlüssel Sie können dies Von AWS verwalteter Schlüssel nicht ändern oder löschen. Wenn Sie einen vom Kunden verwalteten Schlüssel AWS KMS zum Verschlüsseln oder Entschlüsseln von Artefakten im S3-Bucket verwenden, können Sie diesen vom Kunden verwalteten Schlüssel nach Bedarf ändern oder rotieren.

> **Important**
>
> CodePipeline unterstützt nur symmetrische KMS-Schlüssel. Verwenden Sie keinen asymmetrischen KMS-Schlüssel, um die Daten in Ihrem S3-Bucket zu verschlüsseln.

Konfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte für CodePipeline

Es gibt zwei Möglichkeiten, die serverseitige Verschlüsselung für Amazon S3 S3-Artefakte zu konfigurieren:

- CodePipeline erstellt einen S3-Artefakt-Bucket und wird standardmäßig verwendet Von AWS verwalteter Schlüssel , wenn Sie mit dem Assistenten „Pipeline erstellen“ eine Pipeline erstellen. Das Von AWS verwalteter Schlüssel wird zusammen mit den Objektdaten verschlüsselt und von AWS verwaltet.
- Sie können Ihren eigenen vom Kunden verwalteten Schlüssel erstellen und verwalten.

> **Important**
>
> CodePipeline unterstützt nur symmetrische KMS-Schlüssel. Verwenden Sie keinen asymmetrischen KMS-Schlüssel, um die Daten in Ihrem S3-Bucket zu verschlüsseln.

Wenn Sie den Standard-S3-Schlüssel verwenden, können Sie diesen nicht ändern oder löschen. Von AWS verwalteter Schlüssel Wenn Sie einen vom Kunden verwalteten Schlüssel AWS KMS zum Verschlüsseln oder Entschlüsseln von Artefakten im S3-Bucket verwenden, können Sie diesen vom Kunden verwalteten Schlüssel nach Bedarf ändern oder rotieren.

Amazon S3 unterstützt Bucket-Richtlinien, die Sie verwenden können, wenn Sie eine serverseitige Verschlüsselung für alle in Ihrem Bucket gespeicherten Objekte benötigen. Beispielsweise verweigert die folgende Bucket-Richtlinie jedem die `s3:PutObject`-Berechtigung zum Hochladen von Objekten, wenn die Anfrage nicht den `x-amz-server-side-encryption`-Header enthält, der eine serverseitige Verschlüsselung mit SSE-KMS anfordert.

```
{
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-west-2-89050EXAMPLE/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-west-2-89050EXAMPLE/*",
            "Condition": {
                "Bool": {
```

```
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

Weitere Informationen zur serverseitigen Verschlüsselung finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung und Schützen von Daten mithilfe serverseitiger Verschlüsselung mit in (SSE-KMS) gespeicherten KMS-Schlüsseln. AWS KMS AWS Key Management Service

Weitere Informationen AWS KMS zu finden Sie im Entwicklerhandbuch.AWS Key Management Service

Themen

- Sehen Sie sich Ihre an Von AWS verwalteter Schlüssel
- Konfigurieren Sie die serverseitige Verschlüsselung für S3-Buckets mit oder AWS CloudFormationAWS CLI

Sehen Sie sich Ihre an Von AWS verwalteter Schlüssel

Wenn Sie den Assistenten Create Pipeline (Pipeline erstellen) verwenden, um Ihre erste Pipeline zu erstellen, wird in derselben Region, in der Sie die Pipeline erstellt haben, ein S3-Bucket erstellt. Der Bucket wird für das Speichern von Pipeline-Artefakten verwendet. Während der Ausführung einer Pipeline werden Artefakte in den S3-Bucket eingefügt und aus diesem abgerufen. CodePipeline Verwendet standardmäßig serverseitige Verschlüsselung AWS KMS mit dem Von AWS verwalteter Schlüssel für Amazon S3 (dem `aws/s3` Schlüssel). Dieser Von AWS verwalteter Schlüssel wird erstellt und in Ihrem AWS Konto gespeichert. Wenn Artefakte aus dem S3-Bucket abgerufen werden, CodePipeline verwendet denselben SSE-KMS-Prozess, um das Artefakt zu entschlüsseln.

Um Informationen zu Ihrem einzusehen Von AWS verwalteter Schlüssel

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS KMS Konsole.
2. Wenn eine Willkommensseite angezeigt wird, wählen Sie Jetzt loslegen.
3. Wählen Sie im Service-Navigationsbereich AWS -verwaltete Schlüssel aus.
4. Wählen Sie die Region für Ihre Pipeline aus. Wenn die Pipeline beispielsweise in erstellt wurde`us-east-2`, stellen Sie sicher, dass der Filter auf USA Ost (Ohio) eingestellt ist.

Weitere Informationen zu den Regionen und Endpunkten CodePipeline, für die verfügbar sind, finden Sie unter AWS CodePipeline Endpunkte und Kontingente.

5. Wählen Sie in der Liste den Schlüssel mit dem Alias aus, der für Ihre Pipeline verwendet wird (standardmäßig aws/s3). Grundlegende Informationen zum Schlüssel werden angezeigt.

Konfigurieren Sie die serverseitige Verschlüsselung für S3-Buckets mit oder AWS CloudFormationAWS CLI

Wenn Sie AWS CloudFormation oder AWS CLI zum Erstellen einer Pipeline verwenden, müssen Sie die serverseitige Verschlüsselung manuell konfigurieren. Verwenden Sie die obige Beispiel-Bucket-Richtlinie und erstellen Sie dann Ihren eigenen, vom Kunden verwalteten Schlüssel. Sie können anstelle von auch Ihre eigenen Schlüssel verwenden. Von AWS verwalteter Schlüssel Einige Gründe, warum Sie Ihren eigenen Schlüssel wählen sollten, sind unter anderem:

- Sie möchten den Schlüssel nach einem Plan wechseln, um Geschäfts- oder Sicherheitsanforderungen Ihrer Organisation zu erfüllen.
- Sie möchten eine Pipeline erstellen, die mit einem anderen AWS -Konto verknüpfte Ressourcen verwendet. Dies erfordert die Verwendung eines kundenverwalteten Schlüssels. Weitere Informationen finden Sie unter Erstellen Sie eine Pipeline CodePipeline , in der Ressourcen von einem anderen AWS Konto verwendet werden.

Die bewährten Methoden für die Kryptografie raten von einer extensiven Weiterverwendung von Verschlüsselungsschlüsseln ab. Das regelmäßige Wechseln Ihres Schlüssels stellt eine bewährte Methode dar. Um neues kryptografisches Material für Ihre AWS KMS Schlüssel zu erstellen, können Sie einen vom Kunden verwalteten Schlüssel erstellen und dann Ihre Anwendungen oder Aliase so ändern, dass sie den neuen vom Kunden verwalteten Schlüssel verwenden. Oder Sie können die automatische Schlüsselrotation für einen vorhandenen, vom Kunden verwalteten Schlüssel aktivieren.

Informationen zur Rotation Ihres vom Kunden verwalteten Schlüssels finden Sie unter Rotieren von Schlüsseln.

> **Important**
>
> CodePipeline unterstützt nur symmetrische KMS-Schlüssel. Verwenden Sie keinen asymmetrischen KMS-Schlüssel, um die Daten in Ihrem S3-Bucket zu verschlüsseln.

Wird verwendet AWS Secrets Manager , um Datenbankkennwörter oder API-Schlüssel von Drittanbietern zu verfolgen

Wir empfehlen, dass Sie es verwenden, AWS Secrets Manager um Datenbankanmeldedaten, API-Schlüssel und andere Geheimnisse während ihres gesamten Lebenszyklus zu rotieren, zu verwalten und abzurufen. Secrets Manager ermöglicht es Ihnen, hartcodierte Anmeldeinformationen in Ihrem Code (einschließlich Kennwörtern) durch einen API-Aufruf an Secrets Manager zu ersetzen, um das Geheimnis programmgesteuert abzurufen. Weitere Informationen finden Sie unter Was ist AWS Secrets Manager? im AWS Secrets Manager Manager-Benutzerhandbuch.

Für Pipelines, bei denen Sie geheime Parameter (wie OAuth-Anmeldeinformationen) in einer AWS CloudFormation Vorlage übergeben, sollten Sie dynamische Verweise in Ihre Vorlage aufnehmen, die auf die Secrets zugreifen, die Sie in Secrets Manager gespeichert haben. Das Referenz-ID-Muster und Beispiele finden Sie unter Secrets Manager Secrets im AWS CloudFormation Benutzerhandbuch. Ein Beispiel, das dynamische Verweise in einem Vorlagenausschnitt für GitHub Webhook in einer Pipeline verwendet, finden Sie unter Webhook-Ressourcenkonfiguration.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen unterstützen Sie bei der Arbeit mit der Verwaltung von geheimen Schlüsseln.

- Secrets Manager kann Datenbankanmeldedaten automatisch rotieren, z. B. für die Rotation von Amazon RDS-Geheimnissen. Weitere Informationen finden Sie unter Rotation Ihrer AWS Secrets Manager Manager-Geheimnisse im AWS Secrets Manager Manager-Benutzerhandbuch.
- Anweisungen zum Hinzufügen dynamischer Verweise von Secrets Manager zu Ihren AWS CloudFormation -Vorlagen finden Sie unter https://aws.amazon.com/blogs/security/how-to-create-and-retrieve-secrets-managed-in-aws-secrets-manager-using-aws-cloudformation-template/.

Identity and Access Management für AWS CodePipeline

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. CodePipeline IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

Zielgruppe

Die Art und Weise, wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, in der Sie tätig sind. CodePipeline

Dienstbenutzer — Wenn Sie den CodePipeline Dienst für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die erforderlichen Anmeldeinformationen und Berechtigungen zur Verfügung. Wenn Sie für Ihre Arbeit mehr CodePipeline Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie in nicht auf eine Funktion zugreifen können CodePipeline, finden Sie weitere Informationen unterFehlerbehebung für AWS CodePipeline -Identität und -Zugriff.

Serviceadministrator — Wenn Sie in Ihrem Unternehmen für CodePipeline Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf CodePipeline. Es ist Ihre Aufgabe, zu bestimmen,

auf welche CodePipeline Funktionen und Ressourcen Ihre Servicebenutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM nutzen kann CodePipeline, finden Sie unterWie AWS CodePipeline funktioniert mit IAM.

IAM-Administrator — Wenn Sie ein IAM-Administrator sind, möchten Sie vielleicht mehr darüber erfahren, wie Sie Richtlinien schreiben können, um den Zugriff darauf zu verwalten. CodePipeline Beispiele für CodePipeline identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie unter. AWS CodePipeline Beispiele für identitätsbasierte -Richtlinien

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie im IAM-Benutzerhandbuch unter AWS API-Anfragen signieren.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere

Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

Root-Benutzer des AWS-Kontos

Wenn Sie eine erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle Ressourcen im AWS-Services Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb von Ihnen AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management Console indem Sie die Rollen wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon-EC2 aus oder speichert Objekte in Amazon-S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.

- Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
- Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
- Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

- Auf Amazon EC2 ausgeführte Anwendungen — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und API-Anfragen stellen AWS CLI . AWS Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instance-Profil, das an die Instance angehängt ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder

Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service Control Policies (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Entitäten. Root-Benutzer des AWS-Kontos Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations -Benutzerhandbuch.

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Wie AWS CodePipeline funktioniert mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf zu verwalten CodePipeline, sollten Sie wissen, mit welchen IAM-Funktionen Sie verwenden können. CodePipeline Einen allgemeinen Überblick darüber, wie CodePipeline und welche Funktionen mit IAM funktionieren AWS-ServicesAWS-Services , finden Sie im IAM-Benutzerhandbuch unter Diese Funktionen mit IAM.

Themen

- Identitätsbasierte CodePipeline-Richtlinien
- CodePipeline ressourcenbasierte Richtlinien
- Autorisierung auf der Basis von CodePipeline -Tags
- CodePipeline IAM-Rollen

Identitätsbasierte CodePipeline-Richtlinien

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. CodePipeline unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Aktionen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Bei Richtlinienaktionen wird vor der Aktion das folgende Präfix CodePipeline verwendet:`codepipeline:`.

Um beispielsweise jemandem die Berechtigung zum Anzeigen der vorhandenen Pipelines im Konto zu erteilen, fügen Sie die Aktion `codepipeline:GetPipeline` in ihre Richtlinie ein. Richtlinienerklärungen müssen `Action` entweder ein `NotAction` Oder-Element enthalten. CodePipeline definiert einen eigenen Satz von Aktionen, die Aufgaben beschreiben, die Sie mit diesem Dienst ausführen können.

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie wie folgt durch Kommata:

```
"Action": [
      "codepipeline:action1",
      "codepipeline:action2"
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Get` beginnen, einschließlich der folgenden Aktion:

```
"Action": "codepipeline:Get*"
```

Eine Liste der CodePipeline Aktionen finden Sie AWS CodePipeline im IAM-Benutzerhandbuch unter Definierte Aktionen von.

Ressourcen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

CodePipeline Ressourcen und Abläufe

In ist CodePipeline die primäre Ressource eine Pipeline. In einer Richtlinie verwenden Sie einen Amazon-Ressourcennamen (ARN), um die Ressource zu identifizieren, für die die Richtlinie gilt. CodePipeline unterstützt andere Ressourcen, die mit der primären Ressource verwendet werden können, z. B. Stufen, Aktionen und benutzerdefinierte Aktionen. Diese werden als Unterressourcen bezeichnet. Diesen Ressourcen und Subressourcen sind eindeutige Amazon-Ressourcennamen (ARN) zugeordnet. Weitere Informationen zu ARNs finden Sie unter Amazon Resource Names (ARN) and AWS-Service Namespaces in der. Allgemeine Amazon Web Services-Referenz Um den mit Ihrer Pipeline verknüpften Pipeline-ARN abzurufen, finden Sie den Pipeline-ARN unter Einstellungen in der Konsole. Weitere Informationen finden Sie unter Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen.

Ressourcentyp	ARN-Format
Pipeline	*arn:aws:codepipeline: region: konto: Pipeline-Name*
Stufe	*arn:aws:codepipeline: region: konto: Pipeline-Name/Stagenname*
Aktion	*arn:aws:codepipeline: region: konto: Pipeline-Name/ Stagenname/ Aktionsname*
Benutzerdefinierte Aktion	*arn:aws:codepipeline: region: konto:aktionstyp: eigentümer/kategorie/anbieter/version*

Ressourcentyp	ARN-Format
CodePipeline Alle Ressourcen	arn:aws:codepipeline: *
Alle CodePipeline Ressourcen, die dem angegebenen Konto in der angegebenen Region gehören	*arn:aws:codepipeline: region: konto: ****

> **Note**
>
> Die meisten Dienste in AWS behandeln einen Doppelpunkt (:) oder einen Schrägstrich (/) als dasselbe Zeichen in ARNs. CodePipeline Verwendet jedoch eine exakte Übereinstimmung in den Ereignismustern und Regeln. Verwenden Sie also die richtigen ARN-Zeichen zum Erstellen von Ereignismustern, sodass sie mit der ARN-Syntax in der Pipeline übereinstimmen.

In gibt es API-Aufrufe CodePipeline, die Berechtigungen auf Ressourcenebene unterstützen. Berechtigungen auf Ressourcenebene geben an, ob ein API-Aufruf einen Ressourcen-ARN angeben können, oder ob der API-Aufruf nur alle Ressourcen unter Verwendung des Platzhalters angeben kann. Eine CodePipeline Referenz zu Berechtigungen ausführliche Beschreibung der Berechtigungen auf Ressourcenebene und eine Liste der CodePipeline API-Aufrufe, die Berechtigungen auf Ressourcenebene unterstützen, finden Sie unter.

Sie können beispielsweise folgendermaßen eine bestimmte Pipeline (*myPipeline*) in Ihrer Anweisung mit dem ARN angeben:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:myPipeline"
```

Sie können auch alle Pipelines angeben, die zu einem bestimmten Konto gehören, indem Sie das Platzhalterzeichen (*) wie folgt verwenden:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:*"
```

Wenn Sie alle Ressourcen angeben möchten oder wenn eine bestimmte API-Aktion keine ARNs unterstützt, verwenden Sie das Platzhalterzeichen (*) wie folgt im `Resource`-Element:

```
"Resource": "*"
```

> **Note**
>
> Beachten Sie bei der Erstellung von IAM-Richtlinien die standardmäßigen Sicherheitsempfehlungen zur Gewährung von geringsten Rechten, d. h. gewähren Sie nur die Berechtigungen, die für die Ausführung einer Aufgabe erforderlich sind. Wenn ein API-Aufruf ARNs unterstützt, unterstützt er Berechtigungen auf Ressourcenebene, und Sie müssen das Platzhalterzeichen (*) nicht verwenden.

Einige CodePipeline API-Aufrufe akzeptieren mehrere Ressourcen (z. B.). `GetPipeline` Um mehrere Ressourcen in einer einzigen Anweisung anzugeben, trennen Sie die ARNs mit Komma, wie folgt:

```
"Resource": ["arn1", "arn2"]
```

CodePipeline bietet eine Reihe von Operationen für die Arbeit mit den CodePipeline Ressourcen. Eine Liste der verfügbaren Operationen finden Sie unter CodePipeline Referenz zu Berechtigungen.

Bedingungsschlüssel

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

CodePipeline definiert seinen eigenen Satz von Bedingungsschlüsseln und unterstützt auch die Verwendung einiger globaler Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Alle Amazon EC2-Aktionen unterstützen die Bedingungsschlüssel `aws:RequestedRegion` und `ec2:Region`. Weitere Informationen finden Sie unter Beispiel: Einschränken des Zugriffs auf eine bestimmte Region.

Eine Liste der CodePipeline Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für AWS CodePipeline im IAM-Benutzerhandbuch. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Definierte Aktionen von AWS CodePipeline.

Beispiele

Beispiele für CodePipeline identitätsbasierte Richtlinien finden Sie unter. AWS CodePipeline Beispiele für identitätsbasierte -Richtlinien

CodePipeline ressourcenbasierte Richtlinien

CodePipeline unterstützt keine ressourcenbasierten Richtlinien. Es wird jedoch ein Beispiel für eine ressourcenbasierte Richtlinie für den S3-Dienst bereitgestellt, der sich auf. CodePipeline

Beispiele

Beispiele für CodePipeline ressourcenbasierte Richtlinien finden Sie unter Beispiele für eine ressourcenbasierte AWS CodePipeline -Richtlinie

Autorisierung auf der Basis von CodePipeline -Tags

Sie können Tags an CodePipeline Ressourcen anhängen oder Tags in einer Anfrage an übergeben. CodePipeline Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `codepipeline:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden. Weitere Informationen zum Markieren von CodePipeline Ressourcen finden Sie unterMarkieren von Ressourcen.

Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Verwenden von Tags zur Steuerung des Zugriffs auf Ressourcen CodePipeline.

CodePipeline IAM-Rollen

Eine IAM-Rolle ist eine Entität in Ihrem AWS Konto, die über bestimmte Berechtigungen verfügt.

Verwenden temporärer Anmeldeinformationen mit CodePipeline

Sie können temporäre Anmeldeinformationen verwenden, um sich über einen Verbund anzumelden, eine IAM-Rolle anzunehmen oder eine kontenübergreifende Rolle anzunehmen. Sie erhalten temporäre Sicherheitsanmeldedaten, indem Sie AWS STS API-Operationen wie AssumeRoleoder aufrufen GetFederationToken.

CodePipeline unterstützt die Verwendung temporärer Anmeldeinformationen.

Servicerollen

CodePipeline ermöglicht es einem Dienst, in Ihrem Namen eine Servicerolle zu übernehmen. Diese Rolle gewährt dem Service Zugriff auf Ressourcen in anderen Diensten, um eine Aktion in Ihrem Namen auszuführen. Servicerollen werden in Ihrem IAM-Konto angezeigt und gehören zum Konto. Dies bedeutet, dass ein IAM-Administrator die Berechtigungen für diese Rolle ändern kann. Dies kann jedoch die Funktionalität des Dienstes beeinträchtigen.

CodePipeline unterstützt Servicerollen.

AWS CodePipeline Beispiele für identitätsbasierte -Richtlinien

Standardmäßig sind IAM-Benutzer und -Rollen nicht berechtigt, CodePipeline Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mit der AWS Management Console AWS

CLI, oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern und Rollen die Berechtigung zum Ausführen bestimmter API-Operationen für die angegebenen Ressourcen gewähren, die diese benötigen. Der Administrator muss diese Richtlinien anschließend den IAM-Benutzern oder -Gruppen anfügen, die diese Berechtigungen benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von Richtlinien auf der JSON-Registerkarte im IAM-Benutzerhandbuch.

Informationen zum Erstellen einer Pipeline, die Ressourcen von einem anderen Konto verwendet, sowie die entsprechenden Beispielrichtlinien finden Sie unterErstellen Sie eine Pipeline CodePipeline , in der Ressourcen von einem anderen AWS Konto verwendet werden.

Themen

- Bewährte Methoden für Richtlinien
- Anzeigen von Ressourcen in der Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer
- Beispiele für identitätsbasierte Richtlinien (IAM)
- Verwenden von Tags zur Steuerung des Zugriffs auf Ressourcen CodePipeline
- Erforderliche Berechtigungen für die Verwendung der CodePipeline -Konsole
- AWS verwaltete Richtlinien für AWS CodePipeline
- Beispiele für vom Kunden verwaltete Richtlinien

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand CodePipeline Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden

Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.

- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Anzeigen von Ressourcen in der Konsole

Die CodePipeline Konsole benötigt die `ListRepositories` Erlaubnis, eine Liste von Repositorys für Ihr AWS Konto in der AWS Region anzuzeigen, in der Sie angemeldet sind. Die Konsole umfasst auch eine Funktion Go to resource (Zu Ressource wechseln), um schnell ohne Berücksichtigung der Groß- und Kleinschreibung nach Ressourcen zu suchen. Diese Suche wird in Ihrem AWS Konto in der AWS Region durchgeführt, in der Sie angemeldet sind. Die folgenden Ressourcen werden für die folgenden Services angezeigt:

- AWS CodeBuild: Build-Projekte
- AWS CodeCommit: Repositorys
- AWS CodeDeploy: Anwendungen
- AWS CodePipeline: Pipelines

Für diese Suche unter den Ressourcen in allen Services müssen Sie über die folgenden Berechtigungen verfügen:

- CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

Es werden keine Ergebnisse für die Ressourcen eines Service zurückgegeben, wenn Sie nicht über Berechtigungen für diesen Service verfügen. Auch wenn Sie über Berechtigungen zum Anzeigen von Ressourcen verfügen, werden manche Ressourcen nicht zurückgegeben, wenn die Anzeige dieser Ressourcen ausdrücklich verweigert wird (`Deny`).

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder AWS .

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Beispiele für identitätsbasierte Richtlinien (IAM)

Richtlinien können IAM-Identitäten angefügt werden. Sie können z. B. Folgendes tun:

- Ordnen Sie einem Benutzer oder einer Gruppe in Ihrem Konto eine Berechtigungsrichtlinie zu — Um einem Benutzer Berechtigungen zum Anzeigen von Pipelines in der CodePipeline Konsole zu gewähren, können Sie eine Berechtigungsrichtlinie an einen Benutzer oder eine Gruppe anhängen, zu der der Benutzer gehört.
- Einer Rolle eine Berechtigungsrichtlinie zuweisen (kontoübergreifende Berechtigungen gewähren) – Sie können einer IAM-Rolle eine identitätsbasierte Berechtigungsrichtlinie zuweisen, um kontoübergreifende Berechtigungen zu erteilen. Der Administrator in Konto A kann

beispielsweise AWS-Service wie folgt eine Rolle erstellen, um einem anderen Konto (z. B. AWS Konto B) kontenübergreifende Berechtigungen zu gewähren:

1. Der Administrator von Konto A erstellt eine IAM-Rolle und fügt ihr eine Berechtigungsrichtlinie an, die Berechtigungen für Ressourcen in Konto A erteilt.
2. Der Administrator von Konto A weist der Rolle eine Vertrauensrichtlinie zu, die Konto B als den Prinzipal identifiziert, der die Rolle übernehmen kann.
3. Der Administrator von Konto B kann dann die Berechtigungen zur Übernahme der Rolle an alle Benutzer in Konto B delegieren. Auf diese Weise können Benutzer in Konto B Ressourcen in Konto A erstellen oder darauf zugreifen. Der Principal in der Vertrauensrichtlinie kann auch ein AWS-Service Principal sein, wenn Sie einem Benutzer die AWS-Service Rechte zur Übernahme der Rolle erteilen möchten.

Weitere Informationen zum Delegieren von Berechtigungen mithilfe von IAM finden Sie unter Zugriffsverwaltung im IAM-Benutzerhandbuch.

Im Folgenden finden Sie ein Beispiel für eine Berechtigungsrichtlinie, die Berechtigungen zum Deaktivieren und Aktivieren von Übergängen zwischen allen Phasen in der Pipeline gewährt, die `MyFirstPipeline` `us-west-2 region` im folgenden aufgeführt sind:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codepipeline:EnableStageTransition",
        "codepipeline:DisableStageTransition"
      ],
      "Resource" : [
        "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/*"
      ]
    }
  ]
}
```

Das folgende Beispiel zeigt eine Richtlinie im 111222333444-Konto, die es Benutzern ermöglicht, die in der Konsole angegebene `MyFirstPipeline` Pipeline anzuzeigen, aber nicht zu ändern. CodePipeline Diese Richtlinie basiert auf der verwalteten Richtlinie

`AWSCodePipeline_ReadOnlyAccess`, aber da sie spezifisch für die Pipeline `MyFirstPipeline` gilt, kann sie die verwaltete Richtlinie nicht direkt verwenden. Wenn Sie die Richtlinie nicht auf eine bestimmte Pipeline beschränken möchten, sollten Sie in Erwägung ziehen, eine der verwalteten Richtlinien zu verwenden, die von erstellt und verwaltet wurden. CodePipeline Weitere Informationen finden Sie unter Arbeiten mit verwalteten Richtlinien. Sie müssen diese Richtlinie einer IAM-Rolle zuordnen, die Sie für den Zugriff erstellen, z. B. einer Rolle mit dem Namen`CrossAccountPipelineViewers`:

```
{
  "Statement": [
    {
      "Action": [
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListActionExecutions",
        "codepipeline:ListActionTypes",
        "codepipeline:ListPipelines",
        "codepipeline:ListTagsForResource",
        "iam:ListRoles",
        "s3:ListAllMyBuckets",
        "codecommit:ListRepositories",
        "codedeploy:ListApplications",
        "lambda:ListFunctions",
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"
    },
    {
      "Action": [
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListActionExecutions",
        "codepipeline:ListActionTypes",
        "codepipeline:ListPipelines",
        "codepipeline:ListTagsForResource",
```

```
        "iam:ListRoles",
        "s3:GetBucketPolicy",
        "s3:GetObject",
        "s3:ListBucket",
        "codecommit:ListBranches",
        "codedeploy:GetApplication",
        "codedeploy:GetDeploymentGroup",
        "codedeploy:ListDeploymentGroups",
        "elasticbeanstalk:DescribeApplications",
        "elasticbeanstalk:DescribeEnvironments",
        "lambda:GetFunctionConfiguration",
        "opsworks:DescribeApps",
        "opsworks:DescribeLayers",
        "opsworks:DescribeStacks"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:DescribeNotificationRule"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "codestar-notifications:NotificationsForResource": "arn:aws:codepipeline:*"
        }
      }
    }
  ],
  "Version": "2012-10-17"
}
```

Nachdem Sie diese Richtlinie erstellt haben, erstellen Sie die IAM-Rolle im 111222333444-Konto und fügen Sie die Richtlinie dieser Rolle hinzu. Zu den Vertrauensstellungen der Rolle müssen Sie das AWS Konto hinzufügen, das diese Rolle übernehmen soll. Das folgende Beispiel zeigt eine Richtlinie, die es Benutzern des *111111111111-Kontos ermöglicht, Rollen anzunehmen, die im AWS 111222333444-Konto* definiert sind:

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Das folgende Beispiel zeigt eine im 111111111111-Konto erstellte Richtlinie, die es Benutzern ermöglicht, die im *111222333444-Konto angegebene Rolle anzunehmen* AWS : *CrossAccountPipelineViewers*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::111222333444:role/CrossAccountPipelineViewers"
        }
    ]
}
```

Sie können IAM-Richtlinien erstellen, um die Anrufe und Ressourcen einzuschränken, auf die Benutzer in Ihrem Konto Zugriff haben, und diese Richtlinien dann Ihrem Administratorbenutzer zuordnen. Weitere Informationen zum Erstellen von IAM-Rollen und Beispiele für CodePipeline IAM-Richtlinienerklärungen finden Sie unter. Beispiele für vom Kunden verwaltete Richtlinien

Verwenden von Tags zur Steuerung des Zugriffs auf Ressourcen CodePipeline

Bedingungen in IAM-Richtlinienanweisungen sind Teil der Syntax, mit der Sie Berechtigungen für Ressourcen angeben, die für Aktionen erforderlich sind. CodePipeline Das Verwenden von Tags in Bedingungen ist eine Möglichkeit zur Kontrolle des Zugriffs auf Ressourcen und Anfragen. Informationen zum Markieren von CodePipeline Ressourcen finden Sie unter. Markieren von Ressourcen Dieses Thema behandelt die Tag-basierte Zugriffskontrolle.

Wenn Sie IAM-Richtlinien entwerfen, können Sie präzise abgestufte Berechtigungen festlegen, indem Sie Zugriff auf bestimmte Ressourcen gewähren. Mit zunehmender Anzahl der Ressourcen, die

Sie verwalten, wird diese Aufgabe erschwert. Durch Markieren von Ressourcen und Verwenden von Tags in Richtlinienanweisungsbedingungen lässt sich diese Aufgabe vereinfachen. Sie erteilen Massenzugriff auf eine beliebige Ressource mit einem bestimmten Tag. Anschließend wenden Sie dieses Tag während der Erstellung oder zu einem späteren Zeitpunkt wiederholt auf relevante Ressourcen an.

Markierungen können an die Ressource angehängt oder in der Anfrage an Services übergeben werden, die das Markieren unterstützen. In CodePipeline können Ressourcen Tags haben, und einige Aktionen können Tags enthalten. Wenn Sie eine IAM-Richtlinie erstellen, können Sie Tag-Bedingungsschlüssel verwenden, um Folgendes zu kontrollieren:

- Welche Benutzer Aktionen für eine Pipeline-Ressource ausführen können, basierend auf den Tags, über die diese bereits verfügt.
- Welche Tags in der Anforderung einer Aktion übergeben werden können.
- Ob bestimmte Tag-Schlüssel in einer Anforderung verwendet werden können.

Mit String-Bedingungsoperatoren können Sie `Condition`-Elemente erstellen, die den Zugriff basierend auf einen Vergleich eines Schlüssels mit einem Zeichenfolgewert einschränken. Sie können am Ende jeder Bedingung einen Operatornamen mit Ausnahme der Null-Bedingung hinzufügen`IfExists`. Damit geben Sie zum Ausdruck: "Wenn der Richtlinienschlüssel im Anforderungskontext vorhanden ist, wird der Schlüssel den Angaben in der Richtlinie entsprechend verarbeitet. Wenn der Schlüssel nicht vorhanden ist, wird das Bedingungselement als "true" ausgewertet. Sie können z. B. die Einschränkung anhand `StringEqualsIfExists` von Bedingungsschlüsseln vornehmen, die bei anderen Ressourcentypen möglicherweise nicht vorhanden sind.

Die vollständige Syntax und Semantik der Bedingungsschlüssel für Tags finden Sie unter Zugriffskontrolle mithilfe von Tags. Weitere Informationen zu Bedingungsschlüsseln finden Sie in den folgenden Ressourcen. Die CodePipeline Richtlinienbeispiele in diesem Abschnitt orientieren sich an den folgenden Informationen zu Bedingungsschlüsseln und werden durch Beispiele für Nuancen CodePipeline wie die Verschachtelung von Ressourcen erweitert.

- Bedingungsoperatoren für Zeichenfolgen
- AWS-Services die mit IAM funktionieren
- SCP-Syntax
- IAM-JSON-Richtlinienelemente: Condition
- aws: RequestTag /tag-key

- Zustandstasten für CodePipeline

Die folgenden Beispiele zeigen, wie Tag-Bedingungen in Richtlinien für CodePipeline Benutzer angegeben werden.

Example 1: Einschränken von Aktionen basierend auf Tags in der Anforderung

Die Richtlinie für `AWSCodePipeline_FullAccess` verwaltete Benutzer gibt Benutzern uneingeschränkte Rechte, jede CodePipeline Aktion auf einer beliebigen Ressource auszuführen.

Die folgende Richtlinie schränkt diese Befugnis ein und verweigert nicht autorisierten Benutzern die Erlaubnis, Pipelines zu erstellen, in denen bestimmte Tags in der Anfrage aufgeführt sind. Dazu verweigert es die `CreatePipeline`-Aktion, wenn die Anforderung ein Tag mit dem Namen `Project` und einem der Werte `ProjectA` oder `ProjectB` festlegt. (Der Bedingungsschlüssel `aws:RequestTag` wird verwendet, um zu steuern, welche Tags in einer IAM-Anforderung übergeben werden können.)

Im folgenden Beispiel besteht die Absicht der Richtlinie darin, nicht autorisierten Benutzern die Erlaubnis zu verweigern, eine Pipeline mit den angegebenen Tag-Werten zu erstellen. Das Erstellen einer Pipeline erfordert jedoch zusätzlich zu der Pipeline selbst auch den Zugriff auf Ressourcen (z. B. Pipeline-Aktionen und -Phasen). Da in der Richtlinie `'Resource'` angegeben`'*'`, wird die Richtlinie anhand jeder Ressource ausgewertet, die über einen ARN verfügt, und wird erstellt, wenn die Pipeline erstellt wird. Diese zusätzlichen Ressourcen verfügen nicht über den Tag-Bedingungsschlüssel, sodass die `StringEquals` Prüfung fehlschlägt und dem Benutzer nicht die Möglichkeit gewährt wird, eine Pipeline zu erstellen. Um dieses Problem zu beheben, verwenden Sie stattdessen den Bedingungsoperator `StringEqualsIfExists`. Auf diese Weise findet die Prüfung nur dann statt, wenn der Bedingungsschlüssel existiert.

Sie könnten Folgendes wie folgt lesen: „Wenn die geprüfte Ressource über einen `"RequestTag/Project"` Tag-Bedingungsschlüssel verfügt, lassen Sie die Aktion nur zu, wenn der Schlüsselwert mit beginnt`projectA`. Wenn die zu prüfende Ressource nicht über diesen Bedingungsschlüssel verfügt, ist dies belanglos."

Darüber hinaus verhindert die Richtlinie, dass diese nicht autorisierten Benutzer die Ressourcen manipulieren, indem sie den `aws:TagKeys` Bedingungsschlüssel verwenden, um zu verhindern, dass Aktionen zur Änderung von Tags dieselben Tagwerte enthalten. Der Administrator eines Kunden muss diese IAM-Richtlinie zusätzlich zur Richtlinie für verwaltete Benutzer mit Administratorrechten verknüpfen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:CreatePipeline",
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "aws:RequestTag/Project": ["ProjectA", "ProjectB"]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Example 2: Beschränken Sie Tagging-Aktionen auf der Grundlage von Ressourcen-Tags

Die Richtlinie für `AWSCodePipeline_FullAccess` verwaltete Benutzer gibt Benutzern uneingeschränkte Rechte, jede CodePipeline Aktion auf einer beliebigen Ressource auszuführen.

Die folgende Richtlinie beschränkt diese Befugnis und verweigert nicht autorisierten Benutzern die Berechtigung, Aktionen für bestimmte Projekt-Pipelines auszuführen. Hierfür verweigert sie einige Aktionen, wenn die Ressource über ein Tag mit dem Namen `Project` und einem der Werte `ProjectA` oder `ProjectB` verfügt. (Der Bedingungsschlüssel `aws:ResourceTag` wird verwendet, um den Zugriff auf die Ressourcen auf der Grundlage der Tags dieser Ressourcen zu steuern.) Der

Administrator eines Kunden muss diese IAM-Richtlinie nicht autorisierten IAM-Benutzern hinzufügen, zusätzlich zu der verwalteten Benutzerrichtlinie.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": ["ProjectA", "ProjectB"]
        }
      }
    }
  ]
}
```

Example 3: Zulassen von Aktionen basierend auf Tags in der Anforderung

Die folgende Richtlinie gewährt Benutzern die Erlaubnis, Entwicklungspipelines in CodePipeline zu erstellen.

Hierfür werden die Aktionen `CreatePipeline` und `TagResource` zugelassen, wenn die Anforderung ein Tag mit dem Namen `Project` und dem Wert `ProjectA` angibt. Mit anderen Worten, der einzige Tag-Schlüssel, der angegeben werden kann`Project`, ist, und sein `ProjectA` Wert muss.

Der `aws:RequestTag` Bedingungsschlüssel wird verwendet, um zu steuern, welche Tags in einer IAM-Anfrage übergeben werden können. Die Bedingung `aws:TagKeys` stellt sicher, dass bei Tag-Schlüsseln die Groß- und Kleinschreibung beachtet wird. Diese Richtlinie ist nützlich für Benutzer oder Rollen, denen die Richtlinie für `AWSCodePipeline_FullAccess` verwaltete Benutzer nicht zugeordnet ist. Die verwaltete Richtlinie gibt Benutzern uneingeschränkte Rechte, jede CodePipeline Aktion auf einer beliebigen Ressource auszuführen.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:CreatePipeline",
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Project": "ProjectA"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Example 4: Beschränken Sie die Aktionen zum Entfernen von Tags auf der Grundlage von Ressourcen-Tags

Die Richtlinie für `AWSCodePipeline_FullAccess` verwaltete Benutzer gibt Benutzern uneingeschränkte Rechte, jede CodePipeline Aktion auf einer beliebigen Ressource auszuführen.

Die folgende Richtlinie beschränkt diese Befugnis und verweigert nicht autorisierten Benutzern die Berechtigung, Aktionen für bestimmte Projekt-Pipelines auszuführen. Hierfür verweigert sie einige Aktionen, wenn die Ressource über ein Tag mit dem Namen `Project` und einem der Werte `ProjectA` oder `ProjectB` verfügt.

Außerdem verhindert die Richtlinie, dass diese nicht autorisierten Benutzer die Ressourcen manipulieren, indem sie mithilfe des `aws:TagKeys` Bedingungsschlüssels verhindern, dass das Tag vollständig entfernt wird`Project`. Der Administrator eines Kunden muss diese IAM-Richtlinie zusätzlich zur Richtlinie für verwaltete Benutzer oder Rollen mit nicht autorisierten Benutzern oder Rollen verknüpfen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
```

```
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Erforderliche Berechtigungen für die Verwendung der CodePipeline -Konsole

Für die Verwendung CodePipeline in der CodePipeline Konsole benötigen Sie mindestens Berechtigungen für die folgenden Dienste:

- AWS Identity and Access Management
- Amazon Simple Storage Service

Mit diesen Berechtigungen können Sie andere AWS Ressourcen für Ihr AWS Konto beschreiben.

Abhängig von den sonstigen Services, die Sie in Ihre Pipelines integrieren, benötigen Sie möglicherweise Berechtigungen von einem oder mehreren der folgenden Services:

- AWS CodeCommit
- CodeBuild
- AWS CloudFormation
- AWS CodeDeploy
- AWS Elastic Beanstalk
- AWS Lambda
- AWS OpsWorks

Wenn Sie eine IAM-Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Benutzer mit dieser IAM-Richtlinie. Um sicherzustellen, dass diese Benutzer die CodePipeline Konsole weiterhin verwenden können, fügen Sie dem Benutzer auch die `AWSCodePipeline_ReadOnlyAccess` verwaltete Richtlinie bei, wie unter beschriebenAWS verwaltete Richtlinien für AWS CodePipeline.

Sie müssen Benutzern, die die API AWS CLI oder die CodePipeline API aufrufen, keine Mindestberechtigungen für die Konsole gewähren.

AWS verwaltete Richtlinien für AWS CodePipeline

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. AWS Verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele gängige Anwendungsfälle bereitstellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie kundenverwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

> **Important**
>
> Die von AWS verwalteten Richtlinien `AWSCodePipelineFullAccess` und `AWSCodePipelineReadOnlyAccess` wurden ersetzt. Verwenden Sie die `AWSCodePipeline_ReadOnlyAccess` Richtlinien `AWSCodePipeline_FullAccess` und.

AWS verwaltete Richtlinie: **`AWSCodePipeline_FullAccess`**

Dies ist eine Richtlinie, die vollen Zugriff auf gewährt CodePipeline. Informationen zum JSON-Richtliniendokument in der IAM-Konsole finden Sie unter AWSCodePipeline_FullAccess.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `codepipeline`— Erteilt Berechtigungen für. CodePipeline
- `chatbot`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Ressourcen in AWS Chatbot zu verwalten.
- `cloudformation`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Ressourcenstapel zu verwalten. AWS CloudFormation
- `cloudtrail`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Logging-Ressourcen zu verwalten. CloudTrail
- `codebuild`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, auf Build-Ressourcen zuzugreifen. CodeBuild
- `codecommit`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, auf Quellressourcen in zuzugreifen. CodeCommit
- `codedeploy`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, auf Bereitstellungsressourcen in zuzugreifen. CodeDeploy
- `codestar-notifications`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, auf Ressourcen in AWS CodeStar Benachrichtigungen zuzugreifen.
- `ec2`— Erteilt Berechtigungen, um Bereitstellungen CodeCatalyst zur Verwaltung von Elastic Load Balancing in Amazon EC2 zuzulassen.
- `ecr`— Erteilt Berechtigungen für den Zugriff auf Ressourcen in Amazon ECR.
- `elasticbeanstalk`— Erteilt Berechtigungen, die es Principals ermöglichen, auf Ressourcen in Elastic Beanstalk zuzugreifen.
- `iam`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Rollen und Richtlinien in IAM zu verwalten.
- `lambda`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Ressourcen in Lambda zu verwalten.
- `events`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Ressourcen in Events zu verwalten. CloudWatch
- `opsworks`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Ressourcen in zu verwalten. AWS OpsWorks

- `s3`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Ressourcen in Amazon S3 zu verwalten.
- `sns`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Benachrichtigungsressourcen in Amazon SNS zu verwalten.
- `states`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Zustandsmaschinen einzusehen, in denen sie sich befinden. AWS Step Functions Eine Zustandsmaschine besteht aus einer Sammlung von Zuständen, die Aufgaben verwalten und den Übergang zwischen Zuständen ermöglichen.

```
{
    "Statement": [
        {
            "Action": [
                "codepipeline:*",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStacks",
                "cloudformation:ListChangeSets",
                "cloudtrail:DescribeTrails",
                "codebuild:BatchGetProjects",
                "codebuild:CreateProject",
                "codebuild:ListCuratedEnvironmentImages",
                "codebuild:ListProjects",
                "codecommit:ListBranches",
                "codecommit:GetReferences",
                "codecommit:ListRepositories",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecs:ListClusters",
                "ecs:ListServices",
                "elasticbeanstalk:DescribeApplications",
                "elasticbeanstalk:DescribeEnvironments",
                "iam:ListRoles",
                "iam:GetRole",
                "lambda:ListFunctions",
```

```
                "events:ListRules",
                "events:ListTargetsByRule",
                "events:DescribeRule",
                "opsworks:DescribeApps",
                "opsworks:DescribeLayers",
                "opsworks:DescribeStacks",
                "s3:ListAllMyBuckets",
                "sns:ListTopics",
                "codestar-notifications:ListNotificationRules",
                "codestar-notifications:ListTargets",
                "codestar-notifications:ListTagsforResource",
                "codestar-notifications:ListEventTypes",
                "states:ListStateMachines"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Sid": "CodePipelineAuthoringAccess"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketPolicy",
                "s3:GetBucketVersioning",
                "s3:GetObjectVersion",
                "s3:CreateBucket",
                "s3:PutBucketPolicy"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3::*:codepipeline-*",
            "Sid": "CodePipelineArtifactsReadWriteAccess"
        },
        {
            "Action": [
                "cloudtrail:PutEventSelectors",
                "cloudtrail:CreateTrail",
                "cloudtrail:GetEventSelectors",
                "cloudtrail:StartLogging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:cloudtrail:*:*:trail/codepipeline-source-trail",
            "Sid": "CodePipelineSourceTrailReadWriteAccess"
        },
        {
```

```
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/service-role/cwe-role-*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "events.amazonaws.com"
                    ]
                }
            },
            "Sid": "EventsIAMPassRole"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "codepipeline.amazonaws.com"
                    ]
                }
            },
            "Sid": "CodePipelineIAMPassRole"
        },
        {
            "Action": [
                "events:PutRule",
                "events:PutTargets",
                "events:DeleteRule",
                "events:DisableRule",
                "events:RemoveTargets"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:events:*:*:rule/codepipeline-*"
            ],
            "Sid": "CodePipelineEventsReadWriteAccess"
```

```
        },
        {
            "Sid": "CodeStarNotificationsReadWriteAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-notifications:CreateNotificationRule",
                "codestar-notifications:DescribeNotificationRule",
                "codestar-notifications:UpdateNotificationRule",
                "codestar-notifications:DeleteNotificationRule",
                "codestar-notifications:Subscribe",
                "codestar-notifications:Unsubscribe"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "codestar-notifications:NotificationsForResource":
 "arn:aws:codepipeline:*"
                }
            }
        },
        {
            "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:SetTopicAttributes"
            ],
            "Resource": "arn:aws:sns:*:*:codestar-notifications*"
        },
        {
            "Sid": "CodeStarNotificationsChatbotAccess",
            "Effect": "Allow",
            "Action": [
                "chatbot:DescribeSlackChannelConfigurations",
                "chatbot:ListMicrosoftTeamsChannelConfigurations"
            ],
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

AWS verwaltete Richtlinie: AWSCodePipeline_ReadOnlyAccess

Dies ist eine Richtlinie, die nur Lesezugriff auf gewährt. CodePipeline Informationen zum Anzeigen des JSON-Richtliniendokuments in der IAM-Konsole finden Sie unter. AWSCodePipeline_ReadOnlyAccess

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `codepipeline`— Erteilt Berechtigungen für Aktionen in CodePipeline.
- `codestar-notifications`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, auf Ressourcen in AWS CodeStar Benachrichtigungen zuzugreifen.
- `s3`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Ressourcen in Amazon S3 zu verwalten.
- `sns`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Benachrichtigungsressourcen in Amazon SNS zu verwalten.

```
{
    "Statement": [
        {
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:GetPipelineExecution",
                "codepipeline:ListPipelineExecutions",
                "codepipeline:ListActionExecutions",
                "codepipeline:ListActionTypes",
                "codepipeline:ListPipelines",
                "codepipeline:ListTagsForResource",
                "s3:ListAllMyBuckets",
                "codestar-notifications:ListNotificationRules",
                "codestar-notifications:ListEventTypes",
                "codestar-notifications:ListTargets"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
```

```
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketPolicy"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3::*:codepipeline-*"
        },
        {
            "Sid": "CodeStarNotificationsReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-notifications:DescribeNotificationRule"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "codestar-notifications:NotificationsForResource":
 "arn:aws:codepipeline:*"
                }
            }
        }
    ],
    "Version": "2012-10-17"
}
```

AWS verwaltete Richtlinie: **`AWSCodePipelineApproverAccess`**

Dies ist eine Richtlinie, die die Erlaubnis erteilt, eine manuelle Genehmigungsaktion zu genehmigen oder abzulehnen. Informationen zum Anzeigen des JSON-Richtliniendokuments in der IAM-Konsole finden Sie unter.. AWSCodePipelineApproverAccess

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `codepipeline`— Erteilt Berechtigungen für Aktionen in CodePipeline.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:GetPipelineExecution",
                "codepipeline:ListPipelineExecutions",
                "codepipeline:ListPipelines",
                "codepipeline:PutApprovalResult"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

AWS verwaltete Richtlinie: AWSCodePipelineCustomActionAccess

Dies ist eine Richtlinie, die die Erlaubnis erteilt, benutzerdefinierte Aktionen in Jenkins-Ressourcen für Build CodePipeline - oder Testaktionen zu erstellen oder zu integrieren. Informationen zum JSON-Richtliniendokument in der IAM-Konsole finden Sie unter. AWSCodePipelineCustomActionAccess

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `codepipeline`— Erteilt Berechtigungen für Aktionen in CodePipeline.

```
{
    "Statement": [
        {
            "Action": [
                "codepipeline:AcknowledgeJob",
                "codepipeline:GetJobDetails",
                "codepipeline:PollForJobs",
                "codepipeline:PutJobFailureResult",
                "codepipeline:PutJobSuccessResult"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

CodePipeline verwaltete Richtlinien und Benachrichtigungen

CodePipeline unterstützt Benachrichtigungen, mit denen Benutzer über wichtige Änderungen an Pipelines informiert werden können. Zu den verwalteten Richtlinien CodePipeline gehören auch Richtlinienerklärungen für die Benachrichtigungsfunktion. Weitere Informationen finden Sie unter Was sind Benachrichtigungen?.

Berechtigungen in Zusammenhang mit Benachrichtigungen in verwalteten Vollzugriffsrichtlinien

Diese verwaltete Richtlinie gewährt Berechtigungen für CodePipeline die zugehörigen Dienste CodeCommit CodeBuild, CodeDeploy, und AWS CodeStar Benachrichtigungen. Die Richtlinie gewährt auch Berechtigungen, die Sie für die Arbeit mit anderen Services benötigen, die in Ihre Pipelines integriert sind, wie Amazon S3, Elastic Beanstalk CloudTrail, Amazon EC2 und. AWS CloudFormation Benutzer, für die diese verwaltete Richtlinie gilt, können auch Amazon SNS SNS-Themen für Benachrichtigungen erstellen und verwalten, Benutzer für Themen abonnieren und abbestellen, Themen auflisten, die als Ziele für Benachrichtigungsregeln ausgewählt werden sollen, und für Slack konfigurierte AWS Chatbot Clients auflisten.

Die von `AWSCodePipeline_FullAccess` verwaltete Richtlinie enthält die folgenden Anweisungen, um den vollständigen Zugriff auf Benachrichtigungen zu ermöglichen.

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
```

```
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

Berechtigungen in Zusammenhang mit Benachrichtigungen in schreibgeschützten verwalteten Richtlinien

Die verwaltete Richtlinie `AWSCodePipeline_ReadOnlyAccess` enthält die folgenden Anweisungen, um schreibgeschützten Zugriff auf Benachrichtigungen zu ermöglichen. Benutzer, auf die diese Richtlinie angewendet wird, können Benachrichtigungen für Ressourcen anzeigen, diese jedoch nicht erstellen, verwalten oder abonnieren.

```
  {
       "Sid": "CodeStarNotificationsPowerUserAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:DescribeNotificationRule"
       ],
       "Resource": "*",
       "Condition" : {
           "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"}
       }
   },
   {
       "Sid": "CodeStarNotificationsListAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:ListNotificationRules",
           "codestar-notifications:ListEventTypes",
           "codestar-notifications:ListTargets"
       ],
       "Resource": "*"
   }
```

Weitere Informationen zu IAM und Benachrichtigungen finden Sie unter Identity and Access Management für AWS CodeStar Benachrichtigungen.

AWS CodePipeline Aktualisierungen der AWS verwalteten Richtlinien

Hier finden Sie Informationen zu Aktualisierungen AWS verwalteter Richtlinien, die CodePipeline seit Beginn der Nachverfolgung dieser Änderungen durch diesen Dienst vorgenommen wurden. Abonnieren Sie den RSS-Feed auf der Seite CodePipeline Dokumentenverlauf, um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten.

Änderung	Beschreibung	Datum
AWSCodePipeline_FullAccess— Aktualisierungen der bestehenden Richtlinie	CodePipeline Dieser Richtlinie wurde die Erlaubnis hinzugefügt, `ListStacks` in zu unterstützen AWS CloudFormation.	15. März 2024
AWSCodePipeline_FullAccess— Aktualisierungen der bestehenden Richtlinie	Diese Richtlinie wurde aktualisiert, um Berechtigungen für hinzuzufügen AWS Chatbot. Weitere Informationen finden Sie unter CodePipeline verwaltete Richtlinien und Benachrichtigungen.	21. Juni 2023
AWSCodePipeline_FullAccessund AWSCodePipeline_ReadOnlyAccessverwaltete Richtlinien — Aktualisierungen vorhandener Richtlinien	CodePipeline diesen Richtlinien wurde eine Berechtigung hinzugefügt, um mithilfe AWS Chatbot von, einen zusätzlichen Benachrichtigungstyp zu unterstützen`chatbot:ListMicrosoftTeamsChannelConfigurations` .	16. Mai 2023
AWSCodePipelineFullAccess— Veraltet	Diese Richtlinie wurde ersetzt durch `AWSCodePipeline_FullAccess` . Nach dem 17. November 2022 kann diese Richtlinie keinen neuen Benutzern, Gruppen oder Rollen zugewiesen werden. Weitere Informationen finden Sie unter AWS	17. November 2022

Änderung	Beschreibung	Datum
	verwaltete Richtlinien für AWS CodePipeline.	
AWSCodePipelineReadOnlyAccess— Veraltet	Diese Richtlinie wurde ersetzt durch `AWSCodePipeline_ReadOnlyAccess` . Nach dem 17. November 2022 kann diese Richtlinie keinen neuen Benutzern, Gruppen oder Rollen zugewiesen werden. Weitere Informationen finden Sie unter AWS verwaltete Richtlinien für AWS CodePipeline.	17. November 2022
CodePipeline hat begonnen, Änderungen zu verfolgen	CodePipeline hat begonnen, Änderungen für die AWS verwalteten Richtlinien zu verfolgen.	12. März 2021

Beispiele für vom Kunden verwaltete Richtlinien

In diesem Abschnitt finden Sie Beispielbenutzerrichtlinien, die Berechtigungen für verschiedene CodePipeline Aktionen gewähren. Diese Richtlinien funktionieren, wenn Sie die CodePipeline API, AWS SDKs oder die AWS CLI verwenden. Bei Verwendung der Konsole müssen Sie zusätzliche konsolenspezifische Berechtigungen erteilen. Weitere Informationen finden Sie unter Erforderliche Berechtigungen für die Verwendung der CodePipeline -Konsole.

> **Note**
>
> In allen Beispielen werden die Region USA West (Oregon) (`us-west-2`) und fiktive Konto-IDs verwendet.

Beispiele

- Beispiel 1: Gewähren von Berechtigungen zum Abrufen des Status einer Pipeline
- Beispiel 2: Gewähren von Berechtigungen zum Aktivieren und Deaktivieren von Übergängen zwischen Stufen
- Beispiel 3: Gewähren von Berechtigungen zum Abrufen einer Liste mit allen verfügbaren Aktionstypen
- Beispiel 4: Gewähren von Berechtigungen für das Erlauben oder Ablehnen von manuellen Genehmigungsaktionen
- Beispiel 5: Gewähren von Berechtigungen zum Abfragen von Aufträgen für eine benutzerdefinierte Aktion
- Beispiel 6: Eine Richtlinie für die Jenkins-Integration mit anhängen oder bearbeiten AWS CodePipeline
- Beispiel 7: Konfigurieren des kontoübergreifenden Zugriffs auf eine Pipeline
- Beispiel 8: Verwenden von AWS -Ressourcen, die mit einem anderen Konto in einer Pipeline verknüpft sind

Beispiel 1: Gewähren von Berechtigungen zum Abrufen des Status einer Pipeline

Im folgenden Beispiel werden Berechtigungen vergeben, um den Status der Pipeline namens `MyFirstPipeline` abzurufen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipelineState"
            ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"
        }
    ]
}
```

Beispiel 2: Gewähren von Berechtigungen zum Aktivieren und Deaktivieren von Übergängen zwischen Stufen

Im folgenden Beispiel werden Berechtigungen zum Deaktivieren und Aktivieren von Übergängen zwischen allen Stufen in der Pipeline namens `MyFirstPipeline` vergeben:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:DisableStageTransition",
                "codepipeline:EnableStageTransition"
            ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/*"
        }
    ]
}
```

Um dem Benutzer das Deaktivieren und Aktivieren von Übergängen für eine einzelne Stufe in einer Pipeline zu erlauben, müssen Sie die Stufe festlegen. So können Sie z. B. dem Benutzer das Aktivieren und Deaktivieren von Übergängen für eine Stufe mit der Bezeichnung `Staging` in einer Pipeline namens `MyFirstPipeline` erlauben:

```
"Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/Staging"
```

Beispiel 3: Gewähren von Berechtigungen zum Abrufen einer Liste mit allen verfügbaren Aktionstypen

Im folgenden Beispiel werden Berechtigungen zum Abrufen einer Liste mit allen Aktionstypen vergeben, die für Pipelines in der Region `us-west-2` verfügbar sind:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:ListActionTypes"
            ],
```

```
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:actiontype:*"
        }
    ]
}
```

Beispiel 4: Gewähren von Berechtigungen für das Erlauben oder Ablehnen von manuellen Genehmigungsaktionen

Im folgenden Beispiel werden Berechtigungen vergeben, um manuelle Genehmigungsaktionen in einer Stufe mit der Bezeichnung `Staging` in einer Pipeline namens `MyFirstPipeline` zu erlauben oder abzulehnen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PutApprovalResult"
            ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/
Staging/*"
        }
    ]
}
```

Beispiel 5: Gewähren von Berechtigungen zum Abfragen von Aufträgen für eine benutzerdefinierte Aktion

Im folgenden Beispiel werden Berechtigungen für das Pipeline-übergreifende Abfragen von Aufträgen für die benutzerdefinierte Aktion mit der Bezeichnung `TestProvider` vergeben, die ein `Test`-Aktionstyp in der ersten Version ist:

> **Note**
>
> Der Job Worker für eine benutzerdefinierte Aktion ist möglicherweise unter einem anderen AWS Konto konfiguriert oder erfordert eine bestimmte IAM-Rolle, um zu funktionieren.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PollForJobs"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111222333444:actionType:Custom/Test/TestProvider/1"
            ]
        }
    ]
}
```

Beispiel 6: Eine Richtlinie für die Jenkins-Integration mit anhängen oder bearbeiten AWS CodePipeline

Wenn Sie eine Pipeline so konfigurieren, dass sie Jenkins für Build oder Test verwendet, erstellen Sie eine separate Identität für diese Integration und fügen Sie eine IAM-Richtlinie hinzu, die über die Mindestberechtigungen verfügt, die für die Integration zwischen Jenkins und erforderlich sind. CodePipeline Die Richtlinie ist die gleiche wie die verwaltete Richtlinie `AWSCodePipelineCustomActionAccess`. Das folgende Beispiel zeigt eine Richtlinie für die Jenkins-Integration:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:AcknowledgeJob",
                "codepipeline:GetJobDetails",
                "codepipeline:PollForJobs",
                "codepipeline:PutJobFailureResult",
                "codepipeline:PutJobSuccessResult"
            ],
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

Beispiel 7: Konfigurieren des kontoübergreifenden Zugriffs auf eine Pipeline

Sie können den Pipeline-Zugriff für Benutzer und Gruppen in einem anderen AWS -Konto konfigurieren. Die empfohlene Methode besteht darin, eine Rolle in dem Konto zu erstellen, in dem die Pipeline erstellt wurde. Die Rolle sollte es Benutzern des anderen AWS Kontos ermöglichen, diese Rolle anzunehmen und auf die Pipeline zuzugreifen. Weitere Informationen finden Sie unter Anleitung: Kontenübergreifender Zugriff mithilfe von Rollen.

Das folgende Beispiel zeigt eine Richtlinie im 80398EXAMPLE-Konto, die es Benutzern ermöglicht, die `MyFirstPipeline` in der Konsole angegebene Pipeline anzuzeigen, aber nicht zu ändern. CodePipeline Diese Richtlinie basiert auf der verwalteten Richtlinie `AWSCodePipeline_ReadOnlyAccess`, aber da sie spezifisch für die Pipeline `MyFirstPipeline` gilt, kann sie die verwaltete Richtlinie nicht direkt verwenden. Wenn Sie die Richtlinie nicht auf eine bestimmte Pipeline beschränken möchten, sollten Sie in Erwägung ziehen, eine der verwalteten Richtlinien zu verwenden, die von erstellt und verwaltet wurden. CodePipeline Weitere Informationen finden Sie unter Arbeiten mit verwalteten Richtlinien. Sie müssen diese Richtlinie einer IAM-Rolle zuordnen, die Sie für den Zugriff erstellen, z. B. einer Rolle mit dem Namen`CrossAccountPipelineViewers`:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:ListActionTypes",
                "codepipeline:ListPipelines",
                "iam:ListRoles",
                "s3:GetBucketPolicy",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "codedeploy:GetApplication",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "elasticbeanstalk:DescribeApplications",
                "elasticbeanstalk:DescribeEnvironments",
                "lambda:GetFunctionConfiguration",
                "lambda:ListFunctions"
```

```
            ],
            "Resource": "arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline"
        }
    ],
    "Version": "2012-10-17"
}
```

Nachdem Sie diese Richtlinie erstellt haben, erstellen Sie die IAM-Rolle im 80398EXAMPLE-Konto und fügen Sie die Richtlinie dieser Rolle hinzu. Zu den Vertrauensstellungen der Rolle müssen Sie das AWS Konto hinzufügen, das diese Rolle übernimmt. Das folgende Beispiel zeigt eine Richtlinie, die es Benutzern des Kontos *111111111 ermöglicht, Rollen anzunehmen, die im AWS Konto 80398EXAMPLE* definiert sind:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Das folgende Beispiel zeigt eine im Konto *111111111111* erstellte Richtlinie, die es Benutzern ermöglicht, die im AWS Konto 80398EXAMPLE angegebene Rolle anzunehmen: `CrossAccountPipelineViewers`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::80398EXAMPLE:role/CrossAccountPipelineViewers"
        }
    ]
}
```

Beispiel 8: Verwenden von AWS -Ressourcen, die mit einem anderen Konto in einer Pipeline verknüpft sind

Sie können Richtlinien konfigurieren, die es einem Benutzer ermöglichen, eine Pipeline zu erstellen, die Ressourcen in einem anderen Konto verwendet. AWS Dies erfordert die Konfiguration von Richtlinien und Rollen sowohl im Konto, das die Pipeline erstellen wird (AccountA), als auch im Konto, das die Ressourcen erstellt hat, die in der Pipeline verwendet werden sollen (AccountB). Sie müssen auch ein vom Kunden verwaltetes Key-In erstellen AWS Key Management Service , das Sie für den kontoübergreifenden Zugriff verwenden können. Weitere Informationen und step-by-step Beispiele finden Sie unter Erstellen Sie eine Pipeline CodePipeline , in der Ressourcen von einem anderen AWS Konto verwendet werden undKonfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte für CodePipeline.

Das folgende Beispiel zeigt eine Richtlinie, die von AccountA für einen S3-Bucket konfiguriert wurde, der zum Speichern von Pipeline-Artefakten verwendet wird. Die Richtlinie gewährt Zugriff auf AccountB. Im folgenden Beispiel lautet der ARN für AccountB `012ID_ACCOUNT_B`. Der ARN für den S3-Bucket lautet `codepipeline-us-east-2-1234567890`. Ersetzen Sie diese ARNs durch die ARNs für den S3-Bucket und das Konto, für die Sie Zugriff erlauben möchten:

```
{
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
```

```
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": false
                }
            }
        },
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
            },
            "Action": [
                "s3:Get*",
                "s3:Put*"
            ],
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
        },
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890"
        }
    ]
}
```

Das folgende Beispiel zeigt eine von AccountA konfigurierte Richtlinie, die von AccountB ermöglicht, eine Rolle anzunehmen. Diese Richtlinie muss auf die Servicerolle für CodePipeline (`CodePipeline_Service_Role`) angewendet werden. Weitere Informationen zum Anwenden von Richtlinien auf Rollen in IAM finden Sie unter Rolle ändern. Im folgenden Beispiel ist `012ID_ACCOUNT_B` der ARN für AccountB:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": [
            "arn:aws:iam::012ID_ACCOUNT_B:role/*"
```

```
        ]
    }
}
```

Das folgende Beispiel zeigt eine Richtlinie, die von AccountB konfiguriert und auf die EC2-Instance-Rolle für angewendet wurde. CodeDeploy Diese Richtlinie gewährt Zugriff auf den S3-Bucket, der von AccountA zum Speichern von Pipeline-Artefakten verwendet wird (*codepipeline-us-east-2-1234567890*):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*"
            ],
            "Resource": [
                "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::codepipeline-us-east-2-1234567890"
            ]
        }
    ]
}
```

Das folgende Beispiel zeigt eine Richtlinie dafür, AWS KMS wo sich der ARN des vom Kunden verwalteten Schlüssels ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** befindet, der in AccountA erstellt und so konfiguriert ist, dass AccountB ihn verwenden kann:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt",
                "kms:ReEncrypt*",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:us-
east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE"
            ]
        }
    ]
}
```

Das folgende Beispiel zeigt eine Inline-Richtlinie für eine von AccountB erstellte IAM-Rolle (`CrossAccount_Role`), die den Zugriff auf CodeDeploy Aktionen ermöglicht, die für die Pipeline in AccountA erforderlich sind.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:CreateDeployment",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision"
            ],
            "Resource": "*"
        }
    ]
}
```

Das folgende Beispiel zeigt eine Inline-Richtlinie für eine von AccountB erstellte IAM-Rolle (`CrossAccount_Role`), die den Zugriff auf den S3-Bucket ermöglicht, um Eingabeartefakte herunterzuladen und Ausgabeartefakte hochzuladen:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
            ]
        }
    ]
}
```

Weitere Informationen zum Bearbeiten einer Pipeline für kontenübergreifenden Zugriff auf Ressourcen finden Sie unter Schritt 2: Bearbeiten der Pipeline .

Beispiele für eine ressourcenbasierte AWS CodePipeline -Richtlinie

Andere Services, z. B. Amazon-S3, unterstützen auch ressourcenbasierte Berechtigungsrichtlinien. Beispielsweise können Sie einem S3 Bucket eine Richtlinie zuweisen, um die Zugriffsberechtigungen für diesen Bucket zu verwalten. Sie unterstützt zwar CodePipeline keine ressourcenbasierten Richtlinien, speichert aber Artefakte, die in Pipelines verwendet werden sollen, in versionierten S3-Buckets.

Example Um eine Richtlinie für einen S3-Bucket zu erstellen, der als Artefaktspeicher verwendet werden soll CodePipeline

Sie können jeden versionierten S3-Bucket als Artefaktspeicher für verwenden. CodePipeline Wenn Sie den Assistenten Create Pipeline (Pipeline erstellen) zum Erstellen Ihrer ersten Pipeline verwenden, wird dieser S3-Bucket für Sie erstellt, um sicherzustellen, dass alle Objekte, die in den Artefaktspeicher hochgeladen werden, verschlüsselt und Verbindungen zum Bucket sicher sind. Beim Erstellen Ihres eigenen S3-Bucket ist es eine bewährte Methode, die folgende Richtlinie oder ihre Elemente dem Bucket hinzuzufügen. In dieser Richtlinie ist der ARN für den S3-Bucket `codepipeline-us-east-2-1234567890`. Ersetzen Sie diesen ARN durch den ARN für Ihren S3-Bucket:

```
{
```

```
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": false
                }
            }
        }
    ]
}
```

Fehlerbehebung für AWS CodePipeline -Identität und -Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit CodePipeline und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion durchzuführen in CodePipeline
- Ich bin nicht berechtigt, IAM auszuführen: PassRole
- Ich bin Administrator und möchte anderen Zugriff gewähren CodePipeline
- Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine CodePipeline Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion durchzuführen in CodePipeline

Wenn Ihnen AWS Management Console mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion durchzuführen, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator ist die Person, die Ihnen Ihren Benutzernamen und Ihr Passwort bereitgestellt hat.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` IAM-Benutzer versucht, die Konsole zu verwenden, um Details zu einer Pipeline anzuzeigen, aber nicht über die `codepipeline:GetPipeline` entsprechenden Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codepipeline:GetPipeline on resource: my-pipeline
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion `my-pipeline` auf die Ressource `codepipeline:GetPipeline` zugreifen zu können.

Ich bin nicht berechtigt, IAM auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Ausführung der Aktion `iam:PassRole` autorisiert sind, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator ist die Person, die Ihnen Ihren Benutzernamen und Ihr Passwort bereitgestellt hat. Bitten Sie diese Person, Ihre Richtlinien zu aktualisieren, damit Sie eine Rolle an sie weitergeben können. CodePipeline

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstbezogene Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in auszuführen. CodePipeline Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Service-Rolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall bittet Mary ihren Administrator um die Aktualisierung ihrer Richtlinien, um die Aktion `iam:PassRole` ausführen zu können.

Ich bin Administrator und möchte anderen Zugriff gewähren CodePipeline

Um anderen den Zugriff zu ermöglichen CodePipeline, müssen Sie eine IAM-Entität (Benutzer oder Rolle) für die Person oder Anwendung erstellen, die Zugriff benötigt. Sie werden die Anmeldeinformationen für diese Einrichtung verwenden, um auf AWS zuzugreifen. Anschließend müssen Sie der Entität eine Richtlinie hinzufügen, die ihnen die richtigen Berechtigungen gewährt. CodePipeline

Informationen zum Einstieg finden Sie unter Erstellen Ihrer ersten delegierten IAM-Benutzer und -Gruppen im IAM-Benutzerhandbuch.

Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine CodePipeline Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob diese Funktionen CodePipeline unterstützt werden, finden Sie unterWie AWS CodePipeline funktioniert mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

CodePipeline Referenz zu Berechtigungen

Verwenden Sie die folgende Tabelle als Referenz, wenn Sie die Zugriffskontrolle einrichten und Berechtigungsrichtlinien schreiben, die Sie einer IAM-Identität zuordnen können (identitätsbasierte Richtlinien). In der Tabelle sind die einzelnen CodePipeline API-Operationen und die entsprechenden Aktionen aufgeführt, für die Sie Berechtigungen zur Ausführung der Aktion erteilen können. Bei Vorgängen, die Berechtigungen auf Ressourcenebene unterstützen, ist in der Tabelle die AWS Ressource aufgeführt, für die Sie die Berechtigungen erteilen können. Sie geben die Aktionen im Feld `Action` der Richtlinie an.

Mit Berechtigungen auf Ressourcenebene können Sie angeben, für welche Ressourcen Benutzer Aktionen ausführen dürfen. AWS CodePipeline bietet teilweise Unterstützung für Berechtigungen auf Ressourcenebene. Das bedeutet, dass Sie bei einigen AWS CodePipeline API-Aufrufen steuern können, wann Benutzer diese Aktionen verwenden dürfen, je nachdem, welche Bedingungen erfüllt sein müssen, oder welche Ressourcen Benutzer verwenden dürfen. Beispielsweise können Sie Benutzern die Berechtigung erteilen, Ausführungsinformationen für die Pipeline aufzulisten, dies aber nur für eine oder mehrere bestimmte Pipeline bzw. Pipelines.

> **Note**
>
> In der Spalte Ressourcen werden die Ressourcen aufgeführt, die für API-Aufrufe erforderlich sind, die Berechtigungen auf Ressourcenebene unterstützen. Bei API-Aufrufen, die keine Berechtigungen auf Ressourcenebene unterstützen, können Sie den Benutzern die Berechtigung zu ihrer Verwendung erteilen, müssen aber für das Ressourcenelement in der Richtlinienanweisung einen Platzhalter (*) einfügen.

CodePipeline API-Operationen und erforderliche Berechtigungen für Aktionen

AcknowledgeJob

Aktion: `codepipeline:AcknowledgeJob`

Erforderlich, um Informationen zu einem bestimmten Auftrag anzuzeigen und zu sehen, ob dieser vom Auftragsworker erhalten wurde. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

AcknowledgeThirdPartyJob

Aktion: `codepipeline:AcknowledgeThirdPartyJob`

Erforderlich, um zu bestätigen, dass ein Auftragsworker den angegebenen Auftrag erhalten hat. Wird nur für Partneraktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

CreateCustomActionType

Aktion: `codepipeline:CreateCustomActionType`

Erforderlich, um eine neue benutzerdefinierte Aktion zu erstellen, die in allen mit dem AWS Konto verknüpften Pipelines verwendet werden kann. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:`*region*`:`*account*`:actiontype:`*owner*`/`*category*`/`*provider*`/`*versio*

CreatePipeline

Aktion: `codepipeline:CreatePipeline`

Erforderlich zum Erstellen einer Pipeline

Ressourcen:

Pipeline

`arn:aws:codepipeline:`*region*`:`*account*`:`*pipeline-name*

DeleteCustomActionType

Aktion: `codepipeline:DeleteCustomActionType`

Erforderlich, um eine benutzerdefinierte Aktion als gelöscht zu markieren. `PollForJobs` für die benutzerdefinierte Aktion schlägt fehl, nachdem die Aktion zum Löschen markiert wurde. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

DeletePipeline

Aktion: `codepipeline:DeletePipeline`

Erforderlich zum Löschen einer Pipeline

Ressourcen:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

DeleteWebhook

Aktion: `codepipeline:DeleteWebhook`

Erforderlich zum Löschen eines Webhooks.

Ressourcen:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DeregisterWebhookWithThirdParty

Aktion: `codepipeline:DeregisterWebhookWithThirdParty`

Bevor ein Webhook gelöscht wird, muss die Verbindung zwischen dem Webhook, der von erstellt wurde, CodePipeline und dem externen Tool, dessen Ereignisse erkannt werden sollen, entfernt werden. Wird derzeit nur für Webhooks unterstützt, die auf einen Aktionstyp von abzielen. GitHub

Ressourcen:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DisableStageTransition

Aktion: `codepipeline:DisableStageTransition`

Erforderlich, um zu verhindern, dass Artefakte in einer Pipeline zur nächsten Stufe übergehen

Ressourcen:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

EnableStageTransition

Aktion: `codepipeline:EnableStageTransition`

Erforderlich, um Artefakten in einer Pipeline zu ermöglichen, zur einer Stufe in einer Pipeline überzugehen

Ressourcen:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

GetJobDetails

Aktion: `codepipeline:GetJobDetails`

Erforderlich zum Abrufen von Informationen zu einem Auftrag. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen: Keine Ressource erforderlich.

GetPipeline

Aktion: `codepipeline:GetPipeline`

Erforderlich zum Abrufen von Struktur, Phasen, Aktionen und Metadaten einer Pipeline einschließlich des Pipeline-ARN.

Ressourcen:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

GetPipelineExecution

Aktion: `codepipeline:GetPipelineExecution`

Erforderlich zum Abrufen von Informationen zu einer Ausführung einer Pipeline, einschließlich Details zu Artefakten, Pipeline-Ausführungs-ID und Name, Version und Status der Pipeline

Ressourcen:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

GetPipelineState

Aktion: `codepipeline:GetPipelineState`

Erforderlich zum Abrufen von Informationen zum Zustand einer Pipeline, einschließlich Stufen und Aktionen

Ressourcen:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

GetThirdPartyJobDetails

Aktion: `codepipeline:GetThirdPartyJobDetails`

Erforderlich zum Abfragen der Details eines Auftrags für eine Drittanbieter-Aktion. Wird nur für Partneraktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

ListActionTypes

Aktion: `codepipeline:ListActionTypes`

Erforderlich, um eine Zusammenfassung aller mit Ihrem Konto verknüpften CodePipeline Aktionstypen zu generieren.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

ListPipelineExecutions

Aktion: `codepipeline:ListPipelineExecutions`

Erforderlich, um eine Zusammenfassung der neuesten Ausführungen für eine Pipeline zu generieren.

Ressourcen:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

ListPipelines

Aktion: `codepipeline:ListPipelines`

Erforderlich, um eine Übersicht aller Pipelines zu generieren, die mit Ihrem Konto verknüpft sind.

Ressourcen:

Pipeline-ARN mit Platzhalter (Berechtigungen auf Ressourcenebene auf der Ebene des Pipeline-Namens werden nicht unterstützt)

`arn:aws:codepipeline:region:account:*`

ListTagsForResource

Aktion: `codepipeline:ListTagsForResource`

Erforderlich, um Tags für eine angegebene Ressource aufzulisten.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

ListWebhooks

Aktion: `codepipeline:ListWebhooks`

Erforderlich zum Auflisten aller Webhooks im Konto für diese Region.

Ressourcen:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

PollForJobs

Aktion(en): `codepipeline:PollForJobs`

Erforderlich, um Informationen über Jobs abzurufen, auf die reagiert werden CodePipeline kann.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

PollForThirdPartyJobs

Aktion: `codepipeline:PollForThirdPartyJobs`

Erforderlich, um zu bestimmen, ob es Drittanbieter-Aufträge gibt, für die ein Auftragsworker agiert. Wird nur für Partneraktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

PutActionRevision

Aktion: `codepipeline:PutActionRevision`

Erforderlich, um Informationen CodePipeline über neue Versionen an eine Quelle zu melden.

Ressourcen:

Aktion

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

PutApprovalResult

Aktion: `codepipeline:PutApprovalResult`

Erforderlich, um die Antwort auf eine manuelle Genehmigungsanfrage an zu CodePipeline melden. Gültige Antworten sind `Approved` und `Rejected`.

Ressourcen:

Aktion

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

> **Note**
>
> Dieser API-Aufruf unterstützt Berechtigungen auf Ressourcenebene. Es kann jedoch ein Fehler auftreten, wenn Sie die IAM-Konsole oder den Policy Generator nutzen, um Richtlinien mit `"codepipeline:PutApprovalResult"` zu erstellen, die einen Ressourcen-ARN angeben. Wenn ein Fehler auftritt, können Sie auf der JSON-Registerkarte in der IAM-Konsole oder mit der CLI eine Richtlinie erstellen.

PutJobFailureResult

Aktion: `codepipeline:PutJobFailureResult`

Erforderlich zum Melden des Scheiterns eines Auftrags, wie von einem Auftragsworker an die Pipeline zurückgegeben. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

PutJobSuccessResult

Aktion: `codepipeline:PutJobSuccessResult`

Erforderlich zum Melden des Erfolgs eines Auftrags, wie von einem Auftragsworker an die Pipeline zurückgegeben. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

PutThirdPartyJobFailureResult

Aktion: `codepipeline:PutThirdPartyJobFailureResult`

Erforderlich zum Melden des Scheiterns eines Drittanbieter-Auftrags, wie von einem Auftragsworker an die Pipeline zurückgegeben. Wird nur für Partneraktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

PutThirdPartyJobSuccessResult

Aktion: `codepipeline:PutThirdPartyJobSuccessResult`

Erforderlich zum Melden des Erfolgs eines Drittanbieter-Auftrags, wie von einem Auftragsworker an die Pipeline zurückgegeben. Wird nur für Partneraktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

PutWebhook

Aktion: `codepipeline:PutWebhook`

Erforderlich zum Erstellen eines Webhooks.

Ressourcen:

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RegisterWebhookWithThirdParty

Aktion: `codepipeline:RegisterWebhookWithThirdParty`

Ressourcen:

Nachdem ein Webhook erstellt wurde, müssen die unterstützten Drittanbieter konfiguriert werden, um die generierte Webhook-URL aufzurufen.

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RetryStageExecution

Aktion: `codepipeline:RetryStageExecution`

Erforderlich zum Wiederaufnehmen der Pipeline-Ausführung durch Wiederholen der letzten fehlgeschlagenen Aktionen in einer Stufe.

Ressourcen:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

StartPipelineExecution

Aktion: `codepipeline:StartPipelineExecution`

Erforderlich, um die angegebene Pipeline zu starten (insbesondere, um mit der Verarbeitung des letzten Commits an den Quellspeicherort zu beginnen, der als Teil der Pipeline angegeben wurde).

Ressourcen:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

TagResource

Aktion: `codepipeline:TagResource`

Erforderlich, um die angegebene Ressource zu markieren.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UntagResource

Aktion: `codepipeline:UntagResource`

Erforderlich, um die angegebene Ressource zu markieren.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UpdatePipeline

Aktion: `codepipeline:UpdatePipeline`

Erforderlich zum Aktualisieren einer angegebenen Pipeline mit Bearbeitungen oder Änderungen an seiner Struktur.

Ressourcen:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Die CodePipeline Servicerolle verwalten

Die CodePipeline Servicerolle ist mit einer oder mehreren Richtlinien konfiguriert, die den Zugriff auf die von der Pipeline verwendeten AWS Ressourcen steuern. Möglicherweise möchten Sie dieser Rolle weitere Richtlinien zuordnen, die der Rolle zugeordnete Richtlinie bearbeiten oder Richtlinien für andere Servicerollen in konfigurieren AWS. Sie können auch einer Rolle eine Richtlinie anfügen, wenn Sie den kontenübergreifenden Zugriff auf Ihre Pipeline konfigurieren.

Important

Das Modifizieren einer Richtlinienanweisung oder Anfügen einer weiteren Richtlinie zur Rolle kann Pipelines funktionsunfähig machen. Stellen Sie sicher, dass Sie die Auswirkungen verstehen, bevor Sie die Servicerolle für CodePipeline in irgendeiner Weise ändern. Sie sollten Ihre Pipelines nach Durchführung von Änderungen an der Servicerolle unbedingt testen.

Note

In der Konsole werden Servicerollen, die vor September 2018 erstellt wurden, mit dem Namen `oneClick_AWS-CodePipeline-Service_ID-Number` erstellt.

Servicerollen, die nach September 2018 erstellt wurden, verwenden das Namensformat der Servicerolle `AWSCodePipelineServiceRole-`*`Region`*`-`*`Pipeline_Name`*. Bei einer Pipeline mit dem Namen `MyFirstPipeline` in `eu-west-2` benennt die Konsole beispielsweise die Rolle und die Richtlinie`AWSCodePipelineServiceRole-eu-west-2-MyFirstPipeline`.

Entfernen von Berechtigungen aus der CodePipeline-Servicerolle

Sie können die Servicerollenanweisung entsprechend bearbeiten, um den Zugriff auf ungenutzte Ressourcen zu entfernen. Wenn beispielsweise keine Ihrer Pipelines Elastic Beanstalk enthält, können Sie die Richtlinienerklärung bearbeiten, um den Abschnitt zu entfernen, der Zugriff auf Elastic Beanstalk Beanstalk-Ressourcen gewährt.

Ebenso können Sie, wenn keine Ihrer Pipelines Folgendes beinhaltet CodeDeploy, die Richtlinienerklärung bearbeiten, um den Abschnitt zu entfernen, der Zugriff auf Ressourcen gewährt: CodeDeploy

```
    {
    "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetApplicationRevision",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
```

Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle

Bevor Sie sie in Ihren Pipelines verwenden können, müssen Sie Ihre Diensterollen-Richtlinienerklärung mit den Berechtigungen für eine Richtlinie aktualisieren, die noch AWS-Service nicht in der Standardeinstellung enthalten ist.

Dies ist besonders wichtig, wenn die Servicerolle, die Sie für Ihre Pipelines verwenden, erstellt wurde, bevor der Support CodePipeline für eine hinzugefügt wurde. AWS-Service

Die folgende Tabelle zeigt, wann Unterstützung für andere AWS-Services hinzugefügt wurde.

AWS-Service	CodePipeline Datum der Unterstützung
AWS CloudFormation StackSets Aktionen	30. Dezember 2020
CodeCommit vollständiges Klon-Ausgabeartefaktformat	11. November 2020
CodeBuild Batch-Builds	30. Juli 2020
AWS AppConfig	22. Juni 2020
AWS Step Functions	27. Mai 2020
AWS CodeStar Verbindungen	18. Dezember 2019
`CodeDeployToECS` -Aktion	27. November 2018
Amazon ECR	27. November 2018
Servicekatalog	16. Oktober 2018
AWS Device Farm	19. Juli 2018
Amazon ECS	12. Dezember 2017/Update für die Anmeldung zur Autorisierung von Tagging am 21. Juli 2017
CodeCommit	18. April 2016
AWS OpsWorks	2. Juni 2016
AWS CloudFormation	3. November 2016
AWS CodeBuild	1. Dezember 2016
Elastic Beanstalk	Erster Start des Dienstes

Gehen Sie folgendermaßen vor, um Berechtigungen für einen unterstützten Service hinzuzufügen:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.

2. Wählen Sie in der IAM-Konsole im Navigationsbereich Rollen und dann Ihre `AWS-CodePipeline-Service` Rolle aus der Rollenliste aus.
3. Wählen Sie auf der Registerkarte Berechtigungen unter Inline-Richtlinien in der Zeile für Ihre Servicerollenrichtlinie die Option Richtlinie bearbeiten aus.
4. Fügen Sie die erforderlichen Berechtigungen im Feld Richtliniendokument hinzu.

> **Note**
>
> Beachten Sie bei der Erstellung von IAM-Richtlinien die standardmäßigen Sicherheitsempfehlungen zur Gewährung von geringsten Rechten, d. h. gewähren Sie nur die für die Ausführung einer Aufgabe erforderlichen Berechtigungen. Einige API-Aufrufe unterstützen ressourcenbasierte Berechtigungen und lassen Zugriffseinschränkungen zu. In diesem Fall können Sie beispielsweise das Platzhalterzeichen (*) durch einen Ressourcen-ARN oder durch einen Ressourcen-ARN, der einen Platzhalter (*) enthält, ersetzen, um die Berechtigungen beim Aufrufen von `DescribeTasks` und `ListTasks` einzuschränken. Weitere Informationen zum Erstellen einer Richtlinie, die den Zugriff mit den geringsten Rechten gewährt, finden Sie unter. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

Wenn Sie CodeCommit Unterstützung benötigen, fügen Sie Ihrer Richtlinienerklärung beispielsweise Folgendes hinzu:

```
{
  "Effect": "Allow",
  "Action": [
      "codecommit:GetBranch",
      "codecommit:GetCommit",
      "codecommit:UploadArchive",
      "codecommit:GetUploadArchiveStatus",
      "codecommit:CancelUploadArchive"
           ],
  "Resource": "resource_ARN"
},
```

Wenn Sie AWS OpsWorks Unterstützung benötigen, fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "opsworks:CreateDeployment",
        "opsworks:DescribeApps",
        "opsworks:DescribeCommands",
        "opsworks:DescribeDeployments",
        "opsworks:DescribeInstances",
        "opsworks:DescribeStacks",
        "opsworks:UpdateApp",
        "opsworks:UpdateStack"
    ],
    "Resource": "resource_ARN"
},
```

Wenn Sie AWS CloudFormation Unterstützung benötigen, fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStacks",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "iam:PassRole"
    ],
    "Resource": "resource_ARN"
},
```

Beachten Sie, dass die `cloudformation:DescribeStackEvents` Genehmigung optional ist. Dadurch kann die AWS CloudFormation Aktion eine detailliertere Fehlermeldung anzeigen. Diese Berechtigung kann der IAM-Rolle entzogen werden, wenn Sie nicht möchten, dass

Ressourcendetails in den Pipeline-Fehlermeldungen auftauchen. Weitere Informationen finden Sie unter AWS CloudFormation.

Wenn Sie CodeBuild Unterstützung benötigen, fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Die Support für Batch-Builds wurde zu einem späteren Zeitpunkt hinzugefügt. Informationen zu den Berechtigungen zum Hinzufügen der Servicerolle für Batch-Builds finden Sie in Schritt 11.

Wenn Sie AWS Device Farm Unterstützung benötigen, fügen Sie Ihrer Richtlinienerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "devicefarm:ListProjects",
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
    ],
    "Resource": "resource_ARN"
},
```

Wenn Sie Service Catalog-Support benötigen, fügen Sie Ihrer Richtlinienerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "servicecatalog:ListProvisioningArtifacts",
        "servicecatalog:CreateProvisioningArtifact",
        "servicecatalog:DescribeProvisioningArtifact",
        "servicecatalog:DeleteProvisioningArtifact",
        "servicecatalog:UpdateProduct"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:ValidateTemplate"
    ],
    "Resource": "resource_ARN"
}
```

5. Für Amazon ECR-Support fügen Sie Ihrer Richtlinienerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "ecr:DescribeImages"
    ],
    "Resource": "resource_ARN"
},
```

6. Für Amazon ECS sind die folgenden Mindestberechtigungen aufgeführt, die zum Erstellen von Pipelines mit einer Amazon ECS-Bereitstellungsaktion erforderlich sind.

```
{
    "Effect": "Allow",
    "Action": [
        "ecs:DescribeServices",
        "ecs:DescribeTaskDefinition",
        "ecs:DescribeTasks",
        "ecs:ListTasks",
        "ecs:RegisterTaskDefinition",
        "ecs:TagResource",
        "ecs:UpdateService"
```

```
    ],
    "Resource": "resource_ARN"
},
```

Sie können sich für die Verwendung der Tagging-Autorisierung in Amazon ECS entscheiden. Wenn Sie sich anmelden, müssen Sie die folgenden Berechtigungen erteilen:. `ecs:TagResource` Weitere Informationen darüber, wie Sie sich anmelden und feststellen können, ob die Genehmigung erforderlich ist und die Tag-Autorisierung durchgesetzt wird, finden Sie unter Zeitplan für die Tagging-Autorisierung im Amazon Elastic Container Service Developer Guide.

Sie müssen auch die `iam:PassRole` Berechtigungen hinzufügen, um IAM-Rollen für Aufgaben verwenden zu können. Weitere Informationen finden Sie unter IAM-Rolle für die Ausführung von Amazon ECS-Aufgaben und IAM-Rollen für Aufgaben. Verwenden Sie den folgenden Richtlinientext.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [

 "arn:aws:iam::aws_account_ID:role/ecsTaskExecutionRole_or_TaskRole_name"
            ]
        }
    ]
}
```

7. Für die `CodeDeployToECS` Aktion (blaue/grüne Bereitstellungen) sind die folgenden Mindestberechtigungen erforderlich, um Pipelines mit einer blauen/grünen Bereitstellungsaktion CodeDeploy zu Amazon ECS zu erstellen.

```
{
    "Effect": "Allow",
    "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetDeployment",
        "codedeploy:GetApplication",
        "codedeploy:GetApplicationRevision",
        "codedeploy:RegisterApplicationRevision",
```

```
        "codedeploy:GetDeploymentConfig",
        "ecs:RegisterTaskDefinition",
        "ecs:TagResource"
    ],
    "Resource": "resource_ARN"
},
```

Sie können sich für die Verwendung der Tagging-Autorisierung in Amazon ECS entscheiden. Wenn Sie sich anmelden, müssen Sie die folgenden Berechtigungen erteilen:. `ecs:TagResource` Weitere Informationen darüber, wie Sie sich anmelden und feststellen können, ob die Genehmigung erforderlich ist und die Tag-Autorisierung durchgesetzt wird, finden Sie unter Zeitplan für die Tagging-Autorisierung im Amazon Elastic Container Service Developer Guide.

Sie müssen auch die `iam:PassRole` Berechtigungen hinzufügen, um IAM-Rollen für Aufgaben verwenden zu können. Weitere Informationen finden Sie unter IAM-Rolle für die Ausführung von Amazon ECS-Aufgaben und IAM-Rollen für Aufgaben. Verwenden Sie den folgenden Richtlinientext.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [

 "arn:aws:iam::aws_account_ID:role/ecsTaskExecutionRole_or_TaskRole_name"
            ]
        }
    ]
}
```

Sie können der Liste der Dienste unter der `iam:PassedToService` Bedingung auch etwas hinzufügen`ecs-tasks.amazonaws.com`, wie in diesem Beispiel gezeigt.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
```

```
            ],
            "Resource": "resource_ARN",
            "Condition": {
                "StringEqualsIfExists": {
                    "iam:PassedToService": [
                        "cloudformation.amazonaws.com",
                        "elasticbeanstalk.amazonaws.com",
                        "ec2.amazonaws.com",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        },
```

8. Für AWS CodeStar Verbindungen ist die folgende Berechtigung erforderlich, um Pipelines mit einer Quelle zu erstellen, die eine Verbindung verwendet, z. B. Bitbucket Cloud.

```
{
    "Effect": "Allow",
    "Action": [
        "codestar-connections:UseConnection"
    ],
    "Resource": "resource_ARN"
},
```

   Weitere Informationen zu den IAM-Berechtigungen für Verbindungen findest du unter Referenz zu Verbindungsberechtigungen.

9. Für die `StepFunctions` Aktion sind die folgenden Mindestberechtigungen aufgeführt, die zum Erstellen von Pipelines mit einer Aktion zum Aufrufen von Step Functions erforderlich sind.

```
{
    "Effect": "Allow",
    "Action": [
        "states:DescribeStateMachine",
        "states:DescribeExecution",
        "states:StartExecution"
    ],
    "Resource": "resource_ARN"
},
```

10. Für die `AppConfig` Aktion sind die folgenden Mindestberechtigungen aufgeführt, die zum Erstellen von Pipelines mit einer AWS AppConfig Aufrufaktion erforderlich sind.

```
{
    "Effect": "Allow",
    "Action": [
        "appconfig:StartDeployment",
        "appconfig:GetDeployment",
        "appconfig:StopDeployment"
    ],
    "Resource": "resource_ARN"
},
```

11. Wenn Sie CodeBuild Unterstützung für Batch-Builds benötigen, fügen Sie Ihrer Richtlinienerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuildBatches",
        "codebuild:StartBuildBatch"
    ],
    "Resource": "resource_ARN"
},
```

12. Für AWS CloudFormation StackSets Aktionen sind die folgenden Mindestberechtigungen erforderlich.

    - Für die `CloudFormationStackSet` Aktion fügen Sie Ihrer Richtlinienerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackSet",
        "cloudformation:UpdateStackSet",
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:DescribeStackSet",
        "cloudformation:ListStackInstances"
    ],
    "Resource": "resource_ARN"
},
```

- Für die `CloudFormationStackInstances` Aktion fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation"
    ],
    "Resource": "resource_ARN"
},
```

13. Wenn Sie CodeCommit Unterstützung für die Option „Vollständiges Klonen“ benötigen, fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "codecommit:GetRepository"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Um sicherzustellen, dass für Ihre CodeBuild Aktion die Option „Vollständiges Klonen“ mit einer CodeCommit Quelle verwendet werden kann, müssen Sie der Richtlinienerklärung auch die `codecommit:GitPull` entsprechende Berechtigung für die CodeBuild Servicerolle Ihres Projekts hinzufügen.

14. Für Elastic Beanstalk sind die folgenden Mindestberechtigungen für die Erstellung von Pipelines mit einer `ElasticBeanstalk` Bereitstellungsaktion erforderlich.

```
{
    "Effect": "Allow",
    "Action": [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
```

```
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "cloudformation:*",
        "rds:*",
        "sqs:*",
        "ecs:*"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Sie sollten Platzhalter in der Ressourcenrichtlinie durch die Ressourcen für das Konto ersetzen, auf das Sie den Zugriff beschränken möchten. Weitere Informationen zum Erstellen einer Richtlinie, die den Zugriff mit den geringsten Rechten gewährt, finden Sie unter. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

15 Für eine Pipeline, die Sie für CloudWatch Logs konfigurieren möchten, sind die folgenden Mindestberechtigungen aufgeführt, die Sie der Servicerolle hinzufügen müssen. CodePipeline

```
{
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups",
        "logs:PutRetentionPolicy"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Sie sollten Platzhalter in der Ressourcenrichtlinie durch die Ressourcen für das Konto ersetzen, auf das Sie den Zugriff beschränken möchten. Weitere Informationen zum Erstellen einer Richtlinie, die den Zugriff mit den geringsten Rechten gewährt, finden Sie unter. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

16. Wählen Sie Review policy, um sicherzustellen, dass die Richtlinie keine Fehler enthält. Wenn die Richtlinie fehlerfrei ist, wählen Sie Richtlinie anwenden.

Anmeldung und Überwachung CodePipeline

Mithilfe der Anmeldefunktionen können AWS Sie feststellen, welche Aktionen Benutzer in Ihrem Konto ausgeführt haben und welche Ressourcen verwendet wurden. Die Protokolldateien enthalten Folgendes:

- Datum und Uhrzeit der Aktionen.
- Die Quell-IP-Adresse für eine Aktion.
- Welche Aktionen aufgrund unzureichender Berechtigungen fehlgeschlagen sind.

Die Protokollierungsfunktionen sind in den folgenden Bereichen verfügbar AWS-Services:

- AWS CloudTrail kann verwendet werden, um AWS API-Aufrufe und zugehörige Ereignisse zu protokollieren, die von oder im Namen eines getätigt wurden AWS-Konto. Weitere Informationen finden Sie unter CodePipeline API-Aufrufe protokollieren mit AWS CloudTrail.
- Amazon CloudWatch Events kann verwendet werden, um Ihre AWS Cloud Ressourcen und die Anwendungen, auf denen Sie laufen, zu überwachen AWS. Sie können Benachrichtigungen in Amazon CloudWatch Events auf der Grundlage von Metriken erstellen, die Sie definieren. Weitere Informationen finden Sie unter CodePipeline Ereignisse überwachen.

Konformitätsvalidierung für AWS CodePipeline

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen

und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Basisumgebungen beschrieben AWS , bei denen Sicherheit und Compliance im Mittelpunkt stehen.
- Architecting for HIPAA Security and Compliance on Amazon Web Services — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-fähige Anwendungen erstellen AWS können.

> **Note**
>
> AWS-Services Nicht alle sind HIPAA-fähig. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.
- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.

- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Resilienz in AWS CodePipeline

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Sicherheit der Infrastruktur in AWS CodePipeline

Als verwalteter Dienst AWS CodePipeline ist er durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe für den Zugriff CodePipeline über das Netzwerk. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Bewährte Methoden für die Gewährleistung der Sicherheit

CodePipeline bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Sie verwenden Verschlüsselung und Authentifizierung für die Quell-Repositorys, die mit Ihren Pipelines verbunden werden. Dies sind die CodePipeline bewährten Methoden für die Sicherheit:

- Wenn Sie eine Pipeline- oder Aktionskonfiguration erstellen, die Geheimnisse wie Token oder Passwörter enthalten muss, geben Sie keine Geheimnisse direkt in die Aktionskonfiguration oder Standardwerte von Variablen ein, die auf Pipeline-Ebene oder AWS CloudFormation Konfiguration definiert wurden, da die Informationen in Protokollen angezeigt werden. Verwenden Sie Secrets Manager, um Secrets einzurichten und zu speichern, und verwenden Sie dann das referenzierte Secret in der Pipeline- und Aktionskonfiguration, wie unter beschrieben Wird verwendet AWS Secrets Manager , um Datenbankkennwörter oder API-Schlüssel von Drittanbietern zu verfolgen.
- Wenn Sie eine Pipeline erstellen, die einen S3-Quell-Bucket verwendet, konfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte CodePipeline durch Verwalten AWS KMS keys, wie unter beschrieben. Konfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte für CodePipeline
- Wenn Sie einen Jenkins-Build-Anbieter für ein Build oder eine Testaktion Ihrer Pipeline verwenden und dafür einen Jenkins-Aktionsanbieter nutzen, installieren Sie Jenkins auf einer EC2-Instance und konfigurieren Sie ein separates EC2-Instance-Profil. Stellen Sie sicher, dass das Instance-Profil Jenkins nur die AWS Berechtigungen gewährt, die für die Ausführung von Aufgaben für Ihr Projekt erforderlich sind, z. B. das Abrufen von Dateien aus Amazon S3. Eine Anleitung zum Erstellen der Rolle für Ihr Jenkins-Instance-Profil finden Sie unter Erstellen Sie eine IAM-Rolle, die für die Jenkins-Integration verwendet werden soll.

AWS CodePipeline Befehlszeilenreferenz

Verwenden Sie diese Referenz bei der Arbeit mit den AWS CodePipeline Befehlen und als Ergänzung zu den Informationen, die im AWS CLI Benutzerhandbuch und in der AWS CLI Referenz dokumentiert sind.

Bevor Sie das verwenden AWS CLI, stellen Sie sicher, dass Sie die Voraussetzungen unter erfüllenErste Schritte mit CodePipeline.

Führen Sie den folgenden Befehl aus, um eine Liste aller verfügbaren CodePipeline Befehle anzuzeigen:

```
aws codepipeline help
```

Um Informationen zu einem bestimmten CodePipeline Befehl anzuzeigen, führen Sie den folgenden Befehl aus, wobei *Befehlsname* der Name eines der unten aufgeführten Befehle ist (z. B.create-pipeline):

```
aws codepipeline command-name help
```

Lesen Sie einen oder mehrere der folgenden Abschnitte AWS CLI, um sich mit der Verwendung der Befehle in der CodePipeline Erweiterung von vertraut zu machen:

- Erstellen einer benutzerdefinierten Aktion
- Erstellen einer Pipeline (CLI)
- Löschen einer Pipeline (CLI)
- Deaktivieren oder Aktivieren von Übergängen (CLI)
- Anzeigen von der Pipeline-Details und des Verlaufs (CLI)
- Wiederholen fehlgeschlagener Aktionen (CLI)
- Manuelles Starten einer Pipeline (CLI)
- Bearbeiten einer Pipeline (AWS CLI)

Sie können sich auch Beispiele für die Verwendung der meisten dieser Befehle unter CodePipeline Anleitungen ansehen.

CodePipeline Referenz zur Pipeline-Struktur

Standardmäßig AWS CodePipeline hat jede Pipeline, in der Sie erfolgreich erstellen, eine gültige Struktur. Wenn Sie jedoch manuell eine JSON-Datei erstellen oder bearbeiten, um eine Pipeline zu erstellen oder eine Pipeline aus der zu aktualisieren AWS CLI, können Sie versehentlich eine Struktur erstellen, die nicht gültig ist. Die folgende Referenz kann Ihnen dabei helfen, die Anforderungen hinsichtlich der Struktur Ihrer Pipeline besser zu verstehen und Probleme zu beheben. Beachten Sie die Einschränkungen in Kontingente in AWS CodePipeline, die für alle Pipelines gelten.

Themen

Gültige Aktionstypen und Anbieter in CodePipeline

Das Pipeline-Struktur-Format wird verwendet, um Aktionen und Phasen in einer Pipeline zu erstellen. Ein Aktionstyp besteht aus einer Aktionskategorie und einem Anbietertyp.

Die folgenden Aktionskategorien sind gültig in CodePipeline:

- Quelle
- Entwicklung
- Test
- Bereitstellen
- Genehmigung
- Aufrufen

Jede Aktionskategorie verfügt über einen designierten Satz von Anbietern. Jeder Aktionsanbieter, wie Amazon S3, hat einen Anbieternamen, z. B.`S3`, der in dem `Provider` Feld in der Aktionskategorie in Ihrer Pipeline-Struktur verwendet werden muss.

Es gibt drei gültige Werte für das `Owner`-Feld im Abschnitt „Aktionskategorie“ in der Pipeline-Struktur: `AWS`, `ThirdParty` und `Custom`.

Informationen zum Anbieternamen und Eigentümerinformationen für Ihren Aktionsanbieter finden Sie unter Referenz der Aktionsstruktur oder Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp.

Diese Tabelle führt die gültigen Anbieter nach Aktionstyp auf.

> **Note**
>
> Informationen zu Bitbucket Cloud- GitHub, GitHub Enterprise Server- oder GitLab .com-Aktionen findest du im Referenzthema CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen Aktionen.

Gültige Aktionsanbieter nach Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
Quelle	Amazon S3	Amazon S3 S3-Quellaktion
	Amazon ECR	Amazon ECR
	CodeCommit	CodeCommit
	CodeStarSourceConnection (für Bitbucket Cloud-GitHub, GitHub Enterprise Server- oder GitLab .com-Aktionen)	CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen
Entwicklung	CodeBuild	AWS CodeBuild
	Benutzerdefiniert CloudBees	Anzahl der Eingabe- und Ausgabe-A

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
		rtefakte für jeden Aktionstyp
	Benutzerdefiniert Jenkins	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
	Benutzerdefiniert TeamCity	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
Test	CodeBuild	AWS CodeBuild
	AWS Device Farm	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
	ThirdParty GhostInspector	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
	Benutzerdefiniert Jenkins	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
	ThirdParty Micro Focus StormRunner Load	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	ThirdParty Nouvola	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
Bereitstellen	Amazon S3	Amazon S3 S3-Bereitstellungsaktion
	AWS CloudFormation	AWS CloudFormation
	AWS CloudFormation StackSets (beinhaltet die Aktionen `CloudFormationStackSet` und `CloudFormationStackInstances`)	AWS CloudFormation StackSets
	CodeDeploy	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	Amazon ECS	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
	Amazon ECS (blau/grün) (dies ist die `CodeDeployToECS` -Aktion)	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	Elastic Beanstalk	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	AWS AppConfig	AWS AppConfig
	AWS OpsWorks	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	Servicekatalog	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	Amazon Alexa	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
	Benutzerdefiniert XebiaLabs	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
Genehmigung	Manuell	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
Aufrufen	AWS Lambda	AWS Lambda
	AWS Step Functions	AWS Step Functions

Einige Aktionstypen in CodePipeline sind nur in ausgewählten AWS Regionen verfügbar. Es ist möglich, dass ein Aktionstyp in einer AWS Region verfügbar ist, aber ein AWS Anbieter für diesen Aktionstyp ist nicht verfügbar.

Weitere Informationen zu den einzelnen Aktionsanbietern finden Sie unter Integrationen mit CodePipeline Aktionstypen.

In den folgenden Abschnitten finden Sie Beispiele für Anbieterinformationen und Konfigurationseigenschaften für jeden Aktionstyp.

Anforderungen an die Pipeline- und Phasenstruktur in CodePipeline

Eine zweiphasige Pipeline hat die folgende Grundstruktur:

```
{
    "roleArn": "An IAM ARN for a service role, such as arn:aws:iam::80398EXAMPLE:role/
CodePipeline_Service_Role",
```

```
    "stages": [
        {
            "name": "SourceStageName",
            "actions": [
            ... See Anforderungen an die Aktionsstruktur in  CodePipeline ...
            ]
        },
        {
            "name": "NextStageName",
            "actions": [
            ... See Anforderungen an die Aktionsstruktur in  CodePipeline ...
            ]
        }
    ],
    "artifactStore": {
        "type": "S3",
        "location": "The name of the Amazon S3 bucket automatically generated for you
 the first time you create a pipeline
           using the console, such as codepipeline-us-east-2-1234567890, or any Amazon
 S3 bucket you provision for this purpose"
    },
    "name": "YourPipelineName",
    "version": 1
}
```

Die Pipeline-Struktur hat folgende Anforderungen:

- Eine Pipeline muss mindestens zwei Phasen enthalten.
- Die erste Phase einer Pipeline muss mindestens eine Quellaktion enthalten. Sie darf nur Quellaktionen enthalten.
- Nur die erste Phase einer Pipeline kann Quellaktionen enthalten.
- Mindestens eine Phase in jeder Pipeline muss eine Aktion enthalten, die keine Quellaktion ist.
- Alle Namen von Phasen in einer Pipeline müssen eindeutig sein.
- Phasennamen können in der CodePipeline Konsole nicht bearbeitet werden. Wenn Sie einen Phasennamen mithilfe von bearbeiten und die Phase eine Aktion mit einem oder mehreren geheimen Parametern (z. B. einem OAuth-Token) enthält, wird der Wert dieser geheimen Parameter nicht beibehalten. AWS CLI Sie müssen den Wert der Parameter (die in der vom zurückgegebenen JSON durch vier Sternchen maskiert sind AWS CLI) manuell eingeben und sie in die JSON-Struktur aufnehmen.

- Das `artifactStore` Feld enthält den Artefakt-Bucket-Typ und den Speicherort für eine Pipeline mit allen Aktionen in derselben Region. AWS Wenn Sie Aktionen in einer anderen Region als Ihrer Pipeline hinzufügen, wird das `artifactStores` Mapping verwendet, um den Artefakt-Bucket für jede AWS Region aufzulisten, in der Aktionen ausgeführt werden. Wenn Sie eine Pipeline erstellen oder bearbeiten, müssen Sie einen Artefakt-Bucket in der Pipelineregion haben, sowie einen Artefakt-Bucket für jede Region, in der Sie eine Aktion ausführen möchten.

  Das folgende Beispiel zeigt die grundlegende Struktur für eine Pipeline mit regionsübergreifenden Aktionen, die den `artifactStores`-Parameter verwendet:

```
    "pipeline": {
        "name": "YourPipelineName",
        "roleArn": "CodePipeline_Service_Role",
        "artifactStores": {
            "us-east-1": {
                "type": "S3",
                "location": "S3 artifact bucket name, such as codepipeline-us-
east-1-1234567890"
            },
            "us-west-2": {
                "type": "S3",
                "location": "S3 artifact bucket name, such as codepipeline-us-
west-2-1234567890"
            }
        },
        "stages": [
            {

...
```

- Die Pipeline-Metadatenfelder unterscheiden sich von der Pipelinestruktur und können nicht bearbeitet werden. Wenn Sie eine Pipeline aktualisieren, wird das Datum im Metadatenfeld `updated` automatisch geändert.
- Wenn Sie eine Pipeline bearbeiten oder aktualisieren, kann der Pipelinename nicht geändert werden.

> **Note**
>
> Wenn Sie eine vorhandene Pipeline umbenennen möchten, können Sie mit dem CLI-Befehl `get-pipeline` eine JSON-Datei erstellen, die die Struktur Ihrer Pipeline enthält.

Anschließend können Sie mit dem CLI-Befehl `create-pipeline` eine Pipeline mit dieser Struktur erstellen und ihr einen neuen Namen geben.

Die Versionsnummer einer Pipeline wird bei jedem Pipeline-Update automatisch generiert und aktualisiert.

Anforderungen an die Aktionsstruktur in CodePipeline

Eine Aktion hat die folgende anspruchsvolle Struktur:

```
[
                {
                   "inputArtifacts": [
                        An input artifact structure, if supported for the action
 category
                    ],
                   "name": "ActionName",
                   "region": "Region",
                   "namespace": "source_namespace",
                   "actionTypeId": {
                        "category": "An action category",
                        "owner": "AWS",
                        "version": "1"
                        "provider": "A provider type for the action category",
                   },
                   "outputArtifacts": [
                        An output artifact structure, if supported for the action
 category
                   ],
                   "configuration": {
                        Configuration details appropriate to the provider type
                   },
                   "runOrder": A positive integer that indicates the run order within
 the stage,
                }
            ]
```

Eine Liste der Beispiel-`configuration` Details für den jeweiligen Anbietertyp finden Sie unter Konfigurationsdetails nach Anbietertyp.

Die Aktionsstruktur hat folgende Anforderungen:

- Alle Aktionsnamen in einer Phase müssen eindeutig sein.
- Das Eingabeartefakt einer Aktion muss exakt mit dem in einer vorherigen Aktion deklarierten Ausgabeartefakt übereinstimmen. Wenn beispielsweise eine vorherige Aktion die folgende Erklärung enthält:

  ```
  "outputArtifacts": [
      {
      "MyApp"
      }
  ],
  ```

  und keine anderen Ausgabeartefakte vorhanden sind, muss das Eingabeartefakt einer nachfolgenden Aktion folgendermaßen lauten:

  ```
  "inputArtifacts": [
      {
      "MyApp"
      }
  ],
  ```

  Dies trifft für alle Aktionen zu, unabhängig davon, ob sie in derselben Phase oder in folgenden Phasen enthalten sind. Das Eingabeartefakt muss aber nicht die nächste Aktion sein, die strikt auf die Aktion folgt, von der das Ausgabeartefakt bereitgestellt wurde. Parallele Aktionen können unterschiedliche Ausgabeartefaktpakete deklarieren, die im Gegenzug von verschiedenen Folgeaktionen genutzt werden.
- Namen von Ausgabeartefakten müssen in einer Pipeline eindeutig sein. Eine Pipeline kann z. B. eine Aktion mit einem Ausgabeartefakt namens `"MyApp"` und eine weitere Aktion mit einem Ausgabeartefakt namens `"MyBuiltApp"` enthalten. Eine Pipeline kann jedoch nicht zwei Aktionen enthalten, die beide ein Ausgabeartefakt mit dem Namen `"MyApp"` haben.
- Bei regionsübergreifenden Aktionen wird das `Region` Feld verwendet, um die AWS Region anzugeben, in der die Aktionen erstellt werden sollen. Die für diese Aktion erstellten AWS Ressourcen müssen in derselben Region erstellt werden, die `region` im Feld angegeben wurde. Für die folgenden Aktionstypen können Sie keine regionsübergreifenden Aktionen erstellen:
  - Quellaktionen
  - Aktionen von Drittanbietern

  - Aktionen von benutzerdefinierten Anbietern
- Aktionen können mit Variablen konfiguriert werden. Sie legen die Namespace- und Variableninformationen für Ausführungsvariablen im Feld `namespace` fest. Referenzinformationen zu Ausführungsvariablen und Aktionsausgabevariablen finden Sie unter Variablen.
- Für alle derzeit unterstützten Aktionstypen ist die einzig gültige BesitzerzeichenfolgeAWS,`ThirdParty`, oder`Custom`. Weitere Informationen finden Sie in der CodePipeline API-Referenz.
- Der standardmäßige `runOrder`-Wert für eine Aktion ist 1. Der Wert muss eine positive Ganzzahl (natürliche Zahl) sein. Sie können keine Bruchzahlen, Dezimalzahlen, negative Zahlen oder Null verwenden. Für eine serielle Reihenfolge von Aktionen geben Sie die niedrigste Zahl für die erste Aktion und höhere Zahlen für jede der verbleibenden Aktionen in der Abfolge an. Parallele Aktionen geben Sie an, indem Sie dieselbe Ganzzahl für jede Aktion verwenden, die parallel ausgeführt werden soll. In der Konsole können Sie eine serielle Reihenfolge für eine Aktion angeben, indem Sie Aktionsgruppe hinzufügen auf der Ebene der Phase auswählen, auf der sie ausgeführt werden soll, oder Sie können eine parallel Reihenfolge angeben, indem Sie Aktion hinzufügen wählen. Aktionsgruppe bezieht sich auf eine Ausführungsreihenfolge einer oder mehrerer Aktionen auf derselben Ebene.

  Wenn Sie beispielsweise drei Aktionen der Reihe nach in einer Phase ausführen möchten, geben Sie der ersten Aktion den `runOrder`-Wert 1, der zweiten Aktion den `runOrder`-Wert 2 und der dritten den `runOrder`-Wert 3. Wenn Sie jedoch die zweite und dritte Aktion parallel ausführen möchten, geben Sie der ersten Aktion den `runOrder`-Wert 1 und sowohl der zweiten als auch der dritten Aktion den `runOrder`-Wert 2.

  > **Note**
  >
  > Die Nummerierung aufeinanderfolgender Aktionen muss nicht in strenger Reihenfolge sein. Wenn Sie beispielsweise drei Aktionen in einer Reihenfolge haben und die zweite Aktion entfernen möchten, müssen Sie den `runOrder`-Wert der dritten Aktion nicht neu nummerieren. Da der `runOrder`-Wert dieser Aktion (3) höher ist als der `runOrder`-Wert der ersten Aktion (1), wird sie nach der ersten Aktion in der Phase ausgeführt.
- Wenn Sie einen Amazon S3 S3-Bucket als Bereitstellungsort verwenden, geben Sie auch einen Objektschlüssel an. Ein Objektschlüssel kann ein Dateiname (Objekt) oder eine Kombination aus Präfix (Ordnerpfad) und Dateiname sein. Sie können Variablen verwenden, um den Namen des Speicherorts anzugeben, der von der Pipeline verwendet werden soll. Für Amazon S3-

Bereitstellungsaktionen werden in Amazon S3-Objektschlüsseln die folgenden Variablen unterstützt.

Verwenden von Variablen in Amazon S3

Variable	Beispiel für Konsoleneingabe	Ausgang
datetime	js-application/{datetime}.zip	UTC-Zeitstempel im folgenden Format: <JJJJ>-<MM>-TT>_<HH>-<MM>-<SS> Beispiel: js-application/2019-01-10_07-39-57.zip
uuid	js-application/{uuid}.zip	Die UUID ist ein global eindeutiger Bezeichner, der garantiert mit keinem der vorhandenen Bezeichner identisch ist. Die UUID hat das Format (alle Ziffern im Hexadezimalformat): <8-Ziffern>-<4-Ziffern>-4-Ziffern>-<4-Ziffern>-<12-Ziffern> Beispiel: js-application/54a60075-b96a-4bf3-9013-db3a9EXAMPLE.zip

- Dies sind die gültigen `actionTypeId` Kategorien für CodePipeline:
  - `Source`
  - `Build`
  - `Approval`
  - `Deploy`
  - `Test`
  - `Invoke`

  Einige Anbietertypen und Konfigurationsoptionen werden hier angegeben.

- Welche Anbietertypen für eine Aktionskategorie gültig sind, hängt von der Kategorie ab. Beispielsweise ist ein gültiger Anbietertyp für einen Quellaktionstyp `S3`, `GitHub`, `CodeCommit` oder `Amazon ECR`. Dieses Beispiel zeigt die Struktur für eine Quellaktion mit einem `S3`-Anbieter:

```
"actionTypeId": {
  "category": "Source",
  "owner": "AWS",
  "version": "1",
  "provider": "S3"},
```

- Jede Aktion muss eine gültige Aktionskonfiguration aufweisen in Abhängigkeit zum Anbietertyp für diese Aktion. In der folgenden Tabelle sind die erforderlichen Aktionskonfigurationselemente für jeden gültigen Anbietertyp aufgeführt:

  Aktionskonfigurationseigenschaften für Anbietertypen

Name des Anbieters	Anbietername in Aktionstyp	Konfigurationseigenschaften	Erforderliche Eigenschaft?
Amazon S3 (Aktionsanbieter bereitstellen)	Weitere Informationen, einschließlich Beispielen zu Amazon S3 S3-Bereitstellungsaktionsparametern, finden Sie unterAmazon S3 S3-Bereitstellungsaktion.		
Amazon S3 (Quellaktionsanbieter)	Weitere Informationen, einschließlich Beispielen zu Amazon S3 S3-Quellaktionsparametern, finden Sie unterAmazon S3 S3-Quellaktion.		
Amazon ECR	Weitere Informationen, einschließlich Beispielen zu Amazon ECR-Parametern, finden Sie unterAmazon ECR.		
CodeCommit	Weitere Informationen, einschließlich Beispielen zu CodeCommit Parametern, finden Sie unterCodeCommit.		
GitHub	Weitere Informationen, einschließlich Beispielen zu GitHub Parametern, finden Sie unterGitHub Referenz zur Struktur der Quellaktion von Version 1.		

Name des Anbieters	Anbietername in Aktionstyp	Konfigurationseigenschaften	Erforderliche Eigenschaft?
AWS CloudFormation	Weitere Informationen, einschließlich Beispielen zu AWS CloudFormation Parametern, finden Sie unterAWS CloudFormation.		
CodeBuild	Weitere Beschreibungen und Beispiele zu CodeBuild Parametern finden Sie unterAWS CodeBuild.		
CodeDeploy	Weitere Beschreibungen und Beispiele zu CodeDeploy Parametern finden Sie unterAWS CodeDeploy.		
AWS Device Farm	Weitere Beschreibungen und Beispiele zu AWS Device Farm Parametern finden Sie unterAWS Device Farm.		
AWS Elastic Beanstalk	`ElasticBeanstalk`	`ApplicationName`	Erforderlich
		`EnvironmentName`	Erforderlich
AWS Lambda	Weitere Informationen, einschließlich Beispielen zu AWS Lambda Parametern, finden Sie unterAWS Lambda.		
AWS OpsWorks Stacks	`OpsWorks`	`Stack`	Erforderlich
		`Layer`	Optional
		`App`	Erforderlich
Amazon ECS	Weitere Beschreibungen und Beispiele zu Amazon ECS-Parametern finden Sie unterAmazon Elastic Container Service.		
Amazon ECS und CodeDeploy (Blau/Grün)	Weitere Beschreibungen und Beispiele zu Amazon ECS und CodeDeploy Blau/Grün-Parametern finden Sie unterAmazon Elastic Container Service und CodeDeploy Blaugrün.		

Name des Anbieters	Anbietername in Aktionstyp	Konfigurationseigenschaften	Erforderliche Eigenschaft?
Servicekatalog	`ServiceCatalog`	`TemplateFilePath`	Erforderlich
		`ProductVersionName`	Erforderlich
		`ProductType`	Erforderlich
		`ProductVersionDescription`	Optional
		`ProductId`	Erforderlich
Alexa Skills Kit	`AlexaSkillsKit`	`ClientId`	Erforderlich
		`ClientSecret`	Erforderlich
		`RefreshToken`	Erforderlich
		`SkillId`	Erforderlich
Jenkins	Der Name der Aktion, die Sie im CodePipeline Plugin für Jenkins angegeben haben (z. B.) *`MyJenkinsProviderName`*	`ProjectName`	Erforderlich
Manuelle Genehmigung	`Manual`	`CustomData`	Optional
		`ExternalEntityLink`	Optional
		`NotificationArn`	Optional

Themen

- Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
- Standardeinstellungen für den Parameter PollForSourceChanges
- Konfigurationsdetails nach Anbietertyp

Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp

Je nach Aktionstyp können Sie die folgenden Zahlen für Eingabe- und Ausgabeartefakte verwenden:

Aktionstypbedingungen für Artefakte

Eigentümer	Aktionstyp	Anbieter	Gültige Zahl der Eingabeartefakte	Gültige Zahl der Ausgabear tefakte
AWS	Quelle	Amazon S3	0	1
AWS	Quelle	CodeCommit	0	1
AWS	Quelle	Amazon ECR	0	1
ThirdParty	Quelle	GitHub	0	1
AWS	Entwicklung	CodeBuild	1 bis 5	0 bis 5
AWS	Test	CodeBuild	1 bis 5	0 bis 5
AWS	Test	AWS Device Farm	1	0
AWS	Genehmigung	Manuell	0	0
AWS	Bereitstellen	Amazon S3	1	0
AWS	Bereitstellen	AWS CloudForm ation	0 bis 10	0 bis 1
AWS	Bereitstellen	CodeDeploy	1	0
AWS	Bereitstellen	AWS Elastic Beanstalk	1	0
AWS	Bereitstellen	AWS OpsWorks Stacks	1	0
AWS	Bereitstellen	Amazon ECS	1	0

Eigentümer	Aktionstyp	Anbieter	Gültige Zahl der Eingabeartefakte	Gültige Zahl der Ausgabeartefakte
`AWS`	Bereitstellen	Servicekatalog	1	0
`AWS`	Aufrufen	AWS Lambda	0 bis 5	0 bis 5
`ThirdParty`	Bereitstellen	Alexa Skills Kit	1 bis 2	0
`Custom`	Entwicklung	Jenkins	0 bis 5	0 bis 5
`Custom`	Test	Jenkins	0 bis 5	0 bis 5
`Custom`	Jede unterstützte Kategorie	Wie in der benutzerdefinierten Aktion angegeben	0 bis 5	0 bis 5

Standardeinstellungen für den Parameter PollForSourceChanges

Der Standard für den Parameter `PollForSourceChanges` wird von der Methode festgelegt, mit der die Pipeline erstellt wird, wie in der nachfolgenden Tabelle beschrieben. In vielen Fällen ist die Standardeinstellung für den Parameter `PollForSourceChanges` „true“ und muss deaktiviert werden.

Wenn der `PollForSourceChanges`-Parameter standardmäßig „true“ ist, sollten Sie wie folgt vorgehen:

- Fügen Sie den Parameter `PollForSourceChanges` der JSON-Datei oder der AWS CloudFormation -Vorlage hinzu.
- Erstellen Sie Ressourcen zur Änderungserkennung (CloudWatch Eventregel, sofern zutreffend).
- Stellen Sie den `PollForSourceChanges`-Parameter auf "false" ein.

> **Note**
>
> Wenn Sie eine CloudWatch Ereignisregel oder einen Webhook erstellen, müssen Sie den Parameter auf false setzen, um zu verhindern, dass die Pipeline mehr als einmal ausgelöst wird.

Der `PollForSourceChanges` Parameter wird nicht für Amazon ECR-Quellaktionen verwendet.

- PollForSourceChanges Standardwerte für Parameter

Quelle	Erstellungsmethode	Beispiel für „Konfiguration“ JSON-Struktur-Ausgabe
CodeCommit	Pipeline wird mit der Konsole erstellt (und die Konsole erstellt Änderungserkennung sressourcen). Der Parameter wird in der Pipeline-Strukturausgabe angezeigt und ist standardmäßig `false`.	`BranchName": "main", "PollForSourceCha nges": "false", "RepositoryName": "my- repo"`
	Die Pipeline wird mit der CLI oder erstellt AWS CloudFormation, und der `PollForSo urceChanges` Parameter wird nicht in der JSON-Ausgabe angezeigt, sondern auf `true` .[2] gesetzt.	`BranchName": "main", "RepositoryName": "my- repo"`
Amazon S3	Pipeline wird mit der Konsole erstellt (und die Konsole erstellt Änderungserkennung sressourcen). Der Parameter wird in der Pipeline-Strukturausgabe angezeigt und ist standardmäßig `false`.	`"S3Bucket": "my-bucke t", "S3ObjectKey": "object.zip", "PollForSourceChanges" : "false"`

Quelle	Erstellungsmethode	Beispiel für „Konfiguration“ JSON-Struktur-Ausgabe
	Die Pipeline wird mit der CLI oder erstellt AWS CloudFormation, und der `PollForSourceChanges` Parameter wird nicht in der JSON-Ausgabe angezeigt, sondern auf `true` .[2] gesetzt.	`"S3Bucket": "my-bucket", "S3ObjectKey": "object.zip"`
GitHub	Pipeline wird mit der Konsole erstellt (und die Konsole erstellt Änderungserkennung sressourcen). Der Parameter wird in der Pipeline-Strukturausgabe angezeigt und ist standardmäßig `false`.	`"Owner": "MyGitHubAccountName ", "Repo": " MyGitHubRepositoryName " "PollForSourceChanges": "false", "Branch": " main" "OAuthToken": " ****"`
	Die Pipeline wird mit der CLI oder erstellt AWS CloudFormation, und der `PollForSourceChanges` Parameter wird nicht in der JSON-Ausgabe angezeigt, sondern auf `true` .[2] gesetzt.	`"Owner": "MyGitHubAccountName ", "Repo": "MyGitHubRepositoryName ", "Branch": " main", "OAuthToken": " ****"`

[2] Wenn `PollForSourceChanges` zu irgendeinem Zeitpunkt der JSON-Struktur oder der AWS CloudFormation Vorlage hinzugefügt wurde, wird sie wie folgt angezeigt:

```
"PollForSourceChanges": "true",
```

[3] Informationen zu den Ressourcen zur Änderungserkennung, die für jeden Quellanbieter gelten, finden Sie unter Methoden zur Änderungserkennung.

Konfigurationsdetails nach Anbietertyp

In diesem Abschnitt werden gültige `configuration`-Parameter für jeden Aktionsanbieter aufgeführt.

Das folgende Beispiel zeigt eine gültige Konfiguration für eine Bereitstellungsaktion, die Service Catalog verwendet, für eine Pipeline, die in der Konsole ohne separate Konfigurationsdatei erstellt wurde:

```
"configuration": {
  "TemplateFilePath": "S3_template.json",
  "ProductVersionName": "devops S3 v2",
  "ProductType": "CLOUD_FORMATION_TEMPLATE",
  "ProductVersionDescription": "Product version description",
  "ProductId": "prod-example123456"
}
```

Das folgende Beispiel zeigt eine gültige Konfiguration für eine Bereitstellungsaktion, die Service Catalog verwendet, für eine Pipeline, die in der Konsole mit einer separaten `sample_config.json` Konfigurationsdatei erstellt wurde:

```
"configuration": {
  "ConfigurationFilePath": "sample_config.json",
  "ProductId": "prod-example123456"
}
```

Das folgende Beispiel zeigt eine gültige Konfiguration für eine Bereitstellungsaktion, für die das Alexa Skills Kit verwendet wird:

```
"configuration": {
  "ClientId": "amzn1.application-oa2-client.aadEXAMPLE",
  "ClientSecret": "****",
  "RefreshToken": "****",
  "SkillId": "amzn1.ask.skill.22649d8f-0451-4b4b-9ed9-bfb6cEXAMPLE"
}
```

Das folgende Beispiel zeigt eine gültige Konfiguration für eine manuelle Genehmigung:

```
"configuration": {
  "CustomData": "Comments on the manual approval",
  "ExternalEntityLink": "http://my-url.com",
```

```
  "NotificationArn": "arn:aws:sns:us-west-2:12345EXAMPLE:Notification"
}
```

Referenz der Aktionsstruktur

Dieser Abschnitt ist nur eine Referenz für die Aktionskonfiguration. Eine konzeptionelle Übersicht über die Pipeline-Struktur finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

Jeder Aktionsanbieter CodePipeline verwendet eine Reihe von erforderlichen und optionalen Konfigurationsfeldern in der Pipeline-Struktur. Dieser Abschnitt enthält die folgenden Referenzinformationen, geordnet nach Aktionsanbieter:

- Gültige Werte für die `ActionType`-Felder, die im Pipeline-Struktur-Aktionsblock enthalten sind, z. B. `Owner` und `Provider`.
- Beschreibungen und andere Referenzinformationen für die `Configuration`-Parameter (erforderliche und optionale), die im Abschnitt zur Pipeline-Strukturaktion enthalten sind.
- Gültige JSON- und YAML-Beispielaktionsfelder.

Dieser Abschnitt wird regelmäßig mit weiteren Aktionsanbietern aktualisiert. Derzeit stehen Referenzinformationen für die folgenden Aktionsanbieter zur Verfügung:

Themen

- AWS Lambda
- Referenz zur Snyk-Aktionsstruktur
- AWS Step Functions

Amazon ECR

Löst die Pipeline aus, wenn ein neues Image in das Amazon ECR-Repository übertragen wird. Diese Aktion stellt eine Bilddefinitionsdatei bereit, die auf den URI für das Bild verweist, das an Amazon ECR übertragen wurde. Diese Quellaktion wird häufig in Verbindung mit einer anderen Quellaktion verwendet, z. B. CodeCommit um einen Quellspeicherort für alle anderen Quellartefakte zuzulassen. Weitere Informationen finden Sie unter Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy .

Wenn Sie die Konsole verwenden, um Ihre Pipeline zu erstellen oder zu bearbeiten, CodePipeline erstellt sie eine CloudWatch Ereignisregel, die Ihre Pipeline startet, wenn eine Änderung im Repository erfolgt.

Sie müssen bereits ein Amazon ECR-Repository erstellt und ein Image übertragen haben, bevor Sie die Pipeline über eine Amazon ECR-Aktion verbinden.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Aktionserklärung (Beispiel Amazon ECR)
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Source`
- Eigentümer: `AWS`
- Anbieter: `ECR`
- Version: `1`

Konfigurationsparameter

RepositoryName

Erforderlich: Ja

Der Name des Amazon ECR-Repositorys, in das das Bild übertragen wurde.

ImageTag

Erforderlich: Nein

Das Tag, das für das Image verwendet wird.

> **Note**
>
> Wenn kein Wert für `ImageTag` angegeben ist, wird standardmäßig der Wert `latest` verwendet.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0`
- Beschreibung: Eingabe-Artefakte sind für diesen Aktionstyp nicht gültig.

Ausgabeartefakte

- Anzahl der Artefakte: `1`
- Beschreibung: Diese Aktion erzeugt ein Artefakt, das die Datei `imageDetail.json` enthält. Diese Datei enthält den URI für das Image, durch das die Pipeline-Ausführung ausgelöst wurde. Informationen zur `imageDetail.json`-Datei finden Sie unter ImageDetail.json-Datei für Amazon ECS-Bereitstellungsaktionen in Blau/Grün.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Diese Aktion erzeugt Variablen, die als Ausgabevariablen angezeigt werden können, auch wenn die Aktion

keinen Namespace hat. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

Weitere Informationen finden Sie unter Variablen.

RegistryId

Die AWS Konto-ID, die der Registrierung zugeordnet ist, die das Repository enthält.

RepositoryName

Der Name des Amazon ECR-Repositorys, in das das Bild übertragen wurde.

ImageTag

Das Tag, das für das Image verwendet wird.

ImageDigest

Der `sha256`-Digest des Image-Manifests.

ImageURI

Der URI für das Image.

Aktionserklärung (Beispiel Amazon ECR)

YAML

```
Name: Source
Actions:
  - InputArtifacts: []
    ActionTypeId:
      Version: '1'
      Owner: AWS
      Category: Source
      Provider: ECR
    OutputArtifacts:
      - Name: SourceArtifact
    RunOrder: 1
    Configuration:
      ImageTag: latest
      RepositoryName: my-image-repo
```

```
    Name: ImageSource
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "InputArtifacts": [],
            "ActionTypeId": {
                "Version": "1",
                "Owner": "AWS",
                "Category": "Source",
                "Provider": "ECR"
            },
            "OutputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "RunOrder": 1,
            "Configuration": {
                "ImageTag": "latest",
                "RepositoryName": "my-image-repo"
            },
            "Name": "ImageSource"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy — Dieses Tutorial enthält eine Beispiel-App-Spezifikationsdatei sowie eine CodeDeploy Beispielanwendung und eine Bereitstellungsgruppe zum Erstellen einer Pipeline mit einer CodeCommit Amazon ECR-Quelle, die auf Amazon ECS-Instances bereitgestellt wird.

Amazon Elastic Container Service und CodeDeploy Blaugrün

Sie können eine Pipeline konfigurieren AWS CodePipeline , in der Containeranwendungen mithilfe einer blauen/grünen Bereitstellung bereitgestellt werden. In einer blauen/grünen Bereitstellung können Sie neben der alten Version auch eine neue Version Ihrer Anwendung starten und die neue Version testen, bevor Sie den Datenverkehr dorthin umleiten. Sie können auch den Bereitstellungsprozess überwachen und bei Problemen schnell ein Rollback durchführen.

Die abgeschlossene Pipeline erkennt Änderungen an Ihren Images oder Ihrer Aufgabendefinitionsdatei und verwendet CodeDeploy sie, um den Datenverkehr an einen Amazon ECS-Cluster und Load Balancer weiterzuleiten und bereitzustellen. CodeDeploy erstellt einen neuen Listener auf Ihrem Load Balancer, der Ihre neue Aufgabe über einen speziellen Port ansprechen kann. Sie können die Pipeline auch so konfigurieren, dass sie einen Quellspeicherort verwendet, z. B. ein CodeCommit Repository, in dem Ihre Amazon ECS-Aufgabendefinition gespeichert ist.

Bevor Sie Ihre Pipeline erstellen, müssen Sie bereits die Amazon ECS-Ressourcen, die CodeDeploy Ressourcen sowie den Load Balancer und die Zielgruppe erstellt haben. Sie müssen das Bild bereits markiert und in Ihrem Image-Repository gespeichert und die Aufgabendefinition und die Datei in Ihr AppSpec Datei-Repository hochgeladen haben.

> **Note**
>
> In diesem Thema wird die Bereitstellungsaktion von Amazon ECS auf CodeDeploy Blau/Grün für CodePipeline beschrieben. Referenzinformationen zu den standardmäßigen Bereitstellungsaktionen von Amazon ECS finden Sie unterAmazon Elastic Container Service. CodePipeline

Themen

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `CodeDeployToECS`
- Version: `1`

Konfigurationsparameter

ApplicationName

Erforderlich: Ja

Der Name der Anwendung in CodeDeploy. Bevor Sie Ihre Pipeline erstellen, müssen Sie die Anwendung bereits in erstellt haben CodeDeploy.

DeploymentGroupName

Erforderlich: Ja

Die für Amazon ECS-Aufgabensätze angegebene Bereitstellungsgruppe, die Sie für Ihre CodeDeploy Anwendung erstellt haben. Bevor Sie Ihre Pipeline erstellen, müssen Sie die Bereitstellungsgruppe bereits in erstellt haben CodeDeploy.

TaskDefinitionTemplateArtifact

Erforderlich: Ja

Der Name des Eingabeartefakts, das die Aufgabendefinitionsdatei für die Bereitstellungsaktion bereitstellt. Dies ist im Allgemeinen der Name des Ausgabeartefakts aus der Quellaktion. Wenn Sie die Konsole verwenden, lautet der Standardname für das Ausgabeartefakt der Quellaktion. `SourceArtifact`

AppSpecTemplateArtifact

Erforderlich: Ja

Der Name des Eingabeartefakts, das die AppSpec Datei für die Bereitstellungsaktion bereitstellt. Dieser Wert wird aktualisiert, wenn Ihre Pipeline ausgeführt wird. Dies ist im Allgemeinen der Name des Ausgabeartefakts aus der Quellaktion. Wenn Sie die Konsole verwenden, lautet der Standardname für das Ausgabeartefakt der Quellaktion. `SourceArtifact` Für

`TaskDefinition` AppSpec Dateien können Sie den `<TASK_DEFINITION>` Platzhaltertext wie hier gezeigt beibehalten.

AppSpecTemplatePath

Erforderlich: Nein

Der Dateiname der Datei, die am AppSpec Speicherort der Pipeline-Quelldatei gespeichert ist, z. B. das CodeCommit Repository Ihrer Pipeline. Der Standarddateiname ist `appspec.yaml`. Wenn Ihre AppSpec Datei denselben Namen hat und auf der Stammebene in Ihrem Datei-Repository gespeichert ist, müssen Sie den Dateinamen nicht angeben. Wenn der Pfad nicht der Standardpfad ist, geben Sie den Pfad und den Dateinamen ein.

TaskDefinitionTemplatePath

Erforderlich: Nein

Der Dateiname der Aufgabendefinition, die im Quellverzeichnis der Pipeline-Datei gespeichert ist, z. B. im CodeCommit Repository Ihrer Pipeline. Der Standarddateiname ist `taskdef.json`. Wenn Ihre Aufgabendefinitionsdatei denselben Namen hat und auf der Stammebene in Ihrem Datei-Repository gespeichert ist, müssen Sie den Dateinamen nicht angeben. Wenn der Pfad nicht der Standardpfad ist, geben Sie den Pfad und den Dateinamen ein.

Bild <Number>ArtifactName

Erforderlich: Nein

Der Name des Eingabeartefakts, das das Bild für die Bereitstellungsaktion bereitstellt. Dies ist im Allgemeinen das Ausgabeartefakt des Bild-Repositorys, z. B. die Ausgabe der Amazon ECR-Quellaktion.

Verfügbare Werte für `<Number>` sind 1 bis 4.

Bild <Number>ContainerName

Erforderlich: Nein

Der Name des Bilds, das im Bild-Repository verfügbar ist, z. B. im Amazon ECR-Quell-Repository.

Verfügbare Werte für `<Number>` sind 1 bis 4.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1 to 5`

- Beschreibung: Die `CodeDeployToECS` Aktion sucht zuerst nach der Aufgabendefinitionsdatei und der AppSpec Datei im Quelldatei-Repository, danach nach dem Bild im Image-Repository, generiert dann dynamisch eine neue Version der Aufgabendefinition und führt schließlich die AppSpec Befehle aus, um den Tasksatz und den Container im Cluster bereitzustellen.

  Die `CodeDeployToECS` Aktion sucht nach einer `imageDetail.json` Datei, die den Bild-URI dem Bild zuordnet. Wenn Sie eine Änderung an Ihrem Amazon ECR-Image-Repository festschreiben, erstellt die Pipeline-ECR-Quellaktion eine `imageDetail.json` Datei für diesen Commit. Sie können auch manuell eine `imageDetail.json` Datei für eine Pipeline hinzufügen, bei der die Aktion nicht automatisiert ist. Informationen zur `imageDetail.json`-Datei finden Sie unter ImageDetail.json-Datei für Amazon ECS-Bereitstellungsaktionen in Blau/Grün.

  Die `CodeDeployToECS` Aktion generiert dynamisch eine neue Version der Aufgabendefinition. In dieser Phase ersetzt diese Aktion Platzhalter in der Aufgabendefinitionsdatei durch Bild-URI, die aus ImageDetail.json-Dateien abgerufen wurde. Wenn Sie beispielsweise IMAGE1_NAME als ContainerName Parameter Image1 festlegen, sollten Sie den Platzhalter <IMAGE1_NAME>als Wert des Bildfeldes in Ihrer Aufgabendefinitionsdatei angeben. In diesem Fall ersetzt die CodeDeployTo ECS-Aktion den Platzhalter durch die <IMAGE1_NAME>tatsächliche Bild-URI, die aus ImageDetail.json in dem Artefakt abgerufen wurde, das Sie als Image1 angeben. ArtifactName

  Bei Aktualisierungen der Aufgabendefinitionen enthält die Datei die Eigenschaft. CodeDeploy `AppSpec.yaml` `TaskDefinition`

  ```
  TaskDefinition: <TASK_DEFINITION>
  ```

  Diese Eigenschaft wird durch die `CodeDeployToECS` Aktion aktualisiert, nachdem die neue Aufgabendefinition erstellt wurde.

  <TASK_DEFINITION>Für den Wert des `TaskDefinition` Felds muss der Platzhaltertext sein. Die `CodeDeployToECS` Aktion ersetzt diesen Platzhalter durch den tatsächlichen ARN der dynamisch generierten Aufgabendefinition.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Aktionsdeklaration

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: CodeDeployToECS
      Version: '1'
    RunOrder: 1
    Configuration:
      AppSpecTemplateArtifact: SourceArtifact
      ApplicationName: ecs-cd-application
      DeploymentGroupName: ecs-deployment-group
      Image1ArtifactName: MyImage
      Image1ContainerName: IMAGE1_NAME
      TaskDefinitionTemplatePath: taskdef.json
      AppSpecTemplatePath: appspec.yaml
      TaskDefinitionTemplateArtifact: SourceArtifact
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
      - Name: MyImage
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "CodeDeployToECS",
                "Version": "1"
            },
            "RunOrder": 1,
```

```
            "Configuration": {
                "AppSpecTemplateArtifact": "SourceArtifact",
                "ApplicationName": "ecs-cd-application",
                "DeploymentGroupName": "ecs-deployment-group",
                "Image1ArtifactName": "MyImage",
                "Image1ContainerName": "IMAGE1_NAME",
                "TaskDefinitionTemplatePath": "taskdef.json",
                "AppSpecTemplatePath": "appspec.yaml",
                "TaskDefinitionTemplateArtifact": "SourceArtifact"
            },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                },
                {
                    "Name": "MyImage"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy — Dieses Tutorial führt Sie durch die Erstellung der Ressourcen CodeDeploy und die Amazon ECS-Ressourcen, die Sie für eine Blue/Green-Bereitstellung benötigen. Das Tutorial zeigt Ihnen, wie Sie ein Docker-Image an Amazon ECR übertragen und eine Amazon ECS-Aufgabendefinition erstellen, die Ihren Docker-Image-Namen, Container-Namen, Amazon ECS-Servicenamen und Load Balancer-Konfiguration auflistet. Das Tutorial führt Sie anschließend durch die Erstellung der AppSpec Datei und der Pipeline für Ihre Bereitstellung.

> **Note**
>
> In diesem Thema und in der Anleitung wird die Aktion CodeDeploy /ECS blue/green für beschrieben. CodePipeline Informationen zu den ECS-Standardaktionen finden Sie unter Tutorial: Kontinuierlicher Einsatz mit. CodePipeline CodePipeline

- AWS CodeDeploy Benutzerhandbuch — Informationen zur Verwendung des Load Balancers, des Produktions-Listeners, der Zielgruppen und Ihrer Amazon ECS-Anwendung in einer blauen/grünen Bereitstellung finden Sie unter Tutorial: Bereitstellen eines Amazon ECS-Service. Diese Referenzinformationen im AWS CodeDeploy Benutzerhandbuch bieten einen Überblick über Blue/Green-Bereitstellungen mit Amazon ECS und. AWS CodeDeploy
- Amazon Elastic Container Service Developer Guide — Informationen zur Arbeit mit Docker-Images und Containern, ECS-Services und -Clustern sowie ECS-Aufgabensätzen finden Sie unter Was ist Amazon ECS?

Amazon Elastic Container Service

Sie können eine Amazon ECS-Aktion verwenden, um einen Amazon ECS-Service und einen Tasksatz bereitzustellen. Ein Amazon ECS-Service ist eine Container-Anwendung, die in einem Amazon ECS-Cluster bereitgestellt wird. Ein Amazon ECS-Cluster ist eine Sammlung von Instances, die Ihre Container-Anwendung in der Cloud hosten. Die Bereitstellung erfordert eine Aufgabendefinition, die Sie in Amazon ECS erstellen, und eine Image-Definitionsdatei, die zur Bereitstellung des Images CodePipeline verwendet wird.

> **Important**
>
> Die Amazon ECS-Standardbereitstellungsaktion für CodePipeline erstellt eine eigene Revision der Aufgabendefinition, die auf der vom Amazon ECS-Service verwendeten Version basiert. Wenn Sie neue Revisionen für die Aufgabendefinition erstellen, ohne den Amazon ECS-Service zu aktualisieren, ignoriert die Bereitstellungsaktion diese Revisionen.

Bevor Sie Ihre Pipeline erstellen, müssen Sie die Amazon ECS-Ressourcen bereits erstellt, das Bild markiert und in Ihrem Image-Repository gespeichert und die BuildSpec Datei in Ihr Datei-Repository hochgeladen haben.

> **Note**
>
> In diesem Referenzthema wird die Amazon ECS-Standardbereitstellungsaktion für beschrieben CodePipeline. Referenzinformationen zu den Bereitstellungsaktionen von Amazon ECS to CodeDeploy Blue/Green finden Sie CodePipeline unterAmazon Elastic Container Service und CodeDeploy Blaugrün.

Themen

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `ECS`
- Version: `1`

Konfigurationsparameter

ClusterName

Erforderlich: Ja

Der Amazon ECS-Cluster in Amazon ECS.

ServiceName

Erforderlich: Ja

Der Amazon ECS-Service, den Sie in Amazon ECS erstellt haben.

FileName

Erforderlich: Nein

Der Name Ihrer Image-Definitionsdatei, die JSON-Datei, die den Container-Namen Ihres Services beschreibt, sowie das Bild und das Tag. Sie verwenden diese Datei für ECS-Standardbereitstellungen. Weitere Informationen finden Sie unter Input artifacts (Eingabeartefakte) und Datei imagedefinitions.json für Amazon ECS-Standardbereitstellungsaktionen.

DeploymentTimeout

Erforderlich: Nein

Das Zeitlimit für die Amazon ECS-Bereitstellungsaktion in Minuten. Die Zeitüberschreitung kann bis zum maximalen Standard-Timeout für diese Aktion konfiguriert werden. Beispielsweise:

```
"DeploymentTimeout": "15"
```

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Die Aktion sucht nach einer `imagedefinitions.json` Datei im Quelldatei-Repository für die Pipeline. Ein Imagedefinitionsdokument ist eine JSON-Datei, die Ihren Amazon ECS-Container-Namen sowie das Bild und das Tag beschreibt. CodePipeline verwendet die Datei, um das Bild aus Ihrem Bild-Repository wie Amazon ECR abzurufen. Sie können manuell eine `imagedefinitions.json` Datei für eine Pipeline hinzufügen, in der die Aktion nicht automatisiert ist. Informationen zur `imagedefinitions.json`-Datei finden Sie unter Datei imagedefinitions.json für Amazon ECS-Standardbereitstellungsaktionen.

  Für die Aktion ist ein vorhandenes Image erforderlich, das bereits in Ihr Image-Repository übertragen wurde. Da die Image-Zuordnung von der `imagedefinitions.json` Datei bereitgestellt wird, erfordert die Aktion nicht, dass die Amazon ECR-Quelle als Quellaktion in die Pipeline aufgenommen wird.

Ausgabeartefakte

- Anzahl der Artefakte: `0`

- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Aktionsdeklaration

YAML

```
Name: DeployECS
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: ECS
  Version: '1'
RunOrder: 2
Configuration:
  ClusterName: my-ecs-cluster
  ServiceName: sample-app-service
  FileName: imagedefinitions.json
  DeploymentTimeout: '15'
OutputArtifacts: []
InputArtifacts:
  - Name: my-image
```

JSON

```
{
    "Name": "DeployECS",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "ECS",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "ClusterName": "my-ecs-cluster",
        "ServiceName": "sample-app-service",
        "FileName": "imagedefinitions.json",
        "DeploymentTimeout": "15"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
```

```
            "Name": "my-image"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Kontinuierliche Bereitstellung mit CodePipeline — Dieses Tutorial zeigt Ihnen, wie Sie ein Dockerfile erstellen, das Sie in einem Quelldatei-Repository speichern, wie z. CodeCommit Als Nächstes zeigt Ihnen das Tutorial, wie Sie eine CodeBuild BuildSpec Datei integrieren, die Ihr Docker-Image erstellt und an Amazon ECR überträgt und Ihre imagedefinitions.json-Datei erstellt. Schließlich erstellen Sie einen Amazon ECS-Service und eine Aufgabendefinition und dann erstellen Sie Ihre Pipeline mit einer Amazon ECS-Bereitstellungsaktion.

  > **Note**
  >
  > Dieses Thema und dieses Tutorial beschreiben die Amazon ECS-Standardbereitstellungsaktion für CodePipeline. Informationen zu den Bereitstellungsaktionen von Amazon ECS to CodeDeploy Blue/Green finden Sie CodePipeline unterTutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy .

- Amazon Elastic Container Service Developer Guide — Informationen zur Arbeit mit Docker-Images und Containern, Amazon ECS-Services und -Clustern sowie Amazon ECS-Aufgabensätzen finden Sie unter Was ist Amazon ECS?

Amazon S3 S3-Bereitstellungsaktion

Sie verwenden eine Amazon S3 S3-Bereitstellungsaktion, um Dateien in einem Amazon S3 S3-Bucket bereitzustellen, um statische Websites zu hosten oder zu archivieren. Sie können angeben, ob Bereitstellungsdateien vor dem Hochladen in Ihren Bucket extrahiert werden sollen.

> **Note**
>
> In diesem Referenzthema wird die Amazon S3 S3-Bereitstellungsaktion beschrieben, bei der es CodePipeline sich bei der Bereitstellungsplattform um einen Amazon S3 S3-Bucket handelt, der für das Hosting konfiguriert ist. Referenzinformationen zur Amazon S3 S3-Quellaktion finden Sie unterAmazon S3 S3-Quellaktion. CodePipeline

Themen

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `S3`
- Version: `1`

Konfigurationsparameter

BucketName

Erforderlich: Ja

Der Name des Amazon S3 S3-Buckets, in dem Dateien bereitgestellt werden sollen.

Extrahieren

Erforderlich: Ja

Falls wahr, gibt dies an, dass Dateien vor dem Upload extrahiert werden sollen. Andernfalls bleiben die Anwendungsdateien für den Upload gezippt, z. B. im Fall einer gehosteten statischen Website. Wenn der Wert falsch ist, `ObjectKey` ist der erforderlich.

ObjectKey

Bedingt. Erforderlich, wenn `Extract` = false

Der Name des Amazon S3 S3-Objektschlüssels, der das Objekt im S3-Bucket eindeutig identifiziert.

FRAU EncryptionKey ARN

Erforderlich: Nein

Der ARN des AWS KMS Verschlüsselungsschlüssels für den Host-Bucket. Der `KMSEncryptionKeyARN` Parameter verschlüsselt hochgeladene Artefakte mit den bereitgestellten AWS KMS key. Für einen KMS-Schlüssel können Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN verwenden.

> **Note**
>
> Aliase werden nur in dem Konto erkannt, das den KMS-Schlüssel erstellt hat. Für kontoübergreifende Aktionen können Sie zum Identifizieren des Schlüssels nur die Schlüssel-ID oder den Schlüssel-ARN vewenden. Bei kontoübergreifenden Aktionen wird die Rolle des anderen Kontos (AccountB) verwendet, sodass bei Angabe der Schlüssel-ID der Schlüssel des anderen Kontos (AccountB) verwendet wird.

> **Important**
>
> CodePipeline unterstützt nur symmetrische KMS-Schlüssel. Verwenden Sie keinen asymmetrischen KMS-Schlüssel, um die Daten in Ihrem S3-Bucket zu verschlüsseln.

CannedACL

Erforderlich: Nein

Der `CannedACL` Parameter wendet die angegebene gespeicherte ACL auf Objekte an, die in Amazon S3 bereitgestellt werden. Dadurch werden alle auf das Objekt angewandten vorhandenen ACLs überschrieben.

CacheControl

Erforderlich: Nein

Der `CacheControl` Parameter steuert das Caching-Verhalten für Anforderungen/Antworten für Objekte im Bucket. Eine Liste der gültigen Werte finden Sie im `Cache-Control`-Header-Feld für HTTP-Operationen. Um mehrere Werte in `CacheControl` einzugeben, verwenden Sie ein Komma zwischen den einzelnen Werten. Sie können nach jedem Komma ein Leerzeichen hinzufügen (optional), wie in diesem Beispiel für die CLI gezeigt:

```
"CacheControl": "public, max-age=0, no-transform"
```

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1`
- Beschreibung: Die Dateien für die Bereitstellung oder Archivierung werden aus dem Quell-Repository abgerufen, gezippt und von hochgeladen. CodePipeline

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Beispielaktionskonfiguration

Im Folgenden werden Beispiele für die Aktionskonfiguration gezeigt.

Beispielkonfiguration, wenn eingestellt **`Extract`** ist auf **`false`**

Das folgende Beispiel zeigt die Standardaktionskonfiguration, wenn die Aktion so erstellt wird, dass das `Extract` Feld auf gesetzt ist`false`.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
```

```
      Category: Deploy
      Owner: AWS
      Provider: S3
      Version: '1'
    RunOrder: 1
    Configuration:
      BucketName: website-bucket
      Extract: 'false'
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "S3",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BucketName": "website-bucket",
                "Extract": "false"
                },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

Die Beispielkonfiguration wann **`Extract`** ist auf gesetzt **`true`**

Das folgende Beispiel zeigt die Standardaktionskonfiguration, wenn die Aktion so erstellt wird, dass das `Extract` Feld auf gesetzt ist`true`.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: S3
      Version: '1'
    RunOrder: 1
    Configuration:
      BucketName: website-bucket
      Extract: 'true'
      ObjectKey: MyWebsite
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "S3",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BucketName": "website-bucket",
                "Extract": "true",
```

```
                "ObjectKey": "MyWebsite"
                },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen Sie eine Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet— Dieses Tutorial führt Sie durch zwei Beispiele für die Erstellung einer Pipeline mit einer S3-Bereitstellungsaktion. Sie laden Beispieldateien herunter, laden die Dateien in Ihr CodeCommit Repository hoch, erstellen Ihren S3-Bucket und konfigurieren Ihren Bucket für das Hosting. Als Nächstes verwenden Sie die CodePipeline Konsole, um Ihre Pipeline zu erstellen und eine Amazon S3 S3-Bereitstellungskonfiguration festzulegen.
- Amazon S3 S3-Quellaktion— Diese Aktionsreferenz enthält Referenzinformationen und Beispiele für Amazon S3 S3-Quellaktionen in CodePipeline.

Amazon S3 S3-Quellaktion

Löst die Pipeline aus, wenn ein neues Objekt zum konfigurierten Bucket und Objektschlüssel hochgeladen wird.

> **Note**
>
> In diesem Referenzthema wird die Amazon S3 S3-Quellaktion beschrieben, bei der es CodePipeline sich bei dem Quellspeicherort um einen Amazon S3 S3-Bucket handelt,

der für die Versionierung konfiguriert ist. Referenzinformationen zur Amazon S3 S3-Bereitstellungsaktion in CodePipeline finden Sie unterAmazon S3 S3-Bereitstellungsaktion.

Sie können einen Amazon S3 S3-Bucket erstellen, den Sie als Quellverzeichnis für Ihre Anwendungsdateien verwenden können.

> **Note**
>
> Wenn Sie den Quell-Bucket erstellen, müssen Sie für den Bucket das Versioning aktivieren. Wenn Sie einen vorhandenen Amazon S3 S3-Bucket verwenden möchten, finden Sie weitere Informationen unter Versionierung verwenden, um die Versionierung für einen vorhandenen Bucket zu aktivieren.

Wenn Sie die Konsole verwenden, um Ihre Pipeline zu erstellen oder zu bearbeiten, CodePipeline erstellt sie eine CloudWatch Ereignisregel, die Ihre Pipeline startet, wenn eine Änderung im S3-Quell-Bucket erfolgt.

Sie müssen bereits einen Amazon S3 S3-Quell-Bucket erstellt und die Quelldateien als einzelne ZIP-Datei hochgeladen haben, bevor Sie die Pipeline über eine Amazon S3 S3-Aktion verbinden.

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, können Sie Ihre Quelldatei (en) in eine einzige ZIP-Datei komprimieren und die ZIP-Datei in Ihren Quell-Bucket hochladen. Sie können auch eine einzelne Datei ungezippt hochladen, aber nachgelagerte Aktionen, die eine ZIP-Datei erwarten, schlagen dann fehl.

Themen

- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Source`
- Eigentümer: `AWS`
- Anbieter: `S3`
- Version: `1`

Konfigurationsparameter

S3 Bucket

Erforderlich: Ja

Der Name des Amazon S3 S3-Buckets, in dem Quelländerungen erkannt werden sollen.

S3 ObjectKey

Erforderlich: Ja

Der Name des Amazon S3 S3-Objektschlüssels, bei dem Quelländerungen erkannt werden sollen.

AllowOverrideForS3 ObjectKey

Erforderlich: Nein

`AllowOverrideForS3ObjectKey`steuert, ob Quellenüberschreibungen von die bereits `S3ObjectKey` in der Quelle konfigurierte Aktion überschreiben `StartPipelineExecution` können. Weitere Informationen zu Quellenüberschreibungen mit dem S3-Objektschlüssel finden Sie unter. Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung

> **Important**
>
> Wenn Sie diesen Parameter weglassen`AllowOverrideForS3ObjectKey`, ist CodePipeline standardmäßig die Fähigkeit aktiviert, S3 ObjectKey in der Quellaktion zu überschreiben, indem Sie diesen Parameter auf setzen. `false`

Gültige Werte für diesen Parameter sind:

- `true`: Falls gesetzt, kann der vorkonfigurierte S3-Objektschlüssel während einer Pipeline-Ausführung durch Überschreibungen der Quellversion außer Kraft gesetzt werden.

  > **Note**
  >
  > Wenn Sie allen CodePipeline Benutzern die Möglichkeit geben möchten, den vorkonfigurierten S3-Objektschlüssel zu überschreiben, während sie eine neue Pipeline-Ausführung starten, müssen Sie auf einstellen. `AllowOverrideForS3ObjectKey` `true`

- `false`:

  Wenn diese Option gesetzt ist, CodePipeline kann der S3-Objektschlüssel nicht mithilfe von Quellrevisionsüberschreibungen überschrieben werden. Dies ist auch der Standardwert für diesen Parameter.

PollForSourceChanges

Erforderlich: Nein

`PollForSourceChanges`steuert, ob der Amazon S3 S3-Quell-Bucket nach Quelländerungen CodePipeline abgefragt wird. Wir empfehlen, stattdessen CloudWatch Events zu verwenden und Quelländerungen CloudTrail zu erkennen. Weitere Informationen zur Konfiguration von CloudWatch Ereignissen finden Sie unter Migrieren Sie Polling-Pipelines mit einer S3-Quelle und CloudTrail -Trail (CLI) oderMigrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (Vorlage)AWS CloudFormation.

> **Important**
>
> Wenn Sie CloudWatch Ereignisse konfigurieren möchten, müssen Sie `PollForSourceChanges` auf einstellen, `false` um doppelte Pipeline-Ausführungen zu vermeiden.

Gültige Werte für diesen Parameter sind:

- `true`: Wenn diese Option gesetzt ist, CodePipeline wird Ihr Quellspeicherort nach Quellenänderungen abgefragt.

> **Note**
>
> Wenn Sie diese Option weglassen`PollForSourceChanges`, wird CodePipeline standardmäßig Ihr Quellort nach Quellenänderungen abgefragt. Dieses Verhalten ist das gleiche, als ob `PollForSourceChanges` enthalten wäre und auf `true` festgelegt würde.

- `false`: Wenn diese Option aktiviert ist, CodePipeline wird Ihr Quellort nicht nach Quellenänderungen abgefragt. Verwenden Sie diese Einstellung, wenn Sie beabsichtigen, eine CloudWatch Ereignisregel zur Erkennung von Quellenänderungen zu konfigurieren.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0`
- Beschreibung: Eingabe-Artefakte sind für diesen Aktionstyp nicht gültig.

Ausgabeartefakte

- Anzahl der Artefakte: `1`
- Beschreibung: Stellt die Artefakte bereit, die in dem Quell-Bucket verfügbar sind, der für die Verbindung mit der Pipeline konfiguriert wurde. Die aus dem Bucket generierten Artefakte sind die Ausgabeartefakte für die Amazon S3 S3-Aktion. Die Amazon S3 S3-Objektmetadaten (ETag und Versions-ID) werden CodePipeline als Quellrevision für die ausgelöste Pipeline-Ausführung angezeigt.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Diese Aktion erzeugt Variablen, die als Ausgabevariablen angezeigt werden können, auch wenn die Aktion keinen Namespace hat. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

Weitere Informationen zu Variablen in CodePipeline finden Sie unterVariablen.

BucketName

Der Name des Amazon S3 S3-Buckets, der sich auf die Quellenänderung bezieht, die die Pipeline ausgelöst hat.

ETag

Das Entitäts-Tag für das Objekt, das sich auf die Quelländerung bezieht, von der die Pipeline ausgelöst wurde. Der ETag ist ein MD5-Hash des Objekts. Das ETag gibt nur Änderungen des Inhalt eines Objekts wieder, nicht der Metadaten eines Objekts.

ObjectKey

Der Name des Amazon S3 S3-Objektschlüssels, der sich auf die Quellenänderung bezieht, die die Pipeline ausgelöst hat.

VersionId

Die Versions-ID für die Version des Objekts, die sich auf die Quelländerung bezieht, von der die Pipeline ausgelöst wurde.

Aktionsdeklaration

YAML

```
Name: Source
Actions:
  - RunOrder: 1
    OutputArtifacts:
      - Name: SourceArtifact
    ActionTypeId:
      Provider: S3
      Owner: AWS
      Version: '1'
      Category: Source
    Region: us-west-2
    Name: Source
    Configuration:
      S3Bucket: my-bucket-oregon
      S3ObjectKey: my-application.zip
      PollForSourceChanges: 'false'
    InputArtifacts: []
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "RunOrder": 1,
            "OutputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "ActionTypeId": {
                "Provider": "S3",
                "Owner": "AWS",
                "Version": "1",
                "Category": "Source"
            },
            "Region": "us-west-2",
            "Name": "Source",
            "Configuration": {
                "S3Bucket": "my-bucket-oregon",
                "S3ObjectKey": "my-application.zip",
                "PollForSourceChanges": "false"
            },
            "InputArtifacts": []
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket)— Dieses Tutorial enthält ein Beispiel für eine App-Spezifikationsdatei sowie eine CodeDeploy Beispielanwendung und eine Bereitstellungsgruppe. Verwenden Sie dieses Tutorial, um eine Pipeline mit einer Amazon S3 S3-Quelle zu erstellen, die auf Amazon EC2 EC2-Instances bereitgestellt wird.

AWS AppConfig

AWS AppConfig ist eine Fähigkeit von AWS Systems Manager. AppConfig unterstützt kontrollierte Bereitstellungen für Anwendungen jeder Größe und umfasst integrierte Validierungsprüfungen und Überwachung. Sie können es AppConfig mit Anwendungen verwenden, die auf Amazon EC2 EC2-Instances AWS Lambda, Containern, mobilen Anwendungen oder IoT-Geräten gehostet werden.

Bei der `AppConfig` Bereitstellungsaktion handelt es sich um eine AWS CodePipeline Aktion, mit der Konfigurationen, die in Ihrem Pipeline-Quellverzeichnis gespeichert sind, für eine bestimmte AppConfig Anwendung, eine bestimmte Umgebung und ein bestimmtes Konfigurationsprofil bereitgestellt werden. Sie verwendet die in einer AppConfig Bereitstellungsstrategie definierten Einstellungen.

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `AppConfig`
- Version: `1`

Konfigurationsparameter

Anwendung

Erforderlich: Ja

Die ID der AWS AppConfig Anwendung mit den Details für Ihre Konfiguration und Bereitstellung.

Umgebung

Erforderlich: Ja

Die ID der AWS AppConfig Umgebung, in der die Konfiguration bereitgestellt wird.

ConfigurationProfile

Erforderlich: Ja

Die ID des AWS AppConfig Konfigurationsprofils, das bereitgestellt werden soll.

InputArtifactConfigurationPath

Erforderlich: Ja

Der Dateipfad der Konfigurationsdaten innerhalb des bereitzustellenden Eingabeartefakts.

DeploymentStrategy

Erforderlich: Nein

Die für die AWS AppConfig Bereitstellung zu verwendende Bereitstellungsstrategie.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Das Eingabeartefakt für die Bereitstellungsaktion.

Ausgabeartefakte

Nicht zutreffend.

Beispielaktionskonfiguration

YAML

```
name: Deploy
actions:
  - name: Deploy
    actionTypeId:
      category: Deploy
      owner: AWS
      provider: AppConfig
      version: '1'
    runOrder: 1
    configuration:
      Application: 2s2qv57
      ConfigurationProfile: PvjrpU
      DeploymentStrategy: frqt7ir
      Environment: 9tm27yd
      InputArtifactConfigurationPath: /
    outputArtifacts: []
```

```
    inputArtifacts:
      - name: SourceArtifact
    region: us-west-2
    namespace: DeployVariables
```

JSON

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "AppConfig",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "Application": "2s2qv57",
                "ConfigurationProfile": "PvjrpU",
                "DeploymentStrategy": "frqt7ir",
                "Environment": "9tm27yd",
                "InputArtifactConfigurationPath": "/"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-west-2",
            "namespace": "DeployVariables"
        }
    ]
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWS AppConfig— Informationen zu AWS AppConfig Bereitstellungen finden Sie im AWS Systems Manager Benutzerhandbuch.
- Tutorial: Eine Pipeline erstellen, die AWS AppConfig als Bereitstellungsanbieter verwendet wird— In diesem Tutorial erfahren Sie, wie Sie einfache Konfigurationsdateien und AppConfig Ressourcen für die Bereitstellung einrichten. Außerdem erfahren Sie, wie Sie mithilfe der Konsole eine Pipeline mit einer AWS AppConfig Bereitstellungsaktion erstellen.

AWS CloudFormation

Führt eine Operation auf einem AWS CloudFormation Stack aus. Ein Stack ist eine Sammlung von AWS Ressourcen, die Sie als eine Einheit verwalten können. Die Ressourcen in einem Stack werden durch die AWS CloudFormation -Vorlage des Stacks definiert. Ein Änderungssatz erstellt einen Vergleich, der angezeigt werden kann, ohne den ursprünglichen Stack zu ändern. Informationen zu den Arten von AWS CloudFormation Aktionen, die mit Stacks und Änderungssätzen ausgeführt werden können, finden Sie unter dem `ActionMode` Parameter.

CodePipeline Ruft die AWS CloudFormation `DescribeStackEvents` API auf, um eine Fehlermeldung für eine AWS CloudFormation Aktion zu erstellen, bei der ein Stack-Vorgang fehlgeschlagen ist. Wenn eine Aktions-IAM-Rolle berechtigt ist, auf diese API zuzugreifen, werden die Details zur ersten ausgefallenen Ressource in die CodePipeline Fehlermeldung aufgenommen. Andernfalls ignoriert die Rollenrichtlinie den Zugriff auf die API und zeigt stattdessen eine allgemeine Fehlermeldung an, CodePipeline wenn die Rollenrichtlinie nicht über die entsprechende Berechtigung verfügt. Dazu muss die `cloudformation:DescribeStackEvents` Berechtigung der Servicerolle oder anderen IAM-Rollen für die Pipeline hinzugefügt werden.

Wenn Sie nicht möchten, dass die Ressourcendetails in den Pipeline-Fehlermeldungen auftauchen, können Sie diese Berechtigung für die Aktions-IAM-Rolle widerrufen, indem Sie die Berechtigung entfernen. `cloudformation:DescribeStackEvents`

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Aktionsdeklaration

- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `CloudFormation`
- Version: `1`

Konfigurationsparameter

ActionMode

Erforderlich: Ja

`ActionMode`ist der Name der Aktion, die für einen Stack oder einen Änderungssatz AWS CloudFormation ausgeführt wird. Die folgenden Aktionsmodi sind verfügbar:

- `CHANGE_SET_EXECUTE` führt einen Änderungssatz für den Ressourcen-Stack aus, der auf einer Reihe von angegebenen Ressourcenaktualisierungen basiert. AWS CloudFormation Beginnt mit dieser Aktion, den Stack zu verändern.
- `CHANGE_SET_REPLACE` erstellt den Änderungssatz, falls er noch nicht existiert, basierend auf dem Stack-Namen und der Vorlage, die Sie übermitteln. Wenn der Änderungssatz vorhanden ist, wird er AWS CloudFormation gelöscht und anschließend ein neuer erstellt.
- `CREATE_UPDATE` erstellt den Stack, wenn er nicht vorhanden ist. Wenn der Stapel vorhanden ist, wird der Stapel AWS CloudFormation aktualisiert. Verwenden Sie diese Aktion zum Aktualisieren von bestehenden Stacks. Im Gegensatz `REPLACE_ON_FAILURE` dazu, wenn der Stack existiert und sich in einem ausgefallenen Zustand befindet, CodePipeline wird der Stapel nicht gelöscht und ersetzt.
- `DELETE_ONLY` löscht einen Stack. Wenn Sie einen Stack angeben, der nicht vorhanden ist, wird die Aktion ohne Löschen eines Stacks erfolgreich abgeschlossen.
- `REPLACE_ON_FAILURE` erstellt einen Stack, falls dieser nicht vorhanden ist. Wenn der Stapel vorhanden ist und sich in einem ausgefallenen Zustand befindet, wird der Stapel AWS CloudFormation gelöscht und anschließend ein neuer Stapel erstellt. Wenn sich der Stack nicht in einem ausgefallenen Zustand befindet, wird er AWS CloudFormation aktualisiert.

Der Stack befindet sich in einem fehlerhaften Zustand, wenn einer der folgenden Statustypen in AWS CloudFormation angezeigt wird:

- `ROLLBACK_FAILED`
- `CREATE_FAILED`
- `DELETE_FAILED`
- `UPDATE_ROLLBACK_FAILED`

Verwenden Sie diese Aktion, um fehlerhafte Stacks automatisch ohne Wiederherstellung oder Fehlerbehebung zu ersetzen.

> **Important**
>
> Wir empfehlen Ihnen, `REPLACE_ON_FAILURE` nur für Testzwecke zu verwenden, da Ihr Stack dadurch möglicherweise gelöscht wird.

StackName

Erforderlich: Ja

`StackName` ist der Name eines vorhandenen Stacks oder eines Stacks, den Sie erstellen möchten.

Funktionen

Required: Conditional

Die Verwendung von `Capabilities` bestätigt, dass die Vorlage möglicherweise über die Funktionen verfügt, einige Ressourcen selbst zu erstellen und zu aktualisieren, und dass diese Funktionen basierend auf den Ressourcentypen in der Vorlage bestimmt werden.

Diese Eigenschaft ist erforderlich, wenn Sie über IAM-Ressourcen in Ihrer Stack-Vorlage verfügen oder einen Stack direkt aus einer Vorlage mit Makros erstellen. Damit die AWS CloudFormation Aktion auf diese Weise erfolgreich ausgeführt werden kann, müssen Sie ausdrücklich bestätigen, dass Sie dies mit einer der folgenden Funktionen wünschen:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`
- `CAPABILITY_AUTO_EXPAND`

Sie können mehr als eine Funktion angeben, indem Sie ein Komma (kein Leerzeichen) zwischen den Funktionen verwenden. Das Beispiel in Aktionsdeklaration zeigt einen Eintrag mit den Eigenschaften CAPABILITY_IAM und CAPABILITY_AUTO_EXPAND.

Weitere Informationen zu `Capabilities` finden Sie UpdateStackin den Eigenschaften unter der AWS CloudFormation API-Referenz.

ChangeSetName

Required: Conditional

`ChangeSetName` ist der Name eines vorhandenen Änderungssatzes oder eines neuen Änderungssatzes, den Sie für den angegebenen Stack erstellen möchten.

Diese Eigenschaft ist für die folgenden Aktionsmodi erforderlich: CHANGE_SET_REPLACE und CHANGE_SET_EXECUTE. Bei allen anderen Aktionsmodi wird die Eigenschaft ignoriert.

RoleArn

Required: Conditional

`RoleArn` ist der ARN der IAM-Servicerolle, die AWS CloudFormation annimmt, wenn es mit Ressourcen im angegebenen Stack arbeitet. `RoleArn` wird nicht angewendet, wenn ein Änderungssatz ausgeführt wird. Wenn Sie den Änderungssatz nicht CodePipeline zur Erstellung verwenden, stellen Sie sicher, dass dem Änderungssatz oder Stack eine zugeordnete Rolle zugewiesen ist.

> **Note**
>
> Diese Rolle muss sich in demselben Konto befinden wie die Rolle für die Aktion, die gerade ausgeführt wird, wie in der Aktionsdeklaration konfiguriert`RoleArn`.

Die Eigenschaft ist für die folgenden Aktionsmodi erforderlich:

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- DELETE_ONLY
- CHANGE_SET_REPLACE

> **Note**
>
> AWS CloudFormation erhält eine mit S3 signierte URL zur Vorlage. Daher ist für diese Vorlage `RoleArn` keine Zugriffsberechtigung auf den Artefakt-Bucket erforderlich. Die Aktion `RoleArn` benötigt jedoch die Erlaubnis, auf den Artefakt-Bucket zuzugreifen, um die signierte URL zu generieren.

TemplatePath

Required: Conditional

`TemplatePath`steht für die AWS CloudFormation Vorlagendatei. Sie schließen die Datei in ein Eingabeartefakt für diese Aktion ein. Der Dateiname folgt diesem Format:

`Artifactname``::`*`TemplateFileName`*

`Artifactname`ist der Name des Eingabeartefakts, wie er in CodePipeline erscheint. So erstellt eine Quellstufe mit dem Artefaktnamen `SourceArtifact` und dem Dateinamen `template-export.json` einen `TemplatePath`-Namen, wie in diesem Beispiel gezeigt:

```
"TemplatePath": "SourceArtifact::template-export.json"
```

Die Eigenschaft ist für die folgenden Aktionsmodi erforderlich:

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- CHANGE_SET_REPLACE

Bei allen anderen Aktionsmodi wird die Eigenschaft ignoriert.

> **Note**
>
> Die AWS CloudFormation Vorlagendatei, die den Vorlagentext enthält, hat eine Mindestlänge von 1 Byte und eine Höchstlänge von 1 MB. Für AWS CloudFormation Bereitstellungsaktionen in CodePipeline beträgt die maximale Größe des Eingabeartefakts immer 256 MB. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline und AWS CloudFormation -Limits.

OutputFileName

Erforderlich: Nein

Verwenden Sie diese Option, `OutputFileName` um einen Namen für die Ausgabedatei anzugeben, z. B.`CreateStackOutput.json`, der dem Pipeline-Ausgabeartefakt für diese Aktion CodePipeline hinzugefügt wird. Die JSON-Datei enthält den Inhalt des `Outputs` Abschnitts aus dem AWS CloudFormation Stapel.

Wenn Sie keinen Namen angeben, wird CodePipeline keine Ausgabedatei oder kein Artefakt generiert.

ParameterOverrides

Erforderlich: Nein

Parameter werden in Ihrer Stack-Vorlage definiert und ermöglichen Ihnen, Werte für sie zum Zeitpunkt der Stack-Erstellung oder -Aktualisierung bereitzustellen. Sie können ein JSON-Objekt verwenden, um Parameterwerte in Ihrer Vorlage festzulegen. (Diese Werte überschreiben die in der Vorlagenkonfigurationsdatei festgelegten Werte.) Weitere Informationen zur Verwendung von Parameterüberschreibungen finden Sie unter Konfigurationseigenschaften (JSON-Objekt).

Wir empfehlen Ihnen, die Vorlagenkonfigurationsdatei zu verwenden, um die meisten Ihrer Parameterwerte anzugeben. Verwenden Sie Parameterüberschreibungen nur für Werte, die erst bekannt sind, wenn die Pipeline ausgeführt wird. Weitere Informationen finden Sie im AWS CloudFormation Benutzerhandbuch unter Verwenden von Funktionen zum Überschreiben von Parametern mit CodePipeline Pipelines.

> **Note**
>
> Alle Parameternamen müssen in der Stack-Vorlage vorhanden sein.

TemplateConfiguration

Erforderlich: Nein

`TemplateConfiguration` ist die Vorlagenkonfigurationsdatei. Sie schließen die Datei in ein Eingabeartefakt für diese Aktion ein. Sie kann Vorlagenparameterwerte und eine Stack-Richtlinie enthalten. Weitere Informationen zum Format der Vorlagenkonfigurationsdatei finden Sie unter AWS CloudFormation Artifacts.

Der Dateiname der Vorlagenkonfiguration hat das folgende Format:

Artifactname::*TemplateConfigurationFileName*

`Artifactname`ist der Name des Eingabeartefakts, wie er in CodePipeline erscheint. So erstellt eine Quellstufe mit dem Artefaktnamen `SourceArtifact` und dem Dateinamen `test-configuration.json` einen `TemplateConfiguration`-Namen, wie in diesem Beispiel gezeigt:

```
"TemplateConfiguration": "SourceArtifact::test-configuration.json"
```

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0 to 10`
- Beschreibung: Als Eingabe akzeptiert die AWS CloudFormation Aktion optional Artefakte für folgende Zwecke:
  - Zur Bereitstellung der auszuführenden Stack-Vorlagendatei. (Weitere Informationen finden Sie unter dem `TemplatePath`-Parameter.)
  - Zur Bereitstellung der zu verwendenden Vorlagenkonfigurationsdatei. (Weitere Informationen finden Sie unter dem `TemplateConfiguration`-Parameter.) Weitere Informationen zum Format der Vorlagenkonfigurationsdatei finden Sie unter AWS CloudFormation Artefakte.
  - Um das Artefakt für eine Lambda-Funktion bereitzustellen, die als Teil des AWS CloudFormation Stacks bereitgestellt werden soll.

Ausgabeartefakte

- Anzahl der Artefakte: `0 to 1`
- Beschreibung: Wenn der `OutputFileName`-Parameter angegeben ist, wird durch diese Aktion ein Ausgabeartefakt erzeugt, das eine JSON-Datei mit dem angegebenen Namen enthält. Die JSON-Datei enthält den Inhalt des Ausgabeabschnitts aus dem AWS CloudFormation -Stack.

  Weitere Informationen zum Ausgabeabschnitt, den Sie für Ihre AWS CloudFormation -Aktion erstellen können, finden Sie unter Ausgaben.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

Bei AWS CloudFormation Aktionen werden Variablen aus beliebigen Werten erzeugt, die im `Outputs` Abschnitt einer Stack-Vorlage angegeben sind. Beachten Sie, dass die einzigen CloudFormation Aktionsmodi, die Ausgaben generieren, diejenigen sind, die zur Erstellung oder Aktualisierung eines Stacks führen, wie z. B. die Erstellung von Stacks, Stack-Aktualisierungen und die Ausführung von Änderungssätzen. Die entsprechenden Aktionsmodi, die Variablen generieren, sind:

- `CHANGE_SET_EXECUTE`
- `CHANGE_SET_REPLACE`
- `CREATE_UPDATE`
- `REPLACE_ON_FAILURE`

Weitere Informationen finden Sie unter Variablen. Ein Tutorial, das Ihnen zeigt, wie Sie eine Pipeline mit einer CloudFormation Bereitstellungsaktion in einer Pipeline erstellen, die CloudFormation Ausgabevariablen verwendet, finden Sie unterTutorial: Eine Pipeline erstellen, die Variablen aus AWS CloudFormation Bereitstellungsaktionen verwendet.

Aktionsdeklaration

YAML

```
Name: ExecuteChangeSet
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormation
  Version: '1'
RunOrder: 2
Configuration:
  ActionMode: CHANGE_SET_EXECUTE
  Capabilities: CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND
```

```
  ChangeSetName: pipeline-changeset
  ParameterOverrides: '{"ProjectId": "my-project","CodeDeployRole":
 "CodeDeploy_Role_ARN"}'
  RoleArn: CloudFormation_Role_ARN
  StackName: my-project--lambda
  TemplateConfiguration: 'my-project--BuildArtifact::template-configuration.json'
  TemplatePath: 'my-project--BuildArtifact::template-export.yml'
OutputArtifacts: []
InputArtifacts:
  - Name: my-project-BuildArtifact
```

JSON

```
{
    "Name": "ExecuteChangeSet",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormation",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "ActionMode": "CHANGE_SET_EXECUTE",
        "Capabilities": "CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND",
        "ChangeSetName": "pipeline-changeset",
        "ParameterOverrides": "{\"ProjectId\": \"my-project\",\"CodeDeployRole\":
 \"CodeDeploy_Role_ARN\"}",
        "RoleArn": "CloudFormation_Role_ARN",
        "StackName": "my-project--lambda",
        "TemplateConfiguration": "my-project--BuildArtifact::template-
configuration.json",
        "TemplatePath": "my-project--BuildArtifact::template-export.yml"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
             "Name": "my-project-BuildArtifact"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Referenz zu den Konfigurationseigenschaften — Dieses Referenzkapitel im AWS CloudFormation Benutzerhandbuch enthält weitere Beschreibungen und Beispiele für diese CodePipeline Parameter.
- AWS CloudFormation API-Referenz — Der CreateStackParameter in der AWS CloudFormation API-Referenz beschreibt Stack-Parameter für AWS CloudFormation Vorlagen.

AWS CloudFormation StackSets

CodePipeline bietet die Möglichkeit, AWS CloudFormation StackSets Operationen als Teil Ihres CI/CD-Prozesses durchzuführen. Sie verwenden ein Stack-Set, um mithilfe einer einzigen Vorlage Stapel in AWS Konten in verschiedenen AWS Regionen zu erstellen. AWS CloudFormation Alle in jedem Stack enthaltenen Ressourcen werden durch die AWS CloudFormation Vorlage des Stack-Sets definiert. Wenn Sie das Stack-Set erstellen, geben Sie die zu verwendende Vorlage sowie alle Parameter und Funktionen an, die für die Vorlage erforderlich sind.

Weitere Informationen zu AWS CloudFormation StackSets Konzepten für finden Sie unter StackSets Konzepte im AWS CloudFormation Benutzerhandbuch.

Sie integrieren Ihre Pipeline AWS CloudFormation StackSets mithilfe von zwei unterschiedlichen Aktionstypen, die Sie zusammen verwenden:

- Die `CloudFormationStackSet` Aktion erstellt oder aktualisiert ein Stack-Set oder Stack-Instances anhand der Vorlage, die im Quellverzeichnis der Pipeline gespeichert ist. Jedes Mal, wenn ein Stack-Set erstellt oder aktualisiert wird, initiiert es eine Bereitstellung dieser Änderungen für bestimmte Instances. In der Konsole können Sie den CloudFormation Stack Set-Aktionsanbieter auswählen, wenn Sie Ihre Pipeline erstellen oder bearbeiten.
- Die `CloudFormationStackInstances` Aktion stellt Änderungen an der `CloudFormationStackSet` Aktion auf bestimmte Instanzen bereit, erstellt neue Stack-Instances und definiert Parameterüberschreibungen für bestimmte Instanzen. In der Konsole können Sie den CloudFormation Stack Instances-Aktionsanbieter auswählen, wenn Sie eine bestehende Pipeline bearbeiten.

Sie können diese Aktionen verwenden, um sie für AWS Zielkonten oder Organisationseinheiten-IDs von AWS Zielorganisationen bereitzustellen.

> **Note**
>
> Um die Bereitstellung für Konten oder Organisationseinheiten-IDs von AWS Zielorganisationen durchzuführen und das vom Service verwaltete Berechtigungsmodell zu verwenden, müssen Sie den vertrauenswürdigen Zugriff zwischen AWS CloudFormation StackSets und AWS Organizations aktivieren. Weitere Informationen finden Sie unter Vertrauenswürdigen Zugriff mit AWS CloudFormation Stacksets aktivieren.

Themen

Wie funktionieren Aktionen AWS CloudFormation StackSets

Eine `CloudFormationStackSet` Aktion erstellt oder aktualisiert Ressourcen, je nachdem, ob die Aktion zum ersten Mal ausgeführt wird.

Die `CloudFormationStackSet` Aktion erstellt oder aktualisiert das Stack-Set und stellt diese Änderungen auf bestimmten Instanzen bereit.

> **Note**
>
> Wenn Sie diese Aktion verwenden, um ein Update durchzuführen, das das Hinzufügen von Stack-Instances beinhaltet, werden die neuen Instanzen zuerst bereitgestellt und das Update wird zuletzt abgeschlossen. Die neuen Instanzen erhalten zuerst die alte Version, und dann wird das Update auf alle Instanzen angewendet.

- Erstellen: Wenn keine Instanzen angegeben sind und das Stack-Set nicht existiert, erstellt die CloudFormationStackSetAktion das Stack-Set, ohne dass Instanzen erstellt werden.
- Update: Wenn die CloudFormationStackSetAktion für ein Stack-Set ausgeführt wird, das bereits erstellt wurde, aktualisiert die Aktion das Stack-Set. Wenn keine Instanzen angegeben sind und das Stack-Set bereits existiert, werden alle Instanzen aktualisiert. Wenn diese Aktion verwendet wird, um bestimmte Instanzen zu aktualisieren, werden alle verbleibenden Instanzen in den Status VERALTET versetzt.

  Sie können die CloudFormationStackSetAktion verwenden, um das Stack-Set auf folgende Weise zu aktualisieren.

  - Aktualisieren Sie die Vorlage auf einigen oder allen Instanzen.
  - Aktualisieren Sie die Parameter für einige oder alle Instanzen.
  - Aktualisieren Sie die Ausführungsrolle für das Stack-Set (diese muss mit der in der Administratorrolle angegebenen Ausführungsrolle übereinstimmen).
  - Ändern Sie das Berechtigungsmodell (nur wenn keine Instanzen erstellt wurden).
  - Aktivieren/Deaktivieren, `AutoDeployment` wenn das Stack-Set-Berechtigungsmodell aktiviert ist. `Service Managed`
  - Agieren Sie als delegierter Administrator in einem Mitgliedskonto, wenn das Stack-Set-Berechtigungsmodell dies ist. `Service Managed`
  - Aktualisieren Sie die Administratorrolle.
  - Aktualisieren Sie die Beschreibung auf dem Stack-Set.
  - Fügen Sie dem Stack-Set-Update Bereitstellungsziele hinzu, um neue Stack-Instances zu erstellen.

Die `CloudFormationStackInstances` Aktion erstellt neue Stack-Instances oder aktualisiert veraltete Stack-Instances. Eine Instanz wird veraltet, wenn ein Stack-Set aktualisiert wird, aber nicht alle Instances darin aktualisiert werden.

- Erstellen: Wenn der Stack bereits existiert, aktualisiert die `CloudFormationStackInstances` Aktion nur Instanzen und erstellt keine Stack-Instanzen.
- Update: Wenn die Vorlage oder die Parameter nach Ausführung der `CloudFormationStackSet` Aktion nur in einigen Fällen aktualisiert wurden, wird der Rest markiert`OUTDATED`. `CloudFormationStackInstances`Aktualisiert in späteren Pipeline-Phasen die restlichen Instanzen im Stack in Wellen, sodass alle Instanzen markiert sind`CURRENT`. Diese Aktion kann

auch verwendet werden, um zusätzliche Instanzen hinzuzufügen oder Parameter für neue oder bestehende Instanzen zu überschreiben.

Im Rahmen eines Updates können mit den `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und neue Bereitstellungsziele angegeben werden, wodurch neue Stack-Instances erstellt werden.

Im Rahmen eines Updates löschen die `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und keine Stack-Sets, Instanzen oder Ressourcen. Wenn die Aktion einen Stack aktualisiert, aber nicht alle zu aktualisierenden Instanzen angibt, werden die Instanzen, die nicht für die Aktualisierung angegeben wurden, aus dem Update entfernt und auf den Status von gesetzt`OUTDATED`.

Während einer Bereitstellung können Stack-Instances auch den Status anzeigen, `OUTDATED` ob die Bereitstellung für Instances fehlgeschlagen ist.

Wie strukturiert man StackSets Aktionen in einer Pipeline

Als bewährte Methode sollten Sie Ihre Pipeline so aufbauen, dass das Stack-Set erstellt und zunächst für eine Teilmenge oder eine einzelne Instanz bereitgestellt wird. Nachdem Sie Ihre Bereitstellung getestet und das generierte Stack-Set angesehen haben, fügen Sie die `CloudFormationStackInstances` Aktion hinzu, sodass die verbleibenden Instanzen erstellt und aktualisiert werden.

Verwenden Sie die Konsole oder die CLI, um die empfohlene Pipeline-Struktur wie folgt zu erstellen:

1. Erstellen Sie eine Pipeline mit einer Quellaktion (erforderlich) und der `CloudFormationStackSet` Aktion als Bereitstellungsaktion. Führen Sie Ihre Pipeline aus.
2. Wenn Ihre Pipeline zum ersten Mal ausgeführt wird, erstellt die `CloudFormationStackSet` Aktion Ihr Stack-Set und mindestens eine erste Instanz. Überprüfen Sie die Erstellung des Stack-Sets und die Bereitstellung auf Ihrer ersten Instance. Bei der ersten Erstellung eines Stack-Sets für das Konto Account-A, bei dem `us-east-1` es sich um die angegebene Region handelt, wird die Stack-Instance beispielsweise mit dem Stack-Set erstellt:

Stack-Instanz	Region	Status
StackInstanceID-1	us-east-1	CURRENT

3. Bearbeiten Sie Ihre Pipeline, um sie `CloudFormationStackInstances` als zweite Bereitstellungsaktion zum Erstellen/Aktualisieren von Stack-Instances für die von Ihnen angegebenen Ziele hinzuzufügen. Beispielsweise werden bei der Erstellung von Stack-Instanzen für ein Konto, `Account-A` in dem die `eu-central-1` Regionen `us-east-2` und die Regionen angegeben sind, die verbleibenden Stack-Instances erstellt und die ursprüngliche Instanz bleibt wie folgt aktualisiert:

Stack-Instanz	Region	Status
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	CURRENT
StackInstanceID-3	eu-central-1	CURRENT

4. Führen Sie Ihre Pipeline nach Bedarf aus, um Ihr Stack-Set zu aktualisieren und Stack-Instances zu aktualisieren oder zu erstellen.

Wenn Sie ein Stack-Update initiieren, bei dem Sie Bereitstellungsziele aus der Aktionskonfiguration entfernt haben, werden die Stack-Instances, die nicht für das Update vorgesehen waren, aus der Bereitstellung entfernt und erhalten den Status VERALTET. Bei einem Stack-Instance-Update für ein Konto, `Account-A` bei dem die `us-east-2` Region aus der Aktionskonfiguration entfernt wurde, werden beispielsweise die verbleibenden Stack-Instances erstellt und die entfernte Instanz wird wie folgt auf VERALTET gesetzt:

Stack-Instanz	Region	Status
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	VERALTET
StackInstanceID-3	eu-central-1	CURRENT

Weitere Informationen zu bewährten Methoden für die Bereitstellung von Stack-Sets finden Sie unter Bewährte Methoden für StackSets im AWS CloudFormation Benutzerhandbuch.

`CloudFormationStackSet`-Aktion

Diese Aktion erstellt oder aktualisiert ein Stack-Set anhand der Vorlage, die im Quellverzeichnis der Pipeline gespeichert ist.

Nachdem Sie ein Stack-Set definiert haben, können Sie Stacks in den in den Konfigurationsparametern angegebenen Zielkonten und Regionen erstellen, aktualisieren oder löschen. Beim Erstellen, Aktualisieren und Löschen von Stacks können Sie weitere Einstellungen angeben, z. B. die Reihenfolge der Regionen für auszuführende Operationen, den Prozentsatz der Fehlertoleranz, ab dem Stackoperationen beendet werden, und die Anzahl der Konten, in denen Operationen an Stacks gleichzeitig ausgeführt werden.

Ein Stack-Set ist eine regionale Ressource. Wenn Sie ein Stack-Set in einer AWS Region erstellen, können Sie von anderen Regionen aus nicht darauf zugreifen.

Wenn diese Aktion als Aktualisierungsaktion für das Stack-Set verwendet wird, sind Aktualisierungen des Stacks nur zulässig, wenn mindestens eine Stack-Instance bereitgestellt wird.

Themen

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `CloudFormationStackSet`
- Version: `1`

Konfigurationsparameter

StackSetName

Erforderlich: Ja

Der Name, der dem Stack-Set zugeordnet werden soll. Dieser Name muss in der Region, in der er erstellt wurde, eindeutig sein.

Der Name darf nur alphanumerische Zeichen und Bindestriche enthalten. Er muss mit einem alphabetischen Zeichen beginnen und maximal 128 Zeichen lang sein.

Beschreibung

Erforderlich: Nein

Eine Beschreibung des StackSets. Sie können dies verwenden, um den Zweck des Stack-Sets oder andere relevante Informationen zu beschreiben.

TemplatePath

Erforderlich: Ja

Der Speicherort der Vorlage, die die Ressourcen im Stack-Set definiert. Dies muss auf eine Vorlage mit einer maximalen Größe von 460.800 Byte verweisen.

Geben Sie den Pfad zum Quellartefaktnamen und zur Vorlagendatei im Format ein`"InputArtifactName::TemplateFileName"`, wie im folgenden Beispiel gezeigt.

```
SourceArtifact::template.txt
```

Parameter

Erforderlich: Nein

Eine Liste von Vorlagenparametern für Ihr Stack-Set, die während einer Bereitstellung aktualisiert werden.

Sie können Parameter als wörtliche Liste oder als Dateipfad angeben:

- Sie können Parameter im folgenden Kurzsyntaxformat eingeben:.
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`

Weitere Informationen zu diesen Datentypen finden Sie unter. Datentypen von Vorlagenparametern

Das folgende Beispiel zeigt einen Parameter, der `BucketName` mit dem Wert my- benannt ist`bucket`.

```
ParameterKey=BucketName,ParameterValue=my-bucket
```

Das folgende Beispiel zeigt einen Eintrag mit mehreren Parametern:

```
ParameterKey=BucketName,ParameterValue=my-bucket
ParameterKey=Asset1,ParameterValue=true
ParameterKey=Asset2,ParameterValue=true
```

- Sie können den Speicherort der Datei eingeben, die eine Liste der im Format eingegebenen Überschreibungen von Vorlagenparametern enthält`"InputArtifactName::ParametersFileName"`, wie im folgenden Beispiel gezeigt.

```
SourceArtifact::parameters.txt
```

Das folgende Beispiel zeigt den Dateiinhalt für`parameters.txt`.

```
[
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    },
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    }
]
```

Funktionen

Erforderlich: Nein

Zeigt an, dass die Vorlage je nach Ressourcentyp in der Vorlage Ressourcen erstellen und aktualisieren kann.

Sie müssen diese Eigenschaft verwenden, wenn Ihre Stack-Vorlage IAM-Ressourcen enthält oder wenn Sie einen Stack direkt aus einer Vorlage erstellen, die Makros enthält. Damit die AWS CloudFormation Aktion auf diese Weise erfolgreich ausgeführt werden kann, müssen Sie eine der folgenden Funktionen verwenden:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`

Sie können mehr als eine Fähigkeit angeben, indem Sie ein Komma und keine Leerzeichen zwischen den Funktionen verwenden. Das Beispiel in Beispiel für CloudFormationStackSeteine Aktionskonfiguration zeigt einen Eintrag mit mehreren Funktionen.

PermissionModel

Erforderlich: Nein

Legt fest, wie IAM-Rollen erstellt und verwaltet werden. Wenn das Feld nicht angegeben ist, wird der Standard verwendet. Weitere Informationen finden Sie unter Berechtigungsmodelle für Stack-Set-Operationen.

Gültige Werte für sind:

- `SELF_MANAGED`(Standard): Sie müssen Administrator- und Ausführungsrollen für die Bereitstellung auf Zielkonten erstellen.
- `SERVICE_MANAGED`: erstellt AWS CloudFormation StackSets automatisch die IAM-Rollen, die für die Bereitstellung auf Konten erforderlich sind, die von AWS Organizations verwaltet werden. Dazu muss ein Konto Mitglied einer Organisation sein.

> **Note**
>
> Dieser Parameter kann nur geändert werden, wenn keine Stack-Instances im Stack-Set vorhanden sind.

AdministrationRoleArn

> **Note**
>
> Da AWS CloudFormation StackSets Operationen über mehrere Konten hinweg ausgeführt werden, müssen Sie die erforderlichen Berechtigungen für diese Konten definieren, bevor Sie das Stack-Set erstellen können.

Erforderlich: Nein

> **Note**
>
> Dieser Parameter ist optional für das SELF_MANAGED-Berechtigungsmodell und wird nicht für das SERVICE_MANAGED-Berechtigungsmodell verwendet.

Der ARN der IAM-Rolle im Administratorkonto, das zur Ausführung von Stack-Set-Vorgängen verwendet wird.

Der Name kann alphanumerische Zeichen und eines der folgenden Zeichen enthalten: _ +=, .@- und keine Leerzeichen. Beim Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden. Dieser Rollenname muss eine Mindestlänge von 20 Zeichen und eine maximale Länge von 2048 Zeichen haben. Rollennamen müssen innerhalb des Kontos eindeutig sein. Der hier angegebene Rollenname muss ein vorhandener Rollenname sein. Wenn Sie den Rollennamen nicht angeben, ist er auf gesetzt AWSCloudFormationStackSetAdministrationRole. Wenn Sie angeben ServiceManaged, dürfen Sie keinen Rollennamen definieren.

ExecutionRoleName

> **Note**
>
> Da AWS CloudFormation StackSets Operationen über mehrere Konten hinweg ausgeführt werden, müssen Sie die erforderlichen Berechtigungen für diese Konten definieren, bevor Sie das Stack-Set erstellen können.

Erforderlich: Nein

> **Note**
>
> Dieser Parameter ist optional für das SELF_MANAGED-Berechtigungsmodell und wird nicht für das SERVICE_MANAGED-Berechtigungsmodell verwendet.

Der Name der IAM-Rolle in den Zielkonten, die zur Ausführung von Stack-Set-Vorgängen verwendet wird. Der Name kann alphanumerische Zeichen und eines der folgenden Zeichen enthalten: _+=, .@- und keine Leerzeichen. Beim Namen wird nicht zwischen

Groß- und Kleinschreibung unterschieden. Dieser Rollenname muss eine Mindestlänge von 1 Zeichen und eine maximale Länge von 64 Zeichen haben. Rollennamen müssen innerhalb des Kontos eindeutig sein. Der hier angegebene Rollenname muss ein vorhandener Rollenname sein. Geben Sie diese Rolle nicht an, wenn Sie benutzerdefinierte Ausführungsrollen verwenden. Wenn Sie den Rollennamen nicht angeben, ist er auf gesetzt`AWSCloudFormationStackSetExecutionRole`. Wenn Sie Service_Managed auf true setzen, dürfen Sie keinen Rollennamen definieren.

OrganizationsAutoDeployment

Erforderlich: Nein

> **Note**
>
> Dieser Parameter ist optional für das SERVICE_MANAGED-Berechtigungsmodell und wird nicht für das SELF_MANAGED-Berechtigungsmodell verwendet.

Beschreibt, ob Konten, die einer Zielorganisation oder Organisationseinheit (OU) hinzugefügt werden, AWS CloudFormation StackSets automatisch für AWS Organizations bereitgestellt werden. Wenn `OrganizationsAutoDeployment` angegeben, geben Sie `DeploymentTargets` und `Regions` nicht an.

> **Note**
>
> Wenn keine Eingabe angegeben ist`OrganizationsAutoDeployment`, ist der Standardwert`Disabled`.

Gültige Werte für sind:

- `Enabled`. Erforderlich: Nein.

  StackSets stellt automatisch zusätzliche Stack-Instances für AWS Unternehmenskonten bereit, die einer Zielorganisation oder Organisationseinheit (OU) in den angegebenen Regionen hinzugefügt werden. Wenn ein Konto aus einer Zielorganisation oder Organisationseinheit entfernt wird, werden Stack-Instances aus dem Konto in den angegebenen Regionen AWS CloudFormation StackSets gelöscht.
- `Disabled`. Erforderlich: Nein.

StackSets stellt nicht automatisch zusätzliche Stack-Instances für AWS Organisationskonten bereit, die einer Zielorganisation oder Organisationseinheit (OU) in den angegebenen Regionen hinzugefügt werden.

- `EnabledWithStackRetention`. Erforderlich: Nein.

  Stack-Ressourcen bleiben erhalten, wenn ein Konto aus einer Zielorganisation oder Organisationseinheit entfernt wird.

DeploymentTargets

Erforderlich: Nein

> **Note**
>
> Für das Berechtigungsmodell SERVICE_MANAGED können Sie entweder die Stamm-ID der Organisation oder die IDs der Organisationseinheiten für Bereitstellungsziele angeben. Für das SELF_MANAGED-Berechtigungsmodell können Sie nur Konten angeben.

> **Note**
>
> Wenn dieser Parameter ausgewählt ist, müssen Sie auch Regionen auswählen.

Eine Liste von AWS Konten oder Organisationseinheiten-IDs, für die Stackset-Instanzen erstellt/aktualisiert werden sollen.

- Konten:

  Sie können Konten als wörtliche Liste oder als Dateipfad angeben:

  - Literal: Geben Sie Parameter im Kurzsyntaxformat ein`account_ID,account_ID`, wie im folgenden Beispiel gezeigt.

    ```
    111111222222,333333444444
    ```

  - Dateipfad: Der Speicherort der Datei, die eine Liste von AWS Konten enthält, für die Stackset-Instanzen erstellt/aktualisiert werden sollen, eingegeben im Format. `InputArtifactName::AccountsFileName` Wenn Sie den Dateipfad verwenden, um

entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat in JSON sein, wie im folgenden Beispiel gezeigt.

```
SourceArtifact::accounts.txt
```

Das folgende Beispiel zeigt den Dateiinhalt für`accounts.txt`.

```
[
    "111111222222"
]
```

Das folgende Beispiel zeigt den Dateiinhalt für den Fall`accounts.txt`, dass mehr als ein Konto aufgeführt wird:

```
[
    "111111222222","333333444444"
]
```

- OrganizationalUnitIds:

> **Note**
>
> Dieser Parameter ist optional für das SERVICE_MANAGED-Berechtigungsmodell und wird nicht für das SELF_MANAGED-Berechtigungsmodell verwendet. Verwenden Sie diesen Wert nicht, wenn Sie auswählen. OrganizationsAutoDeployment

Die AWS Organisationseinheiten, in denen die zugehörigen Stack-Instances aktualisiert werden sollen.

Sie können die IDs der Organisationseinheiten als wörtliche Liste oder als Dateipfad angeben:

- Literal: Geben Sie eine Reihe von durch Kommas getrennten Zeichenfolgen ein, wie im folgenden Beispiel gezeigt.

```
ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
```

- Dateipfad: Der Speicherort der Datei, die eine Liste enthält, OrganizationalUnitIds in der Stackset-Instanzen erstellt oder aktualisiert werden sollen. Wenn Sie den Dateipfad

verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat in JSON sein, wie im folgenden Beispiel gezeigt.

Geben Sie einen Pfad zur Datei im Format ein`InputArtifactName::OrganizationalUnitIdsFileName`.

```
SourceArtifact::OU-IDs.txt
```

Das folgende Beispiel zeigt den Dateiinhalt für`OU-IDs.txt`:

```
[
    "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
]
```

Regionen

Erforderlich: Nein

> **Note**
>
> Wenn dieser Parameter ausgewählt ist, müssen Sie auch auswählen DeploymentTargets.

Eine Liste von AWS Regionen, in denen Stackset-Instanzen erstellt oder aktualisiert werden. Regionen werden in der Reihenfolge aktualisiert, in der sie eingegeben wurden.

Geben Sie eine Liste gültiger AWS Regionen in dem Format ein`Region1,Region2`, wie im folgenden Beispiel gezeigt.

```
us-west-2,us-east-1
```

FailureTolerancePercentage

Erforderlich: Nein

Der Prozentsatz der Konten pro Region, bei denen dieser Stack-Vorgang fehlschlagen kann, bevor der Vorgang in dieser Region AWS CloudFormation beendet wird. Wenn der Vorgang in einer Region gestoppt wird, wird der Vorgang in nachfolgenden Regionen AWS CloudFormation nicht versucht. Bei der Berechnung der Anzahl der Konten auf der Grundlage des angegebenen Prozentsatzes wird auf die nächste ganze Zahl AWS CloudFormation abgerundet.

MaxConcurrentPercentage

Erforderlich: Nein

Der maximale Prozentsatz der Konten, auf denen dieser Vorgang gleichzeitig ausgeführt werden kann. Bei der Berechnung der Anzahl der Konten auf der Grundlage des angegebenen Prozentsatzes wird auf die nächste ganze Zahl AWS CloudFormation abgerundet. Wenn das Abrunden zu Null führen würde, wird AWS CloudFormation die Zahl stattdessen auf Eins gesetzt. Sie verwenden diese Einstellung zwar, um das Maximum anzugeben, aber bei großen Bereitstellungen kann die tatsächliche Anzahl der Konten, auf die gleichzeitig zugegriffen wird, aufgrund der Dienstdrosselung geringer sein.

RegionConcurrencyType

Erforderlich: Nein

Sie können angeben, ob das Stack-Set AWS-Regionen sequentiell oder parallel bereitgestellt werden soll, indem Sie den Bereitstellungsparameter Region Concurrency konfigurieren. Wenn die Regions-Parallelität so angegeben ist, dass Stacks über mehrere gleichzeitig AWS-Regionen bereitgestellt werden, kann dies zu kürzeren Gesamtbereitstellungszeiten führen.

- Parallel: Stack-Set-Bereitstellungen werden zur gleichen Zeit durchgeführt, sofern die Bereitstellungsfehler einer Region eine festgelegte Fehlertoleranz nicht überschreiten.
- Sequentiell: Stack-Set-Bereitstellungen werden nacheinander durchgeführt, sofern die Bereitstellungsfehler einer Region eine festgelegte Fehlertoleranz nicht überschreiten. Die sequenzielle Bereitstellung ist die Standardauswahl.

ConcurrencyMode

Erforderlich: Nein

Im Parallelitätsmodus können Sie wählen, wie sich die Parallelitätsebene bei Stack-Set-Vorgängen verhält, unabhängig davon, ob es sich um eine strikte oder eine weiche Fehlertoleranz handelt. Strikte Fehlertoleranz senkt die Bereitstellungsgeschwindigkeit bei fehlerhaften Stack-Set-Vorgängen, da die Parallelität bei jedem Fehler abnimmt. Bei Soft Failure Tolerance wird die Geschwindigkeit der Bereitstellung priorisiert und gleichzeitig die Sicherheitsfunktionen genutzt. AWS CloudFormation

- `STRICT_FAILURE_TOLERANCE`: Diese Option senkt dynamisch den Parallelitätsgrad, um sicherzustellen, dass die Anzahl ausgefallener Konten niemals eine bestimmte Fehlertoleranz überschreitet. Dies ist das Standardverhalten.

- `SOFT_FAILURE_TOLERANCE`: Diese Option entkoppelt die Fehlertoleranz von der tatsächlichen Parallelität. Dadurch können Stack-Set-Operationen unabhängig von der Anzahl der Fehler auf einer bestimmten Parallelitätsebene ausgeführt werden.

CallAs

Erforderlich: Nein

> **Note**
>
> Dieser Parameter ist für das `SERVICE_MANAGED` Berechtigungsmodell optional und wird nicht für das `SELF_MANAGED` Berechtigungsmodell verwendet.

Gibt an, ob Sie im Verwaltungskonto der Organisation oder als delegierter Administrator in einem Mitgliedskonto agieren.

> **Note**
>
> Wenn dieser Parameter auf gesetzt ist, stellen Sie sicher`DELEGATED_ADMIN`, dass die Pipeline-IAM-Rolle über die entsprechende Berechtigung verfügt`organizations:ListDelegatedAdministrators`. Andernfalls schlägt die Aktion während der Ausführung mit einem Fehler ähnlich dem folgenden fehl:`Account used is not a delegated administrator`.

- `SELF`: Bei der Stackset-Bereitstellung werden vom Dienst verwaltete Berechtigungen verwendet, während Sie mit dem Verwaltungskonto angemeldet sind.
- `DELEGATED_ADMIN`: Bei der Stackset-Bereitstellung werden vom Dienst verwaltete Berechtigungen verwendet, wenn Sie mit einem delegierten Administratorkonto angemeldet sind.

Input artifacts (Eingabeartefakte)

Sie müssen mindestens ein Eingabeartefakt, das die Vorlage für das Stack-Set enthält, in eine Aktion einbeziehen. `CloudFormationStackSet` Sie können weitere Eingabeartefakte für Listen mit Bereitstellungszielen, Konten und Parametern hinzufügen.

- Anzahl der Artefakte: `1 to 3`

- Beschreibung: Sie können Artefakte hinzufügen, um Folgendes bereitzustellen:
  - Die Stack-Vorlagendatei. (Weitere Informationen finden Sie unter dem `TemplatePath`-Parameter.)
  - Die Parameterdatei. (Weitere Informationen finden Sie unter dem `Parameters`-Parameter.)
  - Die Kontendatei. (Weitere Informationen finden Sie unter dem `DeploymentTargets`-Parameter.)

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Ausgabevariablen

Wenn Sie diese Aktion konfigurieren, erzeugt sie Variablen, auf die in der Aktionskonfiguration einer Downstream-Aktion in der Pipeline verwiesen werden kann. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

- StackSetId: Die ID des Stack-Sets.
- OperationId: Die ID der Stack-Set-Operation.

Weitere Informationen finden Sie unter Variablen.

Beispiel für CloudFormationStackSeteine Aktionskonfiguration

Die folgenden Beispiele zeigen die Aktionskonfiguration für die CloudFormationStackSetAktion.

Beispiel für das selbstverwaltete Berechtigungsmodell

Das folgende Beispiel zeigt eine CloudFormationStackSetAktion, bei der das eingegebene Bereitstellungsziel eine AWS Konto-ID ist.

YAML

```
Name: CreateStackSet
ActionTypeId:
```

```
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackSet
  Version: '1'
RunOrder: 1
Configuration:
  DeploymentTargets: '111111222222'
  FailureTolerancePercentage: '20'
  MaxConcurrentPercentage: '25'
  PermissionModel: SELF_MANAGED
  Regions: us-east-1
  StackSetName: my-stackset
  TemplatePath: 'SourceArtifact::template.json'
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: us-west-2
Namespace: DeployVariables
```

JSON

```
{
    "Name": "CreateStackSet",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackSet",
        "Version": "1"
    },
    "RunOrder": 1,
    "Configuration": {
        "DeploymentTargets": "111111222222",
        "FailureTolerancePercentage": "20",
        "MaxConcurrentPercentage": "25",
        "PermissionModel": "SELF_MANAGED",
        "Regions": "us-east-1",
        "StackSetName": "my-stackset",
        "TemplatePath": "SourceArtifact::template.json"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
```

```
        }
    ],
    "Region": "us-west-2",
    "Namespace": "DeployVariables"
}
```

Beispiel für das Modell der vom Service verwalteten Berechtigungen

Das folgende Beispiel zeigt eine CloudFormationStackSetAktion für das vom Service verwaltete Berechtigungsmodell, bei dem die Option für die auto Bereitstellung für AWS Organizations mit Stack-Aufbewahrung aktiviert ist.

YAML

```
Name: Deploy
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackSet
  Version: '1'
RunOrder: 1
Configuration:
  Capabilities: 'CAPABILITY_IAM,CAPABILITY_NAMED_IAM'
  OrganizationsAutoDeployment: EnabledWithStackRetention
  PermissionModel: SERVICE_MANAGED
  StackSetName: stacks-orgs
  TemplatePath: 'SourceArtifact::template.json'
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: eu-central-1
Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackSet",
        "Version": "1"
```

```
    },
    "RunOrder": 1,
    "Configuration": {
        "Capabilities": "CAPABILITY_IAM,CAPABILITY_NAMED_IAM",
        "OrganizationsAutoDeployment": "EnabledWithStackRetention",
        "PermissionModel": "SERVICE_MANAGED",
        "StackSetName": "stacks-orgs",
        "TemplatePath": "SourceArtifact::template.json"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "eu-central-1",
    "Namespace": "DeployVariables"
}
```

CloudFormationStackInstances-Aktion

Diese Aktion erstellt neue Instanzen und stellt Stack-Sets für bestimmte Instanzen bereit. Eine Stack-Instance ist eine Referenz auf einen Stack in einem Zielkonto innerhalb einer Region. Eine Stack-Instance kann auch ohne Stack existieren. Wenn die Stack-Erstellung beispielsweise nicht erfolgreich ist, zeigt die Stack-Instance den Grund für das Fehlschlagen der Stack-Erstellung an. Ein Stack-Instance ist nur einem Stack-Set zugeordnet.

Nach der ersten Erstellung eines Stack-Sets können Sie neue Stack-Instances hinzufügen, indem `CloudFormationStackInstances` Sie Vorlagenparameterwerte können auf Stack-Instance-Ebene bei der Erstellung oder Aktualisierung von Stack-Set-Instance-Vorgängen überschrieben werden.

Jedes Stack-Set hat eine Vorlage und einen Satz von Vorlagenparametern. Wenn Sie die Vorlage oder die Vorlagenparameter aktualisieren, aktualisieren Sie sie für den gesamten Satz. Dann werden alle Instanzstatus auf gesetzt, `OUTDATED` bis die Änderungen für diese Instanz bereitgestellt werden.

Um Parameterwerte für bestimmte Instanzen zu überschreiben, z. B. wenn die Vorlage einen Parameter für `stage` mit dem Wert von enthält`prod`, können Sie den Wert dieses Parameters auf `beta` oder `gamma` überschreiben.

Themen

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `CloudFormationStackInstances`
- Version: `1`

Konfigurationsparameter

StackSetName

Erforderlich: Ja

Der Name, der dem Stack-Set zugeordnet werden soll. Dieser Name muss in der Region, in der er erstellt wurde, eindeutig sein.

Der Name darf nur alphanumerische Zeichen und Bindestriche enthalten. Er muss mit einem alphabetischen Zeichen beginnen und maximal 128 Zeichen lang sein.

DeploymentTargets

Erforderlich: Nein

> **Note**
>
> Für das SERVICE_MANAGED-Berechtigungsmodell können Sie entweder die Stamm-ID der Organisation oder die IDs der Organisationseinheiten für Bereitstellungsziele angeben. Für das SELF_MANAGED-Berechtigungsmodell können Sie nur Konten angeben.

> **Note**
>
> Wenn dieser Parameter ausgewählt ist, müssen Sie auch Regionen auswählen.

Eine Liste von AWS Konten oder Organisationseinheiten-IDs, für die Stackset-Instanzen erstellt/aktualisiert werden sollen.

- Konten:

  Sie können Konten als wörtliche Liste oder als Dateipfad angeben:

  - Literal: Geben Sie Parameter im Kurzsyntaxformat ein`account_ID,account_ID`, wie im folgenden Beispiel gezeigt.

    ```
    111111222222,333333444444
    ```

  - Dateipfad: Der Speicherort der Datei, die eine Liste von AWS Konten enthält, für die Stackset-Instanzen erstellt/aktualisiert werden sollen, eingegeben im Format. `InputArtifactName::AccountsFileName` Wenn Sie den Dateipfad verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat in JSON sein, wie im folgenden Beispiel gezeigt.

    ```
    SourceArtifact::accounts.txt
    ```

    Das folgende Beispiel zeigt den Dateiinhalt für`accounts.txt`:

    ```
    [
        "111111222222"
    ]
    ```

    Das folgende Beispiel zeigt den Dateiinhalt für den Fall`accounts.txt`, dass mehr als ein Konto aufgeführt wird:

    ```
    [
        "111111222222","333333444444"
    ]
    ```

- OrganizationalUnitIds:

> **Note**
>
> Dieser Parameter ist optional für das SERVICE_MANAGED-Berechtigungsmodell und wird nicht für das SELF_MANAGED-Berechtigungsmodell verwendet. Verwenden Sie diesen Wert nicht, wenn Sie auswählen. OrganizationsAutoDeployment

Die AWS Organisationseinheiten, in denen die zugehörigen Stack-Instances aktualisiert werden sollen.

Sie können die IDs der Organisationseinheiten als wörtliche Liste oder als Dateipfad angeben.

- Literal: Geben Sie eine Reihe von durch Kommas getrennten Zeichenfolgen ein, wie im folgenden Beispiel gezeigt.

  ```
  ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
  ```

- Dateipfad: Der Speicherort der Datei, die eine Liste enthält, OrganizationalUnitIds in der Stackset-Instanzen erstellt oder aktualisiert werden sollen. Wenn Sie den Dateipfad verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat in JSON sein, wie im folgenden Beispiel gezeigt.

  Geben Sie einen Pfad zur Datei im Format einInputArtifactName::OrganizationalUnitIdsFileName.

  ```
  SourceArtifact::OU-IDs.txt
  ```

  Das folgende Beispiel zeigt den Dateiinhalt für0U-IDs.txt:

  ```
  [
      "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
  ]
  ```

Regionen

Erforderlich: Ja

> **Note**
>
> Wenn dieser Parameter ausgewählt ist, müssen Sie auch auswählen DeploymentTargets.

Eine Liste von AWS Regionen, in denen Stackset-Instanzen erstellt oder aktualisiert werden. Regionen werden in der Reihenfolge aktualisiert, in der sie eingegeben wurden.

Geben Sie eine Liste gültiger AWS Regionen im Format: ein`Region1,Region2`, wie im folgenden Beispiel gezeigt.

```
us-west-2,us-east-1
```

ParameterOverrides

Erforderlich: Nein

Eine Liste von Stack-Set-Parametern, die Sie in den ausgewählten Stack-Instances überschreiben möchten. Überschriebene Parameterwerte werden auf alle Stack-Instances in den angegebenen Konten und Regionen angewendet.

Sie können Parameter als Literalliste oder als Dateipfad angeben:

- Sie können Parameter im folgenden Kurzsyntaxformat eingeben:. `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV` `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV` Weitere Informationen zu diesen Datentypen finden Sie unter. Datentypen von Vorlagenparametern

  Das folgende Beispiel zeigt einen Parameter, der `BucketName` mit dem Wert my- benannt ist`bucket`.

  ```
  ParameterKey=BucketName,ParameterValue=my-bucket
  ```

  Das folgende Beispiel zeigt einen Eintrag mit mehreren Parametern.

  ```
  ParameterKey=BucketName,ParameterValue=my-bucket
  ParameterKey=Asset1,ParameterValue=true
  ```

```
  ParameterKey=Asset2,ParameterValue=true
```

- Sie können den Speicherort der Datei eingeben, die eine Liste der im Format eingegebenen Überschreibungen von Vorlagenparametern enthält`InputArtifactName::ParameterOverridessFileName`, wie im folgenden Beispiel gezeigt.

  ```
  SourceArtifact::parameter-overrides.txt
  ```

  Das folgende Beispiel zeigt den Dateiinhalt für`parameter-overrides.txt`.

  ```
  [
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      },
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      }
  ]
  ```

FailureTolerancePercentage

Erforderlich: Nein

Der Prozentsatz der Konten pro Region, für die dieser Stack-Vorgang fehlschlagen kann, bevor der Vorgang in dieser Region AWS CloudFormation beendet wird. Wenn der Vorgang in einer Region gestoppt wird, wird der Vorgang in nachfolgenden Regionen AWS CloudFormation nicht versucht. Bei der Berechnung der Anzahl der Konten auf der Grundlage des angegebenen Prozentsatzes wird auf die nächste ganze Zahl AWS CloudFormation abgerundet.

MaxConcurrentPercentage

Erforderlich: Nein

Der maximale Prozentsatz der Konten, für die dieser Vorgang gleichzeitig ausgeführt werden soll. Bei der Berechnung der Anzahl der Konten auf der Grundlage des angegebenen Prozentsatzes wird auf die nächste ganze Zahl AWS CloudFormation abgerundet. Wenn das Abrunden zu Null führen würde, wird AWS CloudFormation die Zahl stattdessen auf Eins gesetzt. Sie geben zwar das Maximum an, aber bei großen Bereitstellungen kann die tatsächliche Anzahl der Konten, auf die gleichzeitig zugegriffen wird, aufgrund der Dienstdrosselung geringer sein.

RegionConcurrencyType

Erforderlich: Nein

Sie können angeben, ob das Stack-Set AWS-Regionen sequentiell oder parallel bereitgestellt werden soll, indem Sie den Bereitstellungsparameter Region Concurrency konfigurieren. Wenn die Regions-Parallelität so angegeben ist, dass Stacks über mehrere gleichzeitig AWS-Regionen bereitgestellt werden, kann dies zu kürzeren Gesamtbereitstellungszeiten führen.

- Parallel: Stack-Set-Bereitstellungen werden zur gleichen Zeit durchgeführt, sofern die Bereitstellungsfehler einer Region eine festgelegte Fehlertoleranz nicht überschreiten.
- Sequentiell: Stack-Set-Bereitstellungen werden nacheinander durchgeführt, sofern die Bereitstellungsfehler einer Region eine festgelegte Fehlertoleranz nicht überschreiten. Die sequenzielle Bereitstellung ist die Standardauswahl.

ConcurrencyMode

Erforderlich: Nein

Im Parallelitätsmodus können Sie wählen, wie sich die Parallelitätsebene bei Stack-Set-Vorgängen verhält, unabhängig davon, ob es sich um eine strikte oder eine weiche Fehlertoleranz handelt. Strikte Fehlertoleranz senkt die Bereitstellungsgeschwindigkeit bei fehlerhaften Stack-Set-Vorgängen, da die Parallelität bei jedem Fehler abnimmt. Bei Soft Failure Tolerance wird die Geschwindigkeit der Bereitstellung priorisiert und gleichzeitig die Sicherheitsfunktionen genutzt. AWS CloudFormation

- `STRICT_FAILURE_TOLERANCE`: Diese Option senkt dynamisch den Parallelitätsgrad, um sicherzustellen, dass die Anzahl ausgefallener Konten niemals eine bestimmte Fehlertoleranz überschreitet. Dies ist das Standardverhalten.
- `SOFT_FAILURE_TOLERANCE`: Diese Option entkoppelt die Fehlertoleranz von der tatsächlichen Parallelität. Dadurch können Stack-Set-Operationen unabhängig von der Anzahl der Fehler auf einer bestimmten Parallelitätsebene ausgeführt werden.

CallAs

Erforderlich: Nein

> **Note**
>
> Dieser Parameter ist für das `SERVICE_MANAGED` Berechtigungsmodell optional und wird nicht für das `SELF_MANAGED` Berechtigungsmodell verwendet.

Gibt an, ob Sie im Verwaltungskonto der Organisation oder als delegierter Administrator in einem Mitgliedskonto agieren.

> **Note**
>
> Wenn dieser Parameter auf gesetzt ist, stellen Sie sicher`DELEGATED_ADMIN`, dass die Pipeline-IAM-Rolle über die entsprechende Berechtigung verfügt`organizations:ListDelegatedAdministrators`. Andernfalls schlägt die Aktion während der Ausführung mit einem Fehler ähnlich dem folgenden fehl:`Account used is not a delegated administrator`.

- `SELF`: Bei der Stackset-Bereitstellung werden vom Dienst verwaltete Berechtigungen verwendet, während Sie mit dem Verwaltungskonto angemeldet sind.
- `DELEGATED_ADMIN`: Bei der Stackset-Bereitstellung werden vom Dienst verwaltete Berechtigungen verwendet, wenn Sie mit einem delegierten Administratorkonto angemeldet sind.

Input artifacts (Eingabeartefakte)

`CloudFormationStackInstances`kann Artefakte enthalten, die Bereitstellungsziele und -parameter auflisten.

- Anzahl der Artefakte: `0 to 2`
- Beschreibung: Als Eingabe akzeptiert die Stack-Set-Aktion optional Artefakte für folgende Zwecke:
  - Um die zu verwendende Parameterdatei bereitzustellen. (Weitere Informationen finden Sie unter dem `ParameterOverrides`-Parameter.)
  - Um die zu verwendende Zielkontendatei bereitzustellen. (Weitere Informationen finden Sie unter dem `DeploymentTargets`-Parameter.)

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

- StackSetId: Die ID des Stack-Sets.
- OperationId: Die ID der Stack-Set-Operation.

Weitere Informationen finden Sie unter Variablen.

Beispielaktionskonfiguration

Die folgenden Beispiele zeigen die Aktionskonfiguration für die CloudFormationStackInstancesAktion.

Beispiel für das selbstverwaltete Berechtigungsmodell

Das folgende Beispiel zeigt eine CloudFormationStackInstancesAktion, bei der das eingegebene Bereitstellungsziel eine AWS-Konto ID 111111222222 ist.

YAML

```
Name: my-instances
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackInstances
  Version: '1'
RunOrder: 2
Configuration:
  DeploymentTargets: '111111222222'
  Regions: 'us-east-1,us-east-2,us-west-1,us-west-2'
  StackSetName: my-stackset
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: us-west-2
```

JSON

```
{
```

```
    "Name": "my-instances",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackInstances",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "DeploymentTargets": "111111222222",
        "Regions": "us-east-1,us-east-2,us-west-1,us-west-2",
        "StackSetName": "my-stackset"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "us-west-2"
}
```

Beispiel für das Modell der vom Service verwalteten Berechtigungen

Das folgende Beispiel zeigt eine CloudFormationStackInstancesAktion für das Modell mit vom Dienst verwalteten Berechtigungen, bei dem das Bereitstellungsziel eine AWS Organisationseinheit-ID `ou-1111-1example` einer Organisation ist.

YAML

```
Name: Instances
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackInstances
  Version: '1'
RunOrder: 2
Configuration:
  DeploymentTargets: ou-1111-1example
  Regions: us-east-1
  StackSetName: my-stackset
OutputArtifacts: []
```

```
InputArtifacts:
  - Name: SourceArtifact
Region: eu-central-1
```

JSON

```
{
    "Name": "Instances",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackInstances",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "DeploymentTargets": "ou-1111-1example",
        "Regions": "us-east-1",
        "StackSetName": "my-stackset"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "eu-central-1"
}
```

Berechtigungsmodelle für Stack-Set-Operationen

Da AWS CloudFormation StackSets Operationen über mehrere Konten hinweg ausgeführt werden, müssen Sie die erforderlichen Berechtigungen für diese Konten definieren, bevor Sie das Stack-Set erstellen können. Sie können Berechtigungen über selbstverwaltete oder vom Service verwaltete Berechtigungen definieren.

Mit selbstverwalteten Berechtigungen erstellen Sie die beiden IAM-Rollen, die für erforderlich sind: eine Administratorrolle wie die AWSCloudFormationStackSetAdministrationRole in dem Konto, in dem Sie das Stack-Set definieren, und eine Ausführungsrolle wie die AWSCloudFormationStackSetExecutionRole in jedem der Konten, in denen Sie Stackset-Instanzen bereitstellen. StackSets Mithilfe dieses Berechtigungsmodells StackSets kann die Bereitstellung für

jedes AWS Konto erfolgen, für das der Benutzer über die Berechtigung zum Erstellen einer IAM-Rolle verfügt. Weitere Informationen finden Sie im AWS CloudFormation Benutzerhandbuch unter Gewähren von selbstverwalteten Berechtigungen.

> **Note**
>
> Da AWS CloudFormation StackSets Operationen über mehrere Konten hinweg ausgeführt werden, müssen Sie die erforderlichen Berechtigungen für diese Konten definieren, bevor Sie das Stack-Set erstellen können.

Mit vom Service verwalteten Berechtigungen können Sie Stack-Instances für Konten bereitstellen, die von AWS Organizations verwaltet werden. Mit diesem Berechtigungsmodell müssen Sie nicht die erforderlichen IAM-Rollen erstellen, da die IAM-Rollen in Ihrem Namen StackSets erstellt werden. Mit diesem Modell können Sie auch automatische Bereitstellungen für Konten aktivieren, die der Organisation in future hinzugefügt werden. Weitere Informationen finden Sie unter Aktivieren des vertrauenswürdigen Zugriffs mit AWS Organizations im AWS CloudFormation Benutzerhandbuch.

Datentypen von Vorlagenparametern

Zu den bei Stack-Set-Operationen verwendeten Vorlagenparametern gehören die folgenden Datentypen. Weitere Informationen finden Sie unter DescribeStackSet.

ParameterKey

- Beschreibung: Der dem Parameter zugeordnete Schlüssel. Wenn Sie keinen Schlüssel und keinen Wert für einen bestimmten Parameter angeben, wird der Standardwert AWS CloudFormation verwendet, der in der Vorlage angegeben ist.
- Beispiel:

```
"ParameterKey=BucketName,ParameterValue=my-bucket"
```

ParameterValue

- Beschreibung: Der dem Parameter zugeordnete Eingabewert.
- Beispiel:

```
"ParameterKey=BucketName,ParameterValue=my-bucket"
```

UsePreviousValue

- Verwenden Sie während einer Stack-Aktualisierung den vorhandenen Parameterwert, den der Stack für einen bestimmten Parameterschlüssel verwendet. Wenn Sie angeben`true`, geben Sie keinen Parameterwert an.
- Beispiel:

```
"ParameterKey=Asset1,UsePreviousValue=true"
```

Jeder Stacksatz hat eine Vorlage und einen Satz von Vorlagenparametern. Wenn Sie die Vorlage oder die Vorlagenparameter aktualisieren, aktualisieren Sie sie für den gesamten Satz. Dann werden alle Instanzstatus auf VERALTET gesetzt, bis die Änderungen für diese Instanz bereitgestellt werden.

Um Parameterwerte für bestimmte Instanzen zu überschreiben, z. B. wenn die Vorlage einen Parameter für `stage` mit dem Wert von enthält`prod`, können Sie den Wert dieses Parameters auf `beta` oder `gamma` überschreiben.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Parametertypen — Dieses Referenzkapitel im AWS CloudFormation Benutzerhandbuch enthält weitere Beschreibungen und Beispiele für CloudFormation Vorlagenparameter.
- Bewährte Methoden — Weitere Informationen zu bewährten Methoden für die Bereitstellung von Stack-Sets finden Sie https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-bestpractices.html im AWS CloudFormation Benutzerhandbuch.
- AWS CloudFormation API-Referenz — Weitere Informationen zu den bei Stack-Set-Vorgängen verwendeten Parametern finden Sie in der AWS CloudFormation API-Referenz auf die folgenden CloudFormation Aktionen:

  - Die CreateStackSetAktion erstellt ein Stack-Set.
  - Die UpdateStackSetAktion aktualisiert das Stack-Set und die zugehörigen Stack-Instances in den angegebenen Konten und Regionen. Selbst wenn der durch die Aktualisierung des Stack-Sets erzeugte Stack-Set-Vorgang fehlschlägt (ganz oder teilweise, unter oder über einer bestimmten Fehlertoleranz), wird das Stack-Set mit diesen Änderungen aktualisiert.

Nachfolgende CreateStackInstances Aufrufe des angegebenen Stack-Sets verwenden das aktualisierte Stack-Set.

- Die CreateStackInstancesAktion erstellt eine Stack-Instanz für alle angegebenen Regionen innerhalb aller angegebenen Konten bei einem selbstverwalteten Berechtigungsmodell oder innerhalb aller angegebenen Bereitstellungsziele bei einem vom Service verwalteten Berechtigungsmodell. Sie können die Parameter für die durch diese Aktion erstellten Instanzen überschreiben. Wenn die Instanzen bereits existieren, werden CreateStackInstances Aufrufe UpdateStackInstances mit denselben Eingabeparametern ausgeführt. Wenn Sie diese Aktion verwenden, um Instanzen zu erstellen, ändert sich dadurch nicht der Status anderer Stack-Instances.
- Durch die UpdateStackInstancesAktion werden Stack-Instances mit dem Stack-Set für alle angegebenen Regionen innerhalb aller angegebenen Konten bei einem selbstverwalteten Berechtigungsmodell oder innerhalb aller angegebenen Bereitstellungsziele bei einem vom Service verwalteten Berechtigungsmodell auf den neuesten Stand gebracht. Sie können die Parameter für die durch diese Aktion aktualisierten Instanzen überschreiben. Wenn Sie diese Aktion verwenden, um eine Teilmenge von Instances zu aktualisieren, ändert sich dadurch nicht der Status anderer Stack-Instances.
- Die DescribeStackSetOperationAktion gibt die Beschreibung des angegebenen Stack-Set-Vorgangs zurück.
- Die DescribeStackSetAktion gibt die Beschreibung des angegebenen Stack-Sets zurück.

AWS CodeBuild

Ermöglicht die Ausführung von Builds und Tests als Teil Ihrer Pipeline. Wenn Sie eine CodeBuild Build- oder Testaktion ausführen, werden die in der Build-Spezifikation angegebenen Befehle innerhalb eines CodeBuild Containers ausgeführt. Alle Artefakte, die als Eingabeartefakte für eine CodeBuild Aktion angegeben sind, sind innerhalb des Containers verfügbar, in dem die Befehle ausgeführt werden. CodeBuild kann entweder eine Build- oder eine Testaktion bereitstellen. Weitere Informationen finden Sie im AWS CodeBuild -Benutzerhandbuch.

Wenn Sie den CodePipeline Assistenten in der Konsole verwenden, um ein Build-Projekt zu erstellen, zeigt das CodeBuild Build-Projekt den Quellanbieter an CodePipeline. Wenn Sie ein Build-Projekt in der CodeBuild Konsole erstellen, können Sie nicht den Quellanbieter angeben CodePipeline , aber wenn Sie die Build-Aktion zu Ihrer Pipeline hinzufügen, wird die Quelle in der CodeBuild Konsole angepasst. Weitere Informationen finden Sie ProjectSourcein der AWS CodeBuild API-Referenz.

Themen

Aktionstyp

- Kategorie: `Build` oder `Test`
- Eigentümer: `AWS`
- Anbieter: `CodeBuild`
- Version: `1`

Konfigurationsparameter

ProjectName

Erforderlich: Ja

`ProjectName`ist der Name des Build-Projekts in CodeBuild.

PrimarySource

Required: Conditional

Der Wert des `PrimarySource` Parameters muss der Name eines der Eingabeartefakte für die Aktion sein. CodeBuild sucht nach der Build-Spezifikationsdatei und führt die Build-Spezifikationsbefehle in dem Verzeichnis aus, das die entpackte Version dieses Artefakts enthält.

Dieser Parameter ist erforderlich, wenn mehrere Eingabeartefakte für eine Aktion angegeben sind. CodeBuild Wenn nur ein Quellartefakt für die Aktion vorhanden ist, verwendet das `PrimarySource`-Artefakt standardmäßig dieses Artefakt.

BatchEnabled

Erforderlich: Nein

Der boolesche Wert des `BatchEnabled` Parameters ermöglicht es der Aktion, mehrere Builds in derselben Build-Ausführung auszuführen.

Wenn diese Option aktiviert ist, ist die `CombineArtifacts` Option verfügbar.

Pipeline-Beispiele mit aktivierten Batch-Builds finden Sie unter CodePipeline Integration mit CodeBuild und Batch-Builds.

CombineArtifacts

Erforderlich: Nein

Der boolesche Wert des `CombineArtifacts` Parameters kombiniert alle Build-Artefakte aus einem Batch-Build in einer einzigen Artefaktdatei für die Build-Aktion.

Um diese Option verwenden zu können, muss der `BatchEnabled` Parameter aktiviert sein.

EnvironmentVariables

Erforderlich: Nein

Der Wert dieses Parameters wird verwendet, um Umgebungsvariablen für die CodeBuild Aktion in Ihrer Pipeline festzulegen. Der Wert für den Parameter `EnvironmentVariables` hat das Format eines JSON-Arrays von Umgebungsvariablenobjekten. Einen Beispielparameter finden Sie unter Aktionsdeklaration (CodeBuild-Beispiel).

Jedes Objekt besteht aus drei Teilen. Alle Teile sind Zeichenfolgen:

- `name`: Der Name oder Schlüssel der Umgebungsvariable.
- `value`: Der Wert der Umgebungsvariable. Wenn Sie den `SECRETS_MANAGER` Typ `PARAMETER_STORE` oder verwenden, muss dieser Wert der Name eines Parameters sein, den Sie bereits im AWS Systems Manager Parameter Store gespeichert haben, bzw. eines Secrets, das Sie bereits in AWS Secrets Manager gespeichert haben.

> **Note**
>
> Es wird dringend davon abgeraten, Umgebungsvariablen zum Speichern sensibler Werte, insbesondere von AWS Anmeldeinformationen, zu verwenden. Wenn Sie die

CodeBuild Konsole oder AWS CLI verwenden, werden Umgebungsvariablen im Klartext angezeigt. Wenn es sich um vertrauliche Werte handelt, sollten Sie stattdessen den Typ `SECRETS_MANAGER` verwenden.

- `type`: (Optional) Der Typ der Umgebungsvariablen. Gültige Werte sind `PARAMETER_STORE`, `SECRETS_MANAGER` oder `PLAINTEXT`. Wenn keine Angabe gemacht wird, gilt der Standardwert `PLAINTEXT`.

> **Note**
>
> Wenn Sie `namevalue`, und `type` für Ihre Umgebungsvariablenkonfiguration eingeben, insbesondere wenn die Umgebungsvariable die Syntax der CodePipeline Ausgabevariablen enthält, sollten Sie die 1000-Zeichen-Grenze für das Wertefeld der Konfiguration nicht überschreiten. Ein Validierungsfehler wird zurückgegeben, wenn dieser Grenzwert überschritten wird.

Weitere Informationen finden Sie EnvironmentVariablein der AWS CodeBuild API-Referenz. Ein Beispiel für eine CodeBuild Aktion mit einer Umgebungsvariablen, die in den Namen der GitHub Verzweigung aufgelöst wird, finden Sie unterBeispiel: Verwenden Sie eine BranchName Variable mit CodeBuild Umgebungsvariablen.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1 to 5`
- Beschreibung: CodeBuild sucht nach der Build-Spezifikationsdatei und führt die Build-Spezifikationsbefehle aus dem Verzeichnis des primären Quellartefakts aus. Wenn mehr als eine Eingabequelle für die CodeBuild Aktion angegeben wurde, muss dieses Artefakt mithilfe des `PrimarySource` Aktionskonfigurationsparameters in festgelegt werden. CodePipeline

  Jedes Eingabeartefakt wird in ein eigenes Verzeichnis extrahiert, dessen Speicherorte in Umgebungsvariablen gespeichert sind. Das Verzeichnis für das primäre Quellartefakt wird mit `$CODEBUILD_SRC_DIR` zur Verfügung gestellt. Die Verzeichnisse für alle anderen Eingabeartefakte werden mit `$CODEBUILD_SRC_DIR_yourInputArtifactName` zur Verfügung gestellt.

> **Note**
>
> Das in Ihrem CodeBuild Projekt konfigurierte Artefakt wird zum Eingabeartefakt, das von der CodeBuild Aktion in Ihrer Pipeline verwendet wird.

Ausgabeartefakte

- Anzahl der Artefakte: `0 to 5`
- Beschreibung: Diese können verwendet werden, um die in der CodeBuild Build-Spezifikationsdatei definierten Artefakte für nachfolgende Aktionen in der Pipeline verfügbar zu machen. Wenn nur ein Ausgabeartefakt definiert ist, kann dieses Artefakt direkt im `artifacts`-Abschnitt der Build-Spezifikationsdatei definiert werden. Wenn mehr als ein Ausgabe-Artefakt angegeben ist, müssen in der Buildspezifikationsdatei alle Artefakte, auf die verwiesen wird, als sekundäre Artefakte definiert werden. Die Namen der Ausgabeartefakte in CodePipeline müssen mit den Artefakt-Identifikatoren in der Build-Spezifikationsdatei übereinstimmen.

> **Note**
>
> Das in Ihrem CodeBuild Projekt konfigurierte Artefakt wird zum CodePipeline Eingabeartefakt in Ihrer Pipeline-Aktion.

Wenn der `CombineArtifacts` Parameter für Batch-Builds ausgewählt ist, enthält der Speicherort des Ausgabe-Artefakts die kombinierten Artefakte aus mehreren Builds, die in derselben Ausführung ausgeführt wurden.

Ausgabevariablen

Diese Aktion erzeugt als Variablen alle Umgebungsvariablen, die als Teil des Builds exportiert wurden. Weitere Informationen zum Exportieren von Umgebungsvariablen finden Sie EnvironmentVariableim AWS CodeBuild API-Leitfaden.

Weitere Informationen zur Verwendung von CodeBuild Umgebungsvariablen in CodePipeline finden Sie in den Beispielen unterCodeBuild Ausgabevariablen für Aktionen. Eine Liste der

Umgebungsvariablen, die Sie verwenden können CodeBuild, finden Sie im AWS CodeBuild Benutzerhandbuch unter Umgebungsvariablen in Build-Umgebungen.

Aktionsdeklaration (CodeBuild-Beispiel)

YAML

```
Name: Build
Actions:
  - Name: PackageExport
    ActionTypeId:
      Category: Build
      Owner: AWS
      Provider: CodeBuild
      Version: '1'
    RunOrder: 1
    Configuration:
      BatchEnabled: 'true'
      CombineArtifacts: 'true'
      ProjectName: my-build-project
      PrimarySource: MyApplicationSource1
      EnvironmentVariables:
 '[{"name":"TEST_VARIABLE","value":"TEST_VALUE","type":"PLAINTEXT"},
{"name":"ParamStoreTest","value":"PARAMETER_NAME","type":"PARAMETER_STORE"}]'
    OutputArtifacts:
      - Name: MyPipeline-BuildArtifact
    InputArtifacts:
      - Name: MyApplicationSource1
      - Name: MyApplicationSource2
```

JSON

```
{
    "Name": "Build",
    "Actions": [
        {
            "Name": "PackageExport",
            "ActionTypeId": {
                "Category": "Build",
                "Owner": "AWS",
                "Provider": "CodeBuild",
                "Version": "1"
```

```
            },
            "RunOrder": 1,
            "Configuration": {
                "BatchEnabled": "true",
                "CombineArtifacts": "true",
                "ProjectName": "my-build-project",
                "PrimarySource": "MyApplicationSource1",
                "EnvironmentVariables": "[{\"name\":\"TEST_VARIABLE\",\"value\":
\"TEST_VALUE\",\"type\":\"PLAINTEXT\"},{\"name\":\"ParamStoreTest\",\"value\":
\"PARAMETER_NAME\",\"type\":\"PARAMETER_STORE\"}]"
            },
            "OutputArtifacts": [
                {
                    "Name": "MyPipeline-BuildArtifact"
                }
            ],
            "InputArtifacts": [
                {
                    "Name": "MyApplicationSource1"
                },
                {
                    "Name": "MyApplicationSource2"
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWS CodeBuild Benutzerhandbuch — Ein Beispiel für eine Pipeline mit einer CodeBuild Aktion finden Sie unter Verwenden CodePipeline , CodeBuild um Code zu testen und Builds auszuführen. Beispiele für Projekte mit mehreren Eingabe- und CodeBuild Ausgabeartefakten finden Sie unter CodePipelineIntegration mit CodeBuild und Beispiel für mehrere Eingabequellen und Ausgabeartefakte und Beispiel für mehrere Eingabequellen und Ausgabeartefakte.
- Tutorial: Erstellen Sie eine Pipeline, die Ihre Android-App erstellt und testet mit AWS Device Farm— Dieses Tutorial enthält ein Beispiel für eine Build-Spezifikationsdatei und eine

Beispielanwendung zum Erstellen einer Pipeline mit einer GitHub Quelle, die eine Android-App mit und erstellt CodeBuild und AWS Device Farm testet.

- Referenz zur Build-Spezifikation für CodeBuild — Dieses Referenzthema enthält Definitionen und Beispiele zum Verständnis von CodeBuild Build-Spezifikationsdateien. Eine Liste der Umgebungsvariablen, die Sie verwenden können CodeBuild, finden Sie im AWS CodeBuild Benutzerhandbuch unter Umgebungsvariablen in Build-Umgebungen.

CodeCommit

Startet die Pipeline, wenn ein neuer Commit für das konfigurierte CodeCommit Repository und den Branch vorgenommen wird.

Wenn Sie die Konsole verwenden, um die Pipeline zu erstellen oder zu bearbeiten, CodePipeline erstellt sie eine CodeCommit CloudWatch Ereignisregel, die Ihre Pipeline startet, wenn eine Änderung im Repository erfolgt.

Sie müssen bereits ein CodeCommit Repository erstellt haben, bevor Sie die Pipeline über eine CodeCommit Aktion verbinden können.

Nachdem eine Codeänderung erkannt wurde, haben Sie die folgenden Optionen, um den Code an nachfolgende Aktionen zu übergeben:

- Standard — Konfiguriert die CodeCommit Quellaktion so, dass sie eine ZIP-Datei mit einer flachen Kopie Ihres Commits ausgibt.
- Vollständiger Klon — Konfiguriert die Quellaktion so, dass sie für nachfolgende Aktionen eine Git-URL-Referenz an das Repository ausgibt.

  Derzeit kann die Git-URL-Referenz nur von CodeBuild Downstream-Aktionen verwendet werden, um das Repo und die zugehörigen Git-Metadaten zu klonen. Der Versuch, eine Git-URL-Referenz an CodeBuild Nichtaktionen zu übergeben, führt zu einem Fehler.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte

- Ausgabevariablen
- Beispielaktionskonfiguration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Source`
- Eigentümer: `AWS`
- Anbieter: `CodeCommit`
- Version: `1`

Konfigurationsparameter

RepositoryName

Erforderlich: Ja

Der Name des Repositorys, in dem Quelländerungen erkannt werden sollen.

BranchName

Erforderlich: Ja

Der Name des Zweigs, in dem Quelländerungen erkannt werden sollen.

PollForSourceChanges

Erforderlich: Nein

`PollForSourceChanges`steuert, ob das CodeCommit Repository nach Quellenänderungen CodePipeline abfragt. Wir empfehlen, stattdessen CloudWatch Ereignisse zu verwenden, um Quellenänderungen zu erkennen. Weitere Informationen zur Konfiguration von CloudWatch Ereignissen finden Sie unter Migrieren von Abfrage-Pipelines (CodeCommit Quelle) (CLI) oderMigrieren Sie Abfrage-Pipelines (CodeCommit Quelle) (AWS CloudFormation Vorlage).

> **Important**
>
> Wenn Sie beabsichtigen, eine CloudWatch Ereignisregel zu konfigurieren, müssen Sie „`PollForSourceChangesauf`“ festlegen, `false` um doppelte Pipeline-Ausführungen zu vermeiden.

Gültige Werte für diesen Parameter sind:

- `true`: Wenn diese Option gesetzt ist, CodePipeline wird Ihr Repository nach Quellenänderungen abgefragt.

  > **Note**
  >
  > Wenn Sie diese Option weglassen`PollForSourceChanges`, wird CodePipeline standardmäßig Ihr Repository nach Quelländerungen abgefragt. Dieses Verhalten ist das gleiche, als ob `PollForSourceChanges` enthalten wäre und auf `true` festgelegt würde.

- `false`: Falls gesetzt, fragt dein Repository CodePipeline nicht nach Quellenänderungen ab. Verwenden Sie diese Einstellung, wenn Sie beabsichtigen, eine CloudWatch Ereignisregel zur Erkennung von Quelländerungen zu konfigurieren.

OutputArtifactFormat

Erforderlich: Nein

Das Format des Ausgabeartefakts. Werte können entweder `CODEBUILD_CLONE_REF` oder `CODE_ZIP` sein. Wenn dieser Parameter nicht angegeben wird, lautet der Standardwert `CODE_ZIP`.

> **Important**
>
> Die `CODEBUILD_CLONE_REF` Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.
> Wenn Sie diese Option wählen, müssen Sie Ihrer `codecommit:GitPull` CodeBuild Servicerolle die Berechtigung hinzufügen, wie unter beschriebenFügen Sie CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen hinzu. Sie müssen Ihrer CodePipeline Servicerolle auch die `codecommit:GetRepository` Berechtigung hinzufügen, wie unter beschriebenHinzufügen von Berechtigungen zur CodePipeline-

Servicerolle. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiges Klonen verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer CodeCommit Pipeline-Quelle verwenden.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0`
- Beschreibung: Eingabe-Artefakte sind für diesen Aktionstyp nicht gültig.

Ausgabeartefakte

- Anzahl der Artefakte: 1
- Beschreibung: Das Ausgabe-Artefakt dieser Aktion ist eine ZIP-Datei, die den Inhalt des konfigurierten Repositorys und Zweigs beim Commit enthält, der als Quellrevision für die Pipeline-Ausführung angegeben wurde. Die aus dem Repository generierten Artefakte sind die Ausgabeartefakte für die CodeCommit Aktion. Die Quellcode-Commit-ID wird CodePipeline als Quellrevision für die ausgelöste Pipeline-Ausführung angezeigt.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Diese Aktion erzeugt Variablen, die als Ausgabevariablen angezeigt werden können, auch wenn die Aktion keinen Namespace hat. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

Weitere Informationen finden Sie unter Variablen.

CommitId

Die CodeCommit Commit-ID, die die Pipeline-Ausführung ausgelöst hat. Commit-IDs sind der vollständige SHA des Commits.

CommitMessage

Die Beschreibungsmeldung (wenn vorhanden), die dem Commit zugeordnet ist, der die Pipeline-Ausführung ausgelöst hat.

RepositoryName

Der Name des CodeCommit Repositorys, in dem der Commit, der die Pipeline ausgelöst hat, vorgenommen wurde.

BranchName

Der Name des Branches für das CodeCommit Repository, in dem die Quellenänderung vorgenommen wurde.

AuthorDate

Das Datum im Zeitstempelformat, an dem der Commit erstellt wurde.

Weitere Informationen zum Unterschied zwischen einem Autor und einem Committer in Git finden Sie unter Viewing the Commit History in Pro Git von Scott Chacon und Ben Straub.

CommitterDate

Das Datum im Zeitstempelformat, an dem der Commit durchgeführt wurde.

Weitere Informationen zum Unterschied zwischen einem Autor und einem Committer in Git finden Sie unter Viewing the Commit History in Pro Git von Scott Chacon und Ben Straub.

Beispielaktionskonfiguration

Beispiel für das Standardformat für Ausgabeartefakte

YAML

```
Actions:
  - OutputArtifacts:
      - Name: Artifact_MyWebsiteStack
    InputArtifacts: []
    Name: source
    Configuration:
      RepositoryName: MyWebsite
      BranchName: main
      PollForSourceChanges: 'false'
    RunOrder: 1
    ActionTypeId:
      Version: '1'
```

```
      Provider: CodeCommit
      Category: Source
      Owner: AWS
  Name: Source
```

JSON

```
{
    "Actions": [
        {
            "OutputArtifacts": [
                {
                    "Name": "Artifact_MyWebsiteStack"
                }
            ],
            "InputArtifacts": [],
            "Name": "source",
            "Configuration": {
                "RepositoryName": "MyWebsite",
                "BranchName": "main",
                "PollForSourceChanges": "false"
            },
            "RunOrder": 1,
            "ActionTypeId": {
                "Version": "1",
                "Provider": "CodeCommit",
                "Category": "Source",
                "Owner": "AWS"
            }
        }
    ],
    "Name": "Source"
},
```

Beispiel für das vollständige Clone-Ausgabeartefaktformat

YAML

```
name: Source
actionTypeId:
  category: Source
  owner: AWS
```

```
  provider: CodeCommit
  version: '1'
runOrder: 1
configuration:
  BranchName: main
  OutputArtifactFormat: CODEBUILD_CLONE_REF
  PollForSourceChanges: 'false'
  RepositoryName: MyWebsite
outputArtifacts:
  - name: SourceArtifact
inputArtifacts: []
region: us-west-2
namespace: SourceVariables
```

JSON

```
{
    "name": "Source",
    "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "provider": "CodeCommit",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {
        "BranchName": "main",
        "OutputArtifactFormat": "CODEBUILD_CLONE_REF",
        "PollForSourceChanges": "false",
        "RepositoryName": "MyWebsite"
    },
    "outputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "inputArtifacts": [],
    "region": "us-west-2",
    "namespace": "SourceVariables"
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository)— Dieses Tutorial enthält ein Beispiel für eine App-Spezifikationsdatei sowie eine CodeDeploy Beispielanwendung und eine Bereitstellungsgruppe. Verwenden Sie dieses Tutorial, um eine Pipeline mit einer CodeCommit Quelle zu erstellen, die auf Amazon EC2 EC2-Instances bereitgestellt wird.

AWS CodeDeploy

Sie verwenden eine AWS CodeDeploy Aktion, um Anwendungscode für Ihre Bereitstellungsflotte bereitzustellen. Ihre Bereitstellungsflotte kann aus Amazon EC2 EC2-Instances, lokalen Instances oder beidem bestehen.

> **Note**
>
> In diesem Referenzthema wird die CodeDeploy Bereitstellungsaktion für Anwendungen beschrieben, CodePipeline bei denen es sich bei der Bereitstellungsplattform um Amazon EC2 handelt. Referenzinformationen zu den Bereitstellungsaktionen von Amazon Elastic Container Service to CodeDeploy Blue/Green finden Sie CodePipeline unterAmazon Elastic Container Service und CodeDeploy Blaugrün.

Themen

Aktionstyp

- Kategorie: `Deploy`

- Eigentümer: `AWS`
- Anbieter: `CodeDeploy`
- Version: `1`

Konfigurationsparameter

ApplicationName

Erforderlich: Ja

Der Name der Anwendung, in CodeDeploy der Sie erstellt haben.

DeploymentGroupName

Erforderlich: Ja

Die Bereitstellungsgruppe, in der Sie erstellt haben CodeDeploy.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1`
- Beschreibung: Die AppSpec Datei, CodeDeploy anhand derer Folgendes ermittelt wird:
  - Was Sie aus Ihrer Anwendungsrevision in Amazon S3 oder auf Ihren Instances installieren müssen GitHub.
  - Welche Lebenszyklusereignis-Hooks als Reaktion auf Bereitstellungslebenszyklusereignisse ausgeführt werden sollen.

  Weitere Informationen zur AppSpec Datei finden Sie in der CodeDeploy AppSpec Dateireferenz.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Aktionsdeklaration

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: CodeDeploy
      Version: '1'
    RunOrder: 1
    Configuration:
      ApplicationName: my-application
      DeploymentGroupName: my-deployment-group
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "CodeDeploy",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "ApplicationName": "my-application",
                "DeploymentGroupName": "my-deployment-group"
            },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
```

```
                    "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket)— Dieses Tutorial führt Sie Schritt für Schritt durch die Erstellung eines Quell-Buckets, von EC2-Instances und CodeDeploy Ressourcen für die Bereitstellung einer Beispielanwendung. Anschließend erstellen Sie Ihre Pipeline mit einer CodeDeploy Bereitstellungsaktion, die den in Ihrem S3-Bucket verwalteten Code auf Ihrer Amazon EC2 EC2-Instance bereitstellt.
- Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository)— Dieses Tutorial führt Sie Schritt für Schritt durch die Erstellung Ihres CodeCommit Quell-Repositorys, Ihrer EC2-Instances und der CodeDeploy Ressourcen für die Bereitstellung einer Beispielanwendung. Anschließend erstellen Sie Ihre Pipeline mit einer CodeDeploy Bereitstellungsaktion, die Code aus Ihrem CodeCommit Repository auf Ihrer Amazon EC2 EC2-Instance bereitstellt.
- CodeDeploy AppSpec Dateireferenz — Dieses Referenzkapitel im AWS CodeDeploy Benutzerhandbuch enthält Referenzinformationen und Beispiele für CodeDeploy AppSpec Dateien.

CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen

Quellaktionen für Verbindungen werden von unterstützt. AWS CodeConnections CodeConnections ermöglicht es Ihnen, Verbindungen zwischen AWS Ressourcen und Repositorys von Drittanbietern herzustellen und zu verwalten, z. GitHub Startet eine Pipeline, wenn ein neuer Commit in einem Quellcode-Repository eines Drittanbieters vorgenommen wird. Die Quellaktion ruft Codeänderungen

ab, wenn eine Pipeline manuell ausgeführt wird oder wenn ein Webhook-Ereignis vom Quellanbieter gesendet wird.

Sie können Aktionen in Ihrer Pipeline so konfigurieren, dass sie eine Git-Konfiguration verwenden, mit der Sie Ihre Pipeline mit Triggern starten können. Weitere Informationen zur Konfiguration der Pipeline-Trigger zum Filtern mit Triggern finden Sie unterTrigger für Code-Push- oder Pull-Anfragen filtern.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Afrika (Kapstadt), Naher Osten (Bahrain), Naher Osten (VAE), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv) oder AWS GovCloud (US-West) nicht verfügbar. Hinweise zu anderen verfügbaren Aktionen finden Sie unter. Produkt- und Serviceintegrationen mit CodePipeline Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Verbindungen können Ihre AWS Ressourcen mit den folgenden Repositorys von Drittanbietern verknüpfen:

- Bitbucket Cloud (über die Bitbucket-Provider-Option in der CodePipeline Konsole oder den `Bitbucket` Anbieter in der CLI)

  > **Note**
  >
  > Sie können Verbindungen mit einem Bitbucket-Cloud-Repository erstellen. Installierte Bitbucket-Anbietertypen wie Bitbucket Server werden nicht unterstützt.

- > **Note**
  >
  > Wenn du einen Bitbucket-Workspace verwendest, benötigst du Administratorzugriff, um die Verbindung herzustellen.

- GitHub und GitHub Enterprise Cloud (über die Provider-Option GitHub (Version 2) in der CodePipeline Konsole oder den `GitHub` Anbieter in der CLI)

> **Note**
>
> Wenn sich Ihr Repository in einer GitHub Organisation befindet, müssen Sie der Eigentümer der Organisation sein, um die Verbindung herzustellen. Wenn Sie ein Repository verwenden, das sich nicht in einer Organisation befindet, müssen Sie der Eigentümer des Repositorys sein.

- GitHub Enterprise Server (über die GitHub Enterprise Server Provider-Option in der CodePipeline Konsole oder den `GitHub Enterprise Server` Anbieter in der CLI)
- GitLab.com (über die GitLabProvider-Option in der CodePipeline Konsole oder den `GitLab` Anbieter in der CLI)

> **Note**
>
> Sie können Verbindungen zu einem Repository erstellen GitLab, in dem Sie die Rolle „Besitzer“ haben, und dann kann die Verbindung mit dem Repository mit Ressourcen wie verwendet werden CodePipeline. Bei Repositorys in Gruppen müssen Sie nicht der Gruppenbesitzer sein.

- Selbstverwaltete Installation für GitLab (Enterprise Edition oder Community Edition) (über die Option GitLab Self-Managed Provider in der CodePipeline Konsole oder den `GitLabSelfManaged` Provider in der CLI)

> **Note**
>
> Jede Verbindung unterstützt alle Repositorys, die Sie bei diesem Anbieter haben. Sie müssen nur für jeden Anbietertyp eine neue Verbindung erstellen.

Mithilfe von Verbindungen kann Ihre Pipeline Quelländerungen über die Installations-App des Drittanbieters erkennen. Webhooks werden beispielsweise zum Abonnieren von GitHub Ereignistypen verwendet und können in einer Organisation, einem Repository oder einer GitHub App installiert werden. Ihre Verbindung installiert einen Repository-Webhook in Ihrer GitHub App, der Ereignisse vom Typ GitHub Push abonniert.

Nachdem eine Codeänderung erkannt wurde, haben Sie die folgenden Optionen, um den Code an nachfolgende Aktionen zu übergeben:

- Standard: Wie bei anderen existierenden CodePipeline Quellaktionen `CodeStarSourceConnection` kann auch hier eine ZIP-Datei mit einer oberflächlichen Kopie deines Commits ausgegeben werden.
- Vollständiger Klon: `CodeStarSourceConnection` kann auch so konfiguriert werden, dass eine URL-Referenz auf das Repository für nachfolgende Aktionen ausgegeben wird.

  Derzeit kann die Git-URL-Referenz nur von CodeBuild Downstream-Aktionen verwendet werden, um das Repo und die zugehörigen Git-Metadaten zu klonen. Der Versuch, eine Git-URL-Referenz an CodeBuild Nichtaktionen zu übergeben, führt zu einem Fehler.

CodePipeline fordert Sie auf, die AWS Connector-Installations-App zu Ihrem Drittanbieterkonto hinzuzufügen, wenn Sie eine Verbindung herstellen. Sie müssen Ihr Drittanbieter-Konto und Ihr Repository bereits erstellt haben, bevor Sie über die `CodeStarSourceConnection` Aktion eine Verbindung herstellen können.

> **Note**
>
> Informationen zum Erstellen oder Anhängen einer Richtlinie mit den für die Verwendung von AWS CodeStar Verbindungen erforderlichen Berechtigungen an Ihre Rolle finden Sie unter Referenz zu Verbindungsberechtigungen. Je nachdem, wann Ihre CodePipeline Servicerolle erstellt wurde, müssen Sie möglicherweise ihre Berechtigungen aktualisieren, um AWS CodeStar Verbindungen zu unterstützen. Anweisungen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.

> **Note**
>
> Um Verbindungen in Europa (Mailand) nutzen zu können AWS-Region, müssen Sie:
>
> 1. Regionsspezifische App installieren
> 2. Region aktivieren
>
> Diese regionsspezifische App unterstützt Verbindungen in der Region Europa (Mailand). Sie ist auf der Website des Drittanbieters veröffentlicht und von der bestehenden App

getrennt, die Verbindungen für andere Regionen unterstützt. Durch die Installation dieser App autorisieren Sie Drittanbieter, Ihre Daten nur für diese Region an den Dienst weiterzugeben, und Sie können diese Autorisierung jederzeit widerrufen, indem Sie die App deinstallieren. Der Dienst verarbeitet oder speichert Ihre Daten nicht, es sei denn, Sie aktivieren die Region. Durch die Aktivierung dieser Region gewähren Sie unserem Dienst die Erlaubnis, Ihre Daten zu verarbeiten und zu speichern.
Auch wenn die Region nicht aktiviert ist, können Drittanbieter Ihre Daten trotzdem mit unserem Dienst teilen, wenn die regionsspezifische App installiert bleibt. Achten Sie also darauf, die App zu deinstallieren, sobald Sie die Region deaktivieren. Weitere Informationen finden Sie unter Aktivieren einer Region.

Themen

Aktionstyp

- Kategorie: `Source`
- Eigentümer: `AWS`
- Anbieter: `CodeStarSourceConnection`
- Version: `1`

Konfigurationsparameter

ConnectionArn

Erforderlich: Ja

Der Verbindungs-ARN, der für den Quellanbieter konfiguriert und authentifiziert ist.

FullRepositoryId

Erforderlich: Ja

Der Besitzer und der Name des Repositorys, in dem Quelländerungen erkannt werden sollen.

Beispiel: `some-user/my-repo`

> **Important**
>
> Sie müssen die korrekte Groß- und Kleinschreibung für den FullRepositoryIdWert beibehalten. Wenn Ihr Benutzername beispielsweise `some-user` und der Repo-Name lautet`My-Repo`, ist der empfohlene Wert von FullRepositoryId. `some-user/My-Repo`

BranchName

Erforderlich: Ja

Der Name des Zweigs, in dem Quelländerungen erkannt werden sollen.

OutputArtifactFormat

Erforderlich: Nein

Gibt das Format des Ausgabeartefakt an. Kann `CODEBUILD_CLONE_REF` oder `CODE_ZIP` sein. Wenn dieser Parameter nicht angegeben wird, lautet der Standardwert `CODE_ZIP`.

> **Important**
>
> Die `CODEBUILD_CLONE_REF` Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.
> Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschriebenFügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, Enterprise Server oder .com GitHub hinzu GitHub GitLab. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiges Klonen verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

DetectChanges

Erforderlich: Nein

Steuert den automatischen Start Ihrer Pipeline, wenn ein neuer Commit für das konfigurierte Repository und den Branch vorgenommen wird. Falls nicht angegeben, ist der Standardwert`true`, und das Feld wird standardmäßig nicht angezeigt. Gültige Werte für diesen Parameter sind:

- `true`: Startet Ihre Pipeline CodePipeline automatisch bei neuen Commits.
- `false`: CodePipeline startet deine Pipeline nicht bei neuen Commits.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0`
- Beschreibung: Eingabe-Artefakte sind für diesen Aktionstyp nicht gültig.

Ausgabeartefakte

- Anzahl der Artefakte: `1`
- Beschreibung: Die aus dem Repository generierten Artefakte sind die Ausgabeartefakte für die `CodeStarSourceConnection`-Aktion. Die Quellcode-Commit-ID wird CodePipeline als Quellrevision für die ausgelöste Pipeline-Ausführung angezeigt. Sie können das Ausgabeartefakt dieser Aktion konfigurieren in:
  - Eine ZIP-Datei, die den Inhalt des konfigurierten Repositorys und Zweigs beim Commit enthält, der als Quellrevision für die Pipeline-Ausführung angegeben wurde.
  - Eine JSON-Datei, die einen URL-Verweis auf das Repository enthält, damit nachgeschaltete Aktionen Git-Befehle direkt ausführen können.

> **Important**
>
> Diese Option kann nur von CodeBuild Downstream-Aktionen verwendet werden. Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschriebenProblembehebung CodePipeline. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiges Klonen verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Diese Aktion erzeugt Variablen, die als Ausgabevariablen angezeigt werden können, auch wenn die Aktion keinen Namespace hat. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

Weitere Informationen finden Sie unter Variablen.

AuthorDate

Das Datum im Zeitstempelformat, an dem der Commit erstellt wurde.

BranchName

Der Name des Zweigs für das -Repository, in dem die Quelländerung ausgeführt wurde.

CommitId

Die -Commit-ID, die die Pipeline-Ausführung ausgelöst hat.

CommitMessage

Die Beschreibungsmeldung (wenn vorhanden), die dem Commit zugeordnet ist, der die Pipeline-Ausführung ausgelöst hat.

ConnectionArn

Der Verbindungs-ARN, der für den Quellanbieter konfiguriert und authentifiziert ist.

FullRepositoryName

Der Name des -Repositorys, in dem der Commit, der die Pipeline ausgelöst hat, ausgeführt wurde.

Aktionsdeklaration

Im folgenden Beispiel ist das Ausgabeartefakt auf das Standard-ZIP-Format `CODE_ZIP` für die Verbindung mit ARN `arn:aws:codestar-connections:region:`*`account-id`*`:connection/`*`connection-id`* gesetzt.

YAML

```
Name: Source
```

```
Actions:
  - InputArtifacts: []
    ActionTypeId:
      Version: '1'
      Owner: AWS
      Category: Source
      Provider: CodeStarSourceConnection
    OutputArtifacts:
      - Name: SourceArtifact
    RunOrder: 1
    Configuration:
      ConnectionArn: "arn:aws:codestar-connections:region:account-
id:connection/connection-id"
      FullRepositoryId: "some-user/my-repo"
      BranchName: "main"
      OutputArtifactFormat: "CODE_ZIP"
    Name: ApplicationSource
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "InputArtifacts": [],
            "ActionTypeId": {
                "Version": "1",
                "Owner": "AWS",
                "Category": "Source",
                "Provider": "CodeStarSourceConnection"
            },
            "OutputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "RunOrder": 1,
            "Configuration": {
                "ConnectionArn": "arn:aws:codestar-connections:region:account-
id:connection/connection-id",
                "FullRepositoryId": "some-user/my-repo",
                "BranchName": "main",
                "OutputArtifactFormat": "CODE_ZIP"
```

```
            },
            "Name": "ApplicationSource"
        }
    ]
},
```

Installation der Installations-App und Herstellen einer Verbindung

Wenn Sie die Konsole zum ersten Mal verwenden, um eine neue Verbindung zu einem Repository eines Drittanbieters hinzuzufügen, müssen Sie den CodePipeline Zugriff auf Ihre Repositorys autorisieren. Sie wählen oder erstellen eine Installations-App, mit der Sie eine Verbindung mit dem Konto herstellen können, in dem Sie das Code-Repository eines Drittanbieters erstellt haben.

Wenn Sie die AWS CLI oder eine AWS CloudFormation Vorlage verwenden, müssen Sie den Verbindungs-ARN einer Verbindung angeben, die bereits den Installationshandshake durchlaufen hat. Andernfalls wird die Pipeline nicht ausgelöst.

> **Note**
>
> Für eine `CodeStarSourceConnection` Quellaktion müssen Sie keinen Webhook einrichten und auch keine Standardabfrage verwenden. Die Aktion „Verbindungen“ verwaltet die Erkennung von Quellenänderungen für Sie.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWS::CodeStarConnections::Connection— Die AWS CloudFormation Vorlagenreferenz für die AWS CodeStar Connections-Ressource enthält Parameter und Beispiele für Verbindungen in AWS CloudFormation Vorlagen.
- AWS CodeStarConnections API-Referenz — Die AWS CodeStar Connections API-Referenz enthält Referenzinformationen für die verfügbaren Verbindungsaktionen.
- Die Schritte zum Erstellen einer Pipeline mit Quellaktionen, die von Verbindungen unterstützt werden, finden Sie im Folgenden:
  - Verwende für Bitbucket Cloud die Bitbucket-Option in der Konsole oder die `CodestarSourceConnection` Aktion in der CLI. Siehe Bitbucket Cloud-Verbindungen.

  - Verwenden Sie für GitHub GitHub Enterprise Cloud die GitHubProvider-Option in der Konsole oder die `CodestarSourceConnection` Aktion in der CLI. Siehe GitHub Verbindungen.
  - Verwenden Sie für GitHub Enterprise Server die GitHub Enterprise Server Provider-Option in der Konsole oder die `CodestarSourceConnection` Aktion in der CLI. Siehe GitHub Enterprise Server-Verbindungen.
  - Verwenden Sie für GitLab .com die GitLabProvider-Option in der Konsole oder die `CodestarSourceConnection` Aktion mit dem `GitLab` Anbieter in der CLI. Siehe GitLab.com-Verbindungen.
- Ein Tutorial „Erste Schritte“, in dem eine Pipeline mit einer Bitbucket-Quelle und einer CodeBuild Aktion erstellt wird, findest du unter Erste Schritte mit Verbindungen.
- Ein Tutorial, das dir zeigt, wie du eine Verbindung zu einem GitHub Repository herstellst und die Option „Vollständiges Klonen“ mit einer nachgeschalteten CodeBuild Aktion verwendest, findest du unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

AWS Device Farm

In Ihrer Pipeline können Sie eine Testaktion konfigurieren, mit der AWS Device Farm Ihre Anwendung auf Geräten ausgeführt und getestet wird. Device Farm verwendet Testpools von Geräten und Test-Frameworks, um Anwendungen auf bestimmten Geräten zu testen. Informationen zu den Typen von Testframeworks, die von der Aktion Device Farm unterstützt werden, finden Sie unter Arbeiten mit Testtypen in AWS Device Farm.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Test`
- Eigentümer: `AWS`

- Anbieter: `DeviceFarm`
- Version: `1`

Konfigurationsparameter

AppType

Erforderlich: Ja

Das Betriebssystem und die Art der Anwendung, die Sie testen. Im Folgenden finden Sie eine Liste gültiger Werte:

- `iOS`
- `Android`
- `Web`

ProjectId

Erforderlich: Ja

Die Device Farm Farm-Projekt-ID.

Um Ihre Projekt-ID zu finden, wählen Sie in der Device Farm Farm-Konsole Ihr Projekt aus. Kopieren Sie im Browser die URL Ihres neuen Projekts. Die URL enthält die Projekt-ID. Die Projekt-ID ist der Wert in der URL danach`projects/`. Im folgenden Beispiel lautet die Projekt-ID`eec4905f-98f8-40aa-9afc-4c1cfexample`.

```
https://<region-URL>/devicefarm/home?region=us-west-2#/projects/
eec4905f-98f8-40aa-9afc-4c1cfexample/runs
```

App

Erforderlich: Ja

Der Name und der Speicherort der Anwendungsdatei in Ihrem Eingabeartefakt. Beispiel: `s3-ios-test-1.ipa`

TestSpec

Bedingt: Ja

Der Speicherort der Definitionsdatei der Testspezifikation in Ihrem Eingabeartefakt. Dies ist für den Test im benutzerdefinierten Modus erforderlich.

DevicePoolArn

Erforderlich: Ja

Der Gerätefarm-Gerätepool-ARN.

Geben Sie mit der AWS CLI den folgenden Befehl ein, um die verfügbaren Gerätepool-ARNs für das Projekt abzurufen, einschließlich des ARN für Top-Geräte:

```
aws devicefarm list-device-pools --arn arn:aws:devicefarm:us-
west-2:account_ID:project:project_ID
```

TestType

Erforderlich: Ja

Gibt das unterstützte Test-Framework für Ihren Test an. Im Folgenden finden Sie eine Liste gültiger Werte für`TestType`:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_KNOTEN
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- EINGEBAUTETER_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> Die folgenden Testtypen werden von der Aktion in CodePipeline:`WEB_PERFORMANCE_PROFILE`, und nicht unterstützt. `REMOTE_ACCESS_RECORD` `REMOTE_ACCESS_REPLAY`

Informationen zu Device Farm-Testtypen finden Sie unter Arbeiten mit Testtypen in AWS Device Farm.

RadioBluetoothEnabled

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob Bluetooth zu Beginn des Tests aktiviert werden soll.

RecordAppPerformanceData

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob während des Tests Geräteleistungsdaten wie CPU-, FPS- und Speicherleistung aufgezeichnet werden sollen.

RecordVideo

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob während des Tests Videos aufgenommen werden sollen.

RadioWifiEnabled

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob Wi-Fi zu Beginn des Tests aktiviert werden soll.

RadioNfcEnabled

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob NFC zu Beginn des Tests aktiviert werden soll.

RadioGpsEnabled

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob GPS zu Beginn des Tests aktiviert werden soll.

Test

Erforderlich: Nein

Der Name und der Pfad der Testspezifikationsdatei an Ihrem Quellspeicherort. Der Pfad ist relativ zum Stamm des Eingabeartefakts für Ihren Test.

FuzzEventCount

Erforderlich: Nein

Die Anzahl der Benutzeroberflächenereignisse, die der Fuzz-Test durchführen soll, zwischen 1 und 10.000.

FuzzEventThrottle

Erforderlich: Nein

Die Anzahl der Millisekunden, die der Fuzz-Test warten muss, bevor das nächste Benutzeroberflächenereignis ausgeführt wird, zwischen 1 und 1.000.

FuzzRandomizerSeed

Erforderlich: Nein

Ein Startwert für den Fuzz-Test, der für die Randomisierung von Benutzeroberflächenereignissen verwendet wird. Die Verwendung derselben Zahl für nachfolgende Fuzz-Tests führt zu identischen Ereignissequenzen.

CustomHostMachineArtifacts

Erforderlich: Nein

Der Speicherort auf dem Host-Computer, an dem benutzerdefinierte Artefakte gespeichert werden.

CustomDeviceArtifacts

Erforderlich: Nein

Der Ort auf dem Gerät, an dem benutzerdefinierte Artefakte gespeichert werden.

UnmeteredDevicesOnly

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob Sie bei der Ausführung von Tests in diesem Schritt nur Ihre Geräte ohne Zähler verwenden möchten.

JobTimeoutMinutes

Erforderlich: Nein

Die Anzahl der Minuten, die ein Testlauf pro Gerät ausgeführt wird, bevor das Timeout überschritten wird.

Breitengrand

Erforderlich: Nein

Der Breitengrad des Geräts, ausgedrückt in Grad des geographischen Koordinatensystems.

Längengrad

Erforderlich: Nein

Der Längengrad des Geräts, ausgedrückt in Grad des geographischen Koordinatensystems.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Die Gruppe von Artefakten, die für die Testaktion verfügbar gemacht werden sollen. Device Farm sucht nach der erstellten Anwendung und den zu verwendenden Testdefinitionen.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Aktionsdeklaration

YAML

```
Name: Test
Actions:
  - Name: TestDeviceFarm
```

```
    ActionTypeId: null
    category: Test
    owner: AWS
    provider: DeviceFarm
    version: '1'
RunOrder: 1
Configuration:
  App: s3-ios-test-1.ipa
  AppType: iOS
  DevicePoolArn: >-
    arn:aws:devicefarm:us-west-2::devicepool:0EXAMPLE-d7d7-48a5-ba5c-b33d66efa1f5
  ProjectId: eec4905f-98f8-40aa-9afc-4c1cfEXAMPLE
  TestType: APPIUM_PYTHON
  TestSpec: example-spec.yml
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: us-west-2
```

JSON

```
{
    "Name": "Test",
    "Actions": [
        {
            "Name": "TestDeviceFarm",
            "ActionTypeId": null,
            "category": "Test",
            "owner": "AWS",
            "provider": "DeviceFarm",
            "version": "1"
        }
    ],
    "RunOrder": 1,
    "Configuration": {
        "App": "s3-ios-test-1.ipa",
        "AppType": "iOS",
        "DevicePoolArn": "arn:aws:devicefarm:us-west-2::devicepool:0EXAMPLE-
d7d7-48a5-ba5c-b33d66efa1f5",
        "ProjectId": "eec4905f-98f8-40aa-9afc-4c1cfEXAMPLE",
        "TestType": "APPIUM_PYTHON",
        "TestSpec": "example-spec.yml"
    },
```

```
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "us-west-2"
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Arbeiten mit Testtypen in Device Farm — Dieses Referenzkapitel im Device Farm Developer Guide enthält eine genauere Beschreibung der von Device Farm unterstützten Test-Frameworks für Android, iOS und Webanwendungen.
- Aktionen in Device Farm — Die API-Aufrufe und Parameter in der Device Farm API-Referenz können Ihnen bei der Arbeit mit Device Farm Farm-Projekten helfen.
- Tutorial: Erstellen Sie eine Pipeline, die Ihre Android-App erstellt und testet mit AWS Device Farm— Dieses Tutorial enthält eine Beispieldatei mit einer Build-Spezifikationsdatei und eine Beispielanwendung zum Erstellen einer Pipeline mit einer GitHub Quelle, die eine Android-App mit einer Device Farm erstellt CodeBuild und testet.
- Tutorial: Erstellen Sie eine Pipeline, die Ihre iOS-App testet mit AWS Device Farm— Dieses Tutorial enthält eine Beispielanwendung zum Erstellen einer Pipeline mit einer Amazon S3 S3-Quelle, die eine erstellte iOS-App mit Device Farm testet.

AWS Lambda

Ermöglicht es Ihnen, eine Lambda-Funktion als Aktion in Ihrer Pipeline auszuführen. Mit dem Ereignisobjekt, das eine Eingabe für diese Funktion ist, hat die Funktion Zugriff auf die Aktionskonfiguration, Speicherorte der Eingabeartefakte, Speicherorte der Ausgabeartefakte und andere Informationen, die für den Zugriff auf die Artefakte erforderlich sind. Ein Beispielereignis, das an eine Lambda-Aufruffunktion übergeben wurde, finden Sie unter. JSON-Beispielereignis Im Rahmen der Implementierung der Lambda-Funktion muss entweder `PutJobSuccessResult API` oder `PutJobFailureResult API` aufgerufen werden. Andernfalls hängt die Ausführung dieser

Aktion, bis die Aktion abläuft. Wenn Sie Ausgabeartefakte für die Aktion angeben, müssen sie als Teil der Funktionsimplementierung in den S3-Bucket hochgeladen werden.

> **Important**
>
> Protokollieren Sie nicht das JSON-Ereignis, das CodePipeline an Lambda gesendet wird, da dies dazu führen kann, dass Benutzeranmeldeinformationen in CloudWatch Logs protokolliert werden. Die CodePipeline Rolle verwendet ein JSON-Ereignis, um temporäre Anmeldeinformationen im `artifactCredentials` Feld an Lambda zu übergeben. Ein Beispiel für ein Ereignis finden Sie unter JSON-Beispielereignis.

Aktionstyp

- Kategorie: `Invoke`
- Eigentümer: `AWS`
- Anbieter: `Lambda`
- Version: `1`

Konfigurationsparameter

FunctionName

Erforderlich: Ja

`FunctionName`ist der Name der in Lambda erstellten Funktion.

UserParameters

Erforderlich: Nein

Eine Zeichenfolge, die als Eingabe von der Lambda-Funktion verarbeitet werden kann.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0 to 5`
- Beschreibung: Der Satz von Artefakten, die der Lambda-Funktion zur Verfügung gestellt werden sollen.

Ausgabeartefakte

- Anzahl der Artefakte: `0 to 5`
- Beschreibung: Der Satz von Artefakten, die als Ausgabe von der Lambda-Funktion erzeugt werden.

Ausgabevariablen

Diese Aktion erzeugt alle Schlüssel-Wert-Paare, die im `outputVariables` Abschnitt der API-Anfrage enthalten sind, als Variablen. PutJobSuccessResult

Weitere Hinweise zu Variablen in CodePipeline finden Sie unter. Variablen

Beispielaktionskonfiguration

YAML

```
Name: Lambda
Actions:
  - Name: Lambda
    ActionTypeId:
      Category: Invoke
      Owner: AWS
      Provider: Lambda
      Version: '1'
    RunOrder: 1
    Configuration:
      FunctionName: myLambdaFunction
      UserParameters: 'http://192.0.2.4'
    OutputArtifacts: []
    InputArtifacts: []
    Region: us-west-2
```

JSON

```
{
    "Name": "Lambda",
    "Actions": [
        {
            "Name": "Lambda",
```

```
            "ActionTypeId": {
                "Category": "Invoke",
                "Owner": "AWS",
                "Provider": "Lambda",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "FunctionName": "myLambdaFunction",
                "UserParameters": "http://192.0.2.4"
            },
            "OutputArtifacts": [],
            "InputArtifacts": [],
            "Region": "us-west-2"
        }
    ]
},
```

JSON-Beispielereignis

Die Lambda-Aktion sendet ein JSON-Ereignis, das die Job-ID, die Konfiguration der Pipeline-Aktion, die Speicherorte der Eingabe- und Ausgabeartefakte und alle Verschlüsselungsinformationen für die Artefakte enthält. Der Jobworker greift auf diese Details zu, um die Lambda-Aktion abzuschließen. Weitere Informationen finden Sie unter Auftragsdetails. Es folgt ein Beispielereignis.

```
{
    "CodePipeline.job": {
        "id": "11111111-abcd-1111-abcd-111111abcdef",
        "accountId": "111111111111",
        "data": {
            "actionConfiguration": {
                "configuration": {
                    "FunctionName": "MyLambdaFunction",
                    "UserParameters": "input_parameter"
                }
            },
            "inputArtifacts": [
                {
                    "location": {
                        "s3Location": {
                            "bucketName": "bucket_name",
                            "objectKey": "filename"
```

```
                        },
                        "type": "S3"
                    },
                    "revision": null,
                    "name": "ArtifactName"
                }
            ],
            "outputArtifacts": [],
            "artifactCredentials": {
                "secretAccessKey": "secret_key",
                "sessionToken": "session_token",
                "accessKeyId": "access_key_ID"
            },
            "continuationToken": "token_ID",
            "encryptionKey": {
              "id": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
              "type": "KMS"
            }
        }
    }
}
```

Das JSON-Ereignis enthält die folgenden Jobdetails für die Lambda-Aktion in CodePipeline:

- `id`: Die eindeutige, vom System generierte ID des Auftrags.
- `accountId`: Die dem Job zugeordnete AWS Konto-ID.
- `data`: Sonstige Informationen, die für einen Auftragsworker erforderlich sind, um den Auftrag abzuschließen.
  - `actionConfiguration`: Die Aktionsparameter für die Lambda-Aktion. Definitionen finden Sie unter Konfigurationsparameter .
  - `inputArtifacts`: Das für die Aktion bereitgestellte Artefakt.
    - `location`: Der Speicherort des Artefaktspeichers.
      - `s3Location`: Die Informationen zum Speicherort des Eingabeartefakts für die Aktion.
        - `bucketName`: Der Name des Pipeline-Artefaktspeichers für die Aktion (z. B. ein Amazon S3 S3-Bucket mit dem Namen codepipeline-us-east -2-1234567890).
        - `objectKey`: Der Name der Anwendung (z. B. `CodePipelineDemoApplication.zip`).
      - `type`: Der Typ des Artefakts am Speicherort. Derzeit ist `S3` der einzige gültige Artefakttyp.

- `revision`: Die Revisions-ID des Artefakts. Je nach Objekttyp kann dies eine Commit-ID (GitHub) oder eine Revision-ID (Amazon Simple Storage Service) sein. Weitere Informationen finden Sie unter ArtifactRevision.
- `name`: Der Name des zu bearbeitenden Artefakts, z. B. `MyApp`.

- `outputArtifacts`: Die Ausgabe der Aktion.
  - `location`: Der Speicherort des Artefaktspeichers.
    - `s3Location`: Die Speicherortinformationen des Ausgabeartefakts für die Aktion.
      - `bucketName`: Der Name des Pipeline-Artefaktspeichers für die Aktion (z. B. ein Amazon S3 S3-Bucket mit dem Namen codepipeline-us-east -2-1234567890).
      - `objectKey`: Der Name der Anwendung (z. B. `CodePipelineDemoApplication.zip`).
    - `type`: Der Typ des Artefakts am Speicherort. Derzeit ist `S3` der einzige gültige Artefakttyp.
  - `revision`: Die Revisions-ID des Artefakts. Je nach Objekttyp kann dies eine Commit-ID (GitHub) oder eine Revision-ID (Amazon Simple Storage Service) sein. Weitere Informationen finden Sie unter ArtifactRevision.
  - `name`: Der Name der Ausgabe eines Artefakts, z. B. `MyApp`.
- `artifactCredentials`: Die AWS Sitzungsdaten, die für den Zugriff auf Eingabe- und Ausgabeartefakte im Amazon S3 S3-Bucket verwendet werden. Diese Anmeldeinformationen sind temporäre Anmeldeinformationen, die von AWS Security Token Service (AWS STS) ausgestellt werden.
  - `secretAccessKey`: Der geheime Zugriffsschlüssel für die Sitzung.
  - `sessionToken`: Das Token für die Sitzung.
  - `accessKeyId`: Der geheime Zugriffsschlüssel für die Sitzung.
- `continuationToken`: Ein von der Aktion generiertes Token. Zukünftige Aktionen verwenden dieses Token, um die laufende Instance der Aktion zu identifizieren. Wenn die Aktion abgeschlossen ist, sollte kein Fortsetzungs-Token übermittelt werden.
- `encryptionKey`: Der Verschlüsselungsschlüssel, der zum Verschlüsseln der Daten im Artefaktspeicher verwendet wird, z. B. ein AWS KMS Schlüssel. Wenn dieser nicht definiert ist, wird der Standardschlüssel für Amazon Simple Storage Service verwendet.
  - `id`: Die ID, die verwendet wurde, um den Schlüssel zu identifizieren. Für einen AWS KMS -Schlüssel können Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN verwenden.
  - `type`: Der Typ des Verschlüsselungsschlüssels, z. B. ein AWS KMS -Schlüssel.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWS CloudFormation Benutzerhandbuch — Weitere Informationen zu Lambda-Aktionen und AWS CloudFormation Artefakten für Pipelines finden Sie unter Verwenden von Parameterüberschreibungsfunktionen mit CodePipeline Pipelines, Automatisieren der Bereitstellung von Lambda-basierten Anwendungen und Artefakten.AWS CloudFormation
- Rufen Sie eine AWS Lambda Funktion in einer Pipeline auf in CodePipeline— Dieses Verfahren enthält eine Lambda-Beispielfunktion und zeigt Ihnen, wie Sie mithilfe der Konsole eine Pipeline mit einer Lambda-Aufrufaktion erstellen.

Referenz zur Snyk-Aktionsstruktur

Die Snyk-Aktion in CodePipeline automatisiert die Erkennung und Behebung von Sicherheitslücken in Ihrem Open-Source-Code. Du kannst Snyk mit Anwendungsquellcode in deinem Drittanbieter-Repository wie Bitbucket Cloud GitHub oder mit Bildern für Containeranwendungen verwenden. Deine Aktion scannt und meldet die von dir konfigurierten Sicherheitslücken und Warnmeldungen.

> **Note**

Themen

ID des Aktionstyps

- Kategorie: `Invoke`
- Eigentümer: `ThirdParty`
- Anbieter: `Snyk`

- Version: 1

Beispiel:

```
{
    "Category": "Invoke",
    "Owner": "ThirdParty",
    "Provider": "Snyk",
    "Version": "1"
},
```

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Die Dateien, aus denen das Eingabeartefakt für die Aufrufaktion besteht.

Ausgabeartefakte

- Anzahl der Artefakte: 1
- Beschreibung: Die Dateien, aus denen das Ausgabeartefakt für die Aufrufaktion besteht.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Weitere Informationen zur Verwendung von Snyk-Aktionen in CodePipeline finden Sie unter Automatisieren Sie das Scannen von Sicherheitslücken mit Snyk. CodePipeline

AWS Step Functions

Eine AWS CodePipeline Aktion, die Folgendes bewirkt:

- Startet eine AWS Step Functions State-Machine-Ausführung von Ihrer Pipeline aus.
- Stellt dem Zustandsautomaten über eine Eigenschaft in der Aktionskonfiguration oder über eine Datei in einem Pipelineartefakt einen Anfangszustand bereit, der als Eingabe übergeben wird.

- Legt optional ein Ausführungs-ID-Präfix für die Identifizierung von Ausführungen fest, die von der Aktion ausgelöst werden.
- Unterstützt Standard- und Express-Zustandsautomaten .

> **Note**
>
> Die Aktion Step Functions wird auf Lambda ausgeführt und verfügt daher über Artefaktgrößenquoten, die den Artefaktgrößenquoten für Lambda-Funktionen entsprechen. Weitere Informationen finden Sie unter Lambda-Kontingente im Lambda Developer Guide.

Aktionstyp

- Kategorie: `Invoke`
- Eigentümer: `AWS`
- Anbieter: `StepFunctions`
- Version: `1`

Konfigurationsparameter

StateMachineArn

Erforderlich: Ja

Der Amazon-Ressourcenname (ARN) für den Zustandsautomaten, der aufgerufen werden soll.

ExecutionNamePrefix

Erforderlich: Nein

Standardmäßig wird als Ausführungsname für den Zustandsautomaten die Ausführungs-ID der Aktion verwendet. Wenn ein Präfix angegeben wird, wird dieses der Ausführungs-ID der Aktion mit einem Bindestrich vorangestellt und beide Angaben werden zusammen als Ausführungsname für den Zustandsautomaten verwendet.

```
myPrefix-1624a1d1-3699-43f0-8e1e-6bafd7fde791
```

Bei einem Express-Zustandsautomaten sollte der Name nur 0-9, A-Z, a-z, - und _ enthalten.

InputType

Erforderlich: Nein

- Literal (Standard): Wenn angegeben, wird der Wert im Feld Input (Eingabe) direkt als Eingabe an den Zustandsautomaten übergeben.

  Beispieleintrag für das Feld Input (Eingabe), wenn Literal ausgewählt ist:

  ```
  {"action": "test"}
  ```

- FilePath: Der Inhalt einer Datei in dem im Eingabefeld angegebenen Eingabeartefakt wird als Eingabe für die State-Machine-Ausführung verwendet. Ein Eingabeartefakt ist erforderlich, wenn auf gesetzt InputTypeist. FilePath

  Beispieleintrag für das Eingabefeld, wenn ausgewählt FilePathist:

  ```
  assets/input.json
  ```

Eingabe

Required: Conditional

- Literal: Wenn auf Literal (Standard) gesetzt InputTypeist, ist dieses Feld optional.

  Wenn angegeben, wird das Feld Input (Eingabe) direkt als Eingabe für die Ausführung des Zustandsautomaten verwendet. Andernfalls wird der Zustandsautomat mit einem leeren JSON-Objekt (`{}`) aufgerufen.

- FilePath: Wenn auf gesetzt InputTypeist FilePath, ist dieses Feld erforderlich.

  Ein Eingabeartefakt ist auch erforderlich, wenn auf FilePathgesetzt InputTypeist.

  Der Inhalt der Datei im angegebenen Eingabeartefakt wird als Eingabe für die Ausführung des Zustandsautomaten verwendet.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0 to 1`
- Beschreibung: Wenn auf gesetzt InputTypeist FilePath, ist dieses Artefakt erforderlich und wird als Quelle für die Eingabe für die State-Machine-Ausführung verwendet.

Ausgabeartefakte

- Anzahl der Artefakte: `0 to 1`
- Beschreibung
  - Standard-Zustandsautomaten: Wenn vorhanden, wird das Ausgabeartefakt mit der Ausgabe des Zustandsautomaten gefüllt. Dies wird aus der `output` Eigenschaft der Step Functions DescribeExecution Functions-API-Antwort abgerufen, nachdem die State-Machine-Ausführung erfolgreich abgeschlossen wurde.
  - Express-Zustandsautomaten: Nicht unterstützt.

Ausgabevariablen

Durch diese Aktion werden Ausgabevariablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können.

Weitere Informationen finden Sie unter Variablen.

StateMachineArn

Der ARN des Zustandsautomaten.

ExecutionArn

Der ARN der Ausführung des Zustandsautomaten. Nur Standard-Zustandsautomaten.

Beispielaktionskonfiguration

Beispiel für Standardeingabe

YAML

```
Name: ActionName
ActionTypeId:
  Category: Invoke
  Owner: AWS
  Version: 1
  Provider: StepFunctions
OutputArtifacts:
  - Name: myOutputArtifact
Configuration:
```

```
  StateMachineArn: arn:aws:states:us-east-1:111122223333:stateMachine:HelloWorld-
StateMachine
  ExecutionNamePrefix: my-prefix
```

JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Invoke",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "StepFunctions"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "StateMachineArn": "arn:aws:states:us-
east-1:111122223333:stateMachine:HelloWorld-StateMachine",
        "ExecutionNamePrefix": "my-prefix"
    }
}
```

Beispiel für Literaleingabe

YAML

```
Name: ActionName
ActionTypeId:
  Category: Invoke
  Owner: AWS
  Version: 1
  Provider: StepFunctions
OutputArtifacts:
  - Name: myOutputArtifact
Configuration:
  StateMachineArn: arn:aws:states:us-east-1:111122223333:stateMachine:HelloWorld-
StateMachine
  ExecutionNamePrefix: my-prefix
```

```
  Input: '{"action": "test"}'
```

JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Invoke",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "StepFunctions"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "StateMachineArn": "arn:aws:states:us-
east-1:111122223333:stateMachine:HelloWorld-StateMachine",
        "ExecutionNamePrefix": "my-prefix",
        "Input": "{\"action\": \"test\"}"
    }
}
```

Beispiel für Eingabedatei

YAML

```
Name: ActionName
InputArtifacts:
  - Name: myInputArtifact
ActionTypeId:
  Category: Invoke
  Owner: AWS
  Version: 1
  Provider: StepFunctions
OutputArtifacts:
  - Name: myOutputArtifact
Configuration:
  StateMachineArn: 'arn:aws:states:us-east-1:111122223333:stateMachine:HelloWorld-
StateMachine'
```

```
  ExecutionNamePrefix: my-prefix
  InputType: FilePath
  Input: assets/input.json
```

JSON

```
{
    "Name": "ActionName",
    "InputArtifacts": [
        {
            "Name": "myInputArtifact"
        }
    ],
    "ActionTypeId": {
        "Category": "Invoke",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "StepFunctions"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "StateMachineArn": "arn:aws:states:us-
east-1:111122223333:stateMachine:HelloWorld-StateMachine",
        "ExecutionNamePrefix": "my-prefix",
        "InputType": "FilePath",
        "Input": "assets/input.json"
    }
}
```

Behavior

CodePipeline Führt während einer Veröffentlichung die konfigurierte Zustandsmaschine unter Verwendung der Eingabe aus, die in der Aktionskonfiguration angegeben ist.

Wenn auf Literal gesetzt InputTypeist, wird der Inhalt des Konfigurationsfeldes für die Eingabeaktion als Eingabe für die Zustandsmaschine verwendet. Wenn keine Literaleingabe bereitgestellt wird, wird für die Ausführung des Zustandsautomaten ein leeres JSON-Objekt ({}) verwendet. Weitere

Informationen zum Ausführen einer State-Machine-Ausführung ohne Eingabe finden Sie in der Step Functions StartExecution API.

Wenn auf gesetzt InputTypeist FilePath, entpackt die Aktion das Eingabeartefakt und verwendet den Inhalt der Datei, die im Konfigurationsfeld für die Eingabeaktion angegeben ist, als Eingabe für die Zustandsmaschine. Wenn angegeben, FilePathist das Eingabefeld erforderlich und es muss ein Eingabeartefakt vorhanden sein. Andernfalls schlägt die Aktion fehl.

Nach einer erfolgreichen Ausführung des Starts ist das Verhalten für die beiden Arten von Zustandsautomaten, Standard- bzw. Express- unterschiedlich.

Standard-Zustandsautomaten

Wenn die Ausführung der Standardzustandsmaschine erfolgreich gestartet wurde, CodePipeline wird die `DescribeExecution` API abgefragt, bis die Ausführung einen Terminalstatus erreicht. Wenn die Ausführung erfolgreich abgeschlossen wird, wird die Aktion erfolgreich ausgeführt, andernfalls schlägt sie fehl.

Wenn ein Ausgabeartefakt konfiguriert ist, enthält das Artefakt den Rückgabewert des Zustandsautomaten. Dies wird aus der `output` Eigenschaft der Step Functions DescribeExecution Functions-API-Antwort abgerufen, nachdem die State-Machine-Ausführung erfolgreich abgeschlossen wurde. Beachten Sie, dass für diese API Einschränkungen für die Ausgabelänge zwingend gelten.

Fehlerbehandlung

- Wenn der Start einer Ausführung für einen Zustandsautomaten fehlschlägt, schlägt die Ausführung der Aktion ebenfalls fehl.
- Wenn die Ausführung der Zustandsmaschine keinen Terminalstatus erreicht, bevor die CodePipeline Step Functions Functions-Aktion ihr Timeout erreicht (Standard von 7 Tagen), schlägt die Aktionsausführung fehl. Der Zustandsautomat läuft möglicherweise trotz dieses Fehlers weiter. Weitere Informationen zu Timeouts bei der Ausführung von Zustandsmaschinen in Step Functions finden Sie unter Standard- und Express-Workflows.

> **Note**
>
> Sie können für das Konto mit der Aktion eine Kontingenterhöhung für das Timeout von Aufrufaktionen anfordern. Die Kontingenterhöhung gilt dann auch für alle Aktionen dieser Art in allen Regionen für dieses Konto.

- Wenn die Ausführung des Zustandsautomaten mit dem Beendigungsstatus FAILED, TIMED_OUT oder ABORTED endet, schlägt die Ausführung der Aktion fehl.

Express-Zustandsautomaten

Wenn die Ausführung des Express-Zustandsautomaten erfolgreich gestartet wurde, wird die Ausführung der Aufrufaktion erfolgreich abgeschlossen.

Überlegungen zu Aktionen, die für Express-Zustandsautomaten konfiguriert sind:

- Sie können kein Ausgabeartefakt angeben.
- Die Aktion wartet nicht, bis die Ausführung des Zustandsautomaten beendet ist.
- Nachdem die Ausführung der Aktion gestartet wurde CodePipeline, ist die Ausführung der Aktion erfolgreich, auch wenn die Ausführung der Zustandsmaschine fehlschlägt.

Fehlerbehandlung

- Wenn die Ausführung einer Zustandsmaschine CodePipeline nicht gestartet werden kann, schlägt die Ausführung der Aktion fehl. Andernfalls gilt die Aktion sofort ebenfalls als erfolgreich ausgeführt. Die Aktion ist erfolgreich, CodePipeline unabhängig davon, wie lange es dauert, bis die State-Machine-Ausführung abgeschlossen ist oder welches Ergebnis sie hat.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWS Step Functions Entwicklerhandbuch — Informationen zu Zustandsmaschinen, Ausführungen und Eingaben für Zustandsmaschinen finden Sie im AWS Step Functions Entwicklerhandbuch.
- Tutorial: Verwenden Sie eine AWS Step Functions Aufrufaktion in einer Pipeline— In diesem Tutorial beginnen Sie mit einem Beispiel für eine Standardstatusmaschine und erfahren, wie Sie mit der Konsole eine Pipeline aktualisieren, indem Sie eine Step Functions Functions-Aufrufaktion hinzufügen.

Referenz zum Integrationsmodell

Es gibt mehrere vorgefertigte Integrationen für Dienste von Drittanbietern, mit deren Hilfe bestehende Kundentools in den Pipeline-Release-Prozess integriert werden können. Partner oder Drittanbieter verwenden ein Integrationsmodell, um Aktionstypen für den Einsatz in zu implementieren. CodePipeline

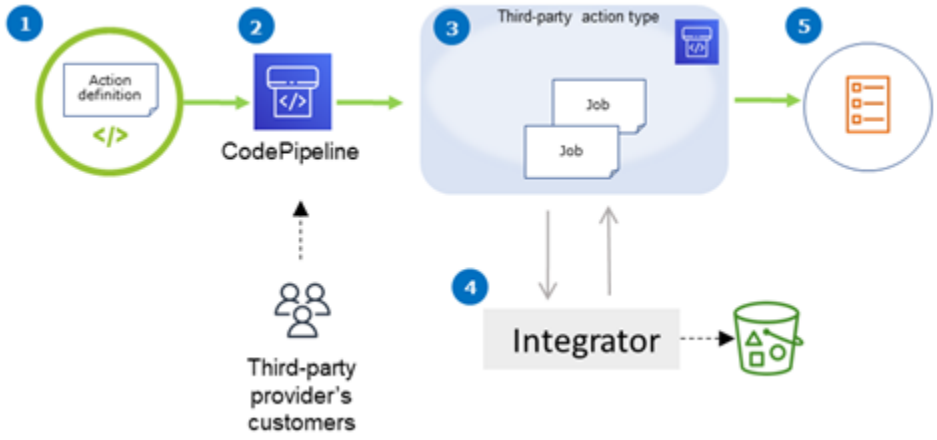
Verwenden Sie diese Referenz, wenn Sie Aktionstypen planen oder damit arbeiten, die mit einem unterstützten Integrationsmodell verwaltet werden CodePipeline.

Um Ihren Aktionstyp eines Drittanbieters als Partnerintegration mit zu zertifizieren CodePipeline, verweisen Sie auf das AWS Partnernetzwerk (APN). Diese Informationen stellen eine Ergänzung zur Referenz dar.AWS CLI

Themen

So funktionieren Aktionstypen von Drittanbietern mit dem Integrator

Sie können Aktionstypen von Drittanbietern zu Kunden-Pipelines hinzufügen, um Aufgaben im Zusammenhang mit Kundenressourcen zu erledigen. Der Integrator verwaltet Jobanfragen und führt die Aktion mit aus. CodePipeline Das folgende Diagramm zeigt einen Aktionstyp eines Drittanbieters, der für Kunden zur Verwendung in ihrer Pipeline erstellt wurde. Nachdem der Kunde die Aktion konfiguriert hat, wird die Aktion ausgeführt und es werden Jobanfragen erstellt, die von der Action Engine des Integrators bearbeitet werden.



Das Diagramm zeigt die folgenden Schritte:

1. Die Aktionsdefinition wird registriert und in verfügbar gemacht CodePipeline. Die Drittanbieter-Aktion ist für Kunden des Drittanbieters verfügbar.
2. Der Kunde des Anbieters wählt und konfiguriert die Aktion in CodePipeline.
3. Die Aktion wird ausgeführt und Jobs werden in die Warteschlange gestellt. CodePipeline Wenn der Job bereit ist CodePipeline, sendet er eine Jobanfrage.
4. Der Integrator (der Job Worker für Polling-APIs von Drittanbietern oder die Lambda-Funktion) nimmt die Jobanfrage entgegen, gibt eine Bestätigung zurück und bearbeitet die Artefakte für die Aktionen.
5. Der Integrator gibt Erfolg/Misserfolge-Ausgabe (der Job Worker verwendet Erfolg/Fehlschlag-APIs oder die Lambda-Funktion sendet Erfolgs-/Fehlschlagsausgaben) mit einem Job-Ergebnis und einem Fortsetzungstoken zurück.

Informationen zu den Schritten, mit denen Sie einen Aktionstyp anfordern, anzeigen und aktualisieren können, finden Sie unter. Mit Aktionstypen arbeiten

Konzepte

In diesem Abschnitt werden die folgenden Begriffe für Aktionstypen von Drittanbietern verwendet:

Aktionstyp

Ein wiederholbarer Prozess, der in Pipelines wiederverwendet werden kann, die dieselben Continuous Delivery-Workloads ausführen. Aktionstypen werden durch ein`Owner`,, `Category` und identifiziert. `Provider` `Version` Beispielsweise:

```
        {

            "Category": "Deploy",
            "Owner": "AWS",
            "Provider": "CodeDeploy",
            "Version": "1"
        },
```

Alle Aktionen desselben Typs haben dieselbe Implementierung.

Aktion

Eine einzelne Instanz eines Aktionstyps, einer der diskreten Prozesse, die innerhalb einer Phase einer Pipeline ablaufen. Dazu gehören in der Regel die Benutzerwerte, die für die Pipeline spezifisch sind, in der diese Aktion ausgeführt wird.

Definition der Aktion

Das Schema für einen Aktionstyp, das die Eigenschaften definiert, die für die Konfiguration der Aktion und der Eingabe-/Ausgabeartefakte erforderlich sind.

Aktionsausführung

Eine Sammlung von Aufträgen, die ausgeführt wurden, um festzustellen, ob die Aktion in der Pipeline des Kunden erfolgreich war oder nicht.

Engine zur Ausführung von Aktionen

Eine Eigenschaft der Konfiguration zur Aktionsausführung, die den Integrationstyp definiert, der von einem Aktionstyp verwendet wird. Gültige Werte sind `JobWorker` und `Lambda`.

Integration

Beschreibt eine Software, die von einem Integrator ausgeführt wird, um einen Aktionstyp zu implementieren. CodePipeline unterstützt zwei Integrationstypen, die den beiden unterstützten Action Engines `JobWorker` und `Lambda` entsprechen.

Integrator

Die Person, der die Implementierung eines Aktionstyps gehört.

Aufgabe

Eine Arbeit mit Pipeline und Kundenkontext zur Ausführung einer Integration. Eine Aktionsausführung besteht aus einem oder mehreren Jobs.

Arbeitsnehmer

Der Dienst, der die Kundeneingaben verarbeitet und einen Job ausführt.

Unterstützte Integrationsmodelle

CodePipeline hat zwei Integrationsmodelle:

- Lambda-Integrationsmodell: Dieses Integrationsmodell ist die bevorzugte Methode, um mit Aktionstypen in CodePipeline zu arbeiten. Das Lambda-Integrationsmodell verwendet eine Lambda-Funktion, um Jobanfragen zu verarbeiten, wenn Ihre Aktion ausgeführt wird.
- Integrationsmodell für Job Worker: Das Job Worker-Integrationsmodell ist das bisher verwendete Modell für Integrationen von Drittanbietern. Das Job-Worker-Integrationsmodell verwendet einen Job Worker, der so konfiguriert ist, dass er die CodePipeline APIs kontaktiert, um Jobanfragen zu bearbeiten, wenn Ihre Aktion ausgeführt wird.

Zum Vergleich werden in der folgenden Tabelle die Funktionen der beiden Modelle beschrieben:

	Lambda-Integrationsmodell	Modell zur Eingliederung von Arbeitern
Beschreibung	Der Integrator schreibt die Integration als Lambda-Funktion, die immer dann aufgerufen wird, CodePipeline wenn ein Job für die Aktion verfügbar ist. Die Lambda-Funktion fragt nicht nach verfügbaren Jobs ab, sondern wartet, bis die nächste Jobanfrage eingeht.	Der Integrator schreibt die Integration als Job Worker, der ständig nach verfügbaren Jobs in den Pipelines des Kunden sucht. Der Jobworker führt dann den Job aus und sendet das Auftragsergebnis mithilfe von APIs zurück an. CodePipeline CodePipeline
Infrastruktur	AWS Lambda	Stellen Sie Job Worker-Code in der Infrastruktur des Integrators bereit, z. B. in Amazon EC2 EC2-Instances.

	Lambda-Integrationsmodell	Modell zur Eingliederung von Arbeitern
Entwicklungsanstrengungen	Die Integration beinhaltet nur die Geschäftslogik.	Die Integration muss zusätzlich zur Geschäftslogik mit CodePipeline APIs interagieren.
Aufwand der Operation	Geringerer operativer Aufwand, da es sich bei der Infrastruktur nur AWS um Ressourcen handelt.	Höherer operativer Aufwand, da der Arbeitsarbeiter seine eigenständige Hardware benötigt.
Max. Job-Laufzeit	Wenn die Integration länger als 15 Minuten aktiv ausgeführt werden muss, kann dieses Modell nicht verwendet werden. Diese Aktion richtet sich an Integratoren, die einen Prozess starten (z. B. einen Build auf dem Code-Artefakt des Kunden initiieren) und nach Abschluss ein Ergebnis zurückgeben müssen. Wir empfehlen dem Integrator nicht, weiterhin darauf zu warten, dass der Build abgeschlossen ist. Geben Sie stattdessen eine Fortsetzung zurück. CodePipelineerstellt in weiteren 30 Sekunden einen neuen Job, wenn eine Fortsetzung vom Code des Integrators empfangen wird, der den Job überprüft, bis er abgeschlossen ist.	Mit diesem Modell können Aufträge mit sehr langer Laufzeit (Stunden/Tage) aufrechterhalten werden.

Lambda-Integrationsmodell

Das unterstützte Lambda-Integrationsmodell umfasst die Erstellung der Lambda-Funktion und die Definition der Ausgabe für den Aktionstyp eines Drittanbieters.

Aktualisieren Sie Ihre Lambda-Funktion, um die Eingabe von zu verarbeiten CodePipeline

Sie können eine neue Lambda-Funktion erstellen. Sie können Ihrer Lambda-Funktion Geschäftslogik hinzufügen, die immer dann ausgeführt wird, wenn in Ihrer Pipeline ein Job für Ihren Aktionstyp verfügbar ist. Wenn Sie beispielsweise den Kontext des Kunden und der Pipeline berücksichtigen, möchten Sie vielleicht einen Build in Ihrem Service für den Kunden starten.

Verwenden Sie die folgenden Parameter, um Ihre Lambda-Funktion so zu aktualisieren, dass sie die Eingabe von CodePipeline verarbeitet.

Format:

- `jobId`:
  - Die eindeutige, vom System generierte ID des Jobs.
  - Typ: Zeichenfolge
  - Muster: [0-9a-f] {8} - [0-9a-f] {4} - [0-9a-f] {4} - [0-9a-f] {4} - [0-9a-f] {12}
- `accountId`:
  - Die ID des AWS Kundenkontos, das bei der Ausführung des Auftrags verwendet werden soll.
  - Typ: Zeichenfolge
  - Muster: [0-9] {12}
- `data`:
  - Andere Informationen über einen Job, die von einer Integration verwendet werden, um den Job abzuschließen.
  - Enthält eine Übersicht der folgenden Elemente:
    - `actionConfiguration`:
      - Die Konfigurationsdaten für die Aktion. Die Felder für die Aktionskonfiguration sind eine Zuordnung von Schlüssel-Wert-Paaren, sodass Ihr Kunde Werte eingeben kann. Die Schlüssel werden durch die Schlüsselparameter in der Aktionstyp-Definitionsdatei bestimmt, wenn Sie Ihre Aktion einrichten. In diesem Beispiel werden die Werte vom Benutzer der Aktion bestimmt, der Informationen in den `Password` Feldern `Username` und angibt.

- Typ: Zuordnung von Zeichenfolge zu Zeichenfolge, optional vorhanden

  Beispiel:

```
"configuration": {
    "Username": "MyUser",
    "Password": "MyPassword"
},
```

- `encryptionKey`:
  - Stellt Informationen über den Schlüssel dar, der zum Verschlüsseln von Daten im Artefaktspeicher verwendet wird, z. B. einen AWS KMS Schlüssel.
  - Inhalt: Typ des Datentyps`encryptionKey`, optional vorhanden
- `inputArtifacts`:
  - Liste mit Informationen über ein Artefakt, an dem gearbeitet werden soll, z. B. ein Test- oder Build-Artefakt.
  - Inhalt: Liste des Datentyps`Artifact`, optional vorhanden
- `outputArtifacts`:
  - Liste mit Informationen über die Ausgabe einer Aktion.
  - Inhalt: Liste des Datentyps`Artifact`, optional vorhanden
- `actionCredentials`:
  - Stellt ein Objekt mit AWS Sitzungsanmeldedaten dar. Bei diesen Anmeldeinformationen handelt es sich um temporäre Anmeldeinformationen, die von ausgestellt werden AWS STS. Sie können verwendet werden, um auf Eingabe- und Ausgabeartefakte im S3-Bucket zuzugreifen, in dem Artefakte für die Pipeline gespeichert werden CodePipeline.

    Diese Anmeldeinformationen haben auch dieselben Berechtigungen wie die angegebene Vorlage für Richtlinienerklärungen in der Aktionstyp-Definitionsdatei.
  - Inhalt: Typ des Datentyps`AWSSessionCredentials`, optional vorhanden
- `actionExecutionId`:
  - Die externe ID der Ausführung der Aktion.
  - Typ: Zeichenfolge
- `continuationToken`:

- Ein vom System generiertes Token, z. B. eine Bereitstellungs-ID, das von einem Job benötigt wird, um den Job asynchron fortzusetzen.
- Typ: Zeichenfolge, optional vorhanden

Datentypen:

- `encryptionKey`:
  - `id`:
    - Die ID, die verwendet wurde, um den Schlüssel zu identifizieren. Für einen AWS KMS Schlüssel können Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN verwenden.
    - Typ: Zeichenfolge
  - `type`:
    - Der Typ des Verschlüsselungsschlüssels, z. B. ein AWS KMS Schlüssel.
    - Typ: Zeichenfolge
    - Zulässige Werte: `KMS`
- `Artifact`:
  - `name`:
    - Der Name des Artefakts.
    - Typ: Zeichenfolge, optional vorhanden
  - `revision`:
    - Die Revisions-ID des Artefakts. Je nach Objekttyp kann dies eine Commit-ID (GitHub) oder eine Revision-ID (Amazon S3) sein.
    - Typ: Zeichenfolge, optional vorhanden
  - `location`:
    - Der Standort eines Artefakts.
    - Inhalt: Typ des Datentyps`ArtifactLocation`, optional vorhanden
- `ArtifactLocation`:
  - `type`:
    - Der Typ des Artefakts am Standort.
    - Typ: Zeichenfolge, optional vorhanden
    - Zulässige Werte: `S3`

  - `s3Location`:
    - Der Speicherort des S3-Buckets, der eine Revision enthält.
    - Inhalt: Typ des Datentyps`S3Location`, optional vorhanden
- `S3Location`:
  - `bucketName`:
    - Der Name des S3-Buckets.
    - Typ: Zeichenfolge
  - `objectKey`:
    - Der Schlüssel des Objekts im S3-Bucket, der das Objekt im Bucket eindeutig identifiziert.
    - Typ: Zeichenfolge
- `AWSSessionCredentials`:
  - `accessKeyId`:
    - Der Zugriffsschlüssel für die Sitzung.
    - Typ: Zeichenfolge
  - `secretAccessKey`:
    - Der geheime Zugriffsschlüssel für die Sitzung.
    - Typ: Zeichenfolge
  - `sessionToken`:
    - Das Token für die Sitzung.
    - Typ: Zeichenfolge

Beispiel:

```
{
    "jobId": "01234567-abcd-abcd-abcd-012345678910",
    "accountId": "012345678910",
    "data": {
        "actionConfiguration": {
            "key1": "value1",
            "key2": "value2"
        },
        "encryptionKey": {
            "id": "123-abc",
            "type": "KMS"
```

```
        },
        "inputArtifacts": [
            {
                "name": "input-art-name",
                "location": {
                    "type": "S3",
                    "s3Location": {
                        "bucketName": "inputBucket",
                        "objectKey": "inputKey"
                    }
                }
            }
        ],
        "outputArtifacts": [
            {
                "name": "output-art-name",
                "location": {
                    "type": "S3",
                    "s3Location": {
                        "bucketName": "outputBucket",
                        "objectKey": "outputKey"
                    }
                }
            }
        ],
        "actionExecutionId": "actionExecutionId",
        "actionCredentials": {
            "accessKeyId": "access-id",
            "secretAccessKey": "secret-id",
            "sessionToken": "session-id"
        },
        "continuationToken": "continueId-xxyyzz"
    }
}
```

Geben Sie die Ergebnisse Ihrer Lambda-Funktion zurück an CodePipeline

Die Job-Worker-Ressource des Integrators muss bei Erfolg, Misserfolg oder Fortführung eine gültige Nutzlast zurückgeben.

Format:

- `result`: Das Ergebnis des Jobs.

- Erforderlich
- Gültige Werte (Groß- und Kleinschreibung wird nicht beachtet):
  - `Success`: Zeigt an, dass ein Job erfolgreich und terminal ist.
  - `Continue`: Zeigt an, dass ein Job erfolgreich ist und fortgesetzt werden muss, z. B. wenn der Job-Worker für dieselbe Aktionsausführung erneut aufgerufen wird.
  - `Fail`: Zeigt an, dass ein Job fehlgeschlagen ist und es sich um ein Terminal handelt.

- `failureType`: Ein Fehlertyp, der einem fehlgeschlagenen Job zugeordnet werden soll.

  Die `failureType` Kategorie für Partneraktionen beschreibt die Art des Fehlers, der bei der Ausführung des Jobs aufgetreten ist. Integratoren legen den Typ zusammen mit der Fehlermeldung fest, wenn das Ergebnis eines Auftragsfehlers zurückgegeben wird. CodePipeline

  - Optional. Erforderlich, wenn das Ergebnis ist`Fail`.
  - Muss Null sein, wenn `result` ist `Success` oder `Continue`
  - Zulässige Werte:
    - ConfigurationError
    - JobFailed
    - PermissionsError
    - RevisionOutOfSync
    - RevisionUnavailable
    - SystemUnavailable

- `continuation`: Fortsetzungsstatus, der an den nächsten Job innerhalb der aktuellen Aktionsausführung übergeben werden soll.
  - Optional. Erforderlich, wenn das Ergebnis ist`Continue`.
  - Muss Null sein, wenn `result` es `Success` oder ist`Fail`.
  - Eigenschaften:
    - `State`: Ein Hash des Status, der übergeben werden soll.
- `status`: Status der Ausführung der Aktion.
  - Optional.
  - Eigenschaften:
    - `ExternalExecutionId`: Eine optionale externe Ausführungs- oder Commit-ID, die dem Job zugeordnet werden soll.

    - `Summary`: Eine optionale Zusammenfassung dessen, was passiert ist. In Ausfallszenarien wird dies zur Fehlermeldung, die dem Benutzer angezeigt wird.
- `outputVariables`: Eine Reihe von Schlüssel/Wert-Paaren, die an die nächste Aktionsausführung übergeben werden.
    - Optional.
    - Muss Null sein, wenn es oder `result` ist`Continue`. `Fail`

Beispiel:

```
{
    "result": "success",
    "failureType": null,
    "continuation": null,
    "status": {
        "externalExecutionId": "my-commit-id-123",
        "summary": "everything is dandy"
    },
    "outputVariables": {
        "FirstOne": "Nice",
        "SecondOne": "Nicest",
        ...
    }
}
```

Verwenden Sie Fortsetzungstoken, um auf Ergebnisse eines asynchronen Prozesses zu warten

Das `continuation` Token ist Teil der Nutzlast und das Ergebnis Ihrer Lambda-Funktion. Es ist eine Möglichkeit, den Status eines Jobs zu übergeben CodePipeline und darauf hinzuweisen, dass der Job fortgesetzt werden muss. Wenn ein Integrator beispielsweise einen Build für den Kunden auf seiner Ressource gestartet hat, wartet er nicht darauf, dass der Build abgeschlossen ist, sondern signalisiert, CodePipeline dass er kein Endergebnis hat, indem er das `result` as zurückgibt `continue` und die eindeutige ID des Builds an das CodePipeline `continuation` as-Token zurückgibt.

> **Note**
>
> Lambda-Funktionen können nur bis zu 15 Minuten ausgeführt werden. Wenn der Job länger ausgeführt werden muss, können Sie Fortsetzungstoken verwenden.

Das CodePipeline Team ruft den Integrator nach 30 Sekunden mit demselben `continuation` Token in seiner Payload auf, sodass es überprüfen kann, ob er abgeschlossen ist. Wenn der Build abgeschlossen ist, gibt der Integrator das Terminal-Erfolg/Fehlschlagen zurück, andernfalls fährt er fort.

Stellen Sie CodePipeline die Berechtigungen bereit, um die Lambda-Funktion des Integrators zur Laufzeit aufzurufen

Sie fügen Ihrer Integrator-Lambda-Funktion Berechtigungen hinzu, um dem CodePipeline Dienst Berechtigungen zu geben, ihn mithilfe des CodePipeline Dienstprinzipals aufzurufen:. `codepipeline.amazonaws.com` Sie können Berechtigungen mithilfe AWS CloudFormation oder der Befehlszeile hinzufügen. Ein Beispiel finden Sie unter Mit Aktionstypen arbeiten.

Modell zur Eingliederung von Arbeitern

Nachdem Sie Ihren allgemeinen Arbeitsablauf entworfen haben, können Sie Ihren Job Worker erstellen. Die Besonderheiten der Drittanbieter-Aktion bestimmen zwar, was für den Job Worker benötigt wird, aber die meisten Job Worker für Drittanbieter-Aktionen verfügen über die folgenden Funktionen:

- Abfrage von Aufträgen aus der Verwendung von CodePipeline . `PollForThirdPartyJobs`
- Bestätigen von Aufträgen und Rückgabe von Ergebnissen bei CodePipeline Verwendung von`AcknowledgeThirdPartyJob`,`PutThirdPartyJobSuccessResult`, und. `PutThirdPartyJobFailureResult`
- Artefakte werden aus dem Amazon S3-Bucket abgerufen und/oder in den Amazon S3 S3-Bucket für die Pipeline gelegt. Um Artefakte aus dem Amazon S3 S3-Bucket herunterzuladen, müssen Sie einen Amazon S3 S3-Client erstellen, der Signature Version 4-Signatur (Sig V4) verwendet. Sig V4 ist erforderlich für AWS KMS.

  Um Artefakte in den Amazon S3 S3-Bucket hochzuladen, müssen Sie auch die Amazon S3 `PutObject` S3-Anfrage so konfigurieren, dass sie Verschlüsselung durch AWS Key Management

Service (AWS KMS) verwendet. AWS KMS verwendet AWS KMS keys. Um zu wissen, ob der Von AWS verwalteter Schlüssel oder ein vom Kunden verwalteter Schlüssel zum Hochladen von Artefakten verwendet werden soll, muss sich Ihr Auftragsmitarbeiter die Auftragsdaten ansehen und die Eigenschaft des Verschlüsselungsschlüssels überprüfen. Wenn die Eigenschaft festgelegt ist, sollten Sie bei der Konfiguration diese vom Kunden verwaltete Schlüssel-ID verwenden AWS KMS. Wenn die Schlüsseleigenschaft Null ist, verwenden Sie die Von AWS verwalteter Schlüssel. CodePipeline verwendet die, Von AWS verwalteter Schlüssel sofern nicht anders konfiguriert.

Ein Beispiel, das zeigt, wie die AWS KMS Parameter in Java oder .NET erstellt werden, finden Sie unter Spezifying the AWS Key Management Service in Amazon S3 Using the AWS SDKs. Weitere Informationen zum Amazon S3 S3-Bucket für CodePipeline finden Sie unterCodePipeline Konzepte .

Wählen und Konfigurieren einer Strategie für die Berechtigungsverwaltung Ihres Auftragsworkers

Um einen Job Worker für Ihre Drittanbieter-Aktion zu entwickeln CodePipeline, benötigen Sie eine Strategie für die Integration von Benutzer- und Rechteverwaltung.

Die einfachste Strategie besteht darin, die Infrastruktur, die Sie für Ihren Job Worker benötigen, hinzuzufügen, indem Sie Amazon EC2 EC2-Instances mit einer AWS Identity and Access Management (IAM-) Instance-Rolle erstellen, sodass Sie die Ressourcen, die Sie für Ihre Integration benötigen, einfach skalieren können. Sie können die integrierte Integration mit verwenden AWS , um die Interaktion zwischen Ihrem Mitarbeiter und zu vereinfachen. CodePipeline

Erfahren Sie mehr über Amazon EC2 und finden Sie heraus, ob es die richtige Wahl für Ihre Integration ist. Weitere Informationen finden Sie unter Amazon EC2 — Virtual Server Hosting. Informationen zum Einrichten einer Amazon EC2 EC2-Instance finden Sie unter Erste Schritte mit Amazon EC2 EC2-Linux-Instances.

Eine weitere Strategie, die Sie in Betracht ziehen sollten, ist die Verwendung eines Identitätsverbunds mit IAM zur Integration Ihres vorhandenen Identitätsanbietersystems und Ihrer Ressourcen. Diese Strategie ist nützlich, wenn Sie bereits über einen Corporate Identity Provider verfügen oder bereits so konfiguriert sind, dass Benutzer, die Web-Identity-Provider verwenden, unterstützt werden. Mithilfe eines Identitätsverbunds können Sie sicheren Zugriff auf AWS Ressourcen gewähren CodePipeline, auch ohne IAM-Benutzer erstellen oder verwalten zu müssen. Sie können Funktionen und Richtlinien für die Sicherheitsanforderungen bezüglich Passwörtern und die Rotation von Anmeldeinformationen

nutzen. Sie können Beispielanwendungen als Vorlage für Ihre selbsterstellten Anwendungen verwenden. Weitere Informationen finden Sie unter Manage Federation.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Im Folgenden finden Sie ein Beispiel für eine Richtlinie, die Sie für die Verwendung mit Ihrem Job Worker eines Drittanbieters erstellen könnten. Diese Richtlinie dient lediglich als Beispiel und wird unverändert bereitgestellt.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:PollForThirdPartyJobs",
        "codepipeline:AcknowledgeThirdPartyJob",
        "codepipeline:GetThirdPartyJobDetails",
        "codepipeline:PutThirdPartyJobSuccessResult",
        "codepipeline:PutThirdPartyJobFailureResult"
      ],
      "Resource": [
        "arn:aws:codepipeline:us-east-2::actionType:ThirdParty/Build/Provider/1/"
      ]
    }
```

```
    ]
}
```

Referenz zu Abbild-Definitionsdateien

Dieser Abschnitt dient nur als Referenz. Informationen zum Erstellen einer Pipeline mit Quell- oder Bereitstellungsaktionen für Container finden Sie unter Erstellen Sie eine Pipeline in CodePipeline.

AWS CodePipeline Jobworker für Container-Aktionen, wie z. B. eine Amazon ECR-Quellaktion oder Amazon ECS-Bereitstellungsaktionen, verwenden Definitionsdateien, um den Image-URI und den Container-Namen der Aufgabendefinition zuzuordnen. Jede Definitionsdatei ist eine JSON-formatierte Datei, die vom Aktionsanbieter wie folgt verwendet wird:

- Amazon ECS-Standardbereitstellungen erfordern eine `imagedefinitions.json` Datei als Eingabe für die Bereitstellungsaktion.
- Für blaue/grüne Bereitstellungen von Amazon ECS ist eine `imageDetail.json` Datei als Eingabe für die Bereitstellungsaktion erforderlich.
  - Amazon ECR-Quellaktionen generieren eine `imageDetail.json`-Datei, die als Ausgabe aus der Quellaktion bereitgestellt wird.

Themen

Datei imagedefinitions.json für Amazon ECS-Standardbereitstellungsaktionen

Ein Imagedefinitionsdokument ist eine JSON-Datei, die Ihren Amazon ECS-Container-Namen sowie das Bild und das Tag beschreibt. Wenn Sie containerbasierte Anwendungen bereitstellen, müssen Sie eine Image-Definitionsdatei generieren, um dem CodePipeline Job-Worker den Amazon ECS-Container und die Image-ID zum Abrufen aus dem Image-Repository, z. B. Amazon ECR, zur Verfügung zu stellen.

> **Note**
>
> Der Standarddateiname für die Datei ist `imagedefinitions.json`. Wenn Sie einen anderen Dateinamen verwenden, müssen Sie diesen bereitstellen, wenn Sie die Pipeline-Bereitstellungsphase erstellen.

Erstellen Sie die Datei `imagedefinitions.json`, wobei Sie Folgendes berücksichtigen:

- Die Datei muss die UTF-8-Kodierung verwenden.
- Die maximale Dateigröße für die Abbild-Definitionsdatei ist 100 KB.
- Sie müssen die Datei als Quell- oder Build-Artefakt erstellen, damit sie ein Eingabeartefakt für die Bereitstellungsaktion ist. Mit anderen Worten, stellen Sie sicher, dass die Datei entweder an Ihren Quellspeicherort, z. B. in Ihr CodeCommit Repository, hochgeladen oder als integriertes Ausgabeartefakt generiert wird.

Die Datei `imagedefinitions.json` stellt den Containernamen und den Abbild-URI bereit. Dieser muss mit dem folgenden Satz von Schlüssel-Wert-Paaren konstruiert werden.

Schlüssel	Wert
Name	*container_name*
imageUri	*imageUri*

Dies ist die JSON-Struktur, wobei der Containername `sample-app`, der Abbild-URI `ecs-repo` und der Tag `latest` ist:

```
[
  {
    "name": "sample-app",
    "imageUri": "11111EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest"
  }
]
```

Sie können die Datei auch so konstruieren, dass mehrere Container-Abbild-Paare aufgelistet werden.

JSON-Struktur:

```
[
  {
    "name": "simple-app",
    "imageUri": "httpd:2.4"
  },
  {
    "name": "simple-app-1",
    "imageUri": "mysql"
  },
  {
    "name": "simple-app-2",
    "imageUri": "java1.8"
  }
]
```

Führen Sie vor Erstellung Ihrer Pipeline die die folgenden Schritte aus, um die `imagedefinitions.json` einzurichten.

1. Planen Sie während der Planung der Bereitstellung Container-basierter Anwendungen für Ihre Pipeline die Quell- und die Build-Phase, sofern vorhanden.
2. Wählen Sie eine der folgenden Optionen aus:

   a. Wenn Ihre Pipeline so erstellt wurde, dass sie die Erstellungsphase überspringt, müssen Sie die JSON-Datei manuell erstellen und in Ihr Quell-Repository hochladen, damit die Quellaktion das Artefakt bereitstellen kann. Erstellen Sie die Datei mit einem Texteditor und geben Sie der Datei einen Namen oder verwenden Sie den Standarddateinamen `imagedefinitions.json`. Verschieben Sie die Abbild-Definitionsdatei in Ihr Quell-Repository.

   > **Note**
   >
   > Wenn es sich bei Ihrem Quell-Repository um einen Amazon S3 S3-Bucket handelt, denken Sie daran, die JSON-Datei zu komprimieren.

   b. Wenn Ihre Pipeline eine Build-Phase enthält, fügen Sie Ihrer Build Spec-Datei einen Befehl hinzu, der die Abbild-Definitionsdatei während der Build-Phase in Ihr Quell-Repository ausgibt. Im folgenden Beispiel wird der Befehl printf verwendet, um eine

`imagedefinitions.json`-Datei zu erstellen. Listen Sie diesen Befehl im Abschnitt `post_build` der Datei `buildspec.yml` auf:

```
printf '[{"name":"container_name","imageUri":"image_URI"}]' >
imagedefinitions.json
```

Sie müssen die Abbild-Definitionsdatei als Ausgabeartefakt in die Datei `buildspec.yml` einfügen.

3. Wenn Sie Ihre Pipeline in der Konsole erstellen, geben Sie auf der Seite Deploy (Bereitstellen) des Assistenten Create Pipeline (Pipeline erstellen) in Image Filename (Name der Abbilddatei) den Namen der Abbild-Definitionsdatei ein.

Ein step-by-step Tutorial zum Erstellen einer Pipeline, die Amazon ECS als Bereitstellungsanbieter verwendet, finden Sie unter Tutorial: Continuous Deployment with CodePipeline.

ImageDetail.json-Datei für Amazon ECS-Bereitstellungsaktionen in Blau/Grün

Ein `imageDetail.json` Dokument ist eine JSON-Datei, die Ihren Amazon ECS-Image-URI beschreibt. Wenn Sie containerbasierte Anwendungen für eine blaue/grüne Bereitstellung bereitstellen, müssen Sie die `imageDetail.json` Datei generieren, um dem Amazon ECS und dem CodeDeploy Jobworker die Image-ID bereitzustellen, die sie aus dem Image-Repository abrufen können, z. B. Amazon ECR.

> **Note**
>
> Der Name der Datei muss `imageDetail.json` sein.

Eine Beschreibung der Aktion und ihrer Parameter finden Sie unter. Amazon Elastic Container Service und CodeDeploy Blaugrün

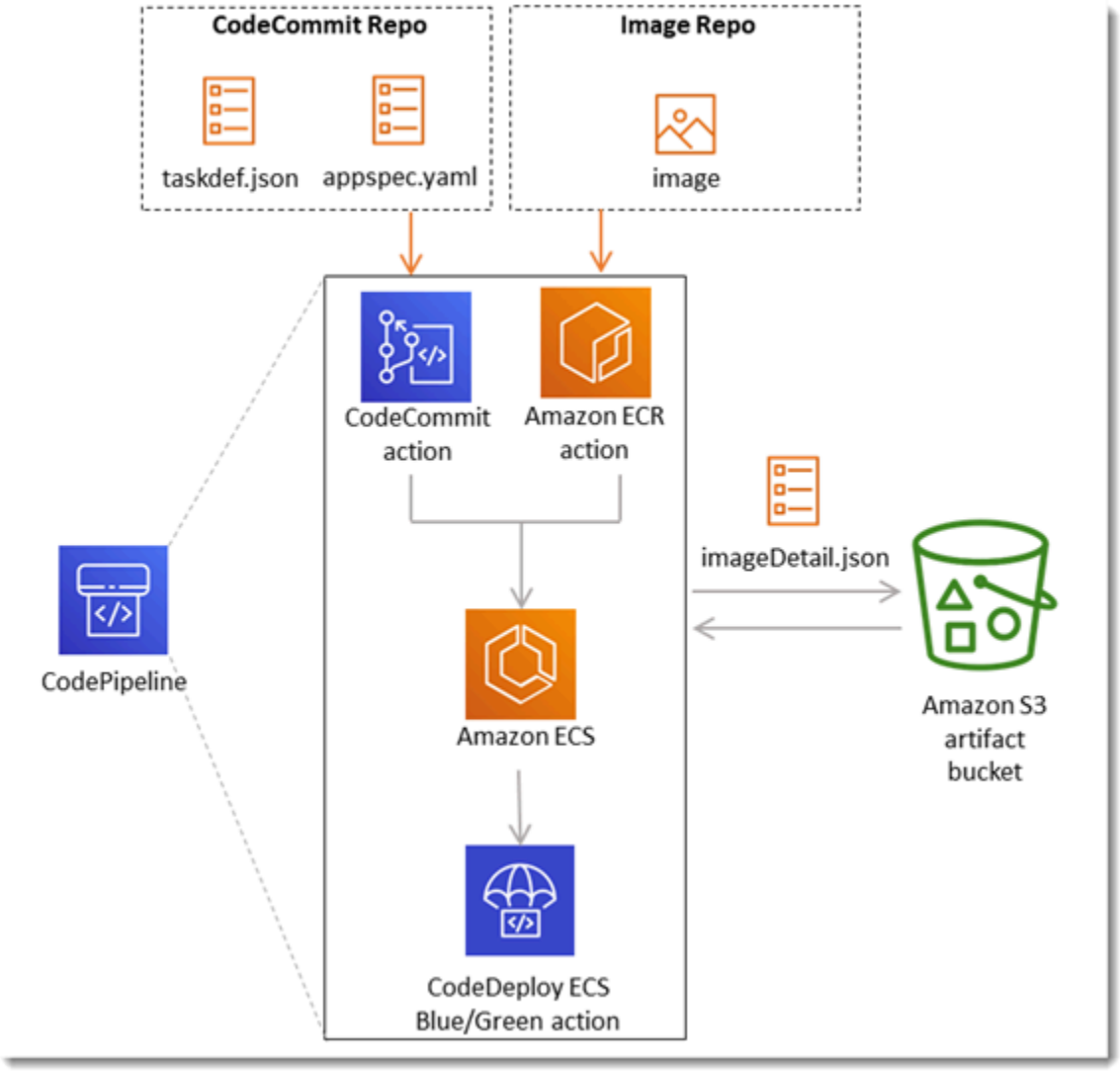
Sie müssen die Datei `imageDetail.json` als Quell- oder Build-Artefakt erstellen, damit sie ein Eingabeartefakt für die Bereitstellungsaktion ist. Sie können eine dieser Methoden verwenden, um die `imageDetail.json`-Datei in der Pipeline bereitzustellen:

- Fügen Sie die `imageDetail.json` Datei an Ihrem Quellspeicherort hinzu, sodass sie in der Pipeline als Eingabe für Ihre Amazon ECS Blue/Green-Bereitstellungsaktion bereitgestellt wird.

  > **Note**
  >
  > Wenn es sich bei Ihrem Quell-Repository um einen Amazon S3 S3-Bucket handelt, denken Sie daran, die JSON-Datei zu komprimieren.

- Amazon ECR-Quellaktionen generieren automatisch eine `imageDetail.json` Datei als Eingabeartefakt für die nächste Aktion.

  > **Note**
  >
  > Da die Amazon ECR-Quellaktion diese Datei erstellt, müssen Pipelines mit einer Amazon ECR-Quellaktion keine Datei manuell bereitstellen. `imageDetail.json`
  > Ein Tutorial zum Erstellen einer Pipeline, die eine Amazon ECR-Quellphase enthält, finden Sie unterTutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung CodeDeploy .



Die `imageDetail.json`-Datei stellt den Abbild-URI bereit. Er muss mit dem Schlüssel-Wert-Paar konstruiert werden.

Schlüssel	Wert
ImageURI	*image_URI*

imageDetail.json

Dies ist die JSON-Struktur, wobei der Abbild-URI `ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-image-repo@sha256:example3` ist:

```
{
```

```
"ImageURI": "ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-image-
repo@sha256:example3"
}
```

imageDetail.json (generated by ECR)

Jedes Mal, wenn eine Änderung in das Image-Repository übertragen wird, wird von der Amazon ECR-Quellaktion automatisch eine `imageDetail.json` Datei generiert. Die von Amazon ECR `imageDetail.json` generierten Quellaktionen werden als Ausgabeartefakt von der Quellaktion zur nächsten Aktion in der Pipeline bereitgestellt.

Dies ist die JSON-Struktur, wobei der Name des Repositorys `dk-image-repo`, der Abbild-URI `ecs-repo` und das Image-Tag `latest` ist:

```
{
    "ImageSizeInBytes": "44728918",
    "ImageDigest":
 "sha256:EXAMPLE1122334455667788990bfea42ea2d3b8a1ee8329ba7e68694950afd3",
    "Version": "1.0",
    "ImagePushedAt": "Mon Jan 21 20:04:00 UTC 2019",
    "RegistryId": "EXAMPLE12233",
    "RepositoryName": "dk-image-repo",
    "ImageURI": "ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-image-
repo@sha256:example3",
    "ImageTags": [
        "latest"
    ]
}
```

Die `imageDetail.json` Datei ordnet den Image-URI und den Container-Namen der Amazon ECS-Aufgabendefinition wie folgt zu:

- `ImageSizeInBytes`: Die Größe des Abbilds im Repository in Bytes.
- `ImageDigest`: Der `sha256`-Digest des Abbildmanifests.
- `Version`: Die Abbildversion.
- `ImagePushedAt`: Datum und Uhrzeit der Push-Übertragung des neuesten Abbilds an das Repository.
- `RegistryId`: Die AWS Konto-ID, die der Registrierung zugeordnet ist, die das Repository enthält.

- `RepositoryName`: Der Name des Amazon ECR-Repositorys, in das das Bild übertragen wurde.
- `ImageURI`: Der URI für das Abbild.
- `ImageTags`: Das Tag für das Abbild.

Führen Sie vor Erstellung Ihrer Pipeline die die folgenden Schritte aus, um die `imageDetail.json` einzurichten.

1. Planen Sie während der Planung der Container-basierten Blau/Grün-Anwendungsbereitstellung für Ihre Pipeline die Quell- und Build-Phase, wenn zutreffend.
2. Wählen Sie eine der folgenden Optionen aus:

   a. Wenn Ihre Pipeline die Erstellungsphase übersprungen hat, müssen Sie die JSON-Datei manuell erstellen und in Ihr Quell-Repository hochladen, z. B. CodeCommit damit die Quellaktion das Artefakt bereitstellen kann. Erstellen Sie die Datei mit einem Texteditor und geben Sie der Datei einen Namen oder verwenden Sie den Standarddateinamen `imageDetail.json`. Übertragen Sie die `imageDetail.json`-Datei per Push zu Ihrem Quell-Repository.

   b. Wenn Ihre Pipeline eine Build-Phase enthält, führen Sie die folgenden Aktionen aus:

      i. Fügen Sie Ihrer Build-Spezifikationsdatei einen Befehl hinzu, der die Abbild-Definitionsdatei während der Build-Phase in Ihr Quell-Repository ausgibt. Im folgenden Beispiel wird der Befehl printf verwendet, um eine `imageDetail.json`-Datei zu erstellen. Listen Sie diesen Befehl im Abschnitt `post_build` der Datei buildspec.yml auf:

      ```
      printf '{"ImageURI":"image_URI"}' > imageDetail.json
      ```

      Sie müssen die Datei `imageDetail.json` als Ausgabeartefakt in die Datei `buildspec.yml` einfügen.

      ii. Fügen Sie die Datei `imageDetail.json` als Artefakt zur Datei `buildspec.yml` hinzu.

      ```
      artifacts:
        files:
          - imageDetail.json
      ```

Variablen

Dieser Abschnitt dient nur als Referenz. Informationen zum Erstellen von Variablen finden Sie unter Arbeiten mit Variablen.

Variablen ermöglichen es Ihnen, Ihre Pipeline-Aktionen mit Werten zu konfigurieren, die zum Zeitpunkt der Pipeline-Ausführung oder der Aktionsausführung bestimmt werden.

Einige Aktionsanbieter erzeugen einen definierten Satz von Variablen. Sie wählen aus Standardvariablenschlüsseln für diesen Aktionsanbieter aus, z. B. eine Commit-ID.

> **Important**
>
> Geben Sie bei der Übergabe geheimer Parameter den Wert nicht direkt ein. Der Wert wird als Klartext gerendert und ist daher lesbar. Verwenden Sie aus Sicherheitsgründen keinen Klartext mit Geheimnissen. Wir empfehlen dringend, die Verwendung AWS Secrets Manager zum Speichern von Geheimnissen zu verwenden.

Um step-by-step Beispiele für die Verwendung von Variablen zu sehen:

- Ein Tutorial mit einer Variablen auf Pipelineebene, die bei der Ausführung der Pipeline übergeben wird, finden Sie unter. Tutorial: Variablen auf Pipeline-Ebene verwenden
- Ein Tutorial mit einer Lambda-Aktion, die Variablen aus einer Upstream-Aktion (CodeCommit) verwendet und Ausgabevariablen generiert, finden Sie unterTutorial: Variablen mit Lambda-Aufrufaktionen verwenden.
- Ein Tutorial mit einer AWS CloudFormation Aktion, die auf Stack-Ausgabevariablen aus einer CloudFormation Upstream-Aktion verweist, finden Sie unterTutorial: Eine Pipeline erstellen, die Variablen aus AWS CloudFormation Bereitstellungsaktionen verwendet.
- Ein Beispiel für eine manuelle Genehmigungsaktion mit Nachrichtentext, der auf Ausgabevariablen verweist, die in die CodeCommit Commit-ID und die Commit-Nachricht aufgelöst werden, finden Sie unterBeispiel: Variablen in manuellen Genehmigungen verwenden.
- Ein Beispiel für eine CodeBuild Aktion mit einer Umgebungsvariablen, die in den GitHub Branch-Namen aufgelöst wird, finden Sie unterBeispiel: Verwenden Sie eine BranchName Variable mit CodeBuild Umgebungsvariablen.
- CodeBuild Aktionen erzeugen alle Umgebungsvariablen, die als Teil des Builds exportiert wurden, als Variablen. Weitere Informationen finden Sie unter CodeBuild Ausgabevariablen für Aktionen.

Variablengrenzwerte

Informationen zu Grenzwerten finden Sie unter Kontingente in AWS CodePipeline.

> **Note**
>
> Wenn Sie die Variablensyntax in die Aktionskonfigurationsfelder eingeben, dürfen Sie das Limit von 1000 Zeichen für die Konfigurationsfelder nicht überschreiten. Ein Validierungsfehler wird zurückgegeben, wenn dieser Grenzwert überschritten wird.

Themen

Konzepte

In diesem Abschnitt werden wichtige Begriffe und Konzepte aufgeführt, die sich auf Variablen und Namespaces beziehen.

Variablen

Variablen sind Schlüssel-Wert-Paare, die Sie für die dynamische Konfiguration von Aktionen in Ihrer Pipeline verwenden können. Derzeit gibt es drei Möglichkeiten, wie diese Variablen verfügbar gemacht werden können:

- Zu Beginn jeder Pipeline-Ausführung gibt es einen Variablensatz, der implizit verfügbar ist. Dieser Satz enthält zurzeit `PipelineExecutionId`, die ID der aktuellen Pipeline-Ausführung.
- Variablen auf Pipeline-Ebene werden definiert, wenn die Pipeline erstellt und zur Laufzeit der Pipeline aufgelöst wird.

  Sie geben Variablen auf Pipeline-Ebene an, wenn die Pipeline erstellt wird, und Sie können Werte zum Zeitpunkt der Pipeline-Ausführung angeben.

- Es gibt Aktionstypen, die Variablensätze erzeugen, wenn sie ausgeführt werden. Sie können sehen, welche Variablen durch eine Aktion erzeugt werden, indem Sie das `outputVariables` Feld überprüfen, das Teil der API ist. ListActionExecutions Die Liste der verfügbaren Schlüsselnamen nach Aktionsanbieter finden Sie unter Für Pipeline-Aktionen verfügbare Variablen. Informationen darüber, welche Variablen die einzelnen Aktionstypen erzeugen, finden Sie unter CodePipeline Referenz der Aktionsstruktur.

Um in Ihrer Aktionskonfiguration auf diese Variablen zu verweisen, müssen Sie die Variablenreferenzsyntax mit dem richtigen Namespace verwenden.

Ein Beispiel für einen Variablen-Workflow finden Sie unter Konfigurieren von Variablen .

Namespaces

Um sicherzustellen, dass Variablen eindeutig referenziert werden können, müssen sie einem Namespace zugewiesen werden. Nachdem Sie einen Satz von Variablen einem Namespace zugewiesen haben, können sie in einer Aktionskonfiguration durch Verwendung des Namespace und Variablenschlüssels mit der folgenden Syntax referenziert werden:

```
#{namespace.variable_key}
```

Es gibt drei Arten von Namespaces, unter denen Variablen zugewiesen werden können:

- Der reservierte Namespace codepipeline

  Dies ist der Namespace, der dem Satz impliziter Variablen zugewiesen ist, die zu Beginn der einzelnen Pipeline-Ausführungen verfügbar sind. Dieser Namespace ist `codepipeline`. Beispiel für eine Variablenreferenz:

  ```
  #{codepipeline.PipelineExecutionId}
  ```

- Der Variablen-Namespace auf Pipeline-Ebene

  Dies ist der Namespace, der Variablen auf Pipeline-Ebene zugewiesen ist. Der Namespace für alle Variablen auf Pipeline-Ebene ist. `variables` Beispiel für eine Variablenreferenz:

  ```
  #{variables.variable_name}
  ```

- Einer Aktion zugewiesener Namespace

Dies ist ein Namespace, den Sie einer Aktion zuweisen. Alle Variablen, die von der Aktion erzeugt werden, werden diesem Namespace zugewiesen. Um die von einer Aktion erzeugten Variablen für die Verwendung in einer nachgeschalteten Aktionskonfiguration verfügbar zu machen, müssen Sie die erzeugende Aktion mit einem Namespace konfigurieren. Namespaces müssen in der Pipeline-Definition eindeutig sein und dürfen nicht mit Artefaktnamen in Konflikt stehen. Dies ist ein Beispiel für eine Variablenreferenz für eine Aktion, die mit dem Namespace `SourceVariables` konfiguriert ist.

```
#{SourceVariables.VersionId}
```

Anwendungsfälle für Variablen

Im Folgenden sind einige der häufigsten Anwendungsfälle für Variablen auf Pipeline-Ebene aufgeführt, anhand derer Sie ermitteln können, wie Sie Variablen für Ihre spezifischen Anforderungen verwenden könnten.

- Variablen auf Pipeline-Ebene sind für CodePipeline Kunden vorgesehen, die jedes Mal dieselbe Pipeline verwenden möchten, mit geringfügigen Abweichungen bei den Eingaben für die Aktionskonfiguration. Jeder Entwickler, der eine Pipeline startet, fügt den Variablenwert in der Benutzeroberfläche hinzu, wenn die Pipeline gestartet wird. Mit dieser Konfiguration übergeben Sie nur Parameter für diese Ausführung.
- Mit Variablen auf Pipeline-Ebene können Sie dynamische Eingaben an Aktionen in der Pipeline übergeben. Sie können Ihre parametrisierten Pipelines zu migrieren, CodePipeline ohne verschiedene Versionen derselben Pipeline verwalten oder komplexe Pipelines erstellen zu müssen.
- Sie können Variablen auf Pipeline-Ebene verwenden, um Eingabeparameter zu übergeben, die es Ihnen ermöglichen, eine Pipeline bei jeder Ausführung wiederzuverwenden, z. B. wenn Sie angeben möchten, welche Version Sie in einer Produktionsumgebung bereitstellen möchten, sodass Sie Pipelines nicht duplizieren müssen.
- Sie können eine einzelne Pipeline verwenden, um Ressourcen für mehrere Build- und Bereitstellungsumgebungen bereitzustellen. Bei einer Pipeline mit einem CodeCommit Repository kann die Bereitstellung beispielsweise von einer bestimmten Branch- und Zielbereitstellungsumgebung aus mit CodeBuild übergebenen CodeDeploy Parametern auf Pipeline-Ebene erfolgen.

Konfigurieren von Variablen

Sie können Variablen auf Pipeline-Ebene oder Aktionsebene in der Pipeline-Struktur konfigurieren.

Variablen auf Pipeline-Ebene konfigurieren

Sie können eine oder mehrere Variablen auf Pipeline-Ebene hinzufügen. Sie können in der Konfiguration von CodePipeline Aktionen auf diesen Wert verweisen. Sie können die Variablennamen, Standardwerte und Beschreibungen hinzufügen, wenn Sie die Pipeline erstellen. Variablen werden zum Zeitpunkt der Ausführung aufgelöst.

> **Note**
>
> Wenn für eine Variable auf Pipeline-Ebene kein Standardwert definiert ist, wird die Variable als erforderlich betrachtet. Sie müssen beim Starten einer Pipeline Überschreibungen für alle erforderlichen Variablen angeben, da andernfalls die Pipelineausführung mit einem Validierungsfehler fehlschlägt.

Sie stellen Variablen auf Pipeline-Ebene mithilfe des Variablen-Attributs in der Pipeline-Struktur bereit. Im folgenden Beispiel `Variable1` hat die Variable den Wert`Value1`.

```
"variables": [
     {
         "name": "Variable1",
         "defaultValue": "Value1",
         "description": "description"
     }
 ]
```

Ein Beispiel für die JSON-Struktur der Pipeline finden Sie unterErstellen Sie eine Pipeline in CodePipeline.

Ein Tutorial mit einer Variablen auf Pipeline-Ebene, die bei der Ausführung der Pipeline übergeben wird, finden Sie unter. Tutorial: Variablen auf Pipeline-Ebene verwenden

Beachten Sie, dass die Verwendung von Variablen auf Pipeline-Ebene in jeder Art von Quellaktion nicht unterstützt wird.

> **Note**
>
> Wenn der `variables` Namespace bereits in einigen Aktionen innerhalb der Pipeline verwendet wird, müssen Sie die Aktionsdefinition aktualisieren und einen anderen Namespace für die widersprüchliche Aktion auswählen.

Variablen auf Aktionsebene konfigurieren

Sie konfigurieren eine Aktion zur Erzeugung von Variablen, indem Sie einen Namespace für die Aktion deklarieren. Bei der Aktion muss es sich bereits um einen Aktionsanbieter handeln, der Variablen generiert. Andernfalls handelt es sich bei den verfügbaren Variablen um Variablen auf Pipeline-Ebene.

Sie deklarieren den Namespace auf eine von zwei Arten:

- Sie geben auf der Seite Edit action (Aktion bearbeiten) in der Konsole in Variable namespace (Variablen-Namespace) einen Namespace ein.
- Sie geben in das `namespace`-Parameterfeld in der JSON-Pipelinestruktur einen Namespace ein.

In diesem Beispiel fügen Sie der CodeCommit Quellaktion den `namespace` Parameter mit dem Namen hinzu`SourceVariables`. Hierdurch wird die Aktion für die Erzeugung der Variablen konfiguriert, die für diesen Aktionsanbieter verfügbar sind, z. B. `CommitId`.

```
{
    "name": "Source",
    "actions": [
        {
            "outputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "name": "Source",
            "namespace": "SourceVariables",
            "configuration": {
                "RepositoryName": "MyRepo",
                "BranchName": "mainline",
                "PollForSourceChanges": "false"
            },
```

```
            "inputArtifacts": [],
            "region": "us-west-2",
            "actionTypeId": {
                "provider": "CodeCommit",
                "category": "Source",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

Als Nächstes konfigurieren Sie die nachgeschaltete Aktion für die Verwendung der von der vorherigen Aktion erzeugten Variablen. Sie tun dies wie folgt:

- Sie geben auf der Seite Edit action (Aktion bearbeiten) der Konsole die Variablensyntax (für die nachgeschaltete Aktion) in die Aktionskonfigurationsfelder ein.
- Sie geben die Variablensyntax (für die nachgeschaltete Aktion) in die Aktionskonfigurationsfelder in der JSON-Pipelinestruktur ein.

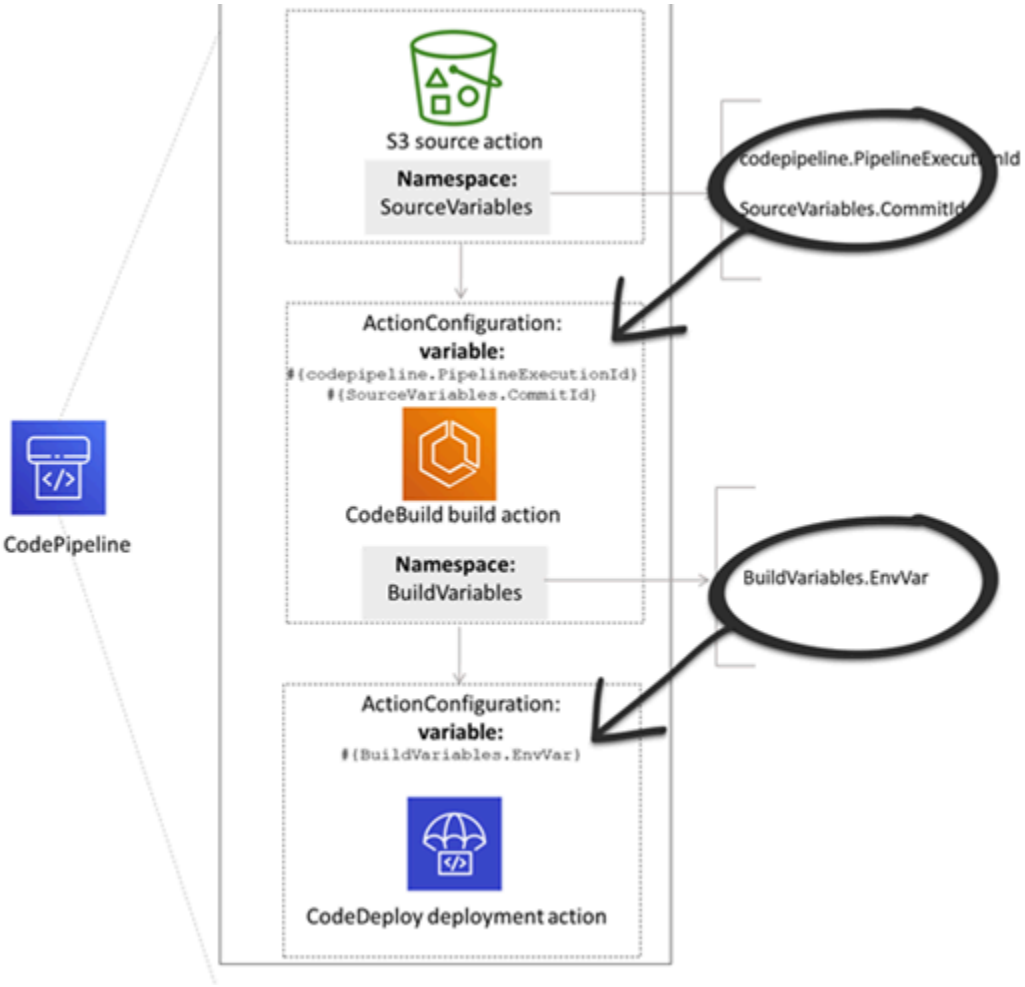
In diesem Beispiel zeigt das Konfigurationsfeld der Build-Aktion Umgebungsvariablen an, die bei Aktionsausführung aktualisiert werden. Das Beispiel gibt den Namespace und die Variable für die Ausführungs-ID mit `#{codepipeline.PipelineExecutionId}` sowie den Namespace und die Variable für die Commit-ID mit **`#{SourceVariables.CommitId}`** an.

```
{
    "name": "Build",
    "actions": [
        {
            "outputArtifacts": [
                {
                    "name": "BuildArtifact"
                }
            ],
            "name": "Build",
            "configuration": {
                "EnvironmentVariables": "[{\"name\":\"Release_ID\",\"value\":
\"#{codepipeline.PipelineExecutionId}\",\"type\":\"PLAINTEXT\"},{\"name\":\"Commit_ID
\",\"value\":\"#{SourceVariables.CommitId}\",\"type\":\"PLAINTEXT\"}]",
                "ProjectName": "env-var-test"
```

```
            },
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-west-2",
            "actionTypeId": {
                "provider": "CodeBuild",
                "category": "Build",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

Variablenauflösung

Jedes Mal, wenn eine Aktion als Teil einer Pipeline-Ausführung ausgeführt wird, stehen die so erzeugten Variablen für jede Aktion zur Verfügung, die nach der erzeugenden Aktion garantiert ausgeführt wird. Um diese Variablen in einer verbrauchenden Aktion zu verwenden, können Sie sie mithilfe der im vorherigen Beispiel gezeigten Syntax zur Konfiguration der verbrauchenden Aktion hinzufügen. CodePipeline Löst vor der Ausführung einer aufwändigen Aktion alle Variablenverweise auf, die in der Konfiguration vorhanden sind, bevor die Ausführung der Aktion eingeleitet wird.



Regeln für Variablen

Die folgenden Regeln helfen Ihnen bei der Konfiguration von Variablen:

- Sie geben den Namespace und die Variable für eine Aktion über eine neue Aktionseigenschaft oder durch Bearbeiten einer Aktion an.
- Wenn Sie den Pipeline-Erstellungsassistenten verwenden, generiert die Konsole für jede mit dem Assistenten erstellte Aktion einen Namespace.
- Wenn kein Namespace angegeben ist, können die von dieser Aktion erzeugten Variablen in keiner Aktionskonfiguration referenziert werden.
- Um von einer Aktion erzeugte Variablen zu referenzieren, muss die referenzierende Aktion nach der Aktion ausgeführt werden, die die Variablen erzeugt. Das bedeutet, dass sie sich entweder in einer späteren Phase als die Aktion befindet, die die Variablen erzeugt, oder in derselben Phase, jedoch in einer höheren Ausführungsrangstufe.

Für Pipeline-Aktionen verfügbare Variablen

Der Aktionsanbieter bestimmt, welche Variablen von der Aktion generiert werden können.

step-by-step Verfahren zur Verwaltung von Variablen finden Sie unter. Arbeiten mit Variablen

Aktionen mit definierten Variablenschlüsseln

Im Gegensatz zu einem Namespace, den Sie wählen können, verwenden die folgenden Aktionen variable Schlüssel, die nicht bearbeitet werden können. Für den Amazon S3 S3-Aktionsanbieter sind beispielsweise nur die Schlüssel `ETag` und die `VersionId` Variablen verfügbar.

Jede Ausführung verfügt außerdem über eine Reihe von CodePipeline -generierten Pipeline-Variablen, die Daten über die Ausführung enthalten, z. B. die Pipeline-Release-ID. Diese Variablen können von jeder Aktion in der Pipeline verbraucht werden.

Themen

- CodePipelineVariable mit der Ausführungs-ID
- Ausgabevariablen für Amazon ECR-Aktionen
- AWS CloudFormation StackSets Aktionsausgabevariablen
- CodeCommit Aktionsausgabevariablen
- CodeStarSourceConnection Aktionsausgabevariablen
- GitHub Aktionsausgabevariablen (GitHub Aktionsversion 1)
- Ausgabevariablen für S3-Aktionen

CodePipelineVariable mit der Ausführungs-ID

CodePipelineAusführungs-ID-Variable

Anbieter	Variablenschlüssel	Beispielwert	Beispiel für eine Variablensyntax
codepipeline	`PipelineExecutionId`	8abc75f0-fbf8-4f4c-bfEXAMPLE	`#{codepipeline.PipelineExecutionId}`

Ausgabevariablen für Amazon ECR-Aktionen

Amazon ECR-Variablen

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`ImageDigest`	sha256:EXAMPLE1122334455	`#{SourceVariables.ImageDigest}`
`ImageTag`	brandneue	`#{SourceVariables.ImageTag}`
`ImageURI`	11111EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest	`#{SourceVariables.ImageURI}`
`RegistryId`	EXAMPLE12233	`#{SourceVariables.RegistryId}`
`RepositoryName`	my-image-repo	`#{SourceVariables.RepositoryName}`

AWS CloudFormation StackSets Aktionsausgabevariablen

AWS CloudFormation StackSets Variablen

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`OperationId`	11111111-2bbb-111-2bbb-11111 Beispiel	`#{DeployVariables.OperationId}`
`StackSetId`	my-stackset:1111aaaa-1111-2222-2bbb-11111 Beispiel	`#{DeployVariables.StackSetId}`

CodeCommit Aktionsausgabevariablen

CodeCommit Variablen

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`AuthorDate`	2019-10-29T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	Entwicklung	`#{SourceVariables.BranchName}`
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Behebung eines Fehlers (maximale Größe 100 KB)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	2019-10-29T03:32:21Z	`#{SourceVariables.CommitterDate}`
`RepositoryName`	myCodeCommitRepo	`#{SourceVariables.RepositoryName}`

CodeStarSourceConnection Aktionsausgabevariablen

`CodeStarSourceConnection`Variablen (Bitbucket Cloud GitHub, GitHub Enterprise Repository und GitLab .com)

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`AuthorDate`	2019-10-29T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	Entwicklung	`#{SourceVariables.BranchName}`

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Behebung eines Fehlers (maximale Größe 100 KB)	`#{SourceVariables.CommitMessage}`
`ConnectionArn`	*arn:aws:codestar-connections:region:konto-id:verbindung/verbindungs-ID*	`#{SourceVariables.ConnectionArn}`
`FullRepositoryName`	Benutzername/ GitHubRepo	`#{SourceVariables.FullRepositoryName}`

GitHub Aktionsausgabevariablen (GitHub Aktionsversion 1)

GitHub Variablen (GitHub Aktionsversion 1)

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`AuthorDate`	2019-10-29T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	Haupt	`#{SourceVariables.BranchName}`
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Behebung eines Fehlers (maximale Größe 100 KB)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	2019-10-29T03:32:21Z	`#{SourceVariables.CommitterDate}`

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`CommitUrl`		`#{SourceVariables.CommitUrl}`
`RepositoryName`	myGitHubRepo	`#{SourceVariables.RepositoryName}`

Ausgabevariablen für S3-Aktionen

S3-Variablen

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`ETag`	example28be1c3	`#{SourceVariables.ETag}`
`VersionId`	exampleta_IUQCv	`#{SourceVariables.VersionId}`

Aktionen mit vom Benutzer konfigurierten Variablentasten

Für CodeBuild AWS CloudFormation, und Lambda-Aktionen werden die variablen Schlüssel vom Benutzer konfiguriert.

Themen

- CloudFormation Aktionsausgabevariablen
- CodeBuild Ausgabevariablen für Aktionen
- Ausgangsvariablen für Lambda-Aktionen

CloudFormation Aktionsausgabevariablen

AWS CloudFormation Variablen

Variablenschlüssel	Beispiel für eine Variablensyntax
Bei AWS CloudFormation Aktionen werden Variablen aus beliebigen Werten erzeugt, die im `Outputs` Abschnitt einer Stack-Vorlage angegeben sind. Beachten Sie, dass die einzigen CloudFormation Aktionsmodi, die Ausgaben generieren, diejenigen sind, die zur Erstellung oder Aktualisierung eines Stacks führen, wie z. B. die Erstellung von Stacks, Stack-Aktualisierungen und die Ausführung von Änderungssätzen. Die entsprechenden Aktionsmodi, die Variablen generieren, sind: • CREATE_UPDATE • CHANGE_SET_EXECUTE • CHANGE_SET_REPLACE • REPLACE_ON_FAILURE Weitere Informationen zu diesen Aktionsmodi finden Sie unterAWS CloudFormation. Ein Tutorial, das Ihnen zeigt, wie Sie eine Pipeline mit einer AWS CloudFormation Bereitstellungsaktion in einer Pipeline erstellen, die AWS CloudFormation Ausgabevariablen verwendet, finden Sie unterTutorial: Eine Pipeline erstellen, die Variablen aus AWS CloudFormation Bereitstellungsaktionen verwendet.	`#{DeployVariables.StackName}`

CodeBuild Ausgabevariablen für Aktionen

CodeBuild Variablen

Variablenschlüssel	Beispiel für eine Variablensyntax
Bei CodeBuild Aktionen werden Variablen aus Werten erzeugt, die von exportierten Umgebungsvariablen generiert	`#{BuildVariables.EnvVar}`

Variablenschlüssel	Beispiel für eine Variablensyntax
wurden. Richten Sie eine CodeBuild Umgebungsvariable ein, indem Sie Ihre CodeBuild Aktion in der Build-Spezifikation bearbeiten CodePipeline oder die Umgebungsvariable zur Build-Spezifikation hinzufügen. Fügen Sie Ihrer CodeBuild Build-Spezifikation Anweisungen hinzu, um die Umgebungsvariable im Abschnitt exportierte Variablen hinzuzufügen. Weitere Informationen finden Sie unter env/exported-variables im Benutzerhandbuch.AWS CodeBuild	

Ausgangsvariablen für Lambda-Aktionen

Lambda-Variablen

Variablenschlüssel	Beispiel für eine Variablensyntax
Die Lambda-Aktion erzeugt als Variablen alle Schlüssel-Wert-Paare, die im `outputVariables` Abschnitt der PutJobSuccessResult API-Anfrage enthalten sind. Ein Tutorial mit einer Lambda-Aktion, die Variablen aus einer Upstream-Aktion (CodeCommit) verwendet und Ausgabevariablen generiert, finden Sie unterTutorial: Variablen mit Lambda-Aufrufaktionen verwenden.	`#{TestVariables.testRunId}`

Arbeiten mit Glob-Mustern in der Syntax

Wenn Sie die Dateien oder Pfade angeben, die in Pipeline-Artefakten oder Quellverzeichnissen verwendet werden, können Sie das Artefakt je nach Aktionstyp angeben. Für die S3-Aktion geben Sie beispielsweise den S3-Objektschlüssel an.

Für Trigger können Sie Filter angeben. Sie können Glob-Muster verwenden, um Filter zu spezifizieren. Im Folgenden sind einige Beispiele aufgeführt.

Wenn die Syntax „glob“ lautet, wird der Zeichenkettendarstellung des Pfads eine eingeschränkte Mustersprache mit einer Syntax zugeordnet, die regulären Ausdrücken ähnelt. Beispielsweise:

- `*.java`Gibt einen Pfad an, der einen Dateinamen darstellt, der auf .java endet
- `*.*`Gibt Dateinamen an, die einen Punkt enthalten
- `*.{java,class}`Gibt Dateinamen an, die mit .java oder .class enden
- `foo.?`Gibt Dateinamen an, die mit foo. beginnen und eine einzelne Zeichenerweiterung haben

Die folgenden Regeln werden verwendet, um Glob-Muster zu interpretieren:

- Um null oder mehr Zeichen einer Namenskomponente in Verzeichnisgrenzen anzugeben, verwenden Sie`*`.
- Um null oder mehr Zeichen einer Namenskomponente anzugeben, die Verzeichnisgrenzen überschreitet, verwenden Sie`**`.
- Um ein Zeichen einer Namenskomponente anzugeben, verwenden Sie`?`.
- Um Zeichen zu maskieren, die andernfalls als Sonderzeichen interpretiert würden, verwenden Sie den umgekehrten Schrägstrich ()\.
- Um ein einzelnes Zeichen aus einer Reihe von Zeichen anzugeben, verwenden Sie. `[ ]`
- Um eine einzelne Datei anzugeben, die sich im Stammverzeichnis Ihres Build-Speicherorts oder Quell-Repository-Speicherorts befindet, verwenden Siemy`-file.jar`.
- Um eine einzelne Datei in einem Unterverzeichnis anzugeben, verwenden Sie `directory/my-file.jar` oder`directory/subdirectory/my-file.jar`.
- Um alle Dateien anzugeben, verwenden Sie`"**"`. Das `**` Glob-Muster gibt an, dass es einer beliebigen Anzahl von Unterverzeichnissen entspricht.

- Um alle Dateien und Verzeichnisse in einem Verzeichnis mit dem Namen anzugeben`directory`, verwenden Sie. `"directory/**"` Das `**` Glob-Muster gibt an, dass es einer beliebigen Anzahl von Unterverzeichnissen entspricht.
- Um alle Dateien in einem Verzeichnis mit dem Namen`directory`, aber nicht in einem seiner Unterverzeichnisse anzugeben, verwenden Sie. `"directory/*"`
- Innerhalb eines Ausdrucks in Klammern stimmen die \ Zeichen`*`, `?` und überein. Das Zeichen (-) entspricht sich selbst, wenn es das erste Zeichen innerhalb der Klammern ist, oder das erste Zeichen nach dem, `!` wenn negierend.
- Die `{ }` Zeichen sind eine Gruppe von Untermustern, wobei die Gruppe übereinstimmt, wenn ein Untermuster in der Gruppe übereinstimmt. Das `","` Zeichen wird verwendet, um die Untermuster zu trennen. Gruppen können nicht verschachtelt werden.

Aktivieren von Pipelines für die empfohlenen Methode zur Änderungserkennung

Wenn Sie über eine Pipeline verfügen, die Abfragen verwendet, um auf Quellenänderungen zu reagieren, können Sie sie so aktualisieren, dass sie die empfohlene Erkennungsmethode verwendet. Einen Migrationsleitfaden mit Anweisungen zur Aktualisierung Ihrer Abfrage-Pipines zur Verwendung der empfohlenen ereignisbasierten Methode zur Erkennung von Änderungen finden Sie unter. Migrieren Sie Abfrage-Pipelines, um die ereignisbasierte Änderungserkennung zu nutzen

Aktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2

AWS CodePipeline In gibt es zwei unterstützte Versionen der GitHub Quellaktion:

- Empfohlen: Die Aktion GitHub Version 2 verwendet die auf der Github-App basierende Authentifizierung, die von einer CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen Ressource unterstützt wird. Sie installiert eine AWS CodeStar Connections-Anwendung in Ihrer GitHub Organisation, sodass Sie den Zugriff verwalten können. GitHub
- Nicht empfohlen: Die Aktion GitHub Version 1 verwendet OAuth-Token zur Authentifizierung GitHub und verwendet einen separaten Webhook, um Änderungen zu erkennen. Dies ist nicht mehr die empfohlene Methode.

> **Note**
>
> Verbindungen sind in den Regionen Asien-Pazifik (Hongkong), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Afrika (Kapstadt), Naher Osten (Bahrain), Naher Osten (VAE), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv) oder AWS GovCloud (USA-West) nicht verfügbar. Weitere verfügbare Aktionen finden Sie unter. Produkt- und Serviceintegrationen mit CodePipeline Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie in der Anmerkung unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.

Die Verwendung der Aktion GitHub Version 2 gegenüber der Aktion GitHub Version 1 bietet einige wichtige Vorteile:

- Bei Verbindungen sind für den Zugriff auf Ihr Repository CodePipeline keine OAuth-Apps oder persönlichen Zugriffstoken mehr erforderlich. Wenn Sie eine Verbindung herstellen, installieren Sie eine GitHub App, die die Authentifizierung für Ihr GitHub Repository verwaltet und Berechtigungen auf Organisationsebene gewährt. Sie müssen OAuth-Token als Benutzer autorisieren, um auf das Repository zuzugreifen. Weitere Informationen zum OAuth-basierten GitHub Zugriff im Gegensatz

zum App-basierten Zugriff finden Sie unter. GitHub https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps

- Wenn Sie Aktionen der GitHub Version 2 in der CLI oder verwalten CloudFormation, müssen Sie Ihr persönliches Zugriffstoken nicht mehr als Geheimnis in Secrets Manager speichern. Sie müssen in Ihrer CodePipeline Aktionskonfiguration nicht mehr dynamisch auf das gespeicherte Geheimnis verweisen. Stattdessen fügen Sie den Verbindungs-ARN zu Ihrer Aktionskonfiguration hinzu. Ein Beispiel für eine Aktionskonfiguration finden Sie unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com und GitLab selbstverwaltete Aktionen.
- Wenn Sie eine Verbindungsressource erstellen, die Sie mit Ihrer Aktion in GitHub Version 2 verwenden möchten CodePipeline, können Sie dieselbe Verbindungsressource verwenden, um Ihrem Repository andere unterstützte Dienste wie CodeGuru Reviewer zuzuordnen.
- In Github Version 2 kannst du Repositorys klonen, um bei nachfolgenden CodeBuild Aktionen auf Git-Metadaten zuzugreifen, während du in Github Version 1 nur die Quelle herunterladen kannst.
- Ein Administrator installiert die App für die Repositorys Ihrer Organisation. Sie müssen keine OAuth-Token mehr nachverfolgen, die von der Person abhängen, die das Token erstellt hat.

Alle in einer Organisation installierten Apps haben Zugriff auf dieselben Repositorys. Um zu ändern, wer auf jedes Repository zugreifen kann, ändern Sie die IAM-Richtlinie für jede Verbindung. Ein Beispiel finden Sie unter Beispiel: Eine nach unten abgegrenzte Richtlinie für die Verwendung von Verbindungen mit einem bestimmten Repository.

Sie können die Schritte in diesem Thema verwenden, um Ihre Quellaktion für GitHub Version 1 zu löschen und eine Quellaktion für GitHub Version 2 aus der Konsole hinzuzufügen. CodePipeline

Themen

- Schritt 1: Ersetzen Sie Ihre GitHub Aktion für Version 1
- Schritt 2: Stellen Sie eine Verbindung her zu GitHub
- Schritt 3: Speichern Sie Ihre GitHub Quellaktion

Schritt 1: Ersetzen Sie Ihre GitHub Aktion für Version 1

Verwenden Sie die Pipeline-Bearbeitungsseite, um Ihre Aktion aus Version 1 durch eine GitHub Aktion aus Version 2 GitHub zu ersetzen.

Um Ihre GitHub Aktion aus Version 1 zu ersetzen

1. Melden Sie sich bei der CodePipeline Konsole an.
2. Wählen Sie Ihre Pipeline und dann Bearbeiten aus. Wählen Sie in Ihrer Quellstufe die Option Phase bearbeiten aus. Es wird eine Meldung angezeigt, in der empfohlen wird, Ihre Aktion zu aktualisieren.
3. Wählen Sie unter Aktionsanbieter die Option GitHub (Version 2) aus.
4. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Option Connect aus GitHub. Fahren Sie mit Schritt 2 fort: Verbindung herstellen zu GitHub.
   - Wenn Sie bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Verbindung aus. Fahren Sie mit Schritt 3 fort: Speichern Sie die Quellaktion für Ihre Verbindung.

Schritt 2: Stellen Sie eine Verbindung her zu GitHub

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die GitHub Seite Connect angezeigt.

Um eine Verbindung herzustellen zu GitHub

1. Unter GitHub Verbindungseinstellungen wird Ihr Verbindungsname unter Verbindungsname angezeigt.

   Wählen Sie unter GitHub Apps eine App-Installation aus oder wählen Sie Neue App installieren, um eine zu erstellen.

   > **Note**
   >
   > Sie installieren eine App für alle Verbindungen mit einem bestimmten Anbieter. Wenn Sie die GitHub App bereits installiert haben, wählen Sie sie aus und überspringen Sie diesen Schritt.

2. Wenn die Autorisierungsseite für GitHub angezeigt wird, melden Sie sich mit Ihren Anmeldeinformationen an und wählen Sie dann, ob Sie fortfahren möchten.

3. Auf der App-Installationsseite wird eine Meldung angezeigt, dass die AWS CodeStar App versucht, eine Verbindung zu Ihrem GitHub Konto herzustellen.

> **Note**
>
> Sie installieren die App nur einmal für jedes GitHub Konto. Wenn Sie die App schon einmal installiert haben, können Sie Configure (Konfiguration) wählen und mit einer Änderungsseite für die App-Installation fortfahren. Alternativ kommen Sie über die Schaltfläche „Back“ (Zurück) zur Konsole zurück.

4. Wählen Sie auf der AWS CodeStar Installationsseite die Option Installieren aus.
5. Auf der GitHub Seite Connect wird die Verbindungs-ID für Ihre neue Installation angezeigt. Wählen Sie Connect aus.

Schritt 3: Speichern Sie Ihre GitHub Quellaktion

Vervollständigen Sie Ihre Aktualisierungen auf der Seite Aktion bearbeiten, um Ihre neue Quellaktion zu speichern.

Um Ihre GitHub Quellaktion zu speichern

1. Geben Sie unter Repository den Namen Ihres Drittanbieter-Repositorys ein. Geben Sie im Feld Branch den Branch ein, in dem Ihre Pipeline Quelländerungen erkennen soll.

> **Note**
>
> Geben Sie im Feld Repository `owner-name/repository-name` wie in diesem Beispiel gezeigt ein:
>
> ```
> my-account/my-repository
> ```

2. Wählen Sie unter Ausgabeartefaktformat das Format für Ihre Artefakte aus.

   - Um die Ausgabeartefakte der GitHub Aktion mit der Standardmethode zu speichern, wählen Sie CodePipeline Standard. Die Aktion greift auf die Dateien aus dem GitHub Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.

- Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone (Vollständiger Klon). Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

  Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschriebenFügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, Enterprise Server oder .com GitHub hinzu GitHub GitLab. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiges Klonen verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

3. Unter Ausgabeartefakte können Sie den Namen des Ausgabeartefakts für diese Aktion beibehalten, z. B. `SourceArtifact` Wählen Sie Fertig, um die Aktionsseite Bearbeiten zu schließen.
4. Wählen Sie „Fertig“, um die Seite zur Bearbeitung der Phase zu schließen. Wählen Sie Speichern, um die Seite zur Bearbeitung der Pipeline zu schließen.

Kontingente in AWS CodePipeline

CodePipeline enthält Kontingente für die Anzahl der Pipelines, Phasen, Aktionen und Webhooks, die ein AWS Konto in jeder AWS Region haben kann. Diese Kontingente gelten pro Region und können erhöht werden. Um eine Erhöhung anzufordern, verwenden Sie die Support-Center-Konsole.

Die Verarbeitung von Anträgen zur Kontingenterhöhung kann bis zu zwei Wochen dauern.

Ressource	Standard
Dauer bis zum Timeout einer Aktion (Dies sind konfigurierbare Timeouts. In der folgenden Tabelle finden Sie nicht konfigurierbare Timeouts)	AWS CloudFormation Bereitstellungsaktion: 3 Tage CodeDeploy und CodeDeploy ECS-Implementierungsmaßnahmen (blau/grün): 5 Tage AWS Lambda Aktion aufrufen: 24 Stunden **Note** Während die Aktion ausgeführt wird, CodePipeline fragt Lambda regelmäßig nach einem Status. Die Lambda-Funktion antwortet mit einem Status, in dem die Ausführung der Aktion entweder erfolgreich, fehlgeschlagen oder in Bearbeitung ist. Wenn die Lambda-Funktion nach 20 Minuten keine Antwort gesendet hat, läuft die Aktion ab. Wenn die Lambda-Funktion während der 20 Minuten geantwortet hat, dass die Aktion noch läuft, CodePipeline startet sie den 20-Minuten-Timer neu und versucht es erneut. Wenn der Vorgang nach 24 Stunden nicht erfolgreich ist, wird

Ressource	Standard
	der Status der Lambda-Aufrufaktion auf „Fehlgeschlagen“ CodePipeline gesetzt. Lambda hat ein separates Timeout für Lambda-Funktionen, das nichts mit dem Aktions-Timeout zu tun hat. CodePipeline Amazon S3 S3-Bereitstellungsaktion: 90 Minuten **Note** Wenn beim Upload auf S3 während der Bereitstellung einer großen ZIP-Datei ein Timeout auftritt, schlägt die Aktion mit einem Timeout-Fehler fehl. Versuchen Sie, die ZIP-Datei in kleinere Dateien aufzuteilen. Standardtimeout für manuelle Genehmigungsaktionen auf Kontoebene: 7 Tage **Note** Das Standard-Timeout für die manuelle Genehmigungsaktion kann für eine bestimmte Aktion in der Pipeline außer Kraft gesetzt werden. Es kann auf 86400 Minuten (60 Tage) mit einem Mindestwert von 5 Minuten konfiguriert werden. Weitere Informationen finden Sie

Ressource	Standard
	ActionDeclarationin der API-Referenz. CodePipeline Wenn konfiguriert, wird dieses Timeout für die Aktion angewendet. Andernfalls wird die Standardeinstellung auf Kontoebene verwendet. Alle anderen Aktionen; 1 Stunde **Note** Das Zeitlimit für Amazon ECS-Bereitstellungsaktionen ist auf bis zu einer Stunde konfigurierbar (das Standard-Timeout).
Maximale Gesamtzahl der Pipelines pro Region in einem Konto AWS	1000 **Note** Pipelines, die für die Polling- oder ereignisbasierte Änderungserkennung konfiguriert sind, werden auf dieses Kontingent angerechnet.

Ressource	Standard
Maximale Anzahl von Pipelines, bei denen Quellenänderungen abgefragt werden sollen, pro Region AWS	300 **Note** Dieses Kontingent ist fest und kann nicht geändert werden. Wenn Sie das Limit für Polling-Pipelines erreichen, können Sie trotzdem zusätzliche Pipelines konfigurieren, die die ereignisbasierte Änderungserkennung verwenden. Weitere Informationen finden Sie unter Quellaktionen und Methoden zur Änderungserkennung.[1]
Maximale Anzahl von Webhooks pro Region in einem Konto AWS	300
Anzahl der benutzerdefinierten Aktionen pro Region in einem Konto AWS	50

[1]Führen Sie basierend auf Ihrem Quellanbieter die folgenden Anweisungen aus, um Ihre Polling-Pipelines zu aktualisieren, um die ereignisbasierte Änderungserkennung zu verwenden:

- Informationen zum Aktualisieren einer CodeCommit Quellaktion finden Sie unterMigrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole).
- Informationen zum Aktualisieren einer Amazon S3 S3-Quellaktion finden Sie unterMigrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole).
- Informationen zum Aktualisieren einer GitHub Quellaktion finden Sie unterMigrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (Konsole).

Die folgenden Kontingente AWS CodePipeline gelten für die Verfügbarkeit in der Region, Benennungsbeschränkungen und zulässige Artefaktgrößen. Diese Kontingente stehen fest und können nicht geändert werden.

Eine Liste der CodePipeline Dienstendpunkte für jede Region finden Sie unter AWS CodePipeline Endpunkte und Kontingente in der AWS Allgemeinen Referenz.

Weitere Informationen zu strukturellen Anforderungen finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

AWS Regionen, in denen Sie eine Pipeline erstellen können	US East (Ohio) USA Ost (Nord-Virginia) USA West (Nordkalifornien) USA West (Oregon) Kanada (Zentral) Europa (Frankfurt) Europa (Zürich) * Israel (Tel Aviv) Europa (Irland) Europe (London) Europa (Mailand) * Europa (Paris) Europa (Spain) Europa (Stockholm) Afrika (Kapstadt) * Asien-Pazifik (Hongkong) * Asien-Pazifik (Hyderabad) Asien-Pazifik (Mumbai) Asien-Pazifik (Tokio)

	Asien-Pazifik (Seoul) Asien-Pazifik (Osaka) Asien-Pazifik (Singapur) Asien-Pazifik (Sydney) Asien-Pazifik (Jakarta) Asien-Pazifik (Melbourne) Südamerika (São Paulo) Naher Osten (Bahrain) * Naher Osten (VAE) AWS GovCloud (US-West) AWS GovCloud (US-Ost)
Zulässige Zeichen in einem Aktionsnamen	Aktionsnamen dürfen nicht mehr als 100 Zeichen enthalten. Zulässige Zeichen sind: Kleinbuchstaben `a` bis `z`, einschließlich. Großbuchstaben `A` bis `Z`, einschließlich. Zahlen `0` bis `9`, einschließlich. Sonderzeichen `.` (Punkt), `@` (At-Zeichen), `-` (Minuszeichen) und `_` (Unterstrich). Andere Zeichen, wie beispielsweise Leerzeichen, sind nicht zulässig.

Zulässige Zeichen in Aktionstypnamen	Aktionstypnamen dürfen nicht mehr als 25 Zeichen enthalten. Zulässige Zeichen sind: Kleinbuchstaben a bis z, einschließlich. Großbuchstaben A bis Z, einschließlich. Zahlen 0 bis 9, inklusive. Sonderzeichen . (Punkt), @ (At-Zeichen), - (Minuszeichen) und _ (Unterstrich). Andere Zeichen, wie beispielsweise Leerzeichen, sind nicht zulässig.
Zulässige Zeichen in Artefaktnamen	Artefaktnamen dürfen 100 Zeichen nicht überschreiten. Zulässige Zeichen sind: Kleinbuchstaben a bis z, einschließlich. Großbuchstaben A bis Z, einschließlich. Zahlen 0 bis 9, einschließlich. Sonderzeichen - (Minuszeichen) und _ (Unterstrich). Andere Zeichen, wie beispielsweise Leerzeichen, sind nicht zulässig.

Zulässige Zeichen in Partneraktionsnamen	Für Namen von Partneraktionen gelten dieselben Benennungskonventionen und Einschränkungen wie für andere Aktionsnamen in CodePipeline. Insbesondere dürfen sie nicht mehr als 100 Zeichen enthalten. Zulässige Zeichen sind: Kleinbuchstaben a bis z, einschließlich. Großbuchstaben A bis Z, einschließlich. Zahlen 0 bis 9, inklusive. Sonderzeichen . (Punkt), @ (At-Zeichen), - (Minuszeichen) und _ (Unterstrich). Andere Zeichen, wie beispielsweise Leerzeich en, sind nicht zulässig.
Zulässige Zeichen in einem Pipeline-Namen	Pipeline-Namen dürfen nicht mehr als 100 Zeichen enthalten. Zulässige Zeichen sind: Kleinbuchstaben a bis z, einschließlich. Großbuchstaben A bis Z, einschließlich. Zahlen 0 bis 9, inklusive. Sonderzeichen . (Punkt), @ (At-Zeichen), - (Minuszeichen) und _ (Unterstrich). Andere Zeichen, wie beispielsweise Leerzeich en, sind nicht zulässig.

Zulässige Zeichen in einem Phasennamen	Phasennamen dürfen nicht mehr als 100 Zeichen enthalten. Zulässige Zeichen sind: Kleinbuchstaben a bis z, einschließlich. Großbuchstaben A bis Z, einschließlich. Zahlen 0 bis 9, inklusive. Sonderzeichen . (Punkt), @ (At-Zeichen), - (Minuszeichen) und _ (Unterstrich). Andere Zeichen, wie beispielsweise Leerzeichen, sind nicht zulässig.
Dauer bis zum Timeout einer Aktion	CodeBuild Aktion erstellen und Aktion testen: 8 Stunden Benutzerdefinierte Aktionen: 24 Stunden Step Functions rufen Aktion auf: 7 Tage
Maximale Länge des Aktionskonfigurationsschlüssels (die CodeBuild Konfigurationsschlüssel lauten beispielsweise `ProjectName` `PrimarySource` , und`EnvironmentVariables`)	50 Zeichen
Die maximale Länge des Aktionskonfigurationswerts (der Wert der `RepositoryName` Konfiguration in der CodeCommit Aktionskonfiguration sollte beispielsweise weniger als 1000 Zeichen lang sein): `"RepositoryName": "my-repo-name-less-than-1000-characters"`)	1 000 Zeichen
Maximale Anzahl von Aktionen pro Pipeline	500

Maximale Anzahl gleichzeitiger Pipeline-Ausführungen pro Pipeline (QUEUED|PARALLEL-Modus)	50
Maximale Anzahl gleichzeitiger Aktionsausführungen pro Pipeline-Ausführung im PARALLEL-Modus	5
Maximale Anzahl von Dateien für ein Amazon S3 S3-Objekt	100 000
Maximalanzahl der Monate, für die die Informationen im Pipeline-Ausführungsverlauf aufbewahrt werden	12
Maximalanzahl paralleler Aktionen in einer Phase	50
Maximalanzahl der sequenziellen Aktionen in einer Phase	50

Maximalgröße der Artefakte in einer Quellstufe	In Amazon S3 S3-Buckets gespeicherte Artefakte: 7 GB In CodeCommit unseren GitHub Repositorien gespeicherte Artefakte: 1 GB Ausnahme: Wenn Sie Anwendungen AWS Elastic Beanstalk zur Bereitstellung verwenden , beträgt die maximale Artefaktgröße immer 512 MB. Ausnahme: Wenn Sie Anwendungen AWS CloudFormation zur Bereitstellung verwenden, beträgt die maximale Artefaktgröße immer 256 MB. Ausnahme: Wenn Sie die Aktion `CodeDeployToECS` zum Bereitstellen von Anwendungen verwenden, beträgt die maximale Artefaktgröße immer 3 MB.
Maximale Größe der JSON-Datei mit Bilddefinitionen, die in Pipelines verwendet wird, die Amazon ECS-Container und -Images bereitstellen	100 KB
Maximale Größe der Eingabeartefakte für Aktionen AWS CloudFormation	256 MB
Maximale Größe der Eingabeartefakte für die `CodeDeployToECS` -Aktion	3 MB
Maximale Größe der Eingabeartefakte für die `Step Functions` -Aktion	Die Aktion Step Functions wird auf Lambda ausgeführt und verfügt daher über Artefaktgrößenquoten, die den Artefaktgrößenquoten für Lambda-Funktionen entsprechen. Weitere Informationen finden Sie unter Lambda-Kontingente im Lambda Developer Guide.

Maximale Größe des JSON-Objekts, das in der Eigenschaft `ParameterOverrides` gespeichert werden kann	Bei einer CodePipeline Bereitstellungsaktion mit dem AWS CloudFormation Anbieter wird die `ParameterOverrides` Eigenschaft verwendet, um ein JSON-Objekt zu speichern , das Werte für die AWS CloudFormation Vorlagenkonfigurationsdatei angibt. Für das JSON-Objekt, das in der Eigenschaft `ParameterOverrides` gespeichert werden kann, gilt eine Maximalgröße von 1 Kilobyte.
Anzahl der Aktionen in einer Phase	Mindestens 1, maximal 50
Anzahl der zulässigen Artefakte für jede Aktion	Die Anzahl der Eingabe- und Ausgabe-A rtefakte, die für jede Aktion zulässig sind, finden Sie unter Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
Anzahl der Phasen in einer Pipeline	Mindestens 2, maximal 50
Pipeline-Tags	Bei Tags muss die Groß- und Kleinschreibung beachtet werden. Maximal 50 pro Ressource.

Pipeline-Tag-Schlüsselnamen	Beliebige Kombination aus Unicode-Buchstaben, Zahlen, Leerstellen und zulässigen UTF-8-Zeichen mit einer Länge zwischen 1 und 128 Zeichen. Zulässige Zeichen sind + - = . _ : / @ Tag-Schlüsselnamen müssen eindeutig sein, und jeder Schlüssel darf nur einen Wert besitzen. Ein Tag darf nicht: • beginnen mit AWS: • nur aus Leerstellen bestehen • mit einem Leerzeichen enden • aus Emojis oder beliebigen der folgenden Zeichen bestehen: ? ^ * [\ ~ ! # $ % & * ( ) > < | " '
Pipeline-Tag-Werte	Beliebige Kombination aus Unicode-Buchstaben, Zahlen, Leerstellen und zulässigen UTF-8-Zeichen mit einer Länge zwischen 1 und 256 Zeichen. Zulässige Zeichen sind + - = . _ : / @ Ein Schlüssel kann nur einen Wert haben, aber viele Schlüssel können den gleichen Wert aufweisen. Ein Tag darf nicht: • beginnen mit AWS: • nur aus Leerstellen bestehen • mit einem Leerzeichen enden • aus Emojis oder beliebigen der folgenden Zeichen bestehen: ? ^ * [\ ~ ! # $ % & * ( ) > < | " '

Auslöser	Eine Pipeline-Definition enthält in der gesamten `pull request` AND-Konfiguration maximal 50 Trigger. `push` Es gibt maximal drei Filter pro Push-Trigger und Pull-Request-Trigger. **Note** Duplikate für Filter im gleichen Ereignistyp-Array sind nicht zulässig. Sie können bis zu 8 Include- und 8 Exclude-Muster, Branches und Dateipfade für jeden Ereignistyp (Push, Pull-Request) hinzufügen. Zulässige Zeichen in Musterwerten umfassen alle Zeichentypen. Für Einschluss- und Ausschlussmuster gibt es eine maximale Länge von 255 Zeichen. Für Tagnamen gibt es eine maximale Länge von 255 Zeichen. Die maximale Größe des `triggers` Arrays sollte 200 KB nicht überschreiten

Filter auslösen	Dateipfade: • Anzahl der Muster: Sie können bis zu 8 Include- und 8 Exclude-Muster hinzufügen. • Größe des Musters: Die Größe jedes Ein- oder Ausschlussmusters kann bis zu 255 Zeichen lang sein. Zweige: • Anzahl der Muster: Sie können bis zu 8 Einschluss- und 8 Ausschlussmuster hinzufügen. • Größe des Musters: Die Größe jedes Ein- oder Ausschlussmusters kann bis zu 255 Zeichen lang sein. Pull-Anfragen: Filialen: • Anzahl der Muster: Sie können bis zu 8 Einschluss- und 8 Ausschlussmuster hinzufügen. • Größe des Musters: Die Größe jedes Ein- oder Ausschlussmusters kann bis zu 255 Zeichen lang sein.

Eindeutigkeit von Namen	Innerhalb eines einzelnen AWS Kontos muss jede Pipeline, die Sie in einer AWS Region erstellen, einen eindeutigen Namen haben. Sie können Namen für Pipelines in verschiedenen AWS Regionen wiederverwenden. Phasennamen müssen innerhalb einer Pipeline eindeutig sein. Aktionsnamen müssen innerhalb einer Phase eindeutig sein.

Kontingente für Ausgabevariablen und Namespaces	Es gibt eine maximale Größenbeschränkung von 122.880 Byte für alle Ausgabevariablen, die für eine bestimmte Aktion kombiniert werden. Es gibt eine maximale Größenbeschränkung von 100 KB für die gesamte aufgelöste Aktionskonfiguration für eine bestimmte Aktion. Bei den Namen der Ausgabevariablen wird zwischen Groß- und Kleinschreibung unterschi eden Bei Namespaces wird zwischen Groß- und Kleinschreibung unterschieden. Zulässige Zeichen sind: • Kleinbuchstaben a bis z, einschließlich. • Großbuchstaben A bis Z, einschließlich. • Zahlen 0 bis 9, inklusive. • Sonderzeichen ^ (Caret), @ (at), - (Minuszei chen), _ (Unterstrich), [(linke Klammer),] (rechte Klammer), * (Sternchen), $ (Dollarze ichen). Andere Zeichen, wie beispielsweise Leerzeichen, sind nicht zulässig.

Kontingente für Variablen auf Pipeline-Ebene	Pro Pipeline gibt es maximal 50 Variablen auf Pipeline-Ebene. Variablennamen für Variablen auf Pipeline-Ebene müssen wie folgt lauten: • Maximallänge 128 Zeichen • Kleinbuchstaben a bis z, einschließlich. • Großbuchstaben A bis Z, einschließlich. • Zahlen 0 bis 9, inklusive. • Sonderzeichen `@\-_]+` Andere Zeichen, wie beispielsweise Leerzeichen, sind nicht zulässig. Für Variablenwerte gibt es eine maximale Länge von 1000 Zeichen Für Variablenwerte sind alle Zeichen zulässig. Für Variablenbeschreibungen gibt es eine maximale Länge von 200 Zeichen.

* Sie müssen diese Region aktivieren, bevor Sie sie verwenden können.

Anhang A: Quellaktionen für GitHub Version 1

Dieser Anhang enthält Informationen zu Version 1 der GitHub Aktion in. CodePipeline

> **Note**
>
> Wir empfehlen zwar nicht, die Aktion GitHub Version 1 zu verwenden, aber bestehende Pipelines mit der Aktion GitHub Version 1 funktionieren weiterhin ohne Auswirkungen. CodePipeline Verwendet bei einer Pipeline mit einer Aktion der GitHub Version 1 OAuth-basierte Token, um eine Verbindung zu Ihrem Repository herzustellen. GitHub Im Gegensatz dazu verwendet die GitHub Aktion (Version 2) eine Verbindungsressource, um AWS Ressourcen mit Ihrem GitHub Repository zu verknüpfen. Die Verbindungsressource verwendet App-basierte Token, um eine Verbindung herzustellen. Weitere Informationen zur Aktualisierung Ihrer Pipeline auf die empfohlene GitHub Aktion, die eine Verbindung verwendet, finden Sie unterAktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2. Weitere Informationen zum OAuth-basierten GitHub Zugriff im Gegensatz zum App-basierten GitHub Zugriff finden Sie unter. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps

Für die Integration GitHub wird eine GitHub OAuth-Anwendung für Ihre Pipeline CodePipeline verwendet. CodePipelineverwendet Webhooks, um die Änderungserkennung für Ihre Pipeline mit der Quellaktion der GitHub Version 1 zu verwalten.

> **Note**
>
> Wenn Sie eine Quellaktion der GitHub Version 2 in konfigurieren AWS CloudFormation, fügen Sie keine GitHub Token-Informationen hinzu und fügen keine Webhook-Ressource hinzu. Sie konfigurieren eine Verbindungsressource wie AWS::CodeStarConnections::Connectionim AWS CloudFormation Benutzerhandbuch beschrieben.

Diese Referenz enthält die folgenden Abschnitte für die Aktion GitHub Version 1:

- Informationen zum Hinzufügen einer Quellaktion und eines Webhooks der GitHub Version 1 zu einer Pipeline finden Sie unterHinzufügen einer Quellaktion für Version 1 GitHub .

- Informationen zu den Konfigurationsparametern und YAML/JSON-Beispielfragmenten für eine Quellaktion der GitHub Version 1 finden Sie unter. GitHub Referenz zur Struktur der Quellaktion von Version 1

Themen

Hinzufügen einer Quellaktion für Version 1 GitHub

Sie fügen Quellaktionen der GitHub Version 1 hinzu, CodePipeline indem Sie:

- Verwenden Sie die CodePipeline Konsole, um den Assistenten zum Erstellen von Pipelines (Erstellen einer Pipeline (Konsole)) oder die Aktionsseite Bearbeiten zu verwenden, um die GitHubAnbieteroption auszuwählen. Die Konsole erstellt einen Webhook, der Ihre Pipeline startet, wenn sich die Quelle ändert.
- Verwenden Sie die CLI, um die Aktionskonfiguration für die `GitHub` Aktion hinzuzufügen und zusätzliche Ressourcen wie folgt zu erstellen:
  - Verwenden Sie die `GitHub` Beispiel-Aktionskonfiguration inGitHub Referenz zur Struktur der Quellaktion von Version 1, um die Aktion zu erstellen, wie unter gezeigtErstellen einer Pipeline (CLI).
  - Regelmäßige Prüfungen werden deaktiviert und die Änderungserkennung manuell erstellt, da die Methode zur Änderungserkennung standardmäßig die Pipeline durch Abfragen der Quelle startet. Sie migrieren Ihre Polling-Pipeline für Aktionen der Version 1 auf Webhooks. GitHub

GitHub Referenz zur Struktur der Quellaktion von Version 1

> **Note**
>
> Wir empfehlen zwar nicht, die Aktion GitHub Version 1 zu verwenden, aber bestehende Pipelines mit der Aktion GitHub Version 1 funktionieren weiterhin ohne Auswirkungen. CodePipeline Verwendet bei einer Pipeline mit einer Quellaktion der GitHub GitHub Version 1 OAuth-basierte Token, um eine Verbindung zu Ihrem Repository herzustellen. GitHub Im Gegensatz dazu verwendet die neue GitHub Aktion (Version 2) eine Verbindungsressource, um AWS Ressourcen mit Ihrem GitHub Repository zu verknüpfen. Die Verbindungsressource

verwendet App-basierte Token, um eine Verbindung herzustellen. Weitere Informationen zur Aktualisierung Ihrer Pipeline auf die empfohlene GitHub Aktion, die eine Verbindung verwendet, finden Sie unterAktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2.

Löst die Pipeline aus, wenn ein neuer Commit für das konfigurierte GitHub Repository und den Branch vorgenommen wird.

CodePipeline Verwendet zur Integration GitHub eine OAuth-Anwendung oder ein persönliches Zugriffstoken für Ihre Pipeline. Wenn Sie die Konsole verwenden, um Ihre Pipeline zu erstellen oder zu bearbeiten, CodePipeline erstellt sie einen GitHub Webhook, der Ihre Pipeline startet, wenn eine Änderung im Repository erfolgt.

Sie müssen bereits ein GitHub Konto und ein Repository erstellt haben, bevor Sie die Pipeline über eine GitHub Aktion verbinden können.

Wenn Sie den Zugriff auf CodePipeline Repositorys einschränken möchten, erstellen Sie ein GitHub Konto und gewähren Sie dem Konto nur Zugriff auf die Repositorys, in die Sie integrieren möchten. CodePipeline Verwenden Sie dieses Konto, wenn Sie die Verwendung von GitHub Repositorys für Quellstufen in Pipelines konfigurieren CodePipeline .

Weitere Informationen finden Sie in der GitHub Entwicklerdokumentation auf der GitHub Website.

Themen

Aktionstyp

- Kategorie: `Source`

- Eigentümer: `ThirdParty`
- Anbieter: `GitHub`
- Version: `1`

Konfigurationsparameter

Eigentümer

Erforderlich: Ja

Der Name des GitHub Benutzers oder der Organisation, dem das GitHub Repository gehört.

Repo

Erforderlich: Ja

Der Name des Repositorys, in dem Quelländerungen erkannt werden sollen.

Verzweigung

Erforderlich: Ja

Der Name des Zweigs, in dem Quelländerungen erkannt werden sollen.

O AuthToken

Erforderlich: Ja

Stellt das GitHub Authentifizierungstoken dar, mit CodePipeline dem Operationen in Ihrem GitHub Repository ausgeführt werden können. Die Eingabe wird stets als Maske mit vier Sternchen angezeigt. Sie stellt einen der folgenden Werte dar:

- Wenn Sie die Konsole verwenden, um die Pipeline zu erstellen, CodePipeline verwendet sie ein OAuth-Token, um die GitHub Verbindung zu registrieren.
- Wenn Sie die Pipeline AWS CLI zum Erstellen der Pipeline verwenden, können Sie in diesem Feld Ihr GitHub persönliches Zugriffstoken übergeben. Ersetzen Sie die Sternchen (****) durch Ihr persönliches Zugriffstoken, von dem Sie kopiert wurden. GitHub Wenn Sie `get-pipeline` ausführen, um die Aktionskonfiguration anzuzeigen, wird dieser Wert durch vier Sternchen maskiert angezeigt.
- Wenn Sie eine AWS CloudFormation Vorlage verwenden, um die Pipeline zu erstellen, müssen Sie das Token zunächst als Secret in speichern. AWS Secrets Manager Sie fügen den Wert für

dieses Feld als dynamischen Verweis auf das gespeicherte Geheimnis in Secrets Manager ein, z. `{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}` B.

Weitere Informationen zu GitHub Bereichen finden Sie in der GitHub Developer API Reference auf der GitHub Website.

PollForSourceChanges

Erforderlich: Nein

`PollForSourceChanges`steuert, ob das GitHub Repository nach Quellenänderungen CodePipeline abfragt. Es wird empfohlen, stattdessen Webhooks zu verwenden, um Quelländerungen zu erkennen. Weitere Informationen zum Konfigurieren von Webhooks finden Sie unter Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (CLI) oder Aktualisierungspipelines für Push-Ereignisse (Quellaktionen der GitHub Version 1) (AWS CloudFormation Vorlage).

> **Important**
>
> Wenn Sie Webhooks konfigurieren möchten, müssen Sie `PollForSourceChanges` auf `false` festlegen, um doppelte Pipeline-Ausführungen zu vermeiden.

Gültige Werte für diesen Parameter sind:

- `True`: Falls gesetzt, CodePipeline fragt dein Repository nach Quellenänderungen ab.

  > **Note**
  >
  > Wenn Sie diese Option weglassen`PollForSourceChanges`, wird CodePipeline standardmäßig Ihr Repository nach Quelländerungen abgefragt. Dieses Verhalten ist das gleiche, als ob `PollForSourceChanges` auf `true` festgelegt würde.

- `False`: Falls gesetzt, fragt dein Repository CodePipeline nicht nach Quellenänderungen ab. Sie verwenden diese Einstellung, wenn Sie einen Webhook konfigurieren möchten, um Quelländerungen zu erkennen.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0`

- Beschreibung: Eingabe-Artefakte sind für diesen Aktionstyp nicht gültig.

Ausgabeartefakte

- Anzahl der Artefakte: 1
- Beschreibung: Das Ausgabe-Artefakt dieser Aktion ist eine ZIP-Datei, die den Inhalt des konfigurierten Repositorys und Zweigs beim Commit enthält, der als Quellrevision für die Pipeline-Ausführung angegeben wurde. Die aus dem Repository generierten Artefakte sind die Ausgabeartefakte für die GitHub Aktion. Die Quellcode-Commit-ID wird CodePipeline als Quellrevision für die ausgelöste Pipeline-Ausführung angezeigt.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Diese Aktion erzeugt Variablen, die als Ausgabevariablen angezeigt werden können, auch wenn die Aktion keinen Namespace hat. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

Weitere Hinweise zu Variablen in CodePipeline finden Sie unterVariablen.

CommitId

Die GitHub Commit-ID, die die Pipeline-Ausführung ausgelöst hat. Commit-IDs sind der vollständige SHA des Commits.

CommitMessage

Die Beschreibungsmeldung (wenn vorhanden), die dem Commit zugeordnet ist, der die Pipeline-Ausführung ausgelöst hat.

CommitUrl

Die URL-Adresse für den Commit, der die Pipeline ausgelöst hat.

RepositoryName

Der Name des GitHub Repositorys, in dem der Commit, der die Pipeline ausgelöst hat, vorgenommen wurde.

BranchName

Der Name des Branches für das GitHub Repository, in dem die Quellenänderung vorgenommen wurde.

AuthorDate

Das Datum im Zeitstempelformat, an dem der Commit erstellt wurde.

Weitere Informationen zum Unterschied zwischen einem Autor und einem Committer in Git finden Sie unter Viewing the Commit History in Pro Git von Scott Chacon und Ben Straub.

CommitterDate

Das Datum im Zeitstempelformat, an dem der Commit durchgeführt wurde.

Weitere Informationen zum Unterschied zwischen einem Autor und einem Committer in Git finden Sie unter Viewing the Commit History in Pro Git von Scott Chacon und Ben Straub.

Aktionsdeklaration (GitHub-Beispiel)

YAML

```
Name: Source
Actions:
  - InputArtifacts: []
    ActionTypeId:
      Version: '1'
      Owner: ThirdParty
      Category: Source
      Provider: GitHub
    OutputArtifacts:
      - Name: SourceArtifact
    RunOrder: 1
    Configuration:
      Owner: MyGitHubAccountName
      Repo: MyGitHubRepositoryName
      PollForSourceChanges: 'false'
      Branch: main
      OAuthToken: '{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}'
    Name: ApplicationSource
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "InputArtifacts": [],
            "ActionTypeId": {
                "Version": "1",
                "Owner": "ThirdParty",
                "Category": "Source",
                "Provider": "GitHub"
            },
            "OutputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "RunOrder": 1,
            "Configuration": {
                "Owner": "MyGitHubAccountName",
                "Repo": "MyGitHubRepositoryName",
                "PollForSourceChanges": "false",
                "Branch": "main",
                "OAuthToken":
 "{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}"
            },
            "Name": "ApplicationSource"
        }
    ]
},
```

Verbindung herstellen zu GitHub (OAuth)

Wenn Sie die Konsole zum ersten Mal verwenden, um einer Pipeline ein GitHub Repository hinzuzufügen, werden Sie aufgefordert, den CodePipeline Zugriff auf Ihre Repositorys zu autorisieren. Das Token erfordert die folgenden Bereiche: GitHub

- Den `repo`- Umfang, der für die volle Kontrolle verwendet wird, um Artefakte aus öffentlichen und privaten Repositorys in eine Pipeline zu lesen und einzufügen.

- Den `admin:repo_hook` Umfang, der für die volle Kontrolle über Repository-Hooks verwendet wird.

Wenn Sie die CLI oder eine AWS CloudFormation Vorlage verwenden, müssen Sie den Wert für ein persönliches Zugriffstoken angeben, das Sie bereits erstellt haben GitHub.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Ressourcenreferenz für das AWS CloudFormation Benutzerhandbuch AWS::CodePipeline::Webhook — Dazu gehören Felddefinitionen, Beispiele und Ausschnitte für die Ressource in. AWS CloudFormation
- Ressourcenreferenz für das AWS CloudFormation User Guide AWS::CodeStar::GitHub Repository — Dazu gehören Felddefinitionen, Beispiele und Ausschnitte für die Ressource in. AWS CloudFormation
- Tutorial: Erstellen Sie eine Pipeline, die Ihre Android-App erstellt und testet mit AWS Device Farm— Dieses Tutorial enthält ein Beispiel für eine Build-Spezifikationsdatei und eine Beispielanwendung zum Erstellen einer Pipeline mit einer Quelle. GitHub Es erstellt und testet eine Android-App mit CodeBuild und AWS Device Farm.

AWS CodePipeline Dokumentverlauf des Benutzerhandbuchs

In der folgenden Tabelle werden die wichtigen Änderungen in den einzelnen Versionen des CodePipeline Benutzerhandbuchs beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

- API-Version: 09. Juli 2015
- Letzte Aktualisierung der Dokumentation: 07. Mai 2024

Änderung	Beschreibung	Datum
Die S3 Quellaktion wurde aktualisiert, um eine neue Option für Quellenüberschreibungen hinzuzufügen	Eine neue Option für benannte Quellüberschreibungen `S3_OBJECT_KEY` ist für die S3 Quellaktion verfügbar. Ein neuer `AllowOverrideForS3ObjectKey` Parameter wurde für die S3-Quellaktion hinzugefügt. Sehen Sie sich die Referenzseite für Amazon S3 S3-Quellaktionen an und starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung.	7. Juni 2024
Aktualisierungen der S3 Quellaktion zum Hinzufügen neuer Ausgabevariablen	Neue Ausgabevariablen, die benannt `ObjectKey` sind `BucketName` und für die S3 Quellaktion verfügbar sind. Weitere Informationen finden Sie auf der Referenzseite für Amazon S3 S3-Quellaktionen.	5. Juni 2024

Support der Kostenanalyse für die Umstellung von Pipelines vom Typ V1 auf Pipelines vom Typ V2	Das `PipelineCostAnalyzer.py` Skript wurde für die Durchführung der Kostenanalyse bei der Umstellung vorhandener Pipelines vom Typ V1 auf Pipelines vom Typ V2 hinzugefügt. Weitere Informationen finden Sie unter Welcher Pipeline-Typ ist der richtige für mich? .	30. Mai 2024
Aktualisierungen der `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und	Der `CallAs` Parameter wurde für die `CloudFormationStackInstances` Aktion `CloudFormationStackSet` und hinzugefügt. Weitere Informationen finden Sie auf der Aktionsreferenzseite.	2. Mai 2024
Support für Rollbacks auf Stufenebene	Sie können eine Phase manuell oder automatisch auf eine vorherige erfolgreiche Pipeline-Ausführung für die Phase zurücksetzen. Weitere Informationen finden Sie unter Konfiguration des Stage-Rollbacks und Konzepte.	26. April 2024
Aktualisierungen der regionalen Verfügbarkeit für Aktionen StackSets und Step Functions Functions-Aktionen	Die Aktionen StackSets und Step Functions sind jetzt in allen Regionen verfügbar, in denen sie CodePipeline verfügbar sind. Siehe AWS CloudFormation StackSets Aktionsreferenz und Aktionsreferenz AWS Step Functions.	27. März 2024

Aktualisierungen der verwalteten Richtlinie	Die AWS verwaltete Richtlinie `AWSCodePipeline_FullAccess` wurde aktualisiert. Weitere Informationen finden Sie unter AWS Verwaltete Richtlinien für AWS CodePipeline.	15. März 2024
Support für konfigurierbares Timeout für manuelle Genehmigungsaktionen	Kontingentinformationen wurden für das neue konfigurierbare Timeout-Feld für manuelle Genehmigungsaktionen hinzugefügt. Weitere Informationen finden Sie unter Kontingente.	15. Februar 2024
Support für Triggerfilterung nach Verzweigungen und Dateipfaden	Support für die Trigger-Konfiguration hinzugefügt, die das Filtern nach Pull-Request-Status, Branches und Dateipfaden für Pipelines vom Typ V2 ermöglicht. Weitere Informationen finden Sie unter Filtern von Triggern auf Code-Push- oder Pull-Requests, Trigger und Filtern nach Feature-Branches, um Ihre Pipeline zu starten, und Kontingente.	8. Februar 2024

Support für neue Pipeline-Ausführungsmodi	Support für die Pipeline-Ausführungsmodi PARALLEL und QUEUED hinzugefügt. Weitere Informationen finden Sie unter Festlegen des Pipeline-Ausführungsmodus, Verarbeitung von Ausführungen im QUEUED-Modus, Verarbeitung von Ausführungen im PARALLELMODUS und Kontingente.	8. Februar 2024
Aktualisierungen der Konsolenseiten zum Anzeigen von Aktionsdetails, zum Überprüfen manueller Genehmigungsaktionen und der Seite „Pipelines auflisten“	Die Konsolenaktualisierungen wurden dokumentiert, darunter die neue Schaltfläche und das neue Dialogfeld „Details anzeigen“, ein neues Dialogfeld für die manuelle Genehmigung und neue Spalten für die letzten Ausführungen auf der Seite „Pipelines auflisten“. Weitere Informationen finden Sie unter Pipelines anzeigen (Konsole), Aktionsdetails in einer Pipeline anzeigen und Genehmigungsaktionen in Pipelines verwalten.	10. Januar 2024

Support für GitLab selbstverwaltete	Support für die Konfiguration von Verbindungen für AWS Ressourcen zur Interaktion mit GitLab selbstverwalteten Ressourcen hinzugefügt. Weitere Informationen finden Sie unter Verbindungen für GitLab selbstverwaltete Geräte.	28. Dezember 2023
Aktualisierungen der Aktionen `CloudFormationStackSet` und `CloudFormationStackInstances`	Der `ConcurrencyMode` Parameter wurde für die `CloudFormationStackInstances` Aktion `CloudFormationStackSet` und hinzugefügt. Weitere Informationen finden Sie auf der Aktionsreferenzseite.	19. Dezember 2023
Aktualisierungen der AWS Device Farm Aktionsparameter in CodePipeline	Die Parameter für die AWS Device Farm Aktion in CodePipeline wurden aktualisiert. Weitere Informationen finden Sie in der AWS Device Farm Aktionsreferenz.	18. Dezember 2023
Support für detaillierte Fehlermeldungen für die AWS CloudFormation Aktion in hinzugefügt CodePipeline	AWS CloudFormation In Aktionsfehlermeldungen können jetzt Details zu Ressourcen angezeigt werden, bei denen Fehler aufgetreten sind. Weitere Informationen finden Sie in der AWS CloudFormation Aktionsreferenz.	15. Dezember 2023

Updates für den Start einer Pipeline mit Quellrevisionsüberschreibungen in CodePipeline	Sie können jetzt eine Pipeline mit einer angegebenen Quellrevision starten. Weitere Informationen finden Sie unter Starten einer Pipeline mit einer Quellrevisionsüberschreibung.	17. November 2023
Neue unterstützte Regionen	CodePipeline ist jetzt in den Regionen Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Naher Osten (VAE), Europa (Spanien) und Israel (Tel Aviv) verfügbar. Das Referenzthema für den Platzhalter-Bucket „Ereignisse“ und das Thema „AWS-Service Endpunkte“ wurden aktualisiert.	13. November 2023
Updates für Event-Felder in Amazon EventBridge	Sie können jetzt aktualisierte Ereignisfelder in Amazon anzeigen EventBridge. Weitere Informationen finden Sie unter CodePipelineEreignisse überwachen.	9. November 2023

Updates für neue Pipelines vom Typ V2, Trigger für Git-Tags und Pipeline-Variablen in CodePipeline	Sie können jetzt einen Pipeline-Typ in CodePipeline auswählen. Für eine Pipeline vom Typ V2 können Sie jetzt eine Trigger-Konfiguration verwenden, um Ihre Pipeline mit Git-Tags zu starten. Bei Pipelines vom Typ V2 kannst du auch Variablen auf Pipeline-Ebene verwenden , um Eingabeparameter für eine Pipeline-Ausführun g zu übergeben. Weitere Informationen finden Sie unter Variablen, Tutorial: Variablen auf Pipeline-Ebene verwenden und Tutorial: Verwenden Sie Git-Tags, um Ihre Pipeline zu starten. Weitere Informationen zu Pipeline-Typen finden Sie unter Pipeline-Typen.	24. Oktober 2023

CodePipeline ermöglicht den erneuten Versuch, alle Aktionen in einer fehlgeschlagenen Phase erneut auszuführen	Bei einer fehlgeschlagenen Phase können Sie die Phase erneut versuchen CodePipeline, ohne die Pipeline erneut auszuführen. Sie tun dies, indem Sie entweder die fehlgeschlagenen Aktionen in einer Phase erneut versuchen, oder indem Sie alle Aktionen in der Phase wiederholen, beginnend mit der ersten Aktion in der Phase. Weitere Informationen finden Sie unter Wiederholen einer fehlgeschlagenen Phase oder fehlgeschlagener Aktionen in einer Phase.	17. Oktober 2023
Support für GitLab Gruppen	Support für die Konfiguration von Verbindungen für AWS Ressourcen zur Interaktion mit GitLab Gruppen hinzugefügt. Weitere Informationen finden Sie unter GitLab Verbindungen.	15. September 2023

CodePipeline unterstützt Verbindungen GitLab zu.com	Sie können Verbindungen verwenden, um AWS Ressourcen für die Interaktion mit GitLab .com zu konfigurieren. Sie können auch die Option „Vollständiges Klonen“ wählen, um Git-Befehle und Metadaten für nachgelagerte Aktionen zu verwenden. Weitere Informationen finden Sie im Referenzthema GitLab Verbindungen und CodeStarSourceConnection Aktionsstruktur.	10. August 2023
Aktualisierung der `CloudFormationStackInstances` Aktion	Der `RegionConcurrencyType` Parameter wurde für die `CloudFormationStackInstances` Aktion hinzugefügt. Informationen zur Aktion finden Sie auf der `CloudFormationStackInstances` Aktionsreferenzseite.	08. August 2023
Aktualisierung der `CloudFormationStackSet` Aktion	Der `RegionConcurrencyType` Parameter wurde für die `CloudFormationStackSet` Aktion hinzugefügt. Informationen zur Aktion finden Sie auf der `CloudFormationStackSet` Aktionsreferenzseite.	24. Juli 2023

Aktualisierungen der verwalteten Richtlinie	Die AWS verwaltete Richtlinie `AWSCodePipeline_FullAccess` wurde aktualisiert. Weitere Informationen finden Sie unter AWS Verwaltete Richtlinien für AWS CodePipeline.	21. Juni 2023
Aktualisierungen der Migrationsverfahren für Polling-Pipelines	Die Verfahren zur Migration (Aktualisierung) von Polling-Pipelines zur Verwendung der ereignisbasierten Änderungserkennung wurden mit den Schritten für Pipelines aktualisiert, die einen Amazon S3 S3-Bucket verwenden, für den Benachrichtigungen aktiviert sind. EventBridge Weitere Informationen finden Sie unter Migrieren von Abfrage-Pipelines zur Verwendung der ereignisbasierten Änderungserkennung.	12. Juni 2023
Aktualisierungen der verwalteten Richtlinien	Die AWS `AWSCodePipeline_FullAccess` verwalteten Richtlinien `AWSCodePipeline_ReadOnlyAccess` wurden mit einer zusätzlichen Berechtigung aktualisiert. Weitere Informationen finden Sie unter AWS CodePipeline Aktualisierungen der AWS verwalteten Richtlinien.	16. Mai 2023

Aktualisierungen verwalteter Richtlinien	Die AWS verwalteten Richtlinien `AWSCodePipelineFullAccess` und `AWSCodePipelineReadOnlyAccess` sind veraltet. Verwenden Sie die Richtlinien `AWSCodePipeline_FullAccess` und. `AWSCodePipeline_ReadOnlyAccess` Siehe AWS CodePipeline Aktualisierungen der AWS verwalteten Richtlinien.	17. November 2022
Aktualisierungen von Verfahren, die Folgendes verwenden CloudTrail	Alle Konsolenprozeduren, CLI-Beispielbefehle AWS CloudFormation sowie Beispielfragmente und Vorlagen für eine Pipeline mit einer S3-Quelle wurden mit der Option aktualisiert, Write und False für Management-Ereignisse in auszuwählen. CloudTrail Die aktualisierten Beispiele finden Sie unter Pipeline starten, Tutorial: Pipeline erstellen mit AWS CloudFormation, Pipelines für die Verwendung von Push-Ereignissen bearbeiten und Polling-Pipelines aktualisieren.	27. April 2022

Neue unterstützte Integration mit Snyk	Sie können die Snyk-Aufr ufaktion verwenden, um die Sicherheitsscans für Ihren Open-Source-Code CodePipel ine zu automatisieren. Weitere Informationen finden Sie in der Snyk-Aktionsreferenz und unter Integrationen.	10. Juni 2021
Neue unterstützte Region Europa (Mailand)	CodePipeline ist jetzt in Europa (Mailand) verfügbar. Die Themen Limits und AWS-Service Endpoints wurden aktualisiert.	27. Januar 2021
Die Änderungserkennung kann für Quellaktionen mit Verbindungen deaktiviert werden	Sie können die CLI oder das SDK verwenden, um eine `CodeStarSourceConn ection` Quellaktion zu aktualisieren, um die automatische Änderungs erkennung für das Quell-Repository zu deaktivieren. Das Referenzthema zur CodeStarSourceConnection Aktionsstruktur wurde mit einer Beschreibung für den `DetectChanges` Parameter aktualisiert.	08. Januar 2021

CodePipeline unterstützt jetzt AWS CloudFormation StackSets Bereitstellungsaktionen	Das neue Tutorial, Tutorial: Eine Pipeline erstellen, die AWS CloudFormation StackSets als Bereitstellungsanbieter verwendet wird, enthält Schritte AWS CloudFormation StackSets zum Erstellen und Aktualisieren Ihrer Stack-Sets und Stack-Instances mit Ihrer Pipeline. Das Referenzthema AWS CloudFormation StackSets Aktionsstruktur wurde ebenfalls hinzugefügt.	30. Dezember 2020
Neue unterstützte Region Asien-Pazifik (Hongkong)	CodePipeline ist jetzt im asiatisch-pazifischen Raum (Hongkong) verfügbar. Die Themen Limits und AWS-Service Endpoints wurden aktualisiert.	22. Dezember 2020
Aktualisierte EventBridge Ereignismuster finden Sie unter CodePipeline	Aktualisierte Ereignismuster und Status für Ereignisse auf Pipeline-, Phasen- und Aktionsebene wurden dem Bereich CodePipeline Überwachungsereignisse hinzugefügt.	21. Dezember 2020

Eingehende Pipeline-Ausführungen anzeigen in CodePipeline	Sie können die Konsole oder die CLI verwenden, um eingehende Ausführungen anzuzeigen. Weitere Informationen finden Sie unter Eingehende Ausführung anzeigen (Konsole) und Status eingehender Ausführung anzeigen (CLI).	16. November 2020
Die CodeCommit Quellaktion in CodePipeline unterstützt die Option „Vollständiges Klonen“	Wenn Sie eine CodeCommit Quellaktion verwenden, können Sie die vollständige Klonoption für die Verwendung von Git-Befehlen und Metadaten für CodeBuild Downstream-Aktionen wählen. Weitere Informationen findest du in der CodeCommit Aktionsreferenz und im Tutorial: Vollständigen Klon mit einer CodeCommit Pipeline-Quelle verwenden.	11. November 2020

CodePipeline unterstützt Verbindungen zu GitHub einem GitHub Enterprise Server	Sie können Verbindungen verwenden, um AWS Ressourcen für die Interaktion mit GitHub GitHub Enterprise Cloud und GitHub Enterprise Server zu konfigurieren. Sie können auch die Option „Vollständiges Klonen“ wählen, um Git-Befehle und Metadaten für nachgelagerte Aktionen zu verwenden. Weitere Informationen finden Sie unter GitHub Verbindungen, GitHub Enterprise Server-Verbindungen und Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden. Wenn Sie bereits über eine Pipeline mit einer GitHub Quellaktion verfügen, finden Sie weitere Informationen unter Aktualisieren einer Quellaktion von GitHub GitHub Version 1 auf eine Quellaktion von Version 2.	30. September 2020
Die CodeBuild Aktion unterstützt das Aktivieren von Batch-Builds in AWS CodePipeline	Für CodeBuild Aktionen in Ihrer Pipeline können Sie Batch-Builds aktivieren, um mehrere Builds in einer einzigen Ausführung auszuführen. Weitere Informationen finden Sie unter Referenz zur CodeBuild Aktionsstruktur und Pipeline erstellen (Konsole).	30. Juli 2020

AWS CodePipeline unterstützt jetzt AWS AppConfig Bereitstellungsaktionen	Das neue Tutorial, Tutorial: Eine Pipeline erstellen, die AWS AppConfig als Bereitstellungsanbieter verwendet wird, enthält Schritte AWS AppConfig zum Bereitstellen von Konfigurationsdateien mit Ihrer Pipeline. Das Referenzthema AWS AppConfig Aktionsstruktur wurde ebenfalls hinzugefügt.	25. Juni 2020
AWS CodePipeline unterstützt jetzt Amazon VPC in AWS GovCloud (US-West)	Sie können jetzt AWS CodePipeline über einen privaten Amazon VPC-Endpunkt in AWS GovCloud (US-West) eine direkte Verbindung herstellen. Weitere Informationen finden Sie unter Verwendung CodePipeline mit Amazon Virtual Private Cloud.	2. Juni 2020
AWS CodePipeline unterstützt jetzt AWS Step Functions Aufrufaktionen	Sie können jetzt eine Pipeline erstellen CodePipeline , die AWS Step Functions als Anbieter für Aufrufaktionen verwendet wird. Das neue Tutorial, Tutorial: Use an AWS Step Functions Invoke Action in a Pipeline, enthält Schritte zum Starten einer State-Machine-Ausführung von Ihrer Pipeline aus. Das Thema AWS Step Functions – Aktionsstrukturreferenz wurde ebenfalls hinzugefügt.	28. Mai 2020

Verbindungen anzeigen, auflisten und aktualisieren	Sie können Verbindungen in der Konsole auflisten, löschen und aktualisieren. Weitere Informationen finden Sie unter Verbindungen auflisten in CodePipeline.	21. Mai 2020
Verbindungen unterstützen das Markieren von Verbindungsressourcen in der CLI	Die Verbindungsressourcen unterstützen jetzt Tagging in der AWS CLI. Verbindungen lassen sich jetzt in integrieren AWS CodeGuru. Informationen finden Sie unter IAM-Berechtigungsreferenz für Verbindungen.	6. Mai 2020
CodePipeline ist jetzt in AWS GovCloud (US-West) verfügbar	Sie können es jetzt CodePipeline in AWS GovCloud (US-West) verwenden. Weitere Informationen finden Sie unter Kontingente.	8. April 2020
Das Thema Kontingente zeigt, welche CodePipeline Dienstkontingente konfigurierbar sind	Das Thema CodePipeline Kontingente wurde neu formatiert. Die Dokumentation zeigt, welche Servicekontingente konfigurierbar sind und welche nicht. Weitere Informationen finden Sie unter Kontingente. AWS CodePipeline	12. März 2020

Das Zeitlimit für Amazon ECS-Bereitstellungsaktionen ist konfigurierbar	Das Zeitlimit für Amazon ECS-Bereitstellungsaktionen ist auf bis zu einer Stunde konfiguri erbar (das Standard-Timeout). Siehe Kontingente in AWS CodePipeline.	5. Februar 2020
In neuen Themen wird beschrieben, wie Sie die Ausführung einer Pipeline beenden können	Sie können die Ausführun g einer Pipeline in beenden CodePipeline. Sie können entweder angeben, dass die Ausführung nach dem Abschluss von in Arbeit befindlichen Aktionen beendet wird, oder Sie können angeben, dass die Ausführun g sofort angehalten und in Arbeit befindliche Aktionen beendet werden sollen. Weitere Informationen finden Sie unter So werden Pipelinea usführungen gestoppt und Eine Pipelineausführung beenden in CodePipeline.	21. Januar 2020

CodePipeline unterstützt Verbindungen	Sie können Verbindungen verwenden, um AWS Ressourcen für die Interaktion mit externen Code-Repositorys zu konfigurieren. Jede Verbindung ist eine Ressource, die von Diensten verwendet werden kann, um beispielsweise eine Verbindung CodePipeline zu einem Repository eines Drittanbieters wie Bitbucket Cloud herzustellen. Weitere Informationen findest du unter Arbeiten mit Verbindungen in CodePipeline.	18. Dezember 2019
Die Themen Sicherheit, Authentifizierung und Zugriffskontrolle wurden aktualisiert	Die Informationen zu Sicherheit, Authentifizierung und Zugriffskontrolle für CodePipeline wurden in einem neuen Sicherheitskapitel zusammengefasst. Weitere Informationen finden Sie unter Sicherheit.	17. Dezember 2019

In neuen Themen wird beschrieben, wie Sie Variablen in Ihren Pipelines verwenden können	Sie können jetzt Namespaces für eine Aktion konfigurieren und jedes Mal Variablen generieren, wenn die Aktion abgeschlossen ist. Sie können nachgeschaltete Aktionen einrichten, um auf diese Namespaces und Variablen zu verweisen. Weitere Informationen hierzu finden Sie unterArbeiten mit Variablen und Variablen.	14. November 2019
In neuen Themen wird beschrieben, wie Pipeline-Ausführungen funktionieren, warum Phasen während einer Ausführung gesperrt sind und wann Pipeline-Ausführungen ersetzt werden	Den Begrüßungsabschnitt wurde eine Reihe von Themen hinzugefügt, in denen beschrieben wird, wie Pipeline-Ausführungen funktionieren, warum Phasen während einer Ausführung gesperrt sind und was passiert, wenn Pipeline-Ausführungen ersetzt werden. Zu diesen Themen gehören eine Liste von Konzepten, ein DevOps Workflow-Beispiel und Empfehlungen zur Strukturierung einer Pipeline. Die folgenden Themen wurden hinzugefügt: Pipeline-Begriffe, DevOps Pipeline-Beispiel und Funktionsweise von Pipeline-Ausführungen.	11. November 2019

CodePipeline unterstützt Benachrichtigungsregeln	Sie können jetzt Benachrichtigungsregeln verwenden, um Benutzer über wichtige Änderungen in Pipelines zu informieren. Weitere Informationen finden Sie unter Erstellen einer Benachrichtigungsregel.	5. November 2019
CodeBuild Umgebungsvariablen sind verfügbar in CodePipeline	Sie können CodeBuild Umgebungsvariablen in der CodeBuild Build-Aktion für Ihre Pipeline festlegen. Sie können den Parameter `EnvironmentVariables` über die Konsole oder die CLI zur Pipeline-Struktur hinzufügen. Das Thema Erstellen einer Pipeline (Konsole) wurde aktualisiert. Die Beispiele für die Aktionskonfiguration in der Aktionsreferenz für CodeBuild wurden ebenfalls aktualisiert.	14. Oktober 2019
Neue Region	CodePipeline ist jetzt in Europa (Stockholm) verfügbar. Die Themen Limits und AWS-Service Endpoints wurden aktualisiert.	5. September 2019

Geben Sie vorgefertigte ACLs und Cache-Steuerung für Amazon S3 S3-Bereitstellungsaktionen an	Sie können jetzt vorgefertigte ACL- und Cache-Kontrolloptionen angeben, wenn Sie eine Amazon S3 S3-Bereitstellungsaktion in erstellen CodePipeline. Die folgenden Themen wurden aktualisiert: Pipeline erstellen (Konsole), Referenz zur CodePipeline Pipeline-Struktur und Tutorial: Pipeline erstellen, die Amazon S3 als Bereitstellungsanbieter verwendet.	27. Juni 2019

Sie können jetzt Tags zu Ressourcen hinzufügen in AWS CodePipeline	Sie können jetzt Tagging verwenden, um AWS CodePipeline Ressourcen wie Pipelines, benutzerdefinierte Aktionen und Webhooks zu verfolgen und zu verwalten . Die folgenden neuen Themen wurden hinzugefü gt: Ressourcen taggen, Verwenden von Tags zur Steuerung des Zugriffs auf CodePipeline Ressource n, Markieren einer Pipeline in CodePipeline, Markieren einer benutzerdefinierten Aktion in und Markieren eines CodePipeline Webhooks in. CodePipeline Die folgenden Themen wurden aktualisiert, um zu zeigen, wie die CLI zum Taggen von Ressourcen verwendet wird: Erstellen einer Pipeline (CLI), Erstellen einer benutzerdefinierten Aktion (CLI) und Erstellen eines Webhooks für eine GitHub Quelle.	15. Mai 2019

Sie können den Verlauf der Ausführung von Aktionen jetzt in einsehen AWS CodePipeline	Sie können jetzt Details zu früheren Ausführungen aller Aktionen in einer Pipeline anzeigen. Dazu gehören Start- und Endzeiten, Dauer, Aktionsausführungs-ID, Status, Ein- und Ausgabe-Artefakt, Standortdetails und Details zu externen Ressourcen. Das Thema Anzeige von Pipeline-Details und -Verlauf wurde um diese Unterstützung erweitert.	20. März 2019
AWS CodePipeline unterstützt jetzt das Veröffentlichen von Anwendungen auf AWS Serverless Application Repository	Sie können jetzt eine Pipeline erstellen CodePipeline , in der Ihre serverlose Anwendung veröffentlicht wird AWS Serverless Application Repository. Das neue Tutorial, Tutorial: Anwendungen auf dem veröffentlichen AWS Serverless Application Repository, enthält Schritte zum Erstellen und Konfigurieren einer Pipeline, mit der Sie Ihre serverlose Anwendung kontinuierlich an den bereitstellen können. AWS Serverless Application Repository	8. März 2019

AWS CodePipeline unterstützt jetzt regionsübergreifende Aktionen in der Konsole	Sie können jetzt regionsübergreifende Aktionen in der AWS CodePipeline Konsole verwalten. Die Aktion „Regionsübergreifende Aktion hinzufügen“ wurde mit den Schritten zum Hinzufügen, Bearbeiten oder Löschen einer Aktion aktualisiert, die sich in einer anderen AWS Region als Ihrer Pipeline befindet. Die Referenzthemen Pipeline erstellen, Pipeline bearbeiten und CodePipeline Pipeline-Struktur wurden aktualisiert.	14. Februar 2019
AWS CodePipeline unterstützt jetzt Amazon S3 S3-Bereitstellungen	Sie können jetzt eine Pipeline erstellen CodePipeline , die Amazon S3 als Bereitstellungsaktionsanbieter verwendet. Ein neues Tutorial, Tutorial: Eine Pipeline erstellen, die Amazon S3 als Bereitstellungsanbieter verwendet, enthält Schritte zum Bereitstellen von Beispieldateien in Ihrem Amazon S3 S3-Bucket mit CodePipeline. Das Referenzthema zur CodePipeline Pipeline-Struktur wurde ebenfalls aktualisiert.	16. Januar 2019

AWS CodePipeline unterstützt jetzt Bereitstellungen von Alexa Skills Kit	Sie können jetzt ein Alexa Skills Kit für den kontinuierlichen Einsatz von Alexa Skills verwenden CodePipeline . Ein neues Tutorial, Tutorial: Eine Pipeline erstellen, die einen Amazon Alexa-Skill bereitstellt, enthält Schritte zum Erstellen von Anmeldeinformationen, mit denen Sie eine Verbindung AWS CodePipeline zu Ihrem Alexa Skills Kit-Entwicklerkonto herstellen können, und zum Erstellen einer Pipeline, die einen Beispiel-Skill bereitstellt. Das Referenzthema zur CodePipeline Pipeline-Struktur wurde aktualisiert.	19. Dezember 2018
AWS CodePipeline unterstützt jetzt Amazon VPC-Endpunkte, die von AWS PrivateLink	Sie können jetzt AWS CodePipeline über einen privaten Endpunkt in Ihrer VPC eine direkte Verbindung herstellen, sodass der gesamte Datenverkehr innerhalb Ihrer VPC und des AWS Netzwerks bleibt. Weitere Informationen finden Sie unter Verwendung CodePipeline mit Amazon Virtual Private Cloud.	6. Dezember 2018

AWS CodePipeline unterstützt jetzt Amazon ECR-Quellaktionen und ECS-to-Deployment-Aktionen CodeDeploy	Sie können jetzt CodePipeline und CodeDeploy mit Amazon ECR und Amazon ECS für die kontinuierliche Bereitstellung containerbasierter Anwendungen verwenden. Das neue Tutorial, Eine Pipeline mit einer Amazon ECR-Quelle und CodeDeploy ECS-to-Deployment erstellen, enthält Schritte zur Verwendung der Konsole zum Erstellen einer Pipeline, die in einem Image-Repository gespeicherte Containeranwendungen in einem Amazon ECS-Cluster mit Datenverkehrs-Routing bereitstellt. CodeDeploy Die Referenzthemen Pipeline erstellen und CodePipeline Pipeline-Struktur wurden aktualisiert.	27. November 2018
AWS CodePipeline unterstützt jetzt regionsübergreifende Aktionen in einer Pipeline	Ein neues Thema, Eine regionsübergreifende Aktion hinzufügen, enthält Schritte zur Verwendung von AWS CLI oder AWS CloudFormation zum Hinzufügen einer Aktion, die sich in einer anderen Region als Ihrer Pipeline befindet. Die Referenzthemen Pipeline erstellen, Pipeline bearbeiten und CodePipeline Pipeline-Struktur wurden aktualisiert.	12. November 2018

AWS CodePipeline lässt sich jetzt in Service Catalog integrieren	Sie können Service Catalog jetzt als Bereitstellungsaktion zu Ihrer Pipeline hinzufügen. Auf diese Weise können Sie eine Pipeline einrichten, um Produktupdates in Service Catalog zu veröffentlichen, wenn Sie eine Änderung in Ihrem Quell-Repository vornehmen. Das Thema Integrationen wurde aktualisiert, um diese Unterstützung für Service Catalog widerzuspiegeln. Dem Abschnitt mit den Tutorials wurden zwei Service AWS CodePipeline Catalog-Tutorials hinzugefügt.	16. Oktober 2018
AWS CodePipeline integriert sich jetzt in AWS Device Farm	Sie können Ihrer Pipeline jetzt AWS Device Farm als Testaktion hinzufügen. Auf diese Weise können Sie eine Pipeline zum Testen von mobilen Anwendungen einrichten. Das Thema Integrationen wurde aktualisiert, um diese Unterstützung für AWS Device Farm widerzuspiegeln. Zwei AWS Device Farm -Tutorials wurden dem Abschnitt AWS CodePipeline - Tutorials hinzugefügt.	19. Juli 2018

AWS CodePipeline Aktualisierungsbenachrichtigungen für das Benutzerhandbuch sind jetzt über RSS verfügbar	Die HTML-Version des CodePipeline Benutzerhandbuchs unterstützt jetzt einen RSS-Feed mit Updates, die auf der Seite „Verlauf der Dokumentationsupdates“ dokumentiert sind. Der RSS-Feed umfasst Aktualisierungen nach dem 30. Juni 2018 und später. Zuvor angekündigte Updates stehen nach wie vor auf der Seite Update-Verlauf der Dokumentation zur Verfügung. Verwenden Sie die RSS-Schaltfläche in der oberen Menüanzeige, um den Feed zu abonnieren.	30. Juni 2018

Frühere Aktualisierungen

In der folgenden Tabelle werden wichtige Änderungen in den einzelnen Versionen des CodePipeline Benutzerhandbuchs vom 30. Juni 2018 und früher beschrieben.

Änderung	Beschreibung	Datum geändert
Verwenden Sie Webhooks, um Quelländerungen in GitHub Pipelines zu erkennen	Wenn Sie eine Pipeline in der Konsole erstellen oder bearbeiten, erstellt CodePipeline Now einen Webhook, der Änderungen an Ihrem GitHub Quell-Repository erkennt und dann Ihre Pipeline startet. Informationen zur Migration Ihrer Pipeline finden Sie unter Konfigurieren Sie Ihre GitHub Pipelines für die Verwendung von Webhooks für die Änderungserkennung. Weitere Informationen finden Sie unter Starten einer Pipeline-Ausführung in. CodePipeline	1. Mai 2018

Änderung	Beschreibung	Datum geändert
Aktualisierte Themen	Wenn Sie eine Pipeline in der Konsole erstellen oder bearbeiten, erstellt CodePipeline Now eine Amazon CloudWatch Events-Regel und einen AWS CloudTrail Trail, der Änderungen an Ihrem Amazon S3 S3-Quell-Bucket erkennt und dann Ihre Pipeline startet. Weitere Informati onen zur Migration Ihrer Pipeline finden Sie unter Quellakti onen und Methoden zur Änderungserkennung. Der Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) wurde aktualisiert und zeigt nun, wie die Amazon CloudWatch Events-Regel und der Amazon Events-Tr ail erstellt werden, wenn Sie eine Amazon S3 S3-Quelle auswählen. Erstellen Sie eine Pipeline in CodePipelineund Eine Pipeline bearbeiten in CodePipeline wurden ebenfalls aktualisiert. Weitere Informationen finden Sie unter Starten Sie eine Pipeline in CodePipeline.	22. März 2018
Aktualisiertes Thema	CodePipeline ist jetzt in Europa (Paris) erhältlich. Das Thema Kontingente in AWS CodePipeline wurde aktualisi ert.	21. Februar 2018

Änderung	Beschreibung	Datum geändert
Aktualisierte Themen	Sie können jetzt Amazon ECS für die kontinuierliche Bereitstellung containerbasierter Anwendungen verwenden CodePipeline . Wenn Sie eine Pipeline erstellen, können Sie Amazon ECS als Bereitstellungsanbieter auswählen. Eine Änderung am Code in Ihrem Quellcodeverwaltungs-Repository veranlasst Ihre Pipeline, ein neues Docker-Image zu erstellen, es in Ihre Container-Registry zu übertragen und dann das aktualisierte Image für einen Amazon ECS-Service bereitzustellen. Die Themen Produkt- und Serviceintegrationen mit CodePipelineErstellen Sie eine Pipeline in CodePipeline, und CodePipeline Referenz zur Pipeline-Struktur wurden aktualisiert, um diese Unterstützung für Amazon ECS widerzuspiegeln.	12. Dezember 2017
Aktualisierte Themen	Wenn Sie eine Pipeline in der Konsole erstellen oder bearbeiten, erstellt CodePipeline Now eine Amazon CloudWatch Events-Regel, die Änderungen an Ihrem CodeCommit Repository erkennt und Ihre Pipeline dann automatisch startet. Weitere Informationen zur Migration Ihrer bestehenden Pipeline finden Sie unter Quellaktionen und Methoden zur Änderungserkennung. Das Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository) wurde aktualisiert und zeigt nun, wie die Regel und Rolle von Amazon CloudWatch Events erstellt werden, wenn Sie ein CodeCommit Repository und einen Branch auswählen. Erstellen Sie eine Pipeline in CodePipelineund Eine Pipeline bearbeiten in CodePipeline wurden ebenfalls aktualisiert. Weitere Informationen finden Sie unter Starten Sie eine Pipeline in CodePipeline.	11. Oktober 2017

Änderung	Beschreibung	Datum geändert
Neue und aktualisierte Themen	CodePipeline bietet jetzt integrierte Unterstützung für Benachrichtigungen über Änderungen des Pipeline-Status über Amazon CloudWatch Events und Amazon Simple Notification Service (Amazon SNS). Es wurde ein neues Tutorial mit dem Titel Tutorial: Richten Sie eine CloudWatch Ereignisregel ein, um E-Mail-Benachrichtigungen für Änderungen des Pipeline-Status zu erhalten hinzugefügt. Weitere Informationen finden Sie unter CodePipeline Ereignisse überwachen.	8. September 2017
Neue und aktualisierte Themen	Sie können jetzt CodePipeline als Ziel für Amazon CloudWatch Events-Aktionen hinzufügen. Amazon CloudWatch Events-Regeln können so eingerichtet werden, dass sie Quellenänderungen erkennen, sodass die Pipeline gestartet wird, sobald diese Änderungen eintreten, oder sie können so eingerichtet werden, dass geplante Pipeline-Ausführungen ausgeführt werden. Es wurden Informationen für die Konfigurationsoption für die PollForSourceChanges Quellaktion hinzugefügt. Weitere Informationen finden Sie unter Starten Sie eine Pipeline in CodePipeline.	5. September 2017
Neue -Regionen	CodePipeline ist jetzt im asiatisch-pazifischen Raum (Seoul) und im asiatisch-pazifischen Raum (Mumbai) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	27. Juli 2017
Neue -Regionen	CodePipeline ist jetzt in den USA West (Nordkalifornien), Kanada (Zentral) und Europa (London) erhältlich. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	29. Juni 2017

Änderung	Beschreibung	Datum geändert
Aktualisierte Themen	Sie können jetzt Details zu allen letzten Ausführungen einer Pipeline statt nur zur letzten Ausführung anzeigen. Diese Details umfassen die Start- und Endzeiten, die Dauer und die Ausführungs-ID. Die Details stehen für maximal 100 Pipeline-Ausführungen während der letzten 12 Monate zur Verfügung. Die Themen Pipelines und Details anzeigen in CodePipeline, CodePipeline Referenz zu Berechtigungen und Kontingente in AWS CodePipeline wurden aktualisiert, um diese Unterstützung darzustellen.	22. Juni 2017
Aktualisiertes Thema	Nouvola wurde der Liste der verfügbaren Aktionen in Testen von Aktionsintegrationen hinzugefügt.	18. Mai 2017
Aktualisierte Themen	Im AWS CodePipeline Assistenten wurde die Seite Schritt 4: Beta in Schritt 4: Deploy umbenannt. Der Standardname der in diesem Schritt erstellten Phase wurde von „Beta“ in „Staging“ geändert. Zahlreiche Themen und Screensho ts wurden aktualisiert, um diese Änderungen widerzusp iegeln.	7. April 2017
Aktualisierte Themen	Sie können jetzt zu jeder Phase einer Pipeline eine Testaktion hinzufügen AWS CodeBuild . Auf diese Weise können Sie Komponententests einfacher für Ihren Code ausführen. AWS CodeBuild Vor dieser Version konnten Sie AWS CodeBuild Komponententests nur als Teil einer Build-Aktion ausführen. Eine Build-Aktion erfordert ein Build-Aus gabeartefakt. Einheitentests produzieren diese in der Regel nicht. Die Themen Produkt- und Serviceintegrationen mit CodePipelineEine Pipeline bearbeiten in CodePipeline, und CodePipeline Referenz zur Pipeline-Struktur wurden aktualisiert, um diese Unterstützung für widerzuspiegeln AWS CodeBuild.	8. März 2017

Änderung	Beschreibung	Datum geändert
Neue und aktualisierte Themen	Das Inhaltsverzeichnis wurde neu organisiert, um Abschnitte für Pipelines, Aktionen und Phasenübergänge einzuschließen. Ein neuer Abschnitt für CodePipeline Tutorials wurde hinzugefügt. Produkt- und Serviceintegrationen mit CodePipeline wurde in kürzere Themen unterteilt, um die Benutzerfreundlichkeit zu verbessern. Ein neuer Abschnitt, Autorisierung und Zugriffskontrolle, bietet umfassende Informationen zur Verwendung von AWS Identity and Access Management (IAM) und CodePipeline zur Sicherung des Zugriffs auf Ihre Ressourcen mithilfe von Anmeldeinformationen. Diese Anmeldeinformationen bieten die erforderlichen Berechtigungen für den Zugriff auf AWS Ressourcen, z. B. für das Einfügen und Abrufen von Artefakten aus Amazon S3 S3-Buckets und das Integrieren von AWS OpsWorks Stacks in Ihre Pipelines.	8. Februar 2017
Neue -Region	CodePipeline ist jetzt im asiatisch-pazifischen Raum (Tokio) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	14. Dezember 2016
Neue -Region	CodePipeline ist jetzt in Südamerika (São Paulo) erhältlich. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	7. Dezember 2016

Änderung	Beschreibung	Datum geändert
Aktualisierte Themen	Sie können es jetzt AWS CodeBuild als Build-Aktion zu jeder Phase einer Pipeline hinzufügen. AWS CodeBuild ist ein vollständig verwalteter Build-Service in der Cloud, der Ihren Quellcode kompiliert, Komponententests durchführt und Artefakte erzeugt, die sofort einsatzbereit sind. Sie können ein vorhandenes Build-Projekt verwenden oder eines in der CodePipeline Konsole erstellen. Die Ausgabe des Build-Projekts kann anschließend als Teil einer Pipeline bereitgestellt werden. Die Themen Produkt- und Serviceintegrationen mit CodePipeline Authentifizierung und Zugriffskontrolle CodePipeline Referenz zur Pipeline-Struktur wurden aktualisiert, um diese Unterstützung für widerzuspiegeln AWS CodeBuild. Erstellen Sie eine Pipeline in CodePipeline Sie können das serverlose Anwendungsmodell jetzt CodePipeline zusammen mit AWS CloudFormation und mit dem AWS serverlosen Anwendungsmodell verwenden, um Ihre serverlosen Anwendungen kontinuierlich bereitzustellen. Das Thema Produkt- und Serviceintegrationen mit CodePipeline wurde aktualisiert, um diese Unterstützung widerzuspiegeln. Produkt- und Serviceintegrationen mit CodePipelinewurde neu organisiert und bietet nun Gruppen AWS - und Partnerangebote nach Aktionstyp.	1. Dezember 2016
Neue -Region	CodePipeline ist jetzt in Europa (Frankfurt) erhältlich. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	16. November 2016

Änderung	Beschreibung	Datum geändert
Aktualisierte Themen	AWS CloudFormation kann jetzt als Bereitstellungsanbieter in Pipelines ausgewählt werden, sodass Sie im Rahmen einer Pipeline-Ausführung Maßnahmen für AWS CloudFormation Stacks und Änderungssätze ergreifen können. Die Themen Produkt- und Serviceintegrationen mit CodePipeline Authentifizierung und Zugriffskontrolle CodePipeline Referenz zur Pipeline-Struktur wurden aktualisiert, um diese Unterstützung für widerzuspiegeln. Erstellen Sie eine Pipeline in CodePipeline AWS CloudFormation	3. November 2016
Neue -Region	CodePipeline ist jetzt in der Region Asien-Pazifik (Sydney) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	26. Oktober 2016
Neue -Region	CodePipeline ist jetzt im asiatisch-pazifischen Raum (Singapur) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	20. Oktober 2016
Neue -Region	CodePipeline ist jetzt in der Region USA Ost (Ohio) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	17. Oktober 2016
Aktualisiertes Thema	Erstellen Sie eine Pipeline in CodePipeline wurde aktualisiert, um die Unterstützung für die Anzeige von Versions-IDs für benutzerdefinierte Aktionen in den Listen Quellanbieter provider und Build-Anbieter widerzuspiegeln.	22. September 2016

Änderung	Beschreibung	Datum geändert
Aktualisiertes Thema	Der Abschnitt Genehmigungsaktionen verwalten in CodePipeline wurde aktualisiert, um eine Optimierung widerzuspiegeln, mit deren Hilfe Personen, die Genehmigungsaktionen prüfen, das Formular Approve or reject the revision direkt aus einer E-Mail-Benachrichtigung heraus öffnen können.	14. September 2016
Neue und aktualisierte Themen	Ein neues Thema, in dem beschrieben wird, wie Sie Details zu Codeänderungen anzeigen können, die derzeit in Ihrer Software-Release-Pipeline vorgenommen werden. Der schnelle Zugriff auf diese Informationen kann bei der Prüfung manueller Genehmigungsaktionen oder der Behebung von Fehlern in Ihrer Pipeline nützlich sein. Ein neuer Abschnitt mit dem Titel Überwachung von Pipelines stellt eine zentrale Stelle für alle Themen im Zusammenhang mit der Überwachung von Status und Fortschritt Ihrer Pipelines dar.	08. September 2016
Neue und aktualisierte Themen	Ein neuer Abschnitt mit dem Titel Genehmigungsaktionen verwalten in CodePipeline stellt Informationen zur Konfiguration und Verwendung manueller Genehmigungsaktionen in Pipelines bereit. Die Themen in diesem Abschnitt enthalten grundlegende Informationen zum Genehmigungsprozess, Anweisungen zum Einrichten der erforderlichen IAM-Berechtigungen, zum Erstellen von Genehmigungsaktionen und zum Genehmigen oder Ablehnen von Genehmigungsaktionen sowie Beispiele für JSON-Daten, die generiert werden, wenn eine Genehmigungsaktion in einer Pipeline erreicht wird.	06. Juli 2016
Neue -Region	CodePipeline ist jetzt in der Region Europa (Irland) verfügbar. Die Themen Kontingente in AWS CodePipeline und Regionen und Endpunkte wurden aktualisiert.	23. Juni 2016

Änderung	Beschreibung	Datum geändert
Neues Thema	Ein neues Thema mit dem Titel Eine fehlgeschlagene Aktion in einer Phase wiederholen wurde hinzugefügt, um zu beschreiben, wie eine fehlgeschlagene Aktion oder eine Gruppe fehlgeschlagener paralleler Aktionen in einer Phase wiederholt werden können.	22. Juni 2016
Aktualisierte Themen	Eine Reihe von Themen, darunter Erstellen Sie eine Pipeline in CodePipeline Authentifizierung und Zugriffsk ontrolle, und, CodePipeline Referenz zur Pipeline-StrukturProdukt- und Serviceintegrationen mit CodePipeline, wurden aktualisiert und bieten nun Unterstützung für die Konfiguration einer Pipeline zur Bereitstellung von Code in Verbindung mit benutzerdefinierten Chef-Koch büchern und -Anwendungen, die in AWS OpsWorks erstellt wurden. CodePipeline Support für AWS OpsWorks ist derzeit nur in der Region USA Ost (Nord-Virginia) (us-east-1) verfügbar.	2. Juni 2016
Neue und aktualisierte Themen	Es wurde ein neues Thema mit dem Titel Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommitRepository) hinzugefügt. Dieses Thema enthält ein Beispiel für eine exemplarische Vorgehensweise, in der gezeigt wird, wie ein CodeCommit Repository und ein Branch als Quellpfad für eine Quellaktion in einer Pipeline verwendet werden. Verschiedene andere Themen wurden aktualisiert, um dieser Integration Rechnung zu tragen CodeCommit, darunter Authentifizierung und Zugriffskontrolle, Produkt- und Serviceintegrationen mit CodePipelineTutorial: Erstellen einer vierstufigen Pipeline, undProblembehebung CodePipeline.	18. April 2016

Änderung	Beschreibung	Datum geändert
Neues Thema	Es wurde ein neues Thema mit dem Titel Rufen Sie eine AWS Lambda Funktion in einer Pipeline auf in CodePipeline hinzugefügt. Dieses Thema enthält AWS Lambda Beispielfunktionen und Schritte zum Hinzufügen von Lambda-Funktionen zu Pipelines.	27. Januar 2016
Aktualisiertes Thema	Zu den ressourcenbasierten Richtlinien für Authentifizierung und Zugriffskontrolle wurde ein neuer Abschnitt hinzugefügt.	22. Januar 2016
Neues Thema	Es wurde ein neues Thema mit dem Titel Produkt- und Serviceintegrationen mit CodePipeline hinzugefügt. Informationen zu Integrationen mit Partnern und anderen wurden in AWS-Services dieses Thema verschoben. Außerdem wurden Links zu Blogs und Videos hinzugefügt.	17. Dezember 2015
Aktualisiertes Thema	Unter Produkt- und Serviceintegrationen mit CodePipeline wurden Details zur Integration in Solano CI hinzugefügt.	17. November 2015
Aktualisiertes Thema	Das CodePipeline Plugin für Jenkins ist jetzt über den Jenkins Plugin Manager als Teil der Plugin-Bibliothek für Jenkins verfügbar. Die Schritte für die Installation des Plug-Ins in Tutorial: Erstellen einer vierstufigen Pipeline wurden aktualisiert.	9. November 2015
Neue -Region	CodePipeline ist jetzt in der Region USA West (Oregon) verfügbar. Das Thema Kontingente in AWS CodePipeline wurde aktualisiert. Außerdem wurden Links zu Regionen und Endpunkte hinzugefügt.	22. Oktober 2015

Änderung	Beschreibung	Datum geändert
Neues Thema	Es wurden zwei neue Themen mit den Titeln Konfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte für CodePipeline und Erstellen Sie eine Pipeline CodePipeline , in der Ressourcen von einem anderen AWS Konto verwendet werden hinzugefügt. Zur Authentifizierung und Zugriffskontrolle wurde ein neuer Abschnitt Beispiel 8: Verwenden von AWS -Ressourcen, die mit einem anderen Konto in einer Pipeline verknüpft sind hinzugefügt.	25. August 2015
Aktualisiertes Thema	Das Thema Erstellen und fügen Sie eine benutzerd efinierte Aktion hinzu in CodePipeline wurde aktualisi ert, um Änderungen der Struktur widerzuspiegeln, darunter `inputArtifactDetails` und `outputArtifactDetails` .	17. August 2015
Aktualisiertes Thema	Das Problembehebung CodePipeline Thema wurde mit überarbeiteten Schritten zur Behebung von Problemen mit der Servicerolle und Elastic Beanstalk aktualisiert.	11. August 2015
Aktualisiertes Thema	Das Thema Authentifizierung und Zugriffskontrolle wurde mit den neuesten Änderungen an der Servicerolle für aktualisiert. CodePipeline	6. August 2015
Neues Thema	Das Thema Problembehebung CodePipeline wurde hinzugefügt. Aktualisierte Schritte wurden für IAM-Rollen und Jenkins hinzugefügt. Tutorial: Erstellen einer vierstufi gen Pipeline	24. Juli 2015
Aktualisiertes Thema	Dem Herunterladen der Beispieldateien in Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) und Tutorial: Erstellen einer vierstufigen Pipeline wurden aktualisierte Schritte hinzugefügt.	22. Juli 2015

Änderung	Beschreibung	Datum geändert
Aktualisiertes Thema	In Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) wurde eine temporäre Umgebung für Probleme beim Herunterladen von Beispieldateien hinzugefügt.	17. Juli 2015
Aktualisiertes Thema	In Kontingente in AWS CodePipeline wurde ein Link hinzugefügt, um auf Informationen dazu zu verweisen, welche Einschränkungen geändert werden können.	15. Juli 2015
Aktualisiertes Thema	Der Abschnitt „Verwaltete Richtlinien“ in „Authentifizierung und Zugriffskontrolle“ wurde aktualisiert.	10. Juli 2015
Erste veröffentlichte Version	Dies ist die erste öffentliche Version des Benutzerhandbuchs CodePipeline .	9. Juli 2015

AWS Glossar

Die neueste AWS Terminologie finden Sie im AWS Glossar in der AWS-Glossar Referenz.

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.