Benutzer-Leitfaden

AWS CodePipeline



API-Version 2015-07-09

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS CodePipeline: Benutzer-Leitfaden

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was CodePipeline ist ... 1
Kontinuierliche Bereitstellung und kontinuierliche Integration ... 1
Was kann ich damit machen CodePipeline? ... 2
Ein kurzer Blick auf CodePipeline ... 3
Wie fange ich an mit CodePipeline? ... 4
Konzepte ... 4
Pipelines ... 5
Pipeline-Ausführungen ... 6
Operationen auf der Bühne ... 8
Aktionsausführungen ... 8
Arten der Ausführung ... 9
Aktionstypen ... 9
-Artefakte ... 9
Quell-Revisionen ... 10
Auslöser ... 10
Variablen ... 11
Bedingungen ... 11
Regeln ... 12
DevOps Beispiel für eine Pipeline ... 12
So funktionieren Pipeline-Ausführungen ... 14
So werden Pipeline-Ausführungen gestartet ... 15
Wie Quellrevisionen bei Pipeline-Ausführungen verarbeitet werden ... 15
So funktionieren Quellüberschreibungen mit dem EventBridge Eingangstransformator ... 16
Pipeline-Ausführungen beenden ... 17
Wie werden Ausführungen im SUPERSEDED-Modus verarbeitet ... 21
Wie werden Ausführungen im QUEUED-Modus verarbeitet ... 22
Wie werden Ausführungen im PARALLEL-MODUS verarbeitet ... 24
Verwaltung des Pipeline-Ablaufs ... 24
Eingabe- und Ausgabe-Artefakte ... 27
Wie funktionieren die Bühnenbedingungen? ... 30
Regeln für die Bedingungen einer Phase ... 34
Pipeline-Typen ... 35
Welcher Pipeline-Typ ist der richtige für mich? ... 35
Erste Schritte ... 41

Melden Sie sich an für ein AWS-Konto ... 41
Wenden Sie eine verwaltete Richtlinie für den Administratorzugriff an auf CodePipeline ... 42
Installieren Sie das AWS CLI ... 43
Öffnen Sie die Konsole für CodePipeline ... 45
Nächste Schritte ... 45
Produkt- und Service-Integrationen ... 46
Integrationen mit Aktionstypen CodePipeline ... 46
Quellaktions-Integrationen ... 46
Erstellen von Aktionsintegrationen ... 54
Testen von Aktionsintegrationen ... 56
Bereitstellungsaktions-Integrationen ... 59
Integration von Genehmigungsaktionen mit Amazon Simple Notification Service ... 65
Aufrufen von Aktionsintegrationen ... 66
Allgemeine Integrationen mit CodePipeline ... 67
Beispiele aus der Community ... 71
Blog-Posts ... 71
Lernprogramme ... 76
Tutorial: Bereitstellung auf Amazon EC2-Instances mit CodePipeline ... 77
Voraussetzungen ... 77
Schritt 1: Amazon EC2 Linux-Instances erstellen ... 78
Schritt 2: Fügen Sie der EC2-Instance-Rolle Artifact-Bucket-Berechtigungen hinzu ... 80
Schritt 3: Füge eine Skriptdatei zu deinem Repository hinzu ... 81
Schritt 4: Deine Pipeline erstellen ... 82
Schritt 5: Testen Sie Ihre Pipeline ... 86
Tutorial: Erstellen Sie ein Docker-Image und übertragen Sie es auf Amazon ECR mit CodePipeline (Typ V2) ... 87
Voraussetzungen ... 88
Schritt 1: Fügen Sie Ihrem Quell-Repository ein Dockerfile hinzu ... 88
Schritt 2: Füge eine imagedefinitions.json-Datei zu deinem Quell-Repository hinzu ... 90
Schritt 3: Deine Pipeline erstellen ... 91
Schritt 4: Testen Ihrer Pipeline ... 97
Tutorial: Bereitstellung auf Amazon EKS mit CodePipeline ... 98
Voraussetzungen ... 98
Schritt 1: (Optional) Erstellen Sie einen Cluster in Amazon EKS ... 99
Schritt 2: Konfigurieren Sie Ihren privaten Cluster in Amazon EKS ... 102
Schritt 3: Aktualisieren Sie die CodePipeline Servicerollenrichtlinie in IAM ... 102

Schritt 4: Erstellen Sie einen Zugriffseintrag für die CodePipeline Servicerolle 105
Schritt 5: Erstellen Sie ein Quell-Repository und fügen Sie die Helm-Chart-Konfigurationsdateien hinzu 106
Schritt 6: Deine Pipeline erstellen 107
Tutorial: Erstellen Sie eine Pipeline, die Befehle mit Compute ausführt (Typ V2) 109
Voraussetzungen 110
Schritt 1: Erstellen Sie Quelldateien und übertragen Sie sie in Ihr GitHub Repository 110
Schritt 2: Erstellen der Pipeline 111
Schritt 3: Führen Sie Ihre Pipeline aus und überprüfen Sie die Build-Befehle 115
Tutorial: Verwenden Sie Git-Tags, um Ihre Pipeline zu starten 116
Voraussetzungen 117
Schritt 1: Öffne CloudShell und klone dein Repository 117
Schritt 2: Erstellen Sie eine Pipeline, die auf Git-Tags ausgelöst wird 118
Schritt 3: Tagge deine Commits zur Veröffentlichung 122
Schritt 4: Änderungen veröffentlichen und Protokolle anzeigen 123
Tutorial: Filtere nach Branch-Namen für Pull Requests, um deine Pipeline zu starten (Typ V2) 124
Voraussetzungen 124
Schritt 1: Erstellen Sie eine Pipeline, um mit einem Pull-Request für bestimmte Branches zu beginnen 125
Schritt 2: Erstelle einen Pull-Request und füge ihn zusammen GitHub.com , um mit der Ausführung deiner Pipeline zu beginnen 128
Tutorial: Verwenden Sie Variablen auf Pipeline-Ebene 129
Voraussetzungen 130
Schritt 1: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt 130
Schritt 2: Änderungen veröffentlichen und Protokolle anzeigen 134
Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) 134
Erstellen eines S3-Buckets 136
Erstellen Sie Windows Server Amazon EC2-Instanzen und installieren Sie den Agenten CodeDeploy 138
Erstellen Sie eine Anwendung in CodeDeploy 140
Erstellen Ihrer ersten Pipeline 142
Hinzufügen einer weiteren Phase 146
Deaktivieren und Aktivieren von Übergängen zwischen Phasen 153
Bereinigen von Ressourcen 154
Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommit Repository) 155

Erstellen Sie ein CodeCommit Repository ... 156
Herunterladen und Push-Übertragen Ihres Codes sowie Durchführen eines Commits für Ihren Code ... 157
Erstellen Sie eine Amazon EC2 Linux-Instance und installieren Sie den Agenten CodeDeploy ... 160
Erstellen Sie eine Anwendung in CodeDeploy ... 162
Erstellen Ihrer ersten Pipeline ... 164
Aktualisiere den Code in deinem CodeCommit Repository ... 167
Bereinigen von Ressourcen ... 169
Weitere Informationen ... 169
Tutorial: Erstellen einer vierstufigen Pipeline ... 170
Erfüllen der Voraussetzungen ... 172
Erstellen Sie eine Pipeline ... 176
Hinzufügen weiterer Phasen ... 178
Bereinigen von Ressourcen ... 182
Tutorial: Richten Sie eine CloudWatch Event-Regel ein, um E-Mail-Benachrichtigungen über Änderungen des Pipeline-Status zu erhalten ... 183
Richten Sie eine E-Mail-Benachrichtigung mit Amazon SNS ein ... 184
Erstellen Sie eine Regel für die Benachrichtigung über CloudWatch Ereignisse für CodePipeline ... 185
Bereinigen von Ressourcen ... 187
Tutorial: Erstellen und testen Sie eine Android-App mit AWS Device Farm ... 187
Für CodePipeline die Verwendung Ihrer Device Farm Farm-Tests konfigurieren ... 188
Tutorial: Testen Sie eine iOS-App mit AWS Device Farm ... 193
Für CodePipeline die Verwendung Ihrer Device Farm Farm-Tests konfigurieren (Beispiel Amazon S3) ... 195
Tutorial: Erstellen Sie eine Pipeline, die im Service Catalog bereitgestellt wird ... 200
Option 1: Bereitstellung im Service Catalog ohne Konfigurationsdatei ... 201
Option 2: Mithilfe einer Konfigurationsdatei im Service Catalog bereitstellen ... 206
Tutorial: Erstellen Sie eine Pipeline mit AWS CloudFormation ... 212
Beispiel 1: Erstellen Sie eine AWS CodeCommit Pipeline mit AWS CloudFormation ... 212
Beispiel 2: Erstellen Sie eine Amazon S3-Pipeline mit AWS CloudFormation ... 215
Tutorial: Erstellen Sie eine Pipeline, die Variablen aus AWS CloudFormation Bereitstellungsaktionen verwendet ... 219
Voraussetzungen: Erstellen Sie eine AWS CloudFormation Servicerolle und ein CodeCommit Repository ... 220

Schritt 1: Laden Sie das Beispiel herunter, bearbeiten Sie es und laden Sie es hoch AWS CloudFormation Vorlage ... 220
Schritt 2: Erstellen der Pipeline ... 222
Schritt 3: Fügen Sie eine hinzu CloudFormation Bereitstellungsaktion zum Erstellen des Änderungssatzes ... 225
Schritt 4: Hinzufügen einer manuellen Genehmigungsaktion ... 226
Schritt 5: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Änderungssatz auszuführen ... 226
Schritt 6: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Stapel zu löschen ... 227
Tutorial: Amazon ECS-Standardbereitstellung mit CodePipeline ... 228
Voraussetzungen ... 229
Schritt 1: Hinzufügen einer Build-Spezifikationsdatei zu Ihrem Quell-Repository ... 232
Schritt 2: Erstellen einer durchgängigen Bereitstellungs-Pipeline ... 234
Schritt 3: Amazon ECR-Berechtigungen zur CodeBuild Rolle hinzufügen ... 236
Schritt 4: Testen Ihrer Pipeline ... 236
Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und -Bereitstellung ECS-to-CodeDeploy ... 237
Voraussetzungen ... 239
Schritt 1: Image erstellen und in ein Amazon ECR-Repository übertragen ... 239
Schritt 2: Erstellen Sie die Aufgabendefinition und die AppSpec Quelldateien und übertragen Sie sie in ein Repository CodeCommit ... 241
Schritt 3: Erstellen Ihres Application Load Balancer und der Zielgruppen ... 245
Schritt 4: Erstellen Sie Ihren Amazon ECS-Cluster und -Service ... 248
Schritt 5: Erstellen Sie Ihre CodeDeploy Anwendung und Bereitstellungsgruppe (ECS-Rechenplattform) ... 251
Schritt 6: Erstellen Ihrer Pipeline ... 252
Schritt 7: Vornehmen einer Änderung an Ihrer Pipeline und Überprüfen der Bereitstellung ... 257
Tutorial: Erstellen einer Pipeline zum Bereitstellen eines Amazon Alexa-Skills ... 257
Voraussetzungen ... 258
Schritt 1: Erstellen eines Alexa Developer Services-LWA-Sicherheitsprofils ... 258
Schritt 2: Erstellen Sie Alexa-Skill-Quelldateien und übertragen Sie sie in Ihr CodeCommit Repository ... 258
Schritt 3: Verwenden von ASK-CLI-Befehlen zum Erstellen eines Aktualisierungs-Tokens ... 261
Schritt 4: Erstellen Ihrer Pipeline ... 261

Schritt 5: Vornehmen einer Änderung an einer beliebigen Quelldatei und Überprüfen der Bereitstellung ... 264
Tutorial: Erstellen Sie eine Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet ... 265
Option 1: Statische Website-Dateien auf Amazon S3 bereitstellen ... 266
Option 2: Stellen Sie erstellte Archivdateien aus einem S3-Quell-Bucket in Amazon S3 bereit ... 271
Tutorial: Veröffentlichen Sie Anwendungen auf dem AWS Serverless Application Repository ... 277
Bevor Sie beginnen ... 278
Schritt 1: Erstellen einer buildspec.yml-Datei ... 279
Schritt 2: Erstellen und Konfigurieren Ihrer Pipeline ... 279
Schritt 3: Bereitstellen der Veröffentlichungsanwendung ... 282
Schritt 4: Erstellen der Veröffentlichungsaktion ... 282
Tutorial: Bereitstellung von Lambda-Funktionen mit CodePipeline ... 283
Voraussetzungen ... 284
Schritt 1: Erstellen Sie Ihre Lambda-Beispielfunktion ... 284
Schritt 2: Laden Sie die Funktionsdatei in Ihr Repository hoch ... 285
Schritt 3: Erstellen Sie Ihre Pipeline ... 285
Schritt 4: Testen Ihrer Pipeline ... 288
Weitere Informationen ... 289
Tutorial: Verwenden von Variablen mit Lambda-Aufrufaktionen ... 289
Voraussetzungen ... 290
Schritt 1: Erstellen einer Lambda-Funktion ... 290
Schritt 2: Fügen Sie Ihrer Pipeline eine Lambda-Aufrufaktion und eine manuelle Genehmigungsaktion hinzu ... 293
Tutorial: Verwenden Sie eine AWS Step Functions Aufruf-Aktion ... 295
Voraussetzung: Erstellen oder Auswählen einer einfachen Pipeline ... 296
Schritt 1: Erstellen des Beispielzustandsautomaten ... 296
Schritt 2: Fügen Sie Ihrer Pipeline eine Aktion zum Aufrufen von Step Functions hinzu ... 297
Tutorial: Erstellen Sie eine Pipeline, die AppConfig als Bereitstellungsanbieter verwendet wird ... 298
Voraussetzungen ... 299
Schritt 1: Erstellen Sie Ihre Ressourcen AWS AppConfig ... 299
Schritt 2: Laden Sie Dateien in Ihren S3-Quell-Bucket hoch ... 300
Schritt 3: Erstellen Ihrer Pipeline ... 301
Schritt 4: Nehmen Sie eine Änderung an einer beliebigen Quelldatei vor und überprüfen Sie die Bereitstellung ... 303

Tutorial: Verwenden Sie den vollständigen Klon mit einer GitHub Pipeline-Quelle 303
Voraussetzungen 304
Schritt 1: Erstellen Sie eine README-Datei 304
Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt 305
Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um Verbindungen zu verwenden 309
Schritt 4: Repository-Befehle in der Build-Ausgabe anzeigen 309
Tutorial: Verwenden Sie den vollständigen Klon mit einer CodeCommit Pipeline-Quelle 310
Voraussetzungen 311
Schritt 1: Erstellen Sie eine README-Datei 311
Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt 311
Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um das Repository zu klonen 314
Schritt 4: Repository-Befehle in der Build-Ausgabe anzeigen 315
Tutorial: Erstellen Sie eine Pipeline mit AWS CloudFormation StackSets Bereitstellungsaktionen 315
Voraussetzungen 316
Schritt 1: Laden Sie das Beispiel hoch AWS CloudFormation Vorlage und Parameterdatei .. 317
Schritt 2: Erstellen der Pipeline 319
Schritt 3: Erste Bereitstellung anzeigen 322
Schritt 4: Fügen Sie eine CloudFormationStackInstances Aktion hinzu 322
Schritt 5: Sehen Sie sich die Stack-Set-Ressourcen für Ihre Bereitstellung an 323
Schritt 6: Aktualisieren Sie Ihr Stack-Set 324
Erstellen Sie eine Variablenprüfregel für eine Pipeline als Eingangsbedingung 325
Voraussetzungen 326
Schritt 1: Erstelle eine Beispiel-Quelldatei und füge sie deinem GitHub Repository hinzu 326
Schritt 2: Erstellen der Pipeline 327
Schritt 2: Bearbeiten Sie die Build-Phase, um die Bedingung und die Regel hinzuzufügen ... 330
Schritt 3: Führen Sie die Pipeline aus und sehen Sie sich die aufgelösten Variablen an 332
Bewährte Methoden und Anwendungsfälle 335
Beispiele für die Verwendung CodePipeline 335
Verwendung CodePipeline mit Amazon S3 S3, AWS CodeCommit, und AWS CodeDeploy 335
Verwendung CodePipeline mit Drittanbieter-Aktionsanbietern (GitHub und Jenkins) 336
Wird verwendet, CodePipeline um Code zu kompilieren, zu erstellen und zu testen mit CodeBuild 337

Verwendung CodePipeline mit Amazon ECS für die kontinuierliche Bereitstellung containerbasierter Anwendungen in der Cloud ... 337
Verwendung CodePipeline mit Elastic Beanstalk für die kontinuierliche Bereitstellung von Webanwendungen in der Cloud ... 337
Verwenden Sie mit CodePipeline AWS Lambda für die kontinuierliche Bereitstellung von Lambda-based und serverlose Anwendungen ... 337
Verwenden Sie mit CodePipeline CloudFormation Vorlagen für die kontinuierliche Bereitstellung in der Cloud ... 338
CodePipeline Mit Amazon VPC verwenden ... 339
Verfügbarkeit ... 339
Erstellen Sie einen VPC-Endpunkt für CodePipeline ... 340
Fehlerbehebung für Ihre VPC-Einrichtung ... 341
Definieren Sie CI/CD Pipelines mit Phasen und Aktionen ... 342
Erstellen Sie eine Pipeline, Phasen und Aktionen ... 343
Erstellen Sie eine benutzerdefinierte Pipeline (Konsole) ... 344
Erstellen einer Pipeline (CLI) ... 359
Erstellen Sie eine Pipeline aus statischen Vorlagen ... 365
Bearbeiten einer Pipeline ... 371
Bearbeiten einer Pipeline (Konsole) ... 372
Bearbeiten einer Pipeline (AWS CLI) ... 376
Pipelines und Details anzeigen ... 381
Pipelines anzeigen (Konsole) ... 381
Aktionsdetails in einer Pipeline (Konsole) anzeigen ... 386
Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen ... 389
Anzeigen von der Pipeline-Details und des Verlaufs (CLI) ... 390
Regelergebnisse für Stufenbedingungen im Ausführungsverlauf anzeigen ... 391
Pipeline löschen ... 394
Löschen einer Pipeline (Konsole) ... 394
Löschen einer Pipeline (CLI) ... 394
Erstellen einer Pipeline, die Ressourcen aus einem anderen Konto verwendet ... 396
Voraussetzung: Erstellen Sie eine AWS KMS Verschlüsselungsschlüssel ... 398
Schritt 1: Einrichten von Kontorichtlinien und Rollen ... 399
Schritt 2: Bearbeiten der Pipeline ... 407
Migrieren Sie Polling-Pipelines, um die ereignisbasierte Änderungserkennung zu verwenden ... 410
Wie migriert man Polling-Pipelines ... 411
Polling-Pipelines in Ihrem Konto anzeigen ... 412

Migrieren Sie Polling-Pipelines mit einer Quelle CodeCommit 417
Migrieren Sie Polling-Pipelines mit einer S3-Quelle, die für Ereignisse aktiviert ist 440
Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail 460
Migrieren Sie Polling-Pipelines für eine GitHub Quellaktion (über die OAuth-App) zu Verbindungen 496
Migrieren Sie Polling-Pipelines für eine GitHub Quellaktion (über die OAuth-App) zu Webhooks 500
Erstellen Sie die CodePipeline Servicerolle 518
Erstellen Sie die CodePipeline Servicerolle (Konsole) 519
Erstellen Sie die CodePipeline Servicerolle (CLI) 520
Taggen von Ressourcen 522
Markieren einer Pipeline 523
Markieren von Pipelines (Konsole) 523
Markieren von Pipelines (CLI) 525
Erstellen einer Benachrichtigungsregel 528
..... 532
Stellen Sie mithilfe von Quellaktionen eine Verbindung zu Quellenanbietern von Erstanbietern her 535
Amazon ECR-Quellaktionen und EventBridge 535
Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (Konsole) 536
Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (CLI) 539
Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (CloudFormation Vorlage) 543
Herstellen einer Verbindung zu Amazon S3-Quellaktionen, die EventBridge Ereignisse verwenden 548
Erstellen Sie Pipelines mit einer für Ereignisse aktivierten S3-Quelle (CLI) 549
Erstellen Sie Pipelines mit einer für Ereignisse aktivierten S3-Quelle (CloudFormation Vorlage) 553
Verbindung zu Amazon S3-Quellaktionen herstellen, die EventBridge und verwenden AWS CloudTrail 568
Erstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (Konsole) 569
Erstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (CLI) 573
Erstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (CloudFormation Vorlage) 579
CodeCommit Quellenaktionen und EventBridge 592
Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (Konsole) 593

Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (CLI) 596
Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (CloudFormation Vorlage) 601
Fügen Sie externe Quellanbieter zu Pipelines hinzu mit AWS CodeConnections 610
DevOps Azure-Verbindungen 610
Stellen Sie eine Verbindung zu Azure DevOps (Konsole) her 612
Verbindung zu Azure DevOps (CLI) herstellen 616
Bitbucket Cloud-Verbindungen 618
Stelle eine Verbindung zu Bitbucket Cloud (Konsole) her 620
Verbindung zu Bitbucket Cloud (CLI) herstellen 623
GitHub Verbindungen 625
Stellen Sie eine Verbindung zu GitHub (Konsole) her 627
Verbindung herstellen zu GitHub (CLI) 631
GitHub Enterprise Server-Verbindungen 633
Stellen Sie eine Verbindung zu GitHub Enterprise Server (Konsole) her 635
Erstellen Sie einen Host und eine Verbindung zu GitHub Enterprise Server (CLI) 639
GitLab.com Verbindungen 643
Stelle eine Verbindung zu GitLab.com (Konsole) her 645
Verbindung herstellen zu GitLab.com (CLI) 649
Verbindungen für Selbstverwalter GitLab 651
Stellen Sie eine Verbindung zur GitLab Selbstverwaltung (Konsole) her 653
Erstellen Sie einen Host und eine Verbindung zu GitLab Self-Managed (CLI) 657
Verwenden Sie eine Verbindung, die mit einem anderen Konto geteilt wird 660
Automatisieren Sie das Starten von Pipelines mithilfe von Triggern und Filtern 661
Überlegungen zu Triggerfiltern 663
Pull-Request-Ereignisse für Trigger nach Anbieter 664
Beispiele für Triggerfilter 665
Trigger beim Code-Push ohne Filter hinzufügen 674
Fügen Sie einen Trigger mit den Ereignistypen Code-Push oder Pull Request hinzu 674
Fügen Sie Filter für Push- und Pull-Request-Ereignistypen hinzu (Konsole) 676
Fügen Sie Filter für Push- und Pull-Request-Ereignistypen (CLI) hinzu 678
Fügen Sie Filter für Push- und Pull-Request-Ereignistypen hinzu (CloudFormation Vorlagen) 681
Trigger hinzufügen, um die Änderungserkennung auszuschalten 682
Manuelles Starten und Stoppen von Pipelines 684
Starten Sie eine Pipeline in CodePipeline 684

Manuelles Starten einer Pipeline ... 686
Starten Sie eine Pipeline nach einem Zeitplan ... 688
Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung ... 691
Pipeline-Ausführung anhalten ... 694
Pipeline-Ausführung beenden (Konsole) ... 695
Stoppen Sie eine eingehende Ausführung (Konsole) ... 699
Pipeline-Ausführung anhalten (CLI) ... 700
Stoppen Sie eine eingehende Ausführung (CLI) ... 701
Zeigen Sie den Verlauf an und legen Sie den Modus für Pipeline-Ausführungen fest ... 703
Hinrichtungen anzeigen ... 703
Ausführungsverlauf einer Pipeline anzeigen (Konsole) ... 703
Anzeige des Ausführungsstatus (Konsole) ... 705
Eine eingehende Ausführung anzeigen (Konsole) ... 707
Anzeigen von Revisionen der Pipeline-Ausführungsquelle (Konsole) ... 708
Anzeigen von Aktionsausführungen (Konsole) ... 710
Anzeigen von Informationen zu Aktionsartefakten und zum Artefaktspeicher (Konsole) ... 711
Anzeigen von der Pipeline-Details und des Verlaufs (CLI) ... 711
Legen Sie den Pipeline-Ausführungsmodus fest oder ändern Sie ihn ... 724
Überlegungen zur Anzeige der Ausführungsmodi ... 724
Überlegungen zum Umschalten zwischen den Ausführungsmodi ... 727
Stellen Sie den Pipeline-Ausführungsmodus ein oder ändern Sie ihn (Konsole) ... 729
Stellen Sie den Pipeline-Ausführungsmodus (CLI) ein ... 729
Stufen rückgängig machen oder erneut versuchen ... 733
Konfiguration des Wiederholungsversuchs für eine fehlgeschlagene Phase oder für fehlgeschlagene Aktionen ... 733
Überlegungen zur Wiederholung der Phase ... 734
Wiederholen Sie eine fehlgeschlagene Phase manuell ... 734
Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler ... 739
Konfiguration des Stagen-Rollbacks ... 745
Überlegungen zu Rollbacks ... 746
Führt eine Stufe manuell zurück ... 746
Konfigurieren Sie eine Phase für das automatische Rollback ... 752
Sehen Sie sich den Rollback-Status in der Ausführungsliste an ... 756
Details zum Rollback-Status anzeigen ... 759
Konfigurieren Sie die Bedingungen für eine Phase ... 764
Anwendungsfälle für Bühnenbedingungen ... 765

Überlegungen zu Ergebnissen, die für die Bedingungen der Phase konfiguriert wurden 766
Überlegungen zu Regeln, die für die Bedingungen der Phase konfiguriert sind 767
Zugangsbedingungen erstellen 767
Eingabebedingungen erstellen — Beispiel für eine CloudWatchAlarm Regel (Konsole) 768
Eingabebedingungen mit Ergebnis und `VariableCheck` Regel überspringen (Konsole) erstellen 769
Eingabebedingungen erstellen (CLI) 771
Zugangsbedingungen (CFN) erstellen 773
Bedingungen bei Ausfall erstellen 774
Bedingungen bei Ausfall erstellen (Konsole) 774
Erstellen von OnFailure-Bedingungen mit einem Wiederholungsergebnis (Beispiel) (Konsole) 776
Bedingungen bei Ausfall erstellen (CLI) 777
Bedingungen bei Ausfall erstellen (CFN) 779
Bedingungen bei Erfolg erstellen 780
Bedingungen bei Erfolg erstellen (Konsole) 780
Bedingungen bei Erfolg erstellen (CLI) 782
Erstellen Sie eine On Success-Bedingung (CFN) 784
Stage-Bedingungen löschen 785
Übergeordnete Stufenbedingungen 786
Verwenden Sie Aktionstypen, benutzerdefinierte Aktionen und Genehmigungsaktionen 788
Mit Aktionstypen arbeiten 788
Fordern Sie einen Aktionstyp an 790
Fügen Sie einer Pipeline (Konsole) einen verfügbaren Aktionstyp hinzu 796
Zeigen Sie einen Aktionstyp an 798
Aktualisieren Sie einen Aktionstyp 800
Erstellen einer benutzerdefinierten Aktion für eine Pipeline 802
Erstellen einer benutzerdefinierten Aktion 804
Erstellen eines Auftragsworkers für Ihre benutzerdefinierte Aktion 808
Hinzufügen einer benutzerdefinierten Aktion zu einer Pipeline 817
Kennzeichnen Sie eine benutzerdefinierte Aktion in CodePipeline 820
Hinzufügen von Tags zu benutzerdefinierten Aktionen 821
Anzeigen von Tags für eine benutzerdefinierte Aktion 821
Bearbeiten von Tags für eine benutzerdefinierte Aktion 822
Entfernen von Tags aus einer benutzerdefinierten Aktion 822
Rufen Sie eine Lambda-Funktion in einer Pipeline auf 823

Schritt 1: Erstellen einer Pipeline ... 825
Schritt 2: Erstellen Sie die Lambda-Funktion ... 826
Schritt 3: Fügen Sie die Lambda-Funktion zu einer Pipeline in der Konsole hinzu CodePipeline ... 831
Schritt 4: Testen Sie die Pipeline mit der Lambda-Funktion ... 832
Schritt 5: Nächste Schritte ... 832
JSON-Beispielereignis ... 833
Weitere Beispielfunktionen ... 835
Fügen Sie einer Phase eine manuelle Genehmigungsaktion hinzu ... 848
Konfigurationsoptionen für manuelle Genehmigungsaktionen ... 849
Setup- und Workflow-Übersicht für Genehmigungsaktionen ... 850
Erteilen Sie einem IAM-Benutzer Genehmigungsberechtigungen in CodePipeline ... 851
Gewähren Sie Amazon SNS-Berechtigungen für eine Servicerolle ... 854
Hinzufügen einer manuellen Genehmigungsaktion ... 856
Erlauben oder Ablehnen einer Genehmigungsaktion ... 861
JSON-Datenformat für manuelle Genehmigungsbenachrichtigungen ... 865
Fügen Sie einer Pipeline eine regionsübergreifende Aktion hinzu ... 866
Verwalte regionsübergreifende Aktionen in einer Pipeline (Konsole) ... 868
Fügen Sie einer Pipeline eine regionsübergreifende Aktion hinzu (CLI) ... 871
Fügen Sie einer Pipeline eine regionsübergreifende Aktion hinzu (CloudFormation) ... 877
Arbeiten mit Variablen ... 879
Konfigurieren von Aktionen für Variablen ... 880
Anzeigen von Ausgabevariablen ... 885
Beispiel: Variablen in manuellen Genehmigungen verwenden ... 888
Beispiel: Verwenden Sie eine BranchName Variable mit CodeBuild Umgebungsvariablen ... 888
Arbeiten mit Phasenübergängen ... 891
Deaktivieren oder Aktivieren von Übergängen (Konsole) ... 891
Deaktivieren oder Aktivieren von Übergängen (CLI) ... 893
Überwachen von Pipelines ... 896
Überwachung von CodePipeline Ereignissen ... 897
Detailtypen ... 899
Pipeline-level Ereignisse ... 901
Stage-level Ereignisse ... 910
Action-level Ereignisse ... 914
Erstellen Sie eine Regel, die bei einem Pipeline-Ereignis eine Benachrichtigung sendet ... 922
Platzhalter-Bucket-Referenz für Ereignisse ... 927

Platzhalter-Bucket-Namen für Ereignisse nach Region ... 928
Protokollierung von AWS CloudTrail-API-Aufrufen mit ... 931
CodePipeline Informationen in CloudTrail ... 931
Grundlegendes zu CodePipeline Protokolldateieinträgen ... 932
CodePipeline CloudWatch Metriken ... 934
PipelineDuration ... 935
FailedPipelineExecutions ... 935
Fehlerbehebung ... 936
Pipeline-Fehler: Eine Pipeline, die konfiguriert ist mit AWS Elastic Beanstalk gibt eine Fehlermeldung zurück: „Die Bereitstellung ist fehlgeschlagen. Die angegebene Rolle verfügt nicht über ausreichende Berechtigungen: Service:AmazonElasticLoadBalancing“ ... 938
Bereitstellungsfehler: Eine Pipeline, die mit einem konfiguriert ist AWS Elastic Beanstalk Die Bereitstellungsaktion bleibt hängen und schlägt nicht fehl, wenn die DescribeEvents "" - Berechtigung fehlt ... 938
Pipeline-Fehler: Eine Quellaktion gibt die Meldung mit unzureichenden Berechtigungen zurück: „Auf den Repository-Namen konnte nicht zugegriffen werden. CodeCommit Überprüfen Sie, ob die Pipeline-IAM-Rolle über ausreichende Berechtigungen für den Zugriff auf das Repository verfügt.“ ... 939
Pipeline-Fehler: Ein Jenkins-Build oder eine Testaktion werden über eine lange Zeit ausgeführt und schlagen dann aufgrund fehlender Anmeldeinformationen oder Berechtigungen fehl. ... 939
Pipeline-Fehler: Eine Pipeline wurde in einer Pipeline erstellt AWS Region, die einen in einem anderen erstellten Bucket verwendet AWS Region gibt ein "InternalError" mit dem Code "JobFailed“ zurück ... 940
Bereitstellungsfehler: Eine ZIP-Datei, die eine WAR-Datei enthält, wurde erfolgreich bereitgestellt für AWS Elastic Beanstalk, aber die Anwendungs-URL meldet einen 404-Fehler „Nicht gefunden“ ... 938
Pipeline-Artefakt-Ordnernamen werden gekürzt ... 941
Fügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub Enterprise Server oder hinzu GitHub GitLab.com ... 942
Fügen Sie CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen hinzu ... 944
Pipeline-Fehler: Bei einer Bereitstellung mit der CodeDeployToECS Aktion wird die folgende Fehlermeldung zurückgegeben: „Ausnahme beim Versuch, die Artefaktdatei für die Aufgabendefinition zu lesen aus: Name des < Quellartefakts“ > ... 945
GitHub Quellaktion (über die OAuth-App): Die Repository-Liste zeigt verschiedene Repositorys ... 945

GitHub Quellaktion (über GitHub App): Die Verbindung für ein Repository konnte nicht hergestellt werden ... 946
Amazon S3 S3-Fehler: Der < ARN für die CodePipeline Servicerolle > erhält S3-Zugriff für den S3-Bucket verweigert < BucketName > ... 946
Pipelines mit Amazon S3, Amazon ECR oder CodeCommit Quelle werden nicht mehr automatisch gestartet ... 949
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“ ... 950
Pipelines, deren Ausführungsmodus in den QUEUED- oder PARALLEL-Modus geändert wurde, schlagen fehl, wenn das Ausführungslimit erreicht ist ... 951
Pipelines im PARALLEL-Modus weisen eine veraltete Pipeline-Definition auf, wenn sie beim Wechsel in den QUEUED- oder SUPERSEDED-Modus bearbeitet werden ... 951
Bei Pipelines, die vom PARALLEL-Modus aus geändert wurden, wird ein früherer Ausführungsmodus angezeigt ... 952
Pipelines mit Verbindungen, die Triggerfilterung nach Dateipfaden verwenden, beginnen möglicherweise nicht bei der Erstellung von Zweigen ... 952
Pipelines mit Verbindungen, die Triggerfilterung nach Dateipfaden verwenden, werden möglicherweise nicht gestartet, wenn das Dateilimit erreicht ist ... 953
CodeCommit oder S3-Quellversionen im PARALLEL-Modus stimmen möglicherweise nicht mit dem Ereignis überein EventBridge ... 953
Die EC2-Bereitstellungsaktion schlägt fehl und es wird die Fehlermeldung `Keine solche Datei` angezeigt ... 954
Die EKS Deploy-Aktion schlägt fehl und es wird die Fehlermeldung „`Cluster nicht erreichbar`“ angezeigt ... 955
Die Pipeline wird nicht für alle Branches ausgelöst, wenn mehrere Quellaktionen auf dasselbe Repository verweisen ... 956
Benötigen Sie Hilfe für ein anderes Problem? ... 956
Sicherheit ... 957
Datenschutz ... 958
Richtlinie für den Datenverkehr zwischen Netzwerken ... 959
Verschlüsselung im Ruhezustand ... 960
Verschlüsselung während der Übertragung ... 960
Verwaltung von Verschlüsselungsschlüsseln ... 960
Konfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte für CodePipeline ... 960

Verwenden Sie AWS Secrets Manager um Datenbankpasswörter oder API-Schlüssel von Drittanbietern zu verfolgen ... 964
Identity and Access Management ... 965
Zielgruppe ... 965
Authentifizierung mit Identitäten ... 965
Verwalten des Zugriffs mit Richtlinien ... 967
Wie AWS CodePipeline funktioniert mit IAM ... 968
Identity-based Beispiele für politische Maßnahmen ... 974
Resource-based Beispiele für politische Maßnahmen ... 1005
Fehlerbehebung ... 1006
Berechtigungsreferenz ... 1008
Verwalten Sie die CodePipeline Servicerolle ... 1019
Vorfallreaktion ... 1026
Compliance-Validierung ... 1026
Ausfallsicherheit ... 1027
Sicherheit der Infrastruktur ... 1027
Bewährte Methoden für die Gewährleistung der Sicherheit ... 1028
Referenz der Pipeline-Struktur ... 1029
Pipeline-Erklärung ... 1033
Name ... 1035
roleArn ... 1036
ArtifactStore ODER ArtifactStores ... 1036
Stufen ... 1037
version ... 1038
executionMode ... 1038
PipelineType ... 1038
variables ... 1039
triggers ... 1039
Metadaten ... 1043
Deklaration der Stufe ... 1044
Name ... 1047
actions ... 1047
conditions ... 1047
Regeln ... 1048
Aktionsdeklaration ... 1048
Name ... 1052

Region ... 1052
roleArn ... 1053
Namespace ... 1053
Aktion TypeId ... 1053
InputArtifacts ... 1055
Ausgabe-Artefakte ... 1056
Konfiguration (nach Aktionsanbieter) ... 1057
Bestellung ausführen ... 1058
Gültige Aktionsanbieter in CodePipeline ... 1059
Gültige Einstellungen für den PollForSourceChanges Parameter ... 1064
Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp ... 1067
Gültige Konfigurationsparameter für jeden Anbietertyp ... 1069
Referenz der Aktionsstruktur ... 1073
Amazon EC2-Aktionsreferenz ... 1074
Aktionstyp ... 1075
Konfigurationsparameter ... 1075
Input artifacts (Eingabeartefakte) ... 1079
Ausgabeartefakte ... 1080
Richtlinienberechtigungen für die EC2-Bereitstellungsaktion ... 1080
Referenz zur Spezifikationsdatei bereitstellen ... 1083
Aktionsdeklaration ... 1084
Aktionsdeklaration mit Beispiel für eine Deploy-Spezifikation ... 1086
Weitere Informationen finden Sie auch unter ... 1087
Referenz zur Amazon ECR-Quellaktion ... 1087
Aktionstyp ... 1088
Konfigurationsparameter ... 1089
Input artifacts (Eingabeartefakte) ... 1089
Ausgabeartefakte ... 1089
Ausgabevariablen ... 1089
Berechtigungen für Servicerollen: Amazon ECR-Aktion ... 1090
Aktionserklärung (Amazon ECR-Beispiel) ... 1091
Weitere Informationen finden Sie auch unter ... 1092
ECRBuildAndPublishAktionsreferenz erstellen ... 1092
Aktionstyp ... 1093
Konfigurationsparameter ... 1093
Input artifacts (Eingabeartefakte) ... 1094

Ausgabeartefakte 1095
Ausgabevariablen 1095
Berechtigungen für Servicerollen: Aktion ECRBuildAndPublish 1095
Aktionsdeklaration 1097
Weitere Informationen finden Sie auch unter 1098
Aktionsreferenz für Amazon ECS und CodeDeploy blaugrüne Bereitstellungen 1098
Aktionstyp 1099
Konfigurationsparameter 1100
Input artifacts (Eingabeartefakte) 1101
Ausgabeartefakte 1102
Berechtigungen für Servicerollen: CodeDeployToECS Aktion 1102
Aktionsdeklaration 1105
Weitere Informationen finden Sie auch unter 1106
Referenz zur Aktion „Amazon Elastic Container Service“ zur Bereitstellung 1107
Aktionstyp 1108
Konfigurationsparameter 1108
Input artifacts (Eingabeartefakte) 1109
Ausgabeartefakte 1110
Berechtigungen für Servicerollen: Amazon ECS-Standardaktion 1110
Aktionsdeklaration 1112
Weitere Informationen finden Sie auch unter 1113
Referenz zur Bereitstellung von Amazon Elastic Kubernetes Service EKS 1114
Aktionstyp 1115
Konfigurationsparameter 1115
Input artifacts (Eingabeartefakte) 1116
Ausgabeartefakte 1117
Umgebungsvariablen 1117
Ausgabevariablen 1117
Berechtigungen für Richtlinien zur Servicerolle 1117
Aktionsdeklaration 1120
Weitere Informationen finden Sie auch unter 1121
AWS Lambda Aktionsreferenz bereitstellen 1121
Aktionstyp 1122
Konfigurationsparameter 1122
Input artifacts (Eingabeartefakte) 1126
Ausgabeartefakte 1126

Ausgabevariablen 1126
Berechtigungen für die Lambda-Bereitstellungsaktion (Richtlinie) für die Aktion „Lambda-Bereitstellung“ 1126
Aktionsdeklaration 1128
Weitere Informationen finden Sie auch unter 1129
Referenz zur Amazon S3-Bereitstellungsaktion 1129
Aktionstyp 1130
Konfigurationsparameter 1130
Input artifacts (Eingabeartefakte) 1132
Ausgabeartefakte 1132
Berechtigungen für Servicerollen: Aktion „S3-Bereitstellung“ 1132
Beispielaktionskonfiguration 1133
Weitere Informationen finden Sie auch unter 1136
Referenz zur Amazon S3-Quellaktion 1136
Aktionstyp 1137
Konfigurationsparameter 1138
Input artifacts (Eingabeartefakte) 1140
Ausgabeartefakte 1140
Ausgabevariablen 1140
Berechtigungen für Servicerollen: S3-Quellaktion 1141
Aktionsdeklaration 1142
Weitere Informationen finden Sie auch unter 1143
AWS AppConfig Aktionsreferenz bereitstellen 1143
Aktionstyp 1143
Konfigurationsparameter 1144
Input artifacts (Eingabeartefakte) 1144
Ausgabeartefakte 1144
Berechtigungen für Servicerollen: Aktion `AppConfig` 1145
Beispielaktionskonfiguration 1145
Weitere Informationen finden Sie auch unter 1147
CloudFormation Aktionsreferenz bereitstellen 1147
Aktionstyp 1148
Konfigurationsparameter 1148
Input artifacts (Eingabeartefakte) 1153
Ausgabeartefakte 1153
Ausgabevariablen 1154

Berechtigungen für Servicerollen: CloudFormation Aktion ... 1154
Aktionsdeklaration ... 1156
Weitere Informationen finden Sie auch unter ... 1157
CloudFormation StackSets ... 1158
Wie CloudFormation StackSets Aktionen funktionieren ... 1159
Wie strukturiert man StackSets Aktionen in einer Pipeline ... 1161
`CloudFormationStackSet`-Aktion ... 1162
Die Aktion CloudFormationStackInstances ... 1177
Berechtigungen für Servicerollen: Aktion `CloudFormationStackSet` ... 1187
Berechtigungen für Servicerollen: `CloudFormationStackInstances` Aktion ... 1188
Berechtigungsmodelle für Stack-Set-Operationen ... 1188
Datentypen für Vorlagenparameter ... 1189
Weitere Informationen finden Sie auch unter ... 1157
AWS CodeBuild Aktionsreferenz erstellen und testen ... 1191
Aktionstyp ... 1192
Konfigurationsparameter ... 1192
Input artifacts (Eingabeartefakte) ... 1195
Ausgabeartefakte ... 1196
Ausgabevariablen ... 1196
Berechtigungen für Servicerollen: CodeBuild Aktion ... 1197
Deklaration einer Aktion (CodeBuild Beispiel) ... 1198
Weitere Informationen finden Sie auch unter ... 1199
AWS CodePipeline Aktionsreferenz aufrufen ... 1200
Aktionstyp ... 1200
Konfigurationsparameter ... 1201
Input artifacts (Eingabeartefakte) ... 1203
Ausgabeartefakte ... 1204
Berechtigungen für die Aktion zum CodePipeline Aufrufen der Aktion „Dienst“ ... 1204
Aktionsdeklaration ... 1204
Weitere Informationen finden Sie auch unter ... 1205
AWS CodeCommit Referenz zur Quellaktion ... 1205
Aktionstyp ... 1207
Konfigurationsparameter ... 1207
Input artifacts (Eingabeartefakte) ... 1208
Ausgabeartefakte ... 1209
Ausgabevariablen ... 1209

Berechtigungen für Servicerollen: CodeCommit Aktion 1210
Beispielaktionskonfiguration 1211
Weitere Informationen finden Sie auch unter 1213
AWS CodeDeploy Aktionsreferenz bereitstellen 1213
Aktionstyp 1214
Konfigurationsparameter 1214
Input artifacts (Eingabeartefakte) 1214
Ausgabeartefakte 1215
Berechtigungen für Servicerollen: AWS CodeDeploy Aktion 1215
Aktionsdeklaration 1216
Weitere Informationen finden Sie auch unter 1217
CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com selbstverwaltete Aktionen GitLab 1218
Aktionstyp 1222
Konfigurationsparameter 1222
Input artifacts (Eingabeartefakte) 1224
Ausgabeartefakte 1224
Ausgabevariablen 1224
Berechtigungen für Servicerollen: AWS CodeConnections action 1225
Aktionsdeklaration 1226
Installation der Installations-App und Herstellen einer Verbindung 1227
Weitere Informationen finden Sie auch unter 1228
Aktionsreferenz für Befehle 1229
Überlegungen zur Aktion „Befehle“ 1230
Dienstrolle, Richtlinien, Berechtigungen 1231
Aktionstyp 1232
Konfigurationsparameter 1232
Input artifacts (Eingabeartefakte) 1237
Ausgabeartefakte 1237
Umgebungsvariablen 1237
Berechtigungen für Servicerollen: Befehle, Aktion 1239
Aktionserklärung (Beispiel) 1239
Weitere Informationen finden Sie auch unter 1241
AWS Device Farm Referenz zur Testaktion 1241
Aktionstyp 1242
Konfigurationsparameter 1242

Input artifacts (Eingabeartefakte) 1246
Ausgabeartefakte 1246
Berechtigungen für Servicerollen: AWS Device Farm action 1246
Aktionsdeklaration 1247
Weitere Informationen finden Sie auch unter 1248
Referenz zur Aktion „Elastic Beanstalk-Bereitstellung“ 1249
Aktionstyp 1249
Konfigurationsparameter 1249
Input artifacts (Eingabeartefakte) 1250
Ausgabeartefakte 1250
Berechtigungen für Servicerollen: Aktion `ElasticBeanstalk` bereitstellen 1250
Aktionsdeklaration 1251
Weitere Informationen finden Sie auch unter 1252
Referenz zum `InspectorScan` Aufrufen der Aktion durch Amazon Inspector 1253
ID des Aktionstyps 1253
Konfigurationsparameter 1254
Input artifacts (Eingabeartefakte) 1257
Ausgabeartefakte 1257
Ausgabevariablen 1257
Berechtigungen für Servicerollen: `InspectorScan` Aktion 1257
Aktionsdeklaration 1258
Weitere Informationen finden Sie auch unter 1259
AWS Lambda Aktionsreferenz aufrufen 1260
Aktionstyp 1260
Konfigurationsparameter 1260
Input artifacts (Eingabeartefakte) 1261
Ausgabeartefakte 1261
Ausgabevariablen 1261
Beispielaktionskonfiguration 1261
JSON-Beispielereignis 1262
Weitere Informationen finden Sie auch unter 1265
AWS OpsWorks Aktionsreferenz bereitstellen 1265
Aktionstyp 1265
Konfigurationsparameter 1266
Input artifacts (Eingabeartefakte) 1266
Ausgabeartefakte 1266

Berechtigungen für Servicerollen: AWS OpsWorks action ... 1266
Beispielaktionskonfiguration ... 1267
Weitere Informationen finden Sie auch unter ... 1268
AWS Service Catalog Aktionsreferenz bereitstellen ... 1268
Aktionstyp ... 1268
Konfigurationsparameter ... 1268
Input artifacts (Eingabeartefakte) ... 1269
Ausgabeartefakte ... 1269
Berechtigungen für Servicerollen: Aktion „Servicekatalog“ ... 1269
Beispiele für Aktionskonfigurationen nach Typ der Konfigurationsdatei ... 1270
Weitere Informationen finden Sie auch unter ... 1271
AWS Step Functions ... 1272
Aktionstyp ... 1265
Konfigurationsparameter ... 1272
Input artifacts (Eingabeartefakte) ... 1274
Ausgabeartefakte ... 1274
Ausgabevariablen ... 1274
Berechtigungen für Servicerollen: `StepFunctions` Aktion ... 1275
Beispielaktionskonfiguration ... 1275
Behavior ... 1278
Weitere Informationen finden Sie auch unter ... 1147
Referenz zur Regelstruktur ... 1281
CloudWatchAlarm ... 1281
Art der Regel ... 1282
Konfigurationsparameter ... 1282
Beispiel für eine Regelkonfiguration ... 1283
Weitere Informationen finden Sie auch unter ... 1284
CodeBuild Regel ... 1284
Richtlinienberechtigungen für Servicerollen ... 1284
Regeltyp ... 1285
Konfigurationsparameter ... 1285
Beispiel für eine Regelkonfiguration ... 1287
Weitere Informationen finden Sie auch unter ... 1288
Befehle ... 1288
Überlegungen zur Befehlsregel ... 1289
Richtlinienberechtigungen für die Servicerolle ... 1284

Art der Regel ... 1285
Konfigurationsparameter ... 1285
Beispiel für eine Regelkonfiguration ... 1287
Weitere Informationen finden Sie auch unter ... 1288
DeploymentWindow ... 1294
Typ der Regel ... 1294
Konfigurationsparameter ... 1294
Beispiel für eine Regelkonfiguration ... 1297
Weitere Informationen finden Sie auch unter ... 1297
LambdaInvoke ... 1298
Regeltyp ... 1282
Konfigurationsparameter ... 1298
Beispiel für eine Regelkonfiguration ... 1299
Weitere Informationen finden Sie auch unter ... 1300
VariableCheck ... 1300
Art der Regel ... 1300
Konfigurationsparameter ... 1300
Beispiel für eine Regelkonfiguration ... 1303
Weitere Informationen finden Sie auch unter ... 1304
Referenz zum Integrationsmodell ... 1305
Wie funktionieren Aktionstypen von Drittanbietern mit dem Integrator ... 1305
Konzepte ... 1306
Unterstützte Integrationsmodelle ... 1308
Lambda-Integrationsmodell ... 1310
Aktualisieren Sie Ihre Lambda-Funktion, um die Eingabe von zu verarbeiten CodePipeline 1310
Geben Sie die Ergebnisse Ihrer Lambda-Funktion zurück an CodePipeline ... 1314
Verwenden Sie Fortsetzungstoken, um auf Ergebnisse eines asynchronen Prozesses zu warten ... 1316
Stellen Sie CodePipeline die Berechtigungen bereit, um die Lambda-Funktion des Integrators zur Laufzeit aufzurufen ... 1317
Integrationsmodell für Berufstätige ... 1317
Wählen und Konfigurieren einer Strategie für die Berechtigungsverwaltung Ihres Auftragsworkers ... 1318
Referenz zu Abbild-Definitionsdateien ... 1321
imagedefinitions.json-Datei für Amazon ECS-Standardbereitstellungsaktionen ... 1321
Detail.json Image-Datei für Amazon blue/green ECS-Bereitstellungsaktionen ... 1324

Referenz zu Variablen 1329
Konzepte 1330
Variablen 1330
Namespaces 1331
Anwendungsfälle für Variablen 1332
Konfigurieren von Variablen 1333
Variablen auf Pipeline-Ebene konfigurieren 1333
Variablen auf Aktionsebene konfigurieren 1334
Variablenauflösung 1336
Regeln für Variablen 1337
Für Pipeline-Aktionen verfügbare Variablen 1338
Aktionen mit definierten Variablenschlüsseln 1338
Aktionen mit benutzerkonfigurierten Variablenschlüsseln 1344
Arbeiten mit Glob-Mustern in der Syntax 1347
Aktivieren von Pipelines für die empfohlenen Methode zur Änderungserkennung 1349
Aktualisieren Sie eine Quellaktion GitHub (über die OAuth-App) auf eine Quellaktion GitHub (über GitHub die App) 1350
Schritt 1: Ersetzen Sie Ihre Aktion (über die OAuth-App) GitHub 1352
Schritt 2: Erstellen Sie eine Verbindung zu GitHub 1352
Schritt 3: Speichern Sie Ihre GitHub Quellaktion 1353
Kontingente 1355
Anhang A: Quellaktionen GitHub (über die OAuth-App) 1372
Hinzufügen einer GitHub Quellaktion (über die OAuth-App) 1373
GitHub Referenz zur Quellaktion (über die OAuth-App) 1374
Aktionstyp 1375
Konfigurationsparameter 1375
Input artifacts (Eingabeartefakte) 1377
Ausgabeartefakte 1377
Ausgabevariablen 1378
Aktionserklärung (GitHub Beispiel) 1379
Verbindung zu GitHub (OAuth) herstellen 1380
Weitere Informationen finden Sie auch unter 1380
Dokumentverlauf 1382
Frühere Aktualisierungen 1420
CodePipeline Feature-Referenz 1433
..... mcdxxxvii

Was ist AWS CodePipeline?

AWS CodePipeline ist ein Service zur kontinuierlichen Bereitstellung, 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 zur kontinuierlichen Veröffentlichung Ihrer Softwareänderungen erforderlich sind. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

Themen

- Kontinuierliche Bereitstellung und kontinuierliche Integration
- Was kann ich damit machen? CodePipeline
- Ein kurzer Blick auf CodePipeline
- Wie fange ich an mit CodePipeline?
- CodePipeline Konzepte
- DevOps Beispiel für eine Pipeline
- So funktionieren Pipeline-Ausführungen
- Eingabe- und Ausgabe-Artefakte
- Wie funktionieren die Bühnenbedingungen?
- Pipeline-Typen
- Welcher Pipeline-Typ ist der richtige für mich?

Kontinuierliche Bereitstellung und kontinuierliche Integration

CodePipeline ist ein Continuous Delivery Service, 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 Standort, 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 Praktizieren von Continuous Integration und 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

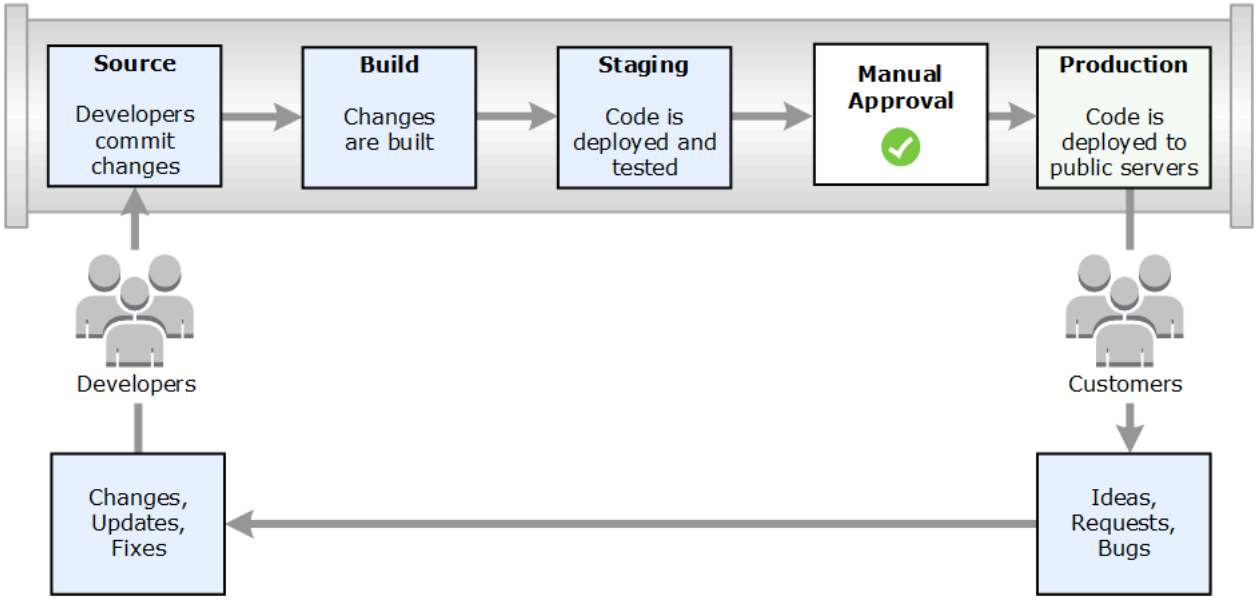
Sie können CodePipeline es verwenden, um Ihre Anwendungen automatisch in der Cloud zu erstellen, zu testen und bereitzustellen. Insbesondere können Sie Folgendes:

- Automatisieren Sie Ihre Release-Prozesse: Automatisiert Ihren Release-Prozess von Anfang bis Ende CodePipeline vollständig, angefangen bei Ihrem Quell-Repository bis hin zu Build, Test und Deployment. 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 gemäß Ihren Kriterien durch.
- 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 von AWS-Services Drittanbietern, die derzeit von unterstützt werden CodePipeline, finden Sie unterProdukt- und Serviceintegrationen mit CodePipeline.

- Fortschritt auf einen Blick: Sie können den Status Ihrer Pipelines in Echtzeit überprüfen, die Details aller Warnmeldungen überprüfen, fehlgeschlagene Phasen oder Aktionen wiederholen, Details zu den Quellrevisionen 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 Veröffentlichungsprozess unter Verwendung von CodePipeline.



In diesem Beispiel entdeckt CodePipeline automatisch die Änderungen, wenn Developer ein Commit für Änderungen für ein Quell-Repository durchführen. 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 Lasttests. Nach erfolgreichem Abschluss dieser Tests und nach Genehmigung einer manuellen Genehmigungsaktion, die der Pipeline hinzugefügt wurde, wird der getestete und genehmigte Code für 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 Dienste bereitstellen. Entwickler können die mitgelieferten

Integrationspunkte auch verwenden, CodePipeline um andere Tools oder Dienste einzubinden, 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 zu beginnen 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 den Schritten in den CodePipeline Tutorials Tutorials folgen.
4. Verwenden Sie es CodePipeline für Ihre neuen oder vorhandenen Projekte, indem Sie den Schritten unter folgenEine Pipeline, Phasen und Aktionen erstellen.

CodePipeline Konzepte

Die Modellierung und Konfiguration Ihres automatisierten Veröffentlichungsprozesses ist einfacher, wenn Sie die in verwendeten Konzepte und Begriffe verstehen AWS CodePipeline. Im Folgenden finden Sie einige Konzepte, die Sie kennen sollten, wenn Sie CodePipeline verwenden.

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

Die folgenden Begriffe werden verwendet in CodePipeline:

Themen

- Pipelines
- Pipeline-Ausführungen
- Operationen auf der Bühne
- Aktionsausführungen
- Arten der Ausführung
- Aktionstypen

- -Artefakte
- Quell-Revisionen
- Auslöser
- Variablen
- Bedingungen
- Regeln

Pipelines

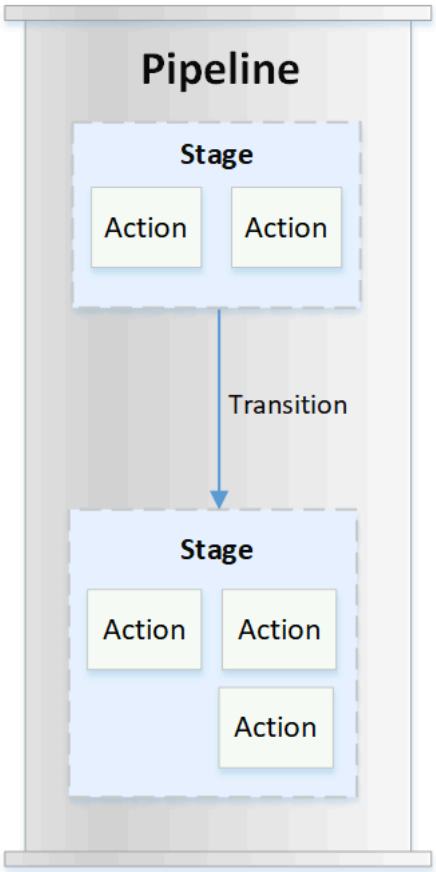
Eine Pipeline ist ein Workflow-Konstrukt, das den Veröffentlichungsprozess von Software-Änderungen beschreibt. Jede Pipeline besteht aus einer Reihe von Phasen.

Stufen

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. Beispielsweise könnte eine Bereitstellungsphase eine Bereitstellungsaktion enthalten, die Code für einen Rechendienst wie Amazon EC2 oder bereitstellt. AWS Lambda

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

Aktionen können seriell oder parallel ausgeführt werden. Informationen zu seriellen und parallelen Aktionen in einer Phase finden Sie in den `runOrder` Informationen unter Anforderungen an die 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. Die Ausführung, die darauf wartet, die besetzte Phase zu betreten, ist eine eingehende Ausführung. Eine eingehende Ausführung kann immer noch fehlschlagen, ersetzt werden oder manuell gestoppt werden. Weitere Hinweise zur Funktionsweise eingehender Ausführungen finden Sie unter. So funktionieren eingehende Hinrichtungen

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 wechselt in die Phase `FAILED` und CodePipeline entsperrt sie. 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 frühere 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 Modus SUPERSEDED eine Ausführung darauf wartet, in eine gesperrte Phase einzutreten, wird sie möglicherweise durch eine neuere Ausführung eingeholt und ersetzt. 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 SUPERSEDED-Modus verarbeitet.

Operationen auf der Bühne

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

Gültige Status für Stufen sind `InProgress`, `Stopping`, `Stopped`, `Succeeded`, und`Failed`. Sie können eine fehlgeschlagene Phase erneut versuchen, es sei denn, die fehlgeschlagene Phase kann nicht erneut versucht werden. Weitere Informationen finden Sie unter StageExecution. Sie können eine Phase auf eine angegebene vorherige erfolgreiche Ausführung zurücksetzen. Eine Phase kann so konfiguriert werden, dass sie bei einem Fehler automatisch rückgängig gemacht wird, wie unter beschriebenKonfiguration des Stage-Rollbacks. 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`, `Abandoned``Succeeded`, oder`Failed`. Weitere Informationen finden Sie unter ActionExecution.

Arten der Ausführung

Bei einer Pipeline- oder Stufenausführung kann es sich entweder um eine Standard- oder eine Rollback-Ausführung handeln.

Bei Standardtypen hat die Ausführung eine eindeutige ID und es handelt sich um einen vollständigen Pipeline-Lauf. Ein Pipeline-Rollback besteht aus einer Phase, die zurückgesetzt werden muss, und einer erfolgreichen Ausführung für die Phase als Zielausführung, zu der das Rollback durchgeführt werden soll. Die Ausführung der Zielpipeline wird verwendet, um Quellrevisionen und Variablen für die erneute Ausführung der Stufe abzurufen.

Aktionstypen

Aktionstypen sind vorkonfigurierte Aktionen, die unter ausgewählt werden können. CodePipeline Der Aktionstyp wird durch seinen Besitzer, Anbieter, Version und Kategorie definiert. Der Aktionstyp bietet benutzerdefinierte Parameter, 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 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 zur weiteren Verarbeitung mithilfe des Pipeline-Artefakt-Buckets an eine andere Aktion 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 Quellrevisionen verarbeitet. Für Repositorys GitHub und CodeCommit Repositorys ist dies der Commit. Bei S3-Buckets oder -Aktionen ist dies die Objektversion.

Sie können eine Pipeline-Ausführung mit einer von Ihnen angegebenen Quellrevision, z. B. einem Commit, starten. Die Ausführung verarbeitet die angegebene Revision und überschreibt die Revision, 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 Anbieter von Quellaktionen in einer Pipeline verfügbar. Bestimmte Trigger hängen vom Quellanbieter für eine Pipeline ab. CloudWatch Ereignisse müssen beispielsweise mit Eventressourcen von Amazon konfiguriert werden, für CloudWatch die 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 eine Art von Trigger, der für Repository-Ereignisse von Drittanbietern konfiguriert ist. WebhookV2 ist beispielsweise ein Triggertyp, mit dem Git-Tags verwendet werden können, um Pipelines mit externen Quellanbietern wie GitHub Enterprise Server GitHub.com 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 dich durch die Verwendung von Git-Tags als Trigger für deine Pipeline führt, findest du unterTutorial: Verwende Git-Tags, um deine Pipeline zu starten.

> **Important**
>
> Bei Pipelines, die länger als 30 Tage inaktiv sind, ist das Polling für die Pipeline deaktiviert. Weitere Informationen finden Sie unter Polling DisabledAt in der Pipeline-Strukturreferenz. Die Schritte zur Migration Ihrer Pipeline vom Polling zur ereignisbasierten Änderungserkennung finden Sie unter Methoden zur Änderungserkennung.

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. Für Variablen, die von einer Aktion ausgegeben werden, ist der Wert verfügbar, nachdem eine Aktion erfolgreich abgeschlossen wurde. Weitere Informationen finden Sie unter Variablenreferenz.

Bedingungen

Eine Bedingung enthält eine Reihe von Regeln, die ausgewertet werden. Wenn alle Regeln in einer Bedingung erfolgreich sind, ist die Bedingung erfüllt. Sie können Bedingungen so konfigurieren, dass, wenn die Kriterien nicht erfüllt sind, das angegebene Ergebnis, z. B. das Nichtbestehen der Stufe, ausgelöst wird. Bedingungen werden auch als Tore bezeichnet, da Sie damit angeben können, wann eine Ausführung eine Phase betritt und durchläuft oder die Phase verlässt, nachdem sie sie durchlaufen hat. Dies ist vergleichbar damit, dass sich eine Verkehrslinie auf einer Fahrbahn an einem geschlossenen Tor versammeln kann und dann die Öffnung des Tores spezifiziert wird, um den Verkehrsfluss in ein Gebiet zu ermöglichen. Zu den Ergebnissen für Bedingungstypen gehören das Nichtbestehen der Stufe oder das Zurücksetzen der Stufe. Mithilfe von Bedingungen können Sie angeben, wann diese Aktionen in einer Pipeline-Phase stattfinden. Sie können Bedingungen zur Laufzeit überschreiben.

Es gibt drei Arten von Bedingungen. Die Teilnahmebedingungen beantworten die Frage „Wenn die Regeln für die Bedingung erfüllt sind, dann betrete die Stufe.“ Die Phase ist gesperrt, wenn die Ausführung in die Phase eintritt, und dann werden die Regeln ausgeführt. Bei den Bedingungen „Bei Ausfall“ wird die Regel aktiviert, wenn eine Phase fehlgeschlagen ist, was dazu führt, dass eine

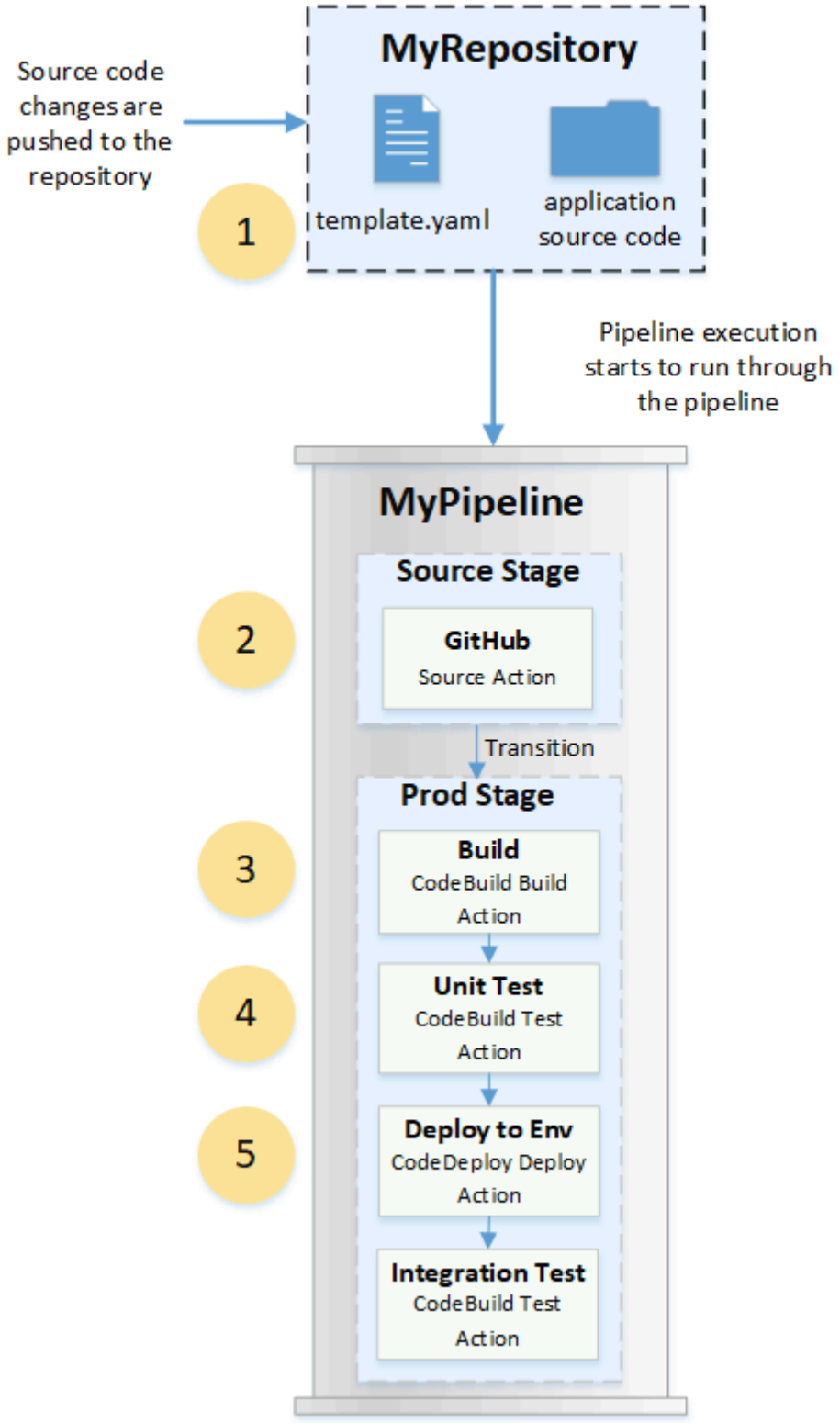
fehlgeschlagene Phase rückgängig gemacht wird. Bei den Bedingungen „Bei Erfolg“ wird die Regel aktiviert, wenn eine Phase erfolgreich ist, z. B. wenn bei einer erfolgreichen Ausführung überprüft wird, ob Alarme vorliegen, bevor der Vorgang fortgesetzt wird. Beispielsweise würde die Bedingung „Bei Erfolg“ zu einem Rollback einer erfolgreichen Phase führen, wenn die CloudWatchAlarm Regel feststellt, dass in der Bereitstellungsumgebung Alarme vorliegen. Weitere Informationen finden Sie unter Wie funktionieren die Bühnenbedingungen?.

Regeln

Bedingungen verwenden eine oder mehrere vorkonfigurierte Regeln, die ausgeführt und Prüfungen durchgeführt werden. Das konfigurierte Ergebnis wird dann verwendet, wenn die Bedingung nicht erfüllt ist. Wenn Sie beispielsweise alle Regeln für eine Regel für eine Eingangsbedingung erfüllen, die den Alarmstatus und die Zeitfenster für die Bereitstellung überprüft, wird eine erfolgreiche Phase ausgelöst, nachdem alle Prüfungen bestanden wurden. Weitere Informationen finden Sie unter Wie funktionieren die Bühnenbedingungen?.

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 eingerichtet und den Quellcode für eine Webanwendung in einem GitHub Repository namens. 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 an das Repository übertragen, CodePipeline erkennt die übertragene Änderung und eine Pipeline-Ausführung beginnt in der Source Stage.
2. Die GitHub Quellaktion wurde erfolgreich abgeschlossen (das heißt, die neuesten Änderungen wurden heruntergeladen und in dem Artefakt-Bucket gespeichert, der für diese Ausführung spezifisch ist). Die von der GitHub Quellaktion erzeugten Ausgabe-Artefakte, bei denen es sich um die Anwendungsdateien aus dem Repository handelt, werden dann als Eingabe-Artefakte verwendet, an denen die Aktionen in der nächsten Phase arbeiten.
3. Die Pipeline-Ausführung wechselt von der Quellphase zur Produktionsphase. Mit der ersten Aktion in der Prod-Phase wird ein Build-Projekt ausgeführt, 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 Prod Stage ist ein Komponententestprojekt, das in der CodeBuild Pipeline erstellt und als Testaktion konfiguriert wurde.
5. Der komponentengetestete Code wird als Nächstes von einer Bereitstellungsaktion in der Produktionsphase bearbeitet, die die Anwendung zu einer Produktionsumgebung bereitstellt. Nach erfolgreichem Abschluss der Bereitstellungsaktion 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 wird. CodePipelineverfolgt jede Pipeline-Ausführung, die beginnt, wenn eine Pipeline manuell gestartet oder eine Änderung am Quellcode vorgenommen wird. CodePipeline verwendet die folgenden Ausführungsmodi, um zu steuern, wie jede Ausführung in der Pipeline abläuft. Weitere Informationen finden Sie unter Den Pipeline-Ausführungsmodus festlegen oder ändern.

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

> **Important**
>
> Für Pipelines im PARALLELMODUS ist ein Stufen-Rollback nicht verfügbar. Ebenso können einer Pipeline im PARALLEL-Modus keine Fehlerbedingungen mit einem Rollback-Ergebnistyp hinzugefügt werden.

So werden Pipeline-Ausführungen gestartet

Sie können eine Ausführung starten, wenn Sie Ihren Quellcode ändern oder die Pipeline manuell starten. Sie können eine Ausführung auch über eine Amazon CloudWatch Events-Regel auslösen, die Sie planen. 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 Quellrevisionen bei Pipeline-Ausführungen verarbeitet werden

Für jede Pipeline-Ausführung, die mit Quellcodeänderungen (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 übertragen, wodurch die Pipeline für Ausführung 1 gestartet wird. In dem Moment, in dem ein zweiter Commit übertragen wird, wird dadurch die Pipeline für Ausführung 2 gestartet.

> **Note**
>
> Bei Pipelines im PARALLEL-Modus mit einer CodeCommit Quelle klont die Quellaktion den HEAD immer zu dem Zeitpunkt, zu dem er gestartet wird, unabhängig vom Commit, der die Pipeline-Ausführung ausgelöst hat. 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 dadurch 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.

So funktionieren Quellüberschreibungen mit dem EventBridge Eingangstransformator

Sie können Overrides verwenden, um eine Pipeline mit einer bestimmten Quell-Revisions-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 der Quellrevision für`revisionType`:

- `COMMIT_ID`

- `IMAGE_DIGEST`
- `S3_OBJECT_VERSION_ID`
- `S3_OBJECT_KEY`

> **Note**
>
> Für die `IMAGE_DIGEST` Typen der Quellversionen `COMMIT_ID` und gilt die Quellrevisions-ID für den gesamten Inhalt des Repositorys in allen Zweigen.

> **Note**
>
> Bei den Quellrevisionen `S3_OBJECT_VERSION_ID` und `S3_OBJECT_KEY` kann jeder der Typen unabhängig voneinander verwendet werden, oder sie können zusammen verwendet werden, um die Quelle mit einer bestimmten ObjectKey VersionID zu überschreiben. Für `S3_OBJECT_KEY` `AllowOverrideForS3ObjectKey` muss der Konfigurationsparameter auf gesetzt werden. `true` Weitere Informationen zu S3-Quellkonfigurationsparametern finden Sie unterKonfigurationsparameter.

Sie können Quellüberschreibungen mithilfe des Eingangstransformators in EventBridge angeben. Verwenden Sie den Eingangstransformator, um die Daten wie folgt zu übergeben:

- Sie können den Eingangstransformator verwenden, um die Daten als JSON-Parameter zu übergeben.
- Sie können den Eingangstransformator verwenden, um Pipeline-Variablen zu übergeben.

Beispiele für die Übergabe der Daten als JSON-Parameter finden Sie unter Amazon ECR-Quellaktionen und Ressourcen EventBridge Verbindung zu Amazon S3-Quellaktionen herstellen, die EventBridge und verwenden AWS CloudTrail für S3 und CodeCommit Quellaktionen und EventBridge für CodeCommit.

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

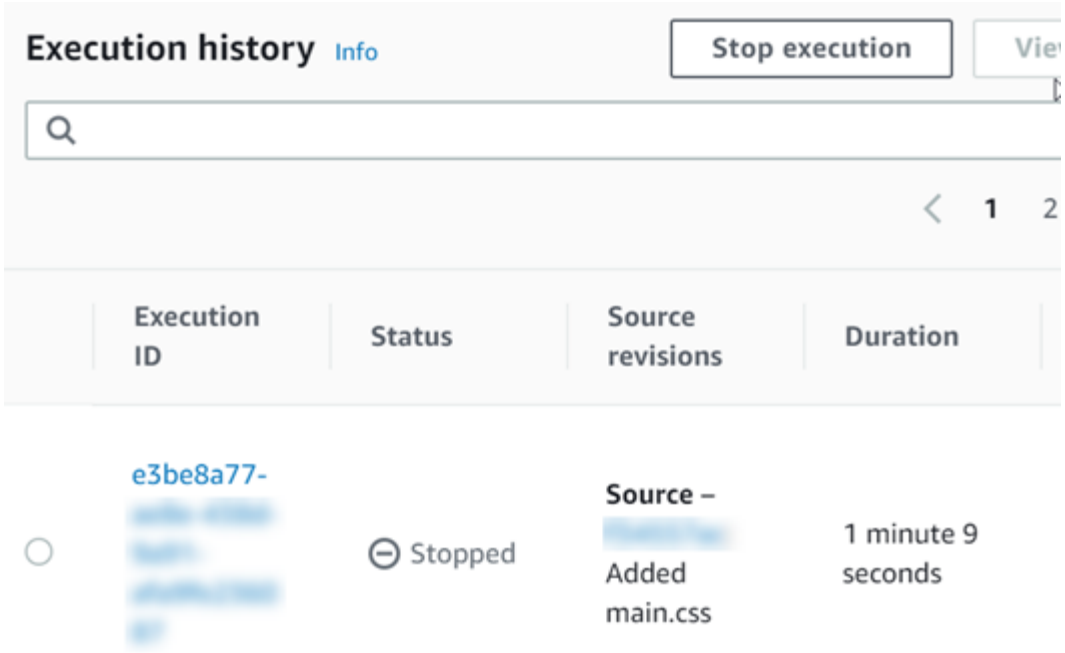
Wenn Sie sich für das Anhalten und Warten entscheiden, wird die ausgewählte Ausführung fortgesetzt, bis die laufenden Aktionen abgeschlossen sind. Beispielsweise wurde die folgende Pipeline-Ausführung angehalten, während die Build-Aktion lief.

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.

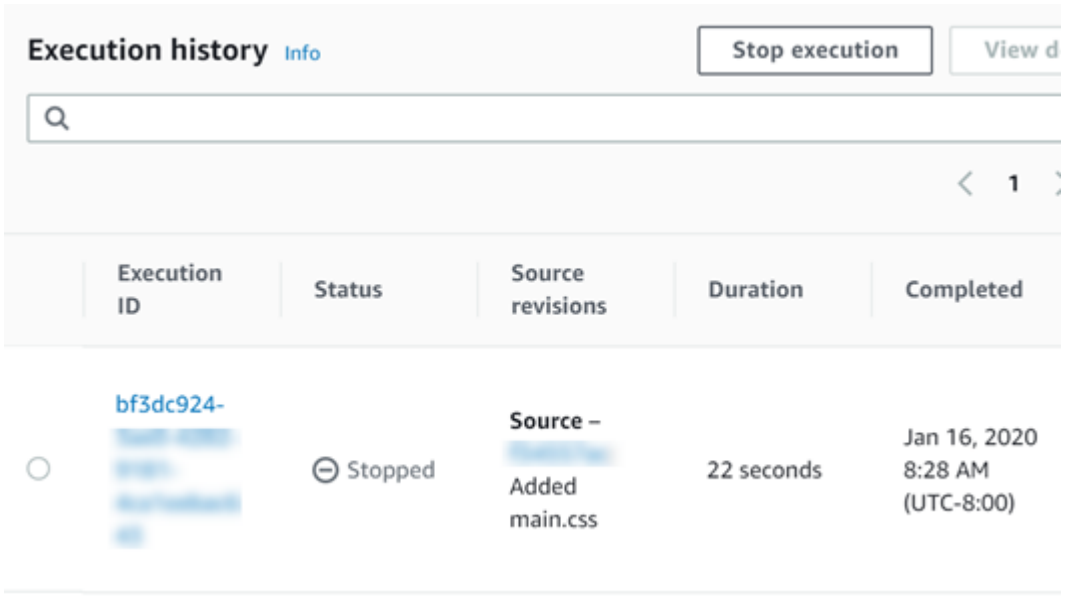
In der Verlaufsansicht wird der Ausführungsstatus Stopped (Beendet) angezeigt, nachdem die laufende Aktion abgeschlossen ist.



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 fehlgeschlagene oder nicht aufeinanderfolgende Aufgaben in Ihrer Pipeline vermeidet. Wenn eine Aktion abgebrochen wird CodePipeline, setzt der Aktionsanbieter alle Aufgaben im Zusammenhang mit der Aktion fort. Im Falle einer CloudFormation Aktion wird die Bereitstellungsaktion in der Pipeline abgebrochen, aber das Stack-Update kann fortgesetzt werden und zu einem fehlgeschlagenen Update führen.

Ein Beispiel für abgebrochene Aktionen, die dazu führen können, dass die Reihenfolge der Aufgaben nicht eingehalten wird: Wenn Sie eine große Datei (1 GB) über eine S3-Bereitstellungsaktion bereitstellen und Sie sich dafür entscheiden, die Aktion zu stoppen und abzubrechen, während die Bereitstellung bereits im Gange ist, wird die Aktion in Amazon S3 abgebrochen CodePipeline, aber in Amazon S3 fortgesetzt. Amazon S3 erhält keine Anweisung zum Abbrechen des Uploads. 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.

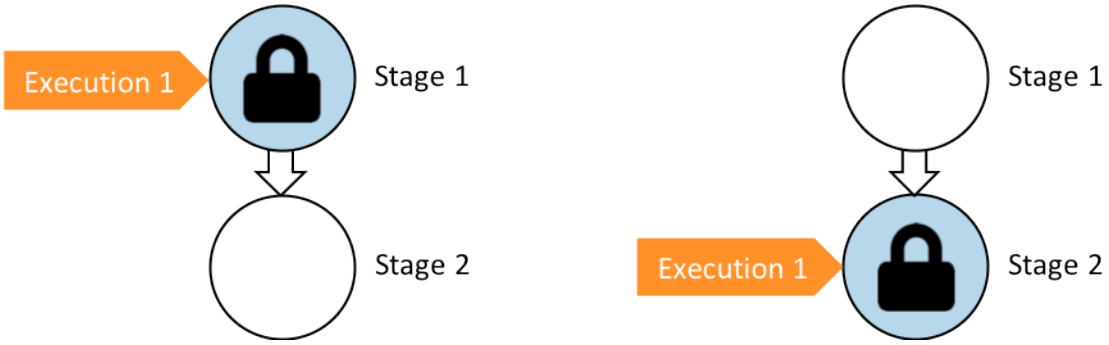
Sie können die Option zum Anhalten und Beenden verwenden, wenn Sie eine benutzerdefinierte Aktion haben. Sie können beispielsweise eine benutzerdefinierte Aktion mit Arbeiten abbrechen, die nicht abgeschlossen werden müssen, bevor Sie eine neue Ausführung zur Behebung eines Fehlers starten.

Wie werden Ausführungen im SUPERSEDED-Modus verarbeitet

Der Standardmodus für die Verarbeitung von Ausführungen ist der SUPERSEDED-Modus. 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.

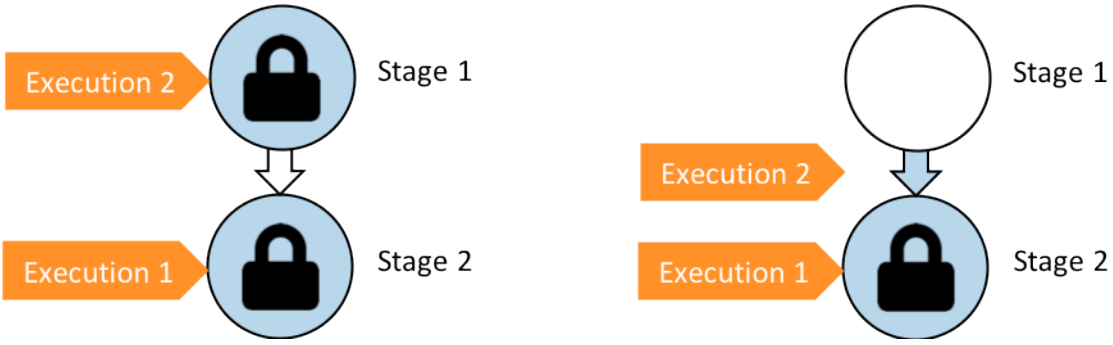


Vorher: Stage 1 is locked as Execution 1 enters. Nachher: Stage 2 is locked as Execution 1 enters.

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.

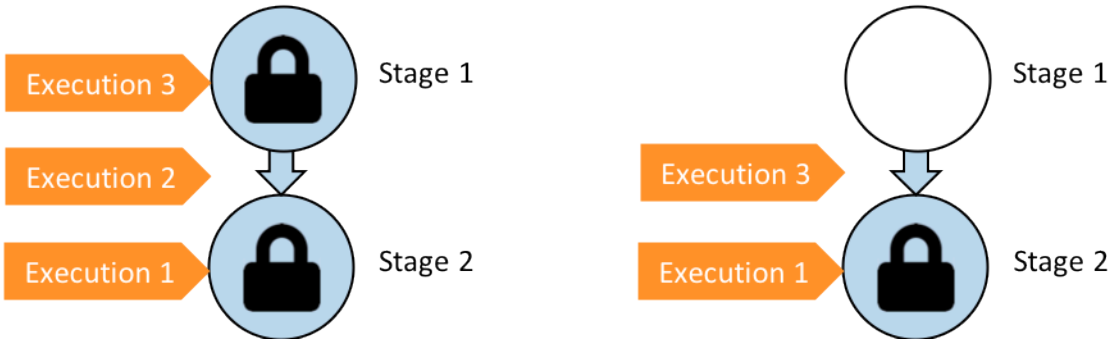
> **Note**
>
> Bevor Sie eine Ausführung beenden, empfehlen wir Ihnen, den Übergang vor der Phase zu deaktivieren. Auf diese Weise wird die Phase aufgrund der angehaltenen Ausführung entsperrt und akzeptiert keine nachfolgende Pipeline-Ausführung.



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.

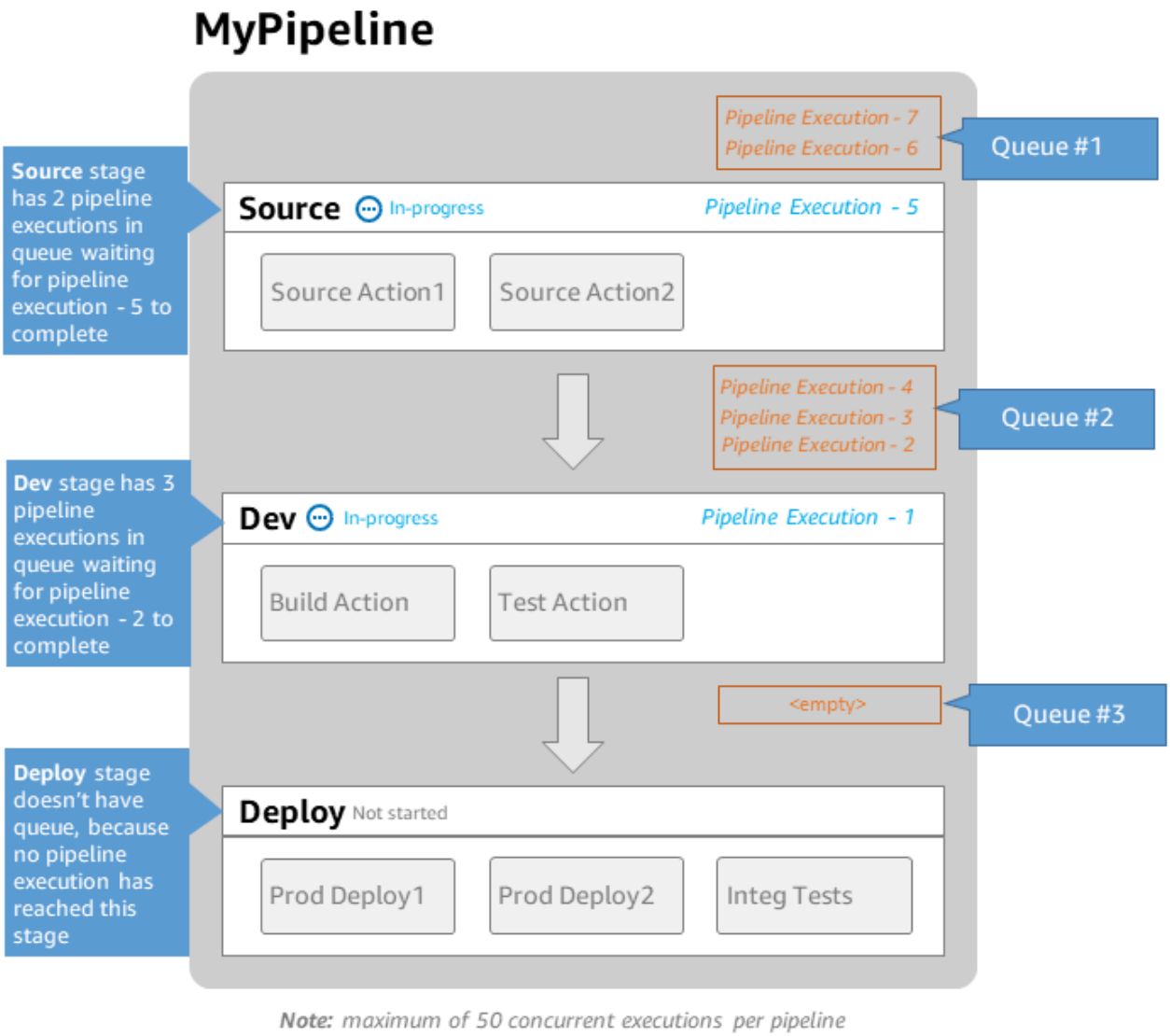
Weitere Informationen zu Überlegungen beim Anzeigen und Umschalten zwischen Ausführungsmodi finden Sie unter. Den Pipeline-Ausführungsmodus festlegen oder ändern Weitere Hinweise zu Kontingenten mit Ausführungsmodi finden Sie unterKontingente in AWS CodePipeline.

Wie werden Ausführungen im QUEUED-Modus verarbeitet

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

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, wird die Phase gesperrt, sodass keine anderen Ausführungen eintreten können. Dieses Verhalten bleibt dasselbe wie im SUPERSEDED-Modus. Wenn die Ausführung die Phase beendet hat, wird die Phase entsperrt und ist bereit für die nächste Ausführung.

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



Weitere Informationen zu Überlegungen zum Anzeigen und Umschalten zwischen Ausführungsmodi finden Sie unter. Den Pipeline-Ausführungsmodus festlegen oder ändern Weitere Hinweise zu Kontingenten mit Ausführungsmodi finden Sie unterKontingente in AWS CodePipeline.

Wie werden Ausführungen im PARALLEL-MODUS verarbeitet

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

Verwenden Sie den PARALLEL-Modus in Entwicklungsumgebungen, in denen jede Funktion ihren eigenen Feature-Branch hat und für Ziele bereitgestellt wird, die nicht von anderen Benutzern gemeinsam genutzt werden.

Weitere Informationen zu Überlegungen zum Anzeigen und Umschalten zwischen Ausführungsmodi finden Sie unterDen Pipeline-Ausführungsmodus festlegen oder ändern. Weitere Hinweise zu Kontingenten mit Ausführungsmodi finden Sie unterKontingente in AWS CodePipeline.

Verwaltung des Pipeline-Ablaufs

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 eintreten. 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, wechselt eine eingehende Ausführung in die Phase 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 Genehmigung erteilt wird (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.

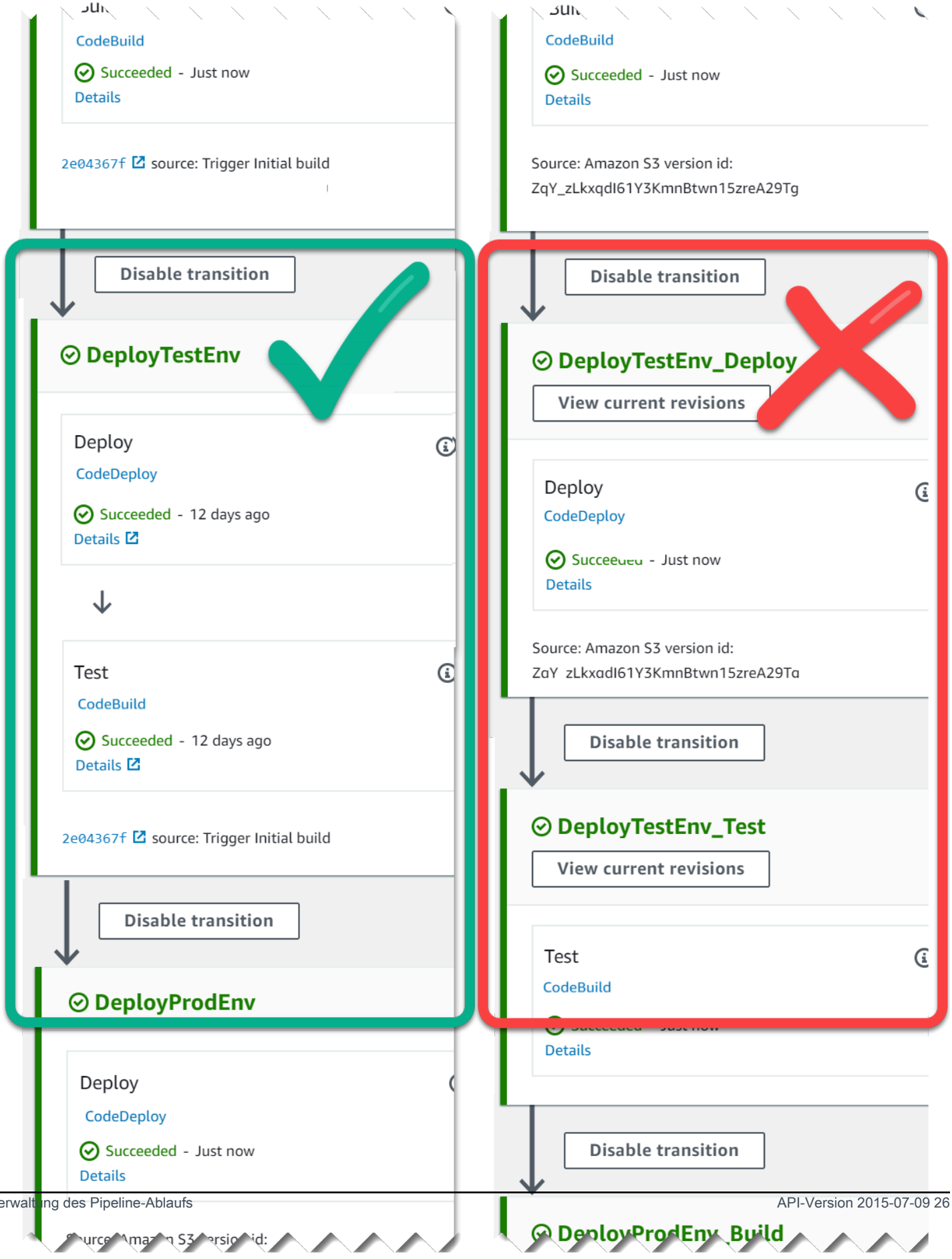
> **Note**
>
> Eine Phase mit einer Genehmigungsaktion ist gesperrt, bis die Genehmigungsaktion genehmigt oder abgelehnt wurde oder eine Zeitüberschreitung eintritt. Eine abgelaufene Genehmigungsaktion wird auf die gleiche Weise wie eine fehlgeschlagene Aktion verarbeitet.

- 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. Dadurch wird die Ausführung wieder aufgenommen (fehlgeschlagene Aktionen werden erneut versucht und, falls sie erfolgreich sind, in der fortgesetzt). stage/pipeline
  - 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 Stufe 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 Hinrichtungen

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

- Eine andere Ausführung hat bereits die nächste Phase erreicht und sie gesperrt.
- Der Übergang zur nächsten Stufe ist deaktiviert.

Sie können einen Übergang deaktivieren, um eine eingehende Ausführung abzuhalten, wenn Sie kontrollieren möchten, ob eine aktuelle Ausführung in nachfolgenden Phasen abgeschlossen werden kann, 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 müssen die folgenden Überlegungen beachtet werden:

- Sobald die Phase „Aktion“, „Transition“ oder „Gesperrt“ verfügbar wird, tritt die laufende eingehende Ausführung in die Phase ein und durchläuft die Pipeline weiter.
- Während die eingehende Ausführung wartet, kann sie manuell gestoppt werden. Eine eingehende Ausführung kann den Status`InProgress`,`Stopped`, oder `Failed` haben.
- Wenn eine eingehende Ausführung gestoppt wurde oder fehlgeschlagen ist, kann sie nicht wiederholt werden, da es keine fehlgeschlagenen Aktionen gibt, die erneut versucht 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 stoppen.

Eingabe- und Ausgabe-Artefakte

CodePipeline lässt sich in Entwicklungstools integrieren, um nach Codeänderungen zu suchen und dann die Entwicklung und Bereitstellung in allen Phasen des Continuous-Delivery-Prozesses durchzuführen. Artefakte sind die Dateien, an denen Aktionen in der Pipeline arbeiten, z. B.

Dateien oder Ordner mit Anwendungscode, Indexseitendateien, Skripts usw. Das Amazon S3-Artefakt für Quellaktionen ist beispielsweise ein Dateiname (oder Dateipfad), in dem die Anwendungsquellcodedateien für die Pipeline-Quellcode-Aktion bereitgestellt werden. Die Dateien werden als ZIP-Datei bereitgestellt, wie zum Beispiel im folgenden Beispiel für einen Artefaktnamen:. `SampleApp_Windows.zip` Das Ausgabeartefakt für die Quellaktion, die Quellcodedateien der Anwendung, ist das Ausgabeartefakt dieser Aktion und das Eingabeartefakt für die nächste Aktion, z. B. eine Build-Aktion. Als weiteres Beispiel könnte eine Build-Aktion Build-Befehle ausführen, die den Anwendungsquellcode für ein Eingabe-Artefakt kompilieren, bei dem es sich um die Anwendungsquellcodedateien handelt. Einzelheiten zu Artefaktparametern, z. B. für die Aktion, finden Sie auf der Referenzseite zur Aktionskonfiguration AWS CodeBuild Aktionsreferenz erstellen und testen 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 die gewählte Quellaktion S3 ist.

Beispiel:

1. CodePipeline löst die Ausführung Ihrer Pipeline aus, wenn ein Commit für das Quell-Repository erfolgt, und stellt das Ausgabe-Artefakt (alle zu erstellenden Dateien) aus dem Source-Stagingbereich 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 (der erstellten Anwendung) wird als Eingabeartefakt in die Deploy-Phase aufgenommen, z. B. in der Staging- oder Produktionsumgebung 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. Für eine zweistufige Pipeline mit einer Quell - und Bereitstellungsphase wählen Sie beispielsweise unter 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 wird in derselben ein Amazon S3-Bucket erstellt, in dem Artikel für AWS-Konto alle AWS-Region Pipelines gespeichert werden. Jedes Mal, wenn Sie die Konsole verwenden, um eine weitere Pipeline in dieser Region zu erstellen, CodePipeline wird ein Ordner für diese Pipeline im Bucket erstellt. Dieser Ordner wird zum Speichern von Artefakten für Ihre Pipeline verwendet, während der automatisierte Veröffentlichungsprozess ausgeführt wird. Dieser Bucket heißt codepipeline-*region*-*12345EXAMPLE-1a2b-3c4d-67890EXAMPLE*, wobei sich die AWS Region *region* befindet, in der Sie die Pipeline erstellt haben, und *12345EXAMPLE-1a2b-3c4d-67890EXAMPLE* ist ein Zufallswert, der auf einer Universally Unique Identifier (UUID) basiert, 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, wird dieser als Standard-Bucket CodePipeline verwendet. Er folgt auch 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. Obwohl der Artefaktname abgeschnitten zu sein scheint, wird er dem Artefakt-Bucket CodePipeline auf eine Weise zugeordnet, die von Artefakten mit verkürzten Namen nicht beeinflusst 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-Region, AWS-Konto und Sie müssen einen Artefakt-Bucket pro Region haben, in der Sie eine Aktion ausführen möchten. Wenn Sie die Konsole verwenden, um eine Pipeline oder regionsübergreifende Aktionen zu erstellen, werden von CodePipeline in den Regionen, in denen Sie Aktionen haben, Standard-Artefakt-Buckets erstellt.

Wenn Sie die verwenden, AWS CLI um eine Pipeline zu erstellen, können Sie die Artefakte für diese Pipeline in jedem Amazon S3-Bucket speichern, solange sich dieser Bucket in demselben Bucket AWS-Konto und AWS-Region wie die Pipeline befindet. Sie können dies tun, wenn Sie Bedenken haben, die für Ihr Konto zulässigen Amazon 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.

Wie funktionieren die Bühnenbedingungen?

Für jede Bedingung, die eine Regel angibt, wird die Regel ausgeführt. Wenn die Bedingung fehlschlägt, führt die Stufe das angegebene Ergebnis aus. Optional geben Sie im Rahmen der Regel auch an, welche Ressourcen in bestimmten Fällen verwendet werden CodePipeline sollen. Beispielsweise verwendet die `CloudWatchAlarm` Regel eine CloudWatch Alarm-Ressource, um die Bedingung zu überprüfen.

Eine Bedingung kann mehreren Regeln entsprechen, und jede Regel kann einen von drei Anbietern angeben.

Der allgemeine Ablauf für die Erstellung von Bedingungen sieht wie folgt aus:

1. Wählen Sie den Bedingungstyp aus den verfügbaren Bedingungstypen unter CodePipeline. Verwenden Sie beispielsweise den Bedingungstyp Bei Erfolg, um eine Phase so einzurichten,

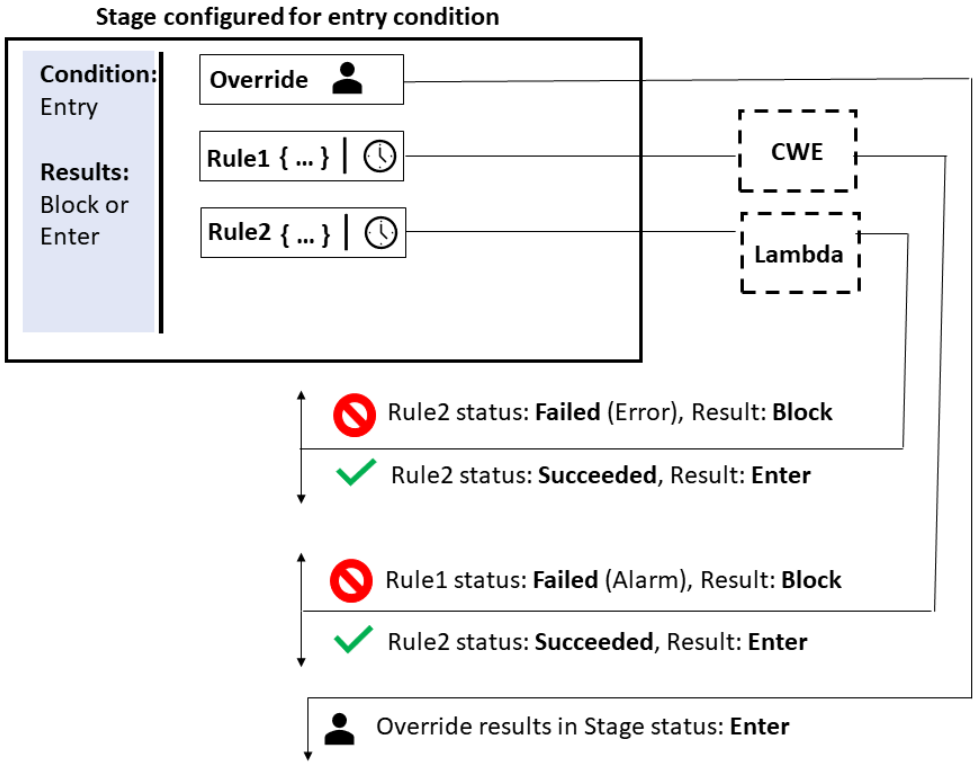
dass nach erfolgreichem Abschluss der Phase eine Reihe von Regeln verwendet werden kann, um Prüfungen durchzuführen, bevor der Vorgang fortgesetzt wird.

2. Wählen Sie die Regel aus. Die `CloudWatchAlarm` Regel sucht beispielsweise nach Alarmen und prüft mithilfe von EB, ob ein vorkonfigurierter Alarmschwellenwert vorliegt. Wenn die Überprüfung erfolgreich ist und der Alarm unter dem Schwellenwert liegt, kann die Phase fortgesetzt werden.
3. Konfigurieren Sie das Ergebnis, z. B. ein Rollback, das verwendet wird, wenn die Regel fehlschlägt.

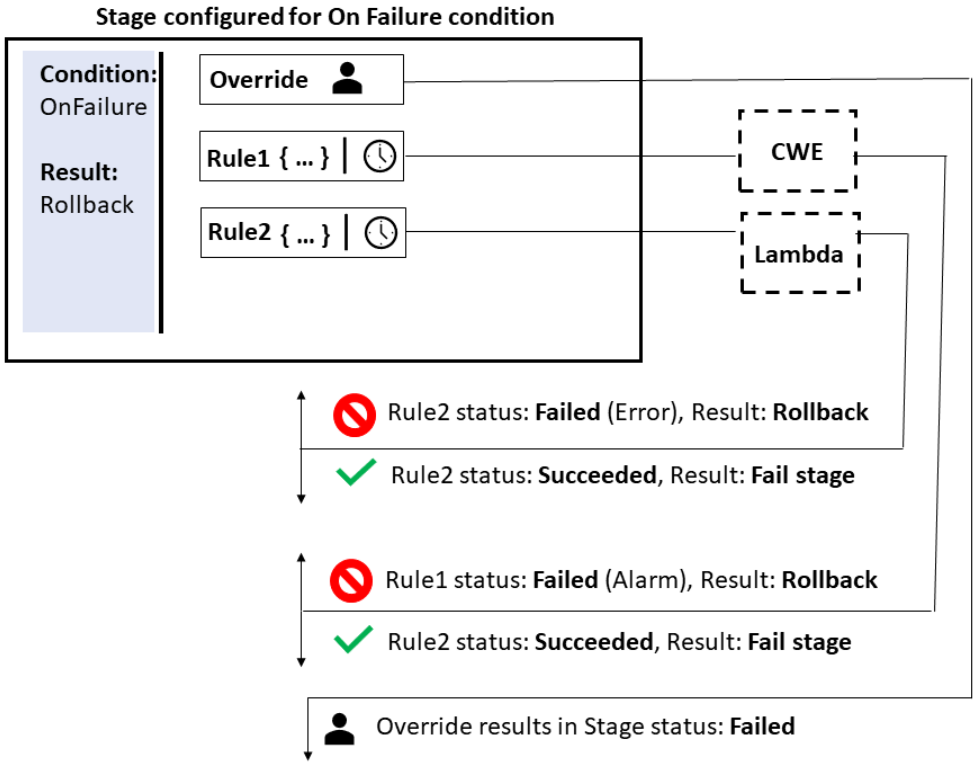
Bedingungen werden für bestimmte Arten von Ausdrücken verwendet, und für jeden gibt es spezifische Optionen für Ergebnisse, die wie folgt verfügbar sind:

- Eingabe — Die Bedingungen für die Durchführung von Prüfungen, die, wenn sie erfüllt sind, den Eintritt in eine Phase ermöglichen. Die Regeln sind mit den folgenden Ergebnisoptionen verknüpft: Fehlgeschlagen oder Überspringen
- Bei Misserfolg — Die Bedingungen, unter denen Prüfungen für die Phase durchgeführt werden, wenn sie fehlschlägt. Regeln verwenden die folgende Ergebnisoption: Rollback
- Bei Erfolg — Die Bedingungen, unter denen überprüft wird, ob die Phase erfolgreich abgeschlossen ist. Regeln werden mit den folgenden Ergebnisoptionen aktiviert: Rollback oder Fail

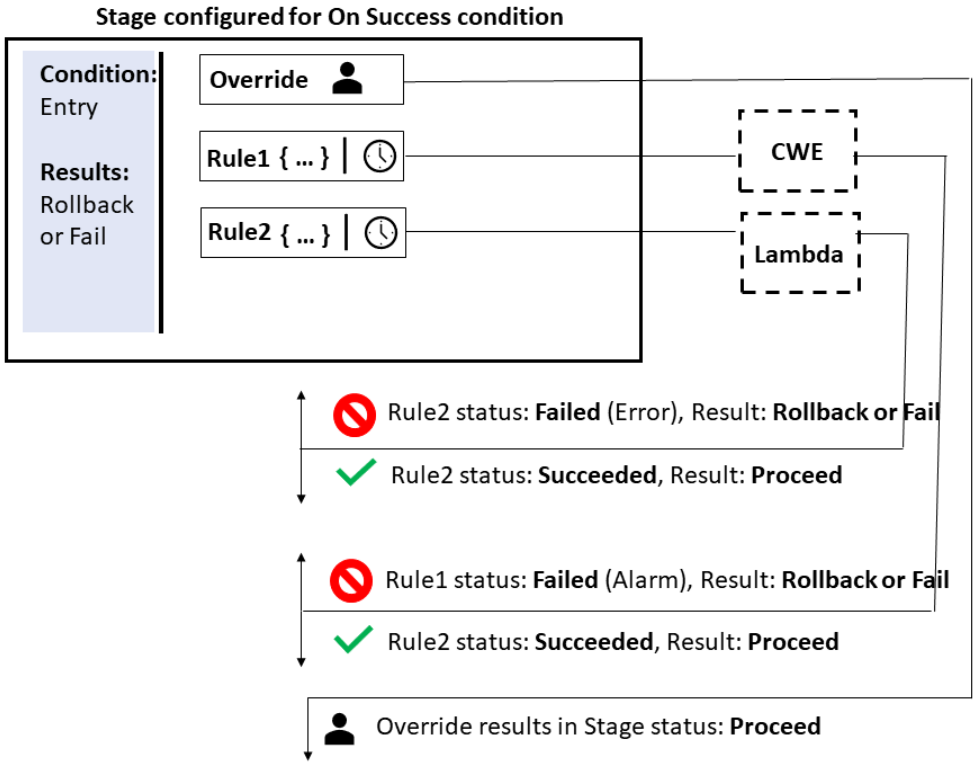
Das folgende Diagramm zeigt einen Beispielablauf für den Bedingungstyp Eingabe in CodePipeline. Bedingungen beantworten die Frage: Was sollte passieren, wenn die Bedingung nicht erfüllt ist, d. h., eine Regel schlägt fehl? Im folgenden Ablauf wird eine Eingabebedingung mit einer LambdaInvoke Regel und einer `CloudWatchAlarm` Regel konfiguriert. Wenn die Regel fehlschlägt, wird das konfigurierte Ergebnis, z. B. Fehlgeschlagen, aktiviert.



Das folgende Diagramm zeigt einen Beispielablauf für den Bedingungstyp On Failure in CodePipeline. Die Bedingungen beantworten die Frage: Was sollte passieren, wenn die Bedingung erfüllt ist, d. h., alle Regeln haben ihre Prüfungen bestanden? Im folgenden Ablauf wird die Bedingung „Bei Ausfall“ mit einer LambdaInvoke Regel und einer `CloudWatchAlarm` Regel konfiguriert. Wenn die Regel erfolgreich ist, wird das konfigurierte Ergebnis, z. B. Fehlgeschlagen, aktiviert.



Das folgende Diagramm zeigt einen Beispielablauf für den Bedingungstyp Bei Erfolg CodePipeline. Die Bedingungen beantworten die Frage: Was sollte passieren, wenn die Bedingung erfüllt ist, d. h., alle Regeln haben ihre Prüfungen bestanden? Im folgenden Ablauf wird die Bedingung „Bei Erfolg“ mit einer `LambdaInvoke` Regel und einer `CloudWatchAlarm` Regel konfiguriert. Wenn die Regel erfolgreich ist, wird das konfigurierte Ergebnis, z. B. Fehlgeschlagen, aktiviert.



Regeln für die Bedingungen einer Phase

Wenn Sie Stufenbedingungen konfigurieren, wählen Sie aus vordefinierten Regeln aus und geben die Ergebnisse für die Regel an. Ein Bedingungsstatus lautet Fehlgeschlagen, wenn eine der Regeln in der Bedingung nicht erfüllt ist, und Erfolgreich, wenn alle Regeln erfolgreich sind. Wie die Kriterien für die Bedingungen „Bei Fehlschlag“ und „Bei Erfolg“ erfüllt werden, hängt vom Typ der Regel ab.

Im Folgenden finden Sie verwaltete Regeln, die Sie den Stufenbedingungen hinzufügen können.

- Bedingungen können mithilfe der Befehlsregel Befehle angeben, die die Regelkriterien für Bedingungen erfüllen. Weitere Informationen zu dieser Regel finden Sie unterBefehle.
- Bedingungen können die AWS DeploymentWindow Regel verwenden, um die genehmigten Bereitstellungszeiten für die Zulassung einer Bereitstellung festzulegen. Die Kriterien für die Regel werden anhand eines bereitgestellten Cron-Ausdrucks für ein Bereitstellungsfenster gemessen. Die Regel ist erfolgreich, wenn Datum und Uhrzeit im Bereitstellungsfenster die Kriterien im Cron-Ausdruck für die Regel erfüllen. Weitere Informationen zu dieser Regel finden Sie unter. DeploymentWindow
- Bedingungen können die AWS Lambda-Regel verwenden, um nach Fehlerzuständen zu suchen, die von konfigurierten Lambda-Funktionen zurückgegeben wurden. Die Regel ist erfüllt,

wenn die Prüfung das Ergebnis der Lambda-Funktion erhält. Ein Fehler der Lambda-Funktion erfüllt die Kriterien für „On Failure“ -Bedingungen. Weitere Hinweise zu dieser Regel finden Sie unterLambdaInvoke.

- Bedingungen können die AWS CloudWatchAlarm Regel verwenden, um nach Alarmen zu suchen, die aufgrund von CloudWatch Ereignissen konfiguriert wurden. Die Regel ist erfüllt, wenn die Prüfung den Alarmstatus OK, ALARM oder INSUFFICIENT_DATA zurückgibt. Für die Bedingungen On Success erfüllen OK und INSUFFICIENT_DATA die Kriterien. ALARM erfüllt die Kriterien für „On Failure“ -Bedingungen. Weitere Hinweise zu dieser Regel finden Sie unterCloudWatchAlarm.
- Bedingungen können die VariableCheck Regel verwenden, um eine Bedingung zu erstellen, bei der die Ausgabevariable mit einem angegebenen Ausdruck verglichen wird. Die Regel besteht die Prüfung, wenn der Variablenwert die Regelkriterien erfüllt, z. B. wenn der Wert gleich oder größer als eine angegebene Ausgabevariable ist. Weitere Hinweise zu dieser Regel finden Sie unterVariableCheck.

Pipeline-Typen

CodePipeline bietet die folgenden Pipeline-Typen, die sich in ihren Eigenschaften und Preisen unterscheiden, sodass Sie die Funktionen und Kosten Ihrer Pipeline 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, zusammen mit zusätzlichen Parametern für die Versionssicherheit und die Triggerkonfiguration.

Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

Einzelheiten zu den Parametern der einzelnen Pipeline-Typen finden Sie auf der CodePipeline Referenz zur Pipeline-Struktur Seite. Hinweise dazu, 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, bei denen die Konfiguration durch die Übergabe von Variablen auf Pipeline-Ebene zur Laufzeit erfolgt • Bereitstellungen, bei denen Pipelines so konfiguriert sind, dass sie mit Git-Tags beginnen
Action-levelVariablen	Unterstützt	Unterstützt
PARALLELER Ausführungsmodus	Nicht unterstützt	Unterstützt
Pipeline-level Variablen	Nicht unterstützt	Unterstützt
AUSFÜHRUNGSMODUS IN DER WARTESCHLANGE	Nicht unterstützt	Unterstützt
Rollback für Pipeline-Phasen	Nicht unterstützt	Unterstützt
Überschreibungen der Quellversion	Nicht unterstützt	Unterstützt
Stufenbedingungen	Nicht unterstützt	Unterstützt
Triggert und filtert Git-Tags, Pull-Requests, Branches oder Dateipfade	Nicht unterstützt	Unterstützt
Die Aktion „Befehle“	Nicht unterstützt	Unterstützt

	Typ V1	Typ V2
Merkmale		
Eingabebedingungen mit dem Befehl Ergebnis überspringen erstellen	Nicht unterstützt	Unterstützt
Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler	Nicht unterstützt	Unterstützt

Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

Sie können ein Python-Skript erstellen und ausführen, um die potenziellen Kosten für die Umstellung einer Pipeline vom Typ V1 auf eine Pipeline vom Typ V2 zu analysieren.

> **Note**
>
> Das folgende Beispielskript dient nur zu Demonstrations- und Evaluierungszwecken. Es ist kein Angebotstool und garantiert nicht die Kosten für Ihre tatsächliche Nutzung einer Pipeline vom Typ V2, und es beinhaltet keine eventuell anfallenden Steuern. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

Um ein Skript zu erstellen und auszuführen, das Ihnen bei der Bewertung der Kosten für die Umstellung einer Pipeline vom Typ V1 auf eine Pipeline vom Typ V2 hilft

1. Laden Sie Python herunter und installieren Sie es.
2. Öffnen Sie ein Terminal-Fenster. Führen Sie den folgenden Befehl aus, um ein neues Python-Skript mit dem Namen zu erstellen PipelineCostAnalyzer.py.

   ```
   vi PipelineCostAnalyzer.py
   ```

3. 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) * 0.002
                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)
```

4. Wechseln Sie vom Terminal oder der Befehlszeile aus in das Verzeichnis, in dem Sie das Analyzer-Skript erstellt haben.

   Führen Sie in diesem Verzeichnis den folgenden Befehl aus, wobei Region der Ort ist, an AWS-Region dem Sie die V1-Pipelines erstellt haben, die Sie analysieren möchten. Sie können optional auch eine bestimmte Pipeline evaluieren, indem Sie ihren Namen angeben:

   ```
   python3 PipelineCostAnalyzer.py region --pipelineName
   ```

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

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

> **Note**
>
> Dieses Skript analysiert alle V1-Pipelines in der angegebenen Region, AWS-Region sofern Sie keinen bestimmten Pipeline-Namen angeben.

5. 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 die geschätzten Kosten dieser Aktionen, wie sie in einer V2-Pipeline ausgeführt wurden.

```
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
```

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

> **Note**
>
> Bei der Skriptausgabe und den zugehörigen Beispielen, die Kosten und andere Informationen zeigen, handelt es sich lediglich um Schätzungen. Sie dienen nur zu Demonstrations- und Bewertungszwecken und garantieren keine tatsächlichen Einsparungen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

Erste Schritte mit CodePipeline

Wenn Sie noch nicht damit vertraut sind 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

- Melden Sie sich an für ein AWS-Konto
- Wenden Sie eine verwaltete Richtlinie für den Administratorzugriff an auf CodePipeline
- Installieren Sie das AWS CLI
- Öffnen Sie die Konsole für CodePipeline
- Nächste Schritte

Melden Sie sich an für ein AWS-Konto

Um loszulegen AWS, benötigen Sie eine AWS-Konto. Es gibt zwei Möglichkeiten, sich anzumelden AWS. Wenn Sie die vollständige Kontrolle über die Kontokonfiguration und den Speicherort Ihrer Inhalte haben möchten oder wenn Sie regulierte Workloads wie HIPAA oder FedRAMP haben, verwenden Sie Sign up for (advanced). AWS Wenn Sie ein neuer Benutzer sind und einen

Anmeldenamen verwenden möchten, den Sie bereits besitzen, um sich anzumelden AWS und eine AWS Umgebung mit vorkonfigurierten Standardeinstellungen in select zu erstellen, verwenden Sie Sign up for. AWS-RegionenAWS

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 Sie 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 Anleitung unter Eine Rolle für einen externen Identitätsanbieter (Verbund) erstellen im IAM-Benutzerhandbuch.

- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter Eine Rolle für einen IAM-Benutzer erstellen 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.

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 das herunter und installieren Sie es AWS CLI. Auf diese Weise können Sie über die CodePipeline Befehlszeile 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 nach dem Namen der Standardregion gefragt werden, geben Sie die Region an, in der Sie die Pipeline erstellen möchten, z. B. `us-east-2` Wenn Sie nach dem standardmäßigen Ausgabeformat gefragt werden, geben Sie `json` an. Beispiel:

```
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 verfügbaren Regionen und Endpunkten finden Sie unter AWS CodePipeline Endpunkte und Kontingente. CodePipeline

Öffnen Sie die Konsole für CodePipeline

- Melden Sie sich bei der an AWS-Managementkonsole 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. Informationen zum Einstieg in die Arbeit mit CodePipeline finden Sie unter CodePipeline Tutorials.

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
- Allgemeine Integrationen mit CodePipeline
- Beispiele aus der Community

Integrationen mit CodePipeline Aktionstypen

Die Integrationsinformationen in diesem Thema sind nach CodePipeline Aktionstypen geordnet.

Themen

- Quellaktions-Integrationen
- Erstellen von Aktionsintegrationen
- Testen von Aktionsintegrationen
- Bereitstellungsaktions-Integrationen
- Integration von Genehmigungsaktionen mit Amazon Simple Notification Service
- Aufrufen von Aktionsintegrationen

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

- Amazon ECR-Quellaktionen
- Amazon S3 S3-Quellaktionen

- Verbindungen zu Bitbucket Cloud GitHub (über GitHub App), GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete
- CodeCommit Quellaktionen
- GitHub (über die OAuth-App) 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:

- Informationen zu Konfigurationsparametern und einem JSON/YAML Beispielausschnitt finden Sie unter Referenz zur Amazon ECR-Quellaktion
- Eine Pipeline, Phasen und Aktionen erstellen
- Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-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 JSON/YAML Beispiel-Snippet finden Sie unter Referenz zur Amazon S3-Quellaktion
- Schritt 1: Erstellen Sie einen S3-Quell-Bucket 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 (über GitHub App), GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete

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

> **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 und GitLab.com 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 Connector-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: • Informationen zu Konfigurationsparametern und einem JSON/YAML Beispielausschnitt finden Sie unter. CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com 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 Connector-App mit Ihrem Code-Repository eines Drittanbieters und verknüpfen sie dann mit Ihrer Verbindung.

	Verwenden Sie die Provider-Option GitHub (via GitHub App) 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 JSON/YAML Beispiel-Snippet finden Sie unter CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com 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 Aktion GitHub (über GitHub App) ist die Quellaktion für GitHub Version 2. Die Aktion GitHub (über die OAuth-App) ist die Aktion der Version 1, die mit der GitHub OAuth-Token-Authentifizierung verwaltet wird. Wir empfehlen zwar nicht, die Aktion GitHub (über die OAuth-App) zu verwenden, aber bestehende Pipelines mit der Aktion GitHub (über die OAuth-App) funktionieren weiterhin ohne Auswirkungen. Sie können jetzt eine CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen Quellaktion in Ihrer Pipeline verwenden, die Ihre Quellaktion mit Apps verwaltet. GitHub GitHub Wenn Sie eine Pipeline haben, die die Aktion GitHub (über die OAuth-App) verwendet, lesen Sie die Schritte zur Aktualisierung, um sie auf die Verwendung einer Aktion GitHub (über GitHub App) in zu aktualisieren. Aktualisieren Sie eine Quellaktion GitHub (über die OAuth-App) auf eine Quellaktion GitHub (über die GitHub App)
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, einrichten, damit Ihre Pipelines auf Code-Repositorys von Drittanbietern zugreifen können. Wenn Sie eine Verbindung herstellen, installieren Sie die Connector-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. 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 JSON/YAML Beispiel-Snippet finden Sie unter CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com 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 Repository als Quelle für Ihren Code verwendet wird. Sie müssen zuvor ein GitLab.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 JSON/YAML Beispielausschnitt finden Sie unter CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com 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 Connector-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 Selbstverwaltung. 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 JSON/YAML Beispiel-Snippet finden Sie unter CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen

- Die Schritte zum Herstellen einer Verbindung mit diesem Anbietertyp finden Sie unter. Verbindungen für GitLab Selbstverwaltung

CodeCommit Quellaktionen

CodeCommit ist ein Versionskontroll-Service, den Sie zum privaten Speichern und Verwalten von Komponenten (z. B. Dokumenten, Quellcode und Binärdateien) in der Cloud verwenden können. Sie können so konfigurieren CodePipeline , dass ein Branch 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:

- Informationen zu Konfigurationsparametern und einem JSON/YAML Beispielausschnitt finden Sie unter. CodeCommit Quellaktionsreferenz
- Tutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen
- 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 (über die OAuth-App) Quellaktionen

Die Aktion GitHub (über die OAuth-App) ist die Aktion der Version 1 GitHub , die mit OAuth Apps verwaltet wird. In verfügbaren Regionen können Sie auch eine CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen Quellaktion in Ihrer Pipeline verwenden, die Ihre GitHub Quellaktion mit Apps verwaltet. GitHub Wenn Sie eine Pipeline haben, die die Aktion GitHub (über die OAuth-App) verwendet, lesen Sie die Schritte zur Aktualisierung, um sie für die Verwendung einer Aktion GitHub (per GitHub App) in zu

aktualisieren. Aktualisieren Sie eine Quellaktion GitHub (über die OAuth-App) auf eine Quellaktion GitHub (über die GitHub App)

> **Note**
>
> Wir empfehlen zwar nicht, die Aktion GitHub (über die OAuth-App) zu verwenden, aber bestehende Pipelines mit der Aktion GitHub (über die OAuth-App) funktionieren weiterhin ohne Auswirkungen.

Weitere Informationen:

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

Erstellen von Aktionsintegrationen

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

Themen

- CodeBuild Aktionen erstellen
- CloudBees Aktionen erstellen
- Amazon ECR Aktionen zum Erstellen und Veröffentlichen
- Jenkins-Build-Aktionen
- TeamCity Aktionen erstellen

CodeBuild Aktionen erstellen

CodeBuild ist ein vollständig verwalteter Build-Service. Sie können damit Ihren Quellcode kompilieren, Einheitentests ausführen und bereitstellbare Artefakte generieren.

Sie können CodeBuild sie als Build-Aktion zur Build-Phase einer Pipeline hinzufügen. Weitere Informationen finden Sie in der Referenz zur CodePipeline Aktionskonfiguration fürAWS CodeBuild Aktionsreferenz erstellen und testen.

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

Weitere Informationen:

- Informationen zu Konfigurationsparametern und einem JSON/YAML Beispielausschnitt finden Sie unter. AWS CodeBuild Aktionsreferenz erstellen und testen
- 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

Amazon ECR Aktionen zum Erstellen und Veröffentlichen

Amazon ECR ist ein AWS Docker-Image-Repository-Service. Sie verwenden Docker-Push- und Pull-Befehle zum Hochladen von Docker-Images auf Ihr Repository.

Sie können die `ECRBuildAndPublish` Aktion zu Ihrer Pipeline hinzufügen, um die Erstellung und Übertragung eines Images zu automatisieren. Weitere Informationen finden Sie in der Referenz zur CodePipeline Aktionskonfiguration für`ECRBuildAndPublish`Aktionsreferenz erstellen.

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 nur Verbindungen akzeptiert. HTTPS/SSL 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

- Spiegelt die Testautomatisierung wider

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 CodePipeline-Referenz für Aktionskonfigurationen für AWS CodeBuild Aktionsreferenz erstellen und testen.

> **Note**
>
> CodeBuild kann auch als Build-Aktion in eine Pipeline aufgenommen werden, wobei ein obligatorisches Build-Ausgabeartefakt erforderlich ist.

Weitere Informationen:

- Informationen zu Konfigurationsparametern und einem JSON/YAML Beispielausschnitt finden Sie unter. AWS CodeBuild Aktionsreferenz erstellen und testen
- 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-zu-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:

- Informationen zu Konfigurationsparametern und einem JSON/YAML Beispielausschnitt finden Sie unter. AWS Device Farm Referenz zur Testaktion
- 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

Spiegelt die Testautomatisierung wider

Reflect ist die AI-powered Testautomatisierungslösung, mit der Sie das Testen vereinfachen und die Herausforderungen manueller Prozesse bewältigen können. Mit der Testautomatisierung ohne Code optimiert Reflect die Erstellung, Ausführung und Wartung von Tests und ermöglicht es Ihnen, robuste, wiederholbare Tests zu erstellen, ohne dass technische Kenntnisse erforderlich sind. Durch die Eliminierung der Komplexität und die Minimierung von Unterbrechungen Ihrer Arbeitsabläufe können Sie Tests beschleunigen und jederzeit sicher hochwertige Anwendungen bereitstellen.

Weitere Informationen:

- AWS CodePipeline Testen Sie die Integration mit Reflect

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 EC2 EC2-Bereitstellungsaktionen
- Amazon Elastic Kubernetes Service EKS-Bereitstellungsaktionen
- Amazon S3 S3-Bereitstellungsaktionen
- AWS AppConfig Aktionen bereitstellen
- CloudFormation Aktionen bereitstellen
- CloudFormation StackSets Aktionen bereitstellen
- Amazon ECS-Bereitstellungsaktionen
- Elastic Beanstalk Beanstalk-Bereitstellungsaktionen
- OpsWorks Aktionen bereitstellen
- Bereitstellungsaktionen für den Service Catalog
- Amazon Alexa Bereitstellungsaktionen
- CodeDeploy Aktionen bereitstellen
- XebiaLabs Aktionen bereitstellen

Amazon EC2 EC2-Bereitstellungsaktionen

Mit Amazon EC2 können Sie Rechenleistung in der Cloud erstellen und verwalten. Sie können eine Aktion zu einer Pipeline hinzufügen, die Amazon EC2 als Bereitstellungsanbieter verwendet, der Ihre Anwendung auf Ihren Instances bereitstellt.

Weitere Informationen:

- Die Aktionsreferenzseite finden Sie unter. Amazon EC2-Aktionsreferenz
- Ein Tutorial finden Sie unter Tutorial: Bereitstellung auf Amazon EC2-Instances mit CodePipeline.

Amazon Elastic Kubernetes Service **EKS-Bereitstellungsaktionen**

Mit Amazon EKS können Sie Kubernetes-Cluster erstellen und verwalten. Sie können eine Aktion zu einer Pipeline hinzufügen, die Amazon EKS als Bereitstellungsanbieter verwendet, der Ihr Image in Ihrem Cluster bereitstellt. Sie können Helm-Vorlagen oder Kubernetes-Manifestdateien verwenden.

Weitere Informationen:

- Die Aktionsreferenzseite finden Sie unter. Referenz zur Amazon Elastic Kubernetes Service EKS-Bereitstellungsaktion
- Ein Tutorial finden Sie unter Tutorial: Bereitstellung auf Amazon EKS mit CodePipeline.

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:

- Eine Pipeline, Phasen und Aktionen erstellen
- 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 Aktionsreferenz bereitstellen
- Tutorial: Eine Pipeline erstellen, die AWS AppConfig als Bereitstellungsanbieter verwendet wird

CloudFormation Aktionen bereitstellen

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 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 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 CloudFormation Stacks und Änderungssätze ausführen. 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 CloudFormation Vorlagen AWS und Parameterdefinitionen angeben, erstellt, bereitgestellt, aktualisiert oder beendet werden.

Weitere Informationen:

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

CloudFormation StackSets Aktionen bereitstellen

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

Sie können CodePipeline with verwenden 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 CloudFormation StackSets Aktionsreferenz bereitstellen
- Tutorial: Erstellen Sie eine Pipeline mit AWS CloudFormation StackSets Aktionen zur Bereitstellung

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 (Blue/Green) -Anbieteraktion in verwenden CodePipeline , um Datenverkehr an Amazon ECS weiterzuleiten und bereitzustellen CodeDeploy.

Weitere Informationen:

- Was ist Amazon ECS?
- Tutorial: Kontinuierliche Bereitstellung mit CodePipeline
- Eine Pipeline, Phasen und Aktionen erstellen
- Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-CodeDeploy

Elastic Beanstalk Beanstalk-Bereitstellungsaktionen

Elastic Beanstalk ist ein Service für die Bereitstellung und Skalierung von Webanwendungen und Services, die mit Java, .NET, PHP, Python, Ruby Node.js, Go und Docker entwickelt wurden,

auf vertrauten Servern wie Apache, Nginx, Passenger und IIS. 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) oder Europa (Zürich) nicht verfügbar. Weitere verfügbare Aktionen finden Sie unterProdukt- und Serviceintegrationen mit CodePipeline.

Weitere Informationen:

- Erste Schritte mit Elastic Beanstalk
- Eine Pipeline, Phasen und Aktionen erstellen

OpsWorks Aktionen bereitstellen

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 OpsWorks bereitstellen.

- Benutzerdefinierte Chef-Kochbücher — OpsWorks verwendet Chef Cookbooks, um Aufgaben wie die Installation und Konfiguration von Paketen und die Bereitstellung von Anwendungen zu erledigen.
- Anwendungen — Eine 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 OpsWorks Stack und die Ebene. Sie können die 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 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.

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

Weitere Informationen:

- Tutorial: Erstellen Sie eine Pipeline, die im Service Catalog bereitgestellt wird
- Eine Pipeline, Phasen und Aktionen erstellen

Amazon Alexa Bereitstellungsaktionen

Mit dem Amazon Alexa Skills Kit können Sie Cloud-basierte Fähigkeiten aufbauen und an Benutzer von Alexa-enabled Geräten 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 auf EC2/on-premises Amazon-Instances, Amazon Elastic Container Service-Rechenplattformen und serverlosen AWS Lambda Rechenplattformen. Sie können ihn so konfigurieren CodePipeline , dass er CodeDeploy zur Bereitstellung Ihres Codes verwendet wird. Sie können die CodeDeploy -Anwendung, -Bereitstellung und -Bereitstellungsgruppe, die in einer Bereitstellungsaktion innerhalb einer Phase verwendet werden sollen, entweder vor Erstellung der Pipeline oder während der Verwendung des Create Pipeline (Pipeline erstellen)-Assistenten erstellen.

Weitere Informationen:

- Schritt 3: Erstellen Sie eine Anwendung in CodeDeploy
- Tutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen

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-Berechtigungen für eine Servicerolle CodePipeline

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

- Amazon Inspector Aktionen aufrufen
- Lambda-Aufrufaktionen
- Step Functions rufen Aktionen auf

Amazon Inspector Aktionen aufrufen

Amazon Inspector ist ein Schwachstellen-Management-Service, der Workloads automatisch erkennt und sie kontinuierlich auf Software-Schwachstellen und unbeabsichtigte Netzwerkbedrohungen überprüft. Amazon Inspector unterstützt mehrere Archivformate, darunter Tar und War, und Amazon Inspector unterstützt Binärdateien wie Rust- und Go-Binärdateien.

Sie können die CodePipeline `InspectorScan` Aktion so konfigurieren, dass Ihr Quellcode oder Ihr Amazon ECR-Image-Repository automatisch nach Sicherheitslücken gescannt wird.

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für Referenz zum `InspectorScan` Aufrufen der Aktion durch Amazon Inspector

Lambda-Aufrufaktionen

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 Aktionsreferenz aufrufen
- Ruft einen auf AWS Lambda Funktion in einer Pipeline in CodePipeline

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 Aktionsreferenz aufrufen
- 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: • Überwachung von CodePipeline Ereignissen • Tutorial: Richten Sie eine CloudWatch Event-Regel ein, um E-Mail-Benachrichtigungen über 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: • Protokollierung von CodePipeline API-Aufrufen 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 einer anderen verwendet werden AWS Konto. • Um eine Pipeline von einem AWS Konto zu erstellen, das einen 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 serversei tige 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. Bei kontoübergreifenden Aktionen können Sie nur die Schlüssel-ID oder den Schlüssel-ARN verwenden, um den Schlüssel zu identifizieren. Cross-account Aktionen beinhalten die Verwendung der Rolle aus dem anderen Konto (AccountB). Wenn Sie also die Schlüssel-ID angeben, wird der Schlüssel des anderen Kontos (AccountB) verwendet.

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

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

- Komplett CI/CD mit AWS CodeCommitAWS CodeBuild, AWS CodeDeploy, und AWS 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

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

  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 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. CloudFormation

  - Laden Sie eine 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 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 CloudFormation

  Wird verwendet 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 Cross-Account 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 CloudFormation Vorlage.

  Veröffentlichung März 2016

- Erkundung des ASP.NET Kernteils 2: Continuous Delivery

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

Veröffentlichung März 2016

- Erstellen Sie mithilfe der AWS CodePipeline Konsole eine Pipeline

  Erfahren Sie in einer exemplarischen Vorgehensweise, wie Sie mithilfe 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 mit der Lambda-Funktion testen, ob Ihre Pipeline erfolgreich abgeschlossen wird.

Veröffentlichung Februar 2016

- AWS Lambda Funktionen ausführen unter Verwenden CodePipeline AWS CloudFormation

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

Veröffentlichung Februar 2016

- Bereitstellung benutzerdefinierter CodePipeline Aktionen in AWS CloudFormation

  Weitere 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 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 Tutorials

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

Themen

- Tutorial: Bereitstellung auf Amazon EC2-Instances mit CodePipeline
- Tutorial: Erstellen Sie ein Docker-Image und übertragen Sie es auf Amazon ECR mit CodePipeline (Typ V2)
- Tutorial: Bereitstellung auf Amazon EKS mit CodePipeline
- Tutorial: Erstellen Sie eine Pipeline, die Befehle mit Compute ausführt (Typ V2)
- Tutorial: Verwende Git-Tags, um deine Pipeline zu starten
- Tutorial: Filtere nach Branch-Namen für Pull-Requests, um deine Pipeline zu starten (Typ V2)
- Tutorial: Variablen auf Pipeline-Ebene verwenden
- Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket)
- Tutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen
- Tutorial: Erstellen einer vierstufigen Pipeline
- Tutorial: Richten Sie eine CloudWatch Event-Regel ein, um E-Mail-Benachrichtigungen über Änderungen des Pipeline-Status zu erhalten
- Tutorial: Erstellen Sie eine Pipeline, die Ihre Android-App erstellt und testet mit AWS Device Farm
- Tutorial: Erstellen Sie eine Pipeline, die Ihre iOS-App testet mit AWS Device Farm
- Tutorial: Erstellen Sie eine Pipeline, die im Service Catalog bereitgestellt wird
- Tutorial: Erstellen Sie eine Pipeline mit AWS CloudFormation
- Tutorial: Erstellen Sie eine Pipeline, die Variablen von verwendet AWS CloudFormation Aktionen zur Bereitstellung
- Tutorial: Amazon ECS-Standardbereitstellung mit CodePipeline
- Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-CodeDeploy
- Tutorial: Erstellen einer Pipeline zum Bereitstellen eines Amazon Alexa-Skills
- Tutorial: Erstellen Sie eine Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet
- Tutorial: Erstellen Sie eine Pipeline, die Ihre serverlose Anwendung auf dem veröffentlicht AWS Serverless Application Repository

- Tutorial: Bereitstellung von Lambda-Funktionen mit CodePipeline
- Tutorial: Variablen mit Lambda-Aufrufaktionen verwenden
- Tutorial: Verwenden Sie eine AWS Step Functions Aufrufaktion in einer Pipeline
- Tutorial: Eine Pipeline erstellen, die AWS AppConfig als Bereitstellungsanbieter verwendet wird
- Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden
- Tutorial: Vollständigen Klon mit einer CodeCommit Pipeline-Quelle verwenden
- Tutorial: Erstellen Sie eine Pipeline mit AWS CloudFormation StackSets Aktionen zur Bereitstellung
- Tutorial: Erstellen Sie eine Variablenprüfregel für eine Pipeline als Eingabebedingung

Tutorial: Bereitstellung auf Amazon EC2-Instances mit CodePipeline

Dieses Tutorial hilft Ihnen dabei, eine Bereitstellungsaktion zu erstellen CodePipeline , mit der Ihr Code auf Instances bereitgestellt wird, die Sie in Amazon EC2 konfiguriert haben.

> **Note**
>
> Im Rahmen der Erstellung einer Pipeline in der Konsole wird ein S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

> **Note**
>
> Die EC2 Bereitstellungsaktion ist nur für Pipelines vom Typ V2 verfügbar.

Voraussetzungen

Es müssen bestimmte Ressourcen zur Verfügung stehen, damit Sie mithilfe dieses Tutorials eine CD-Pipeline erstellen können. Zum Einstieg benötigen Sie Folgendes:

> **Note**
>
> Alle diese Ressourcen sollten in derselben AWS Region erstellt werden.

- Ein Quellcodeverwaltungs-Repository (dieses Tutorial verwendet GitHub), dem Sie eine `script.sh` Beispieldatei hinzufügen werden.
- Sie müssen eine vorhandene CodePipeline Servicerolle verwenden, die mit den Berechtigungen für diese Aktion aktualisiert wurde. Informationen zum Aktualisieren Ihrer Servicerolle finden Sie unterRichtlinienberechtigungen für die EC2-Bereitstellungsaktion.

Nachdem Sie diese Voraussetzungen erfüllt haben, können Sie mit dem Tutorial fortfahren und Ihre CD-Pipeline erstellen.

Schritt 1: Amazon EC2 Linux-Instances erstellen

In diesem Schritt erstellen Sie die Amazon EC2-Instances, auf denen Sie eine Beispielanwendung bereitstellen werden. Erstellen Sie im Rahmen dieses Vorgangs eine Instance-Rolle in IAM, falls Sie in der Region, in der Sie Ressourcen erstellen möchten, noch keine Instance-Rolle erstellt haben.

So erstellen Sie eine Instance-Rolle

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
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-Service die Option aus. Wählen Sie unter Wählen Sie einen Anwendungsfall die Option EC2 aus. Wählen Sie in Select your use case (Anwendungsfall auswählen) die Option EC2 aus. Wählen Sie Weiter aus.
5. Suchen Sie nach der benannten **`AmazonSSMManagedEC2InstanceDefaultPolicy`** Richtlinie und wählen Sie sie aus.
6. Suchen Sie nach der benannten 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.

> **Note**
>
> Sie fügen dieser Rolle Berechtigungen hinzu, um nach der Pipeline-Erstellung den Zugriff auf den S3-Artefakt-Bucket für Ihre Pipeline zu ermöglichen.

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 **`MyInstances`** ein. Dadurch wird der Instance ein Tag-Schlüssel von **`Name`** und ein Tag-Wert von **`MyInstances`** zugewiesen.
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) beschrieben und trägt die Bezeichnung „Kostenloses Kontingent berechtigt“.)
5. Wählen Sie unter Instance-Typ den **`t2.micro`** Typ aus, der für das kostenlose Kontingent in Frage kommt, als Hardwarekonfiguration für Ihre Instance.
6. Wählen Sie unter Schlüsselpaar (Anmeldung) ein Schlüsselpaar aus oder erstellen Sie eines.
7. Vergewissern Sie sich, dass unter Netzwerkeinstellungen der Status Aktiviert lautet.
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`**).

> **Note**
>
> Lassen Sie das Feld Instanzrolle nicht leer, da dadurch eine Standardrolle erstellt wird und die von Ihnen erstellte Rolle nicht ausgewählt wird.

9. Geben Sie unter Zusammenfassung unter Anzahl der Instanzen den Wert ein2.
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 Öffentliches DNS nicht angezeigt wird, wählen Sie das Show/Hide Symbol und dann Public DNS aus.)

Schritt 2: Fügen Sie der EC2-Instance-Rolle Artifact-Bucket-Berechtigungen hinzu

Sie müssen die EC2-Instance-Rolle, die Sie für Ihre Instance erstellt haben, aktualisieren, damit sie auf den Artefakt-Bucket Ihrer Pipeline zugreifen kann.

> **Note**
>
> Wenn Sie die Instance erstellen, erstellen oder verwenden Sie eine vorhandene EC2-Instance-Rolle. Um `Access Denied` Fehler zu vermeiden, müssen Sie der Instance-Rolle S3-Bucket-Berechtigungen hinzufügen, um die Instance-Berechtigungen für den CodePipeline Artefakt-Bucket zu erteilen. Erstellen Sie eine Standardrolle oder aktualisieren Sie Ihre bestehende Rolle mit der `s3:GetObject` Berechtigung, die auf den Artefakt-Bucket für die Region Ihrer Pipeline beschränkt ist.

1. Navigieren Sie in der Konsole zu Ihrer Pipeline. CodePipeline Wählen Sie Einstellungen aus. Sehen Sie sich den Namen und den Speicherort des Artefaktspeichers für eine bestehende Pipeline an. Notieren Sie sich den Amazon-Ressourcennamen (ARN) des Artefakt-Buckets und kopieren Sie ihn.
2. Navigieren Sie zur IAM-Konsole, und wählen Sie Rollen aus. Wählen Sie die Instanzrolle aus, die Sie in Schritt 1 dieses Tutorials erstellt haben.

3. Wählen Sie in der Registerkarte Permissions (Berechtigungen) die Option Add inline policy (Inline-Richtlinie hinzufügen).
4. Fügen Sie dem Richtliniendokument den folgenden JSON-Code hinzu und ersetzen Sie den Wert im `Resource` Feld durch den Bucket-ARN.

```
{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::BucketName"
}
```

5. Wählen Sie Aktualisieren aus.

Schritt 3: Füge eine Skriptdatei zu deinem Repository hinzu

Fügen Sie diesen Beispieltext ein, um Ihre `script.sh` Datei für den Post-Script-Schritt in der Bereitstellung zu erstellen.

```
echo "Hello World!"
```

Um eine hinzuzufügen **`script.sh`** Datei in Ihr Quell-Repository

1. Öffnen Sie einen Texteditor und kopieren Sie dann die obige Datei und fügen Sie sie in eine neue Datei ein.
2. Führen Sie einen Commit und eine Push-Übertragung Ihrer `script.sh`-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 script.sh."
   ```

   c. Führen Sie eine Push-Übertragung für das Commit durch.

```
git push
```

Notieren Sie sich den Pfad in Ihrem Repository.

```
/MyDemoRepo/test/script.sh
```

Schritt 4: Deine Pipeline erstellen

Verwenden Sie den CodePipeline Assistenten, um Ihre Pipeline-Stufen zu erstellen und Ihr Quell-Repository zu verbinden.

So erstellen Sie Ihre Pipeline

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**MyPipeline**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
6. Wählen Sie unter Servicerolle die Option Bestehende Servicerolle verwenden aus, und wählen Sie dann die CodePipeline Servicerolle aus, die mit den erforderlichen Berechtigungen für diese Aktion aktualisiert wurde. Informationen zum Konfigurieren Ihrer CodePipeline Servicerolle für diese Aktion finden Sie unterRichtlinienberechtigungen für die EC2-Bereitstellungsaktion.
7. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
8. Fügen Sie auf der Seite Schritt 3: Quellstufe hinzufügen eine Quellstufe hinzu:

   a. Wählen Sie unter Quellanbieter die Option GitHub (über GitHub App) aus.

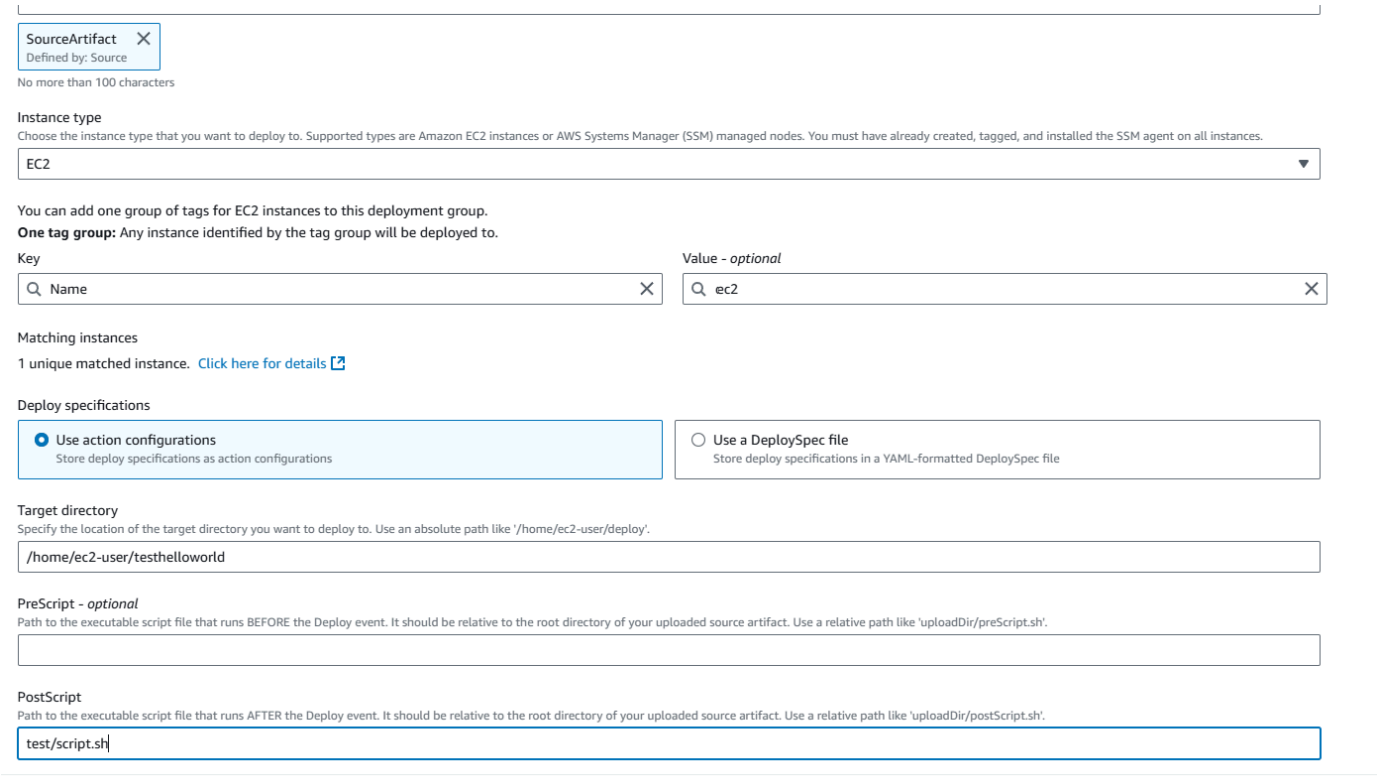
b. Wählen Sie unter Verbindung eine bestehende Verbindung aus oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.

c. Wählen Sie im Feld Repository name (Repositoryname) den Namen Ihres GitHub -Repositorys aus.

Wählen Sie Weiter aus.

9. Wählen Sie auf der Seite Schritt 4: Build-Phase hinzufügen die Option Überspringen aus.

10. Wählen Sie auf der Seite Schritt 5: Bereitstellungsstufe hinzufügen die Option EC2 aus.



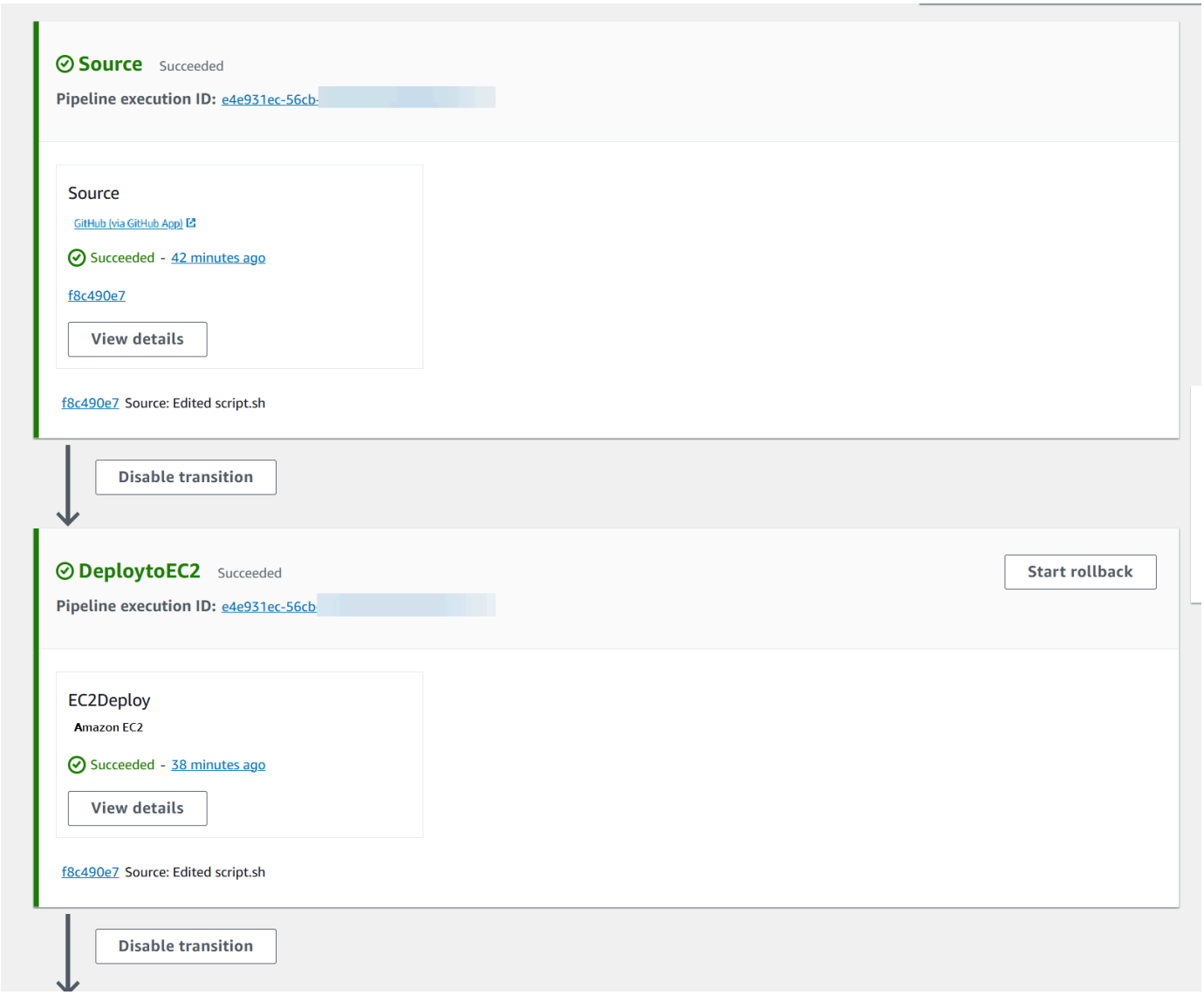
a. Geben Sie für Zielverzeichnis das Verzeichnis auf der Instance ein, auf der Sie die Bereitstellung durchführen möchten, z. B. `/home/ec2-user/testhelloworld`

> **Note**
>
> Geben Sie das Bereitstellungsverzeichnis an, das die Aktion auf der Instanz verwenden soll. Die Aktion automatisiert das Erstellen des angegebenen Verzeichnisses auf der Instanz als Teil der Bereitstellung.

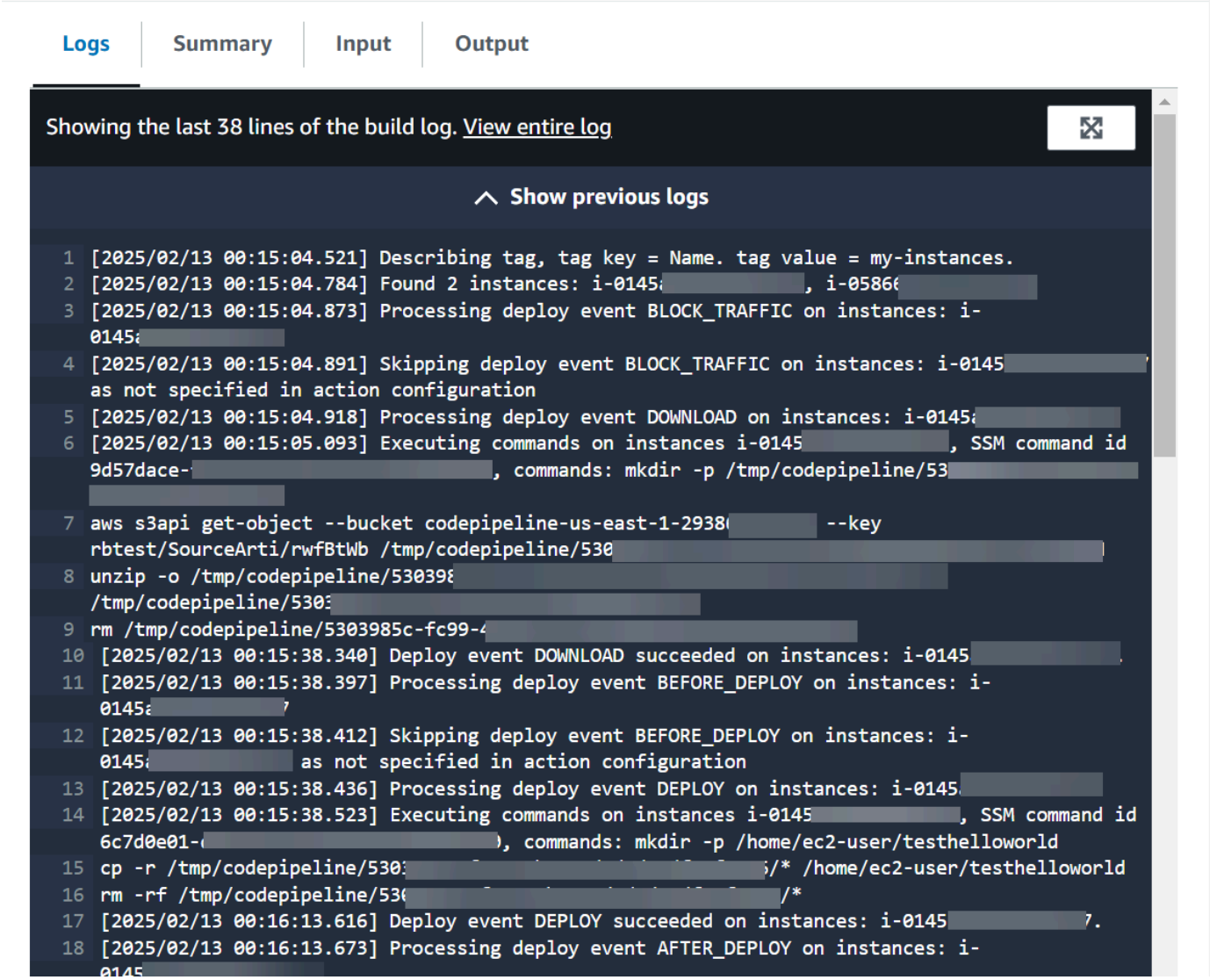
b. Geben Sie für PostScript den Pfad und den Dateinamen für Ihr Skript ein, z. `test/script.sh` B.

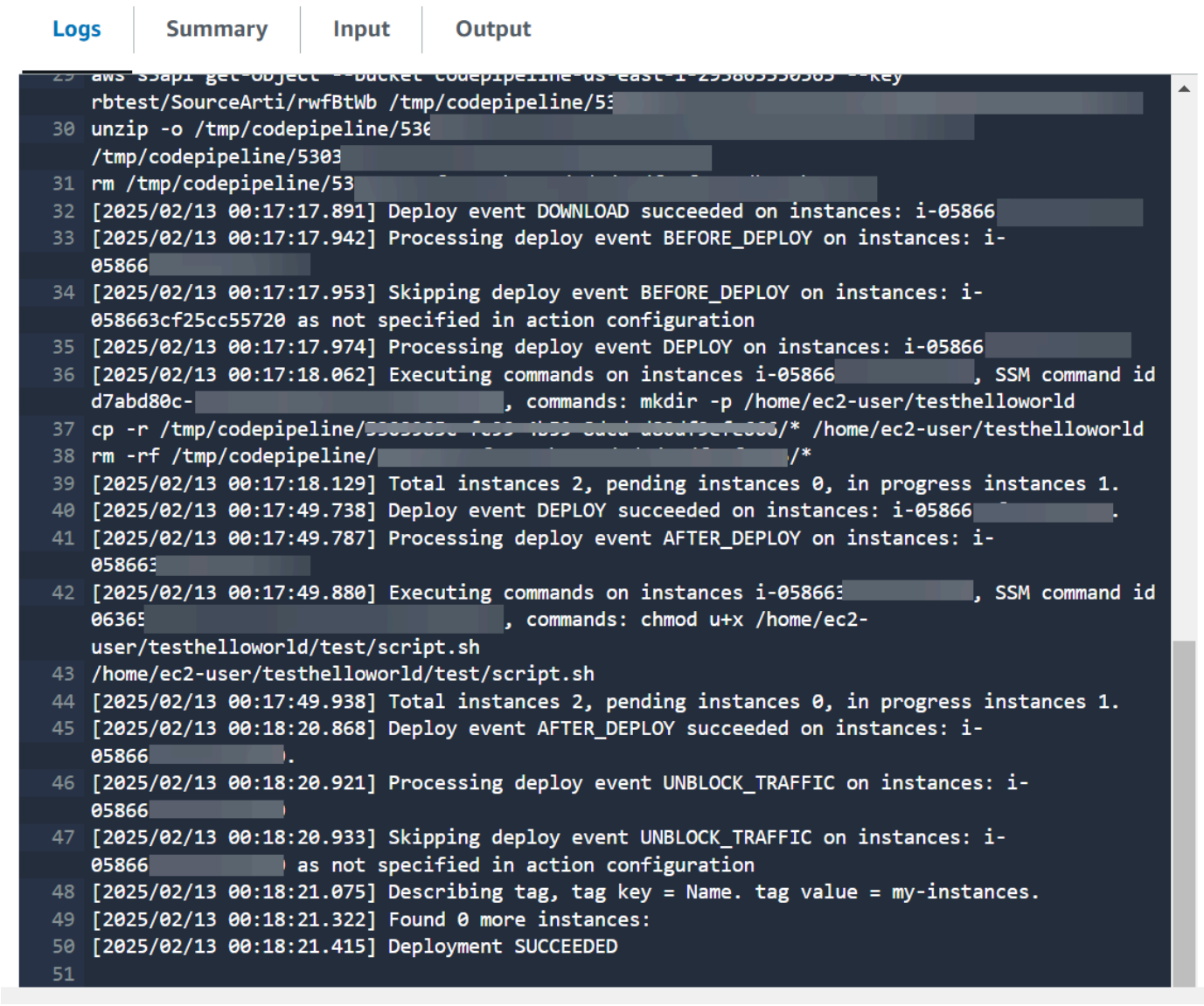
c. Wählen Sie Weiter aus.

11. Überprüfen Sie auf der Seite Step 6: Review Ihre Pipeline-Konfiguration und wählen Sie Create pipeline, um die Pipeline zu erstellen.



12. Nachdem die Pipeline erfolgreich ausgeführt wurde, wählen Sie Details anzeigen, um die Protokolle der Aktion und die Ausgabe der verwalteten Rechenaktion anzuzeigen.


Logs
Summary
Input
Output
Showing the last 38 lines of the build log. View entire log
Show previous logs
1 [2025/02/13 00:15:04.521] Describing tag, tag key = Name. tag value = my-instances.
2 [2025/02/13 00:15:04.784] Found 2 instances: i-0145 , i-0586
3 [2025/02/13 00:15:04.873] Processing deploy event BLOCK_TRAFFIC on instances: i-
0145
4 [2025/02/13 00:15:04.891] Skipping deploy event BLOCK_TRAFFIC on instances: i-0145
as not specified in action configuration
5 [2025/02/13 00:15:04.918] Processing deploy event DOWNLOAD on instances: i-0145
6 [2025/02/13 00:15:05.093] Executing commands on instances i-0145 , SSM command id
9d57dace- , commands: mkdir -p /tmp/codepipeline/53
7 aws s3api get-object --bucket codepipeline-us-east-1-2938 --key
rbtest/SourceArti/rwfBtWb /tmp/codepipeline/530
8 unzip -o /tmp/codepipeline/530398
/tmp/codepipeline/530
9 rm /tmp/codepipeline/5303985c-fc99-4
10 [2025/02/13 00:15:38.340] Deploy event DOWNLOAD succeeded on instances: i-0145
11 [2025/02/13 00:15:38.397] Processing deploy event BEFORE_DEPLOY on instances: i-
0145
12 [2025/02/13 00:15:38.412] Skipping deploy event BEFORE_DEPLOY on instances: i-
0145 as not specified in action configuration
13 [2025/02/13 00:15:38.436] Processing deploy event DEPLOY on instances: i-0145
14 [2025/02/13 00:15:38.523] Executing commands on instances i-0145 , SSM command id
6c7d0e01- , commands: mkdir -p /home/ec2-user/testhelloworld
15 cp -r /tmp/codepipeline/530 /* /home/ec2-user/testhelloworld
16 rm -rf /tmp/codepipeline/53 /*
17 [2025/02/13 00:16:13.616] Deploy event DEPLOY succeeded on instances: i-0145 .
18 [2025/02/13 00:16:13.673] Processing deploy event AFTER_DEPLOY on instances: i-
0145



Schritt 5: Testen Sie Ihre Pipeline

Ihre Pipeline sollte alles enthalten, was Sie für eine durchgängige native AWS kontinuierliche Bereitstellung benötigen. 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 Ihre Aktion stellt das Skript auf Ihren Instances bereit.
5. Weitere Informationen zur Problembehandlung finden Sie unterDie EC2-Bereitstellungsaktion schlägt fehl und es wird die Fehlermeldung `Keine solche Datei` angezeigt.

Tutorial: Erstellen Sie ein Docker-Image und übertragen Sie es auf Amazon ECR mit CodePipeline (Typ V2)

Dieses Tutorial hilft Ihnen dabei, eine Build-Aktion zu erstellen CodePipeline , die Ihr Docker-Image nach einer Änderung Ihres Quellcodes ausführt und an Amazon ECR überträgt. In diesem Tutorial erfahren Sie auch, wie Sie eine Amazon ECS-Bereitstellungsaktion hinzufügen, mit der Ihr gepushtes Image bereitgestellt wird.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline in der Konsole wird ein S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

> **Note**
>
> Dieses Tutorial behandelt die ECRBuildAndPublish Build-Aktion für eine CodePipeline Pipeline mit einem GitHub Quell-Repository und einer Amazon ECS-Standardaktion für die Bereitstellung in einem Amazon ECS-Cluster. Ein Tutorial, das eine Pipeline mit einem ECR-Image-Repository als Quelle für eine Amazon ECS-Aktion zur CodeDeploy blue/green Bereitstellung verwendet CodePipeline, finden Sie unterTutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-CodeDeploy.

> **Important**
>
> Diese Aktion verwendet CodePipeline Managed CodeBuild Compute, um Befehle in einer Build-Umgebung auszuführen. Für die Ausführung der Befehlsaktion fallen separate Gebühren in an AWS CodeBuild.

Voraussetzungen

Es müssen bestimmte Ressourcen zur Verfügung stehen, damit Sie mithilfe dieses Tutorials eine CD-Pipeline erstellen können. Zum Einstieg benötigen Sie Folgendes:

> **Note**
>
> Alle diese Ressourcen sollten in derselben AWS Region erstellt werden.

- Ein Quellcodeverwaltungs-Repository (dieses Tutorial verwendet GitHub), dem Sie für dieses Tutorial Folgendes hinzufügen werden:
  - In Schritt 1 fügen Sie Ihrem Quell-Repository ein Beispiel-Dockerfile als Eingabeartefakt für die ECRBuildAndPublish Build-Aktion hinzu. CodePipeline
  - In Schritt 2 fügen Sie Ihrem Quell-Repository eine Beispieldatei imagedefinitions.json als Voraussetzung für die Amazon ECS-Standardbereitstellungsaktion in hinzu. CodePipeline
- Ein Amazon ECR-Image-Repository, das ein Image enthält, das Sie aus Ihrem Dockerfile erstellt haben. Weitere Informationen finden Sie unter Erstellen eines Repositorys und Übertragen eines Images im Amazon Elastic Container Registry Benutzerhandbuch.
- Ein Amazon ECS-Cluster und -Service, die in derselben Region wie das Image-Repository erstellt wurden. Weitere Informationen finden Sie unter Creating a Cluster und 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: Fügen Sie Ihrem Quell-Repository ein Dockerfile hinzu

In diesem Tutorial wird die ECRBuildAndPublish Aktion verwendet, um Ihr Docker-Image zu erstellen und das Image an Amazon ECR zu übertragen. Die verwaltete Rechenaktion wird CodePipeline

verwendet CodeBuild , um die Befehle für die ECR-Anmeldung und den Image-Push auszuführen. Sie müssen Ihrem Quellcode-Repository keine `buildspec.yml` Datei hinzufügen, um zu erfahren, CodeBuild wie das geht. Für dieses Beispiel stellen Sie das Dockerfile in Ihrem Repository nur wie folgt bereit.

Fügen Sie diesen Beispieltext ein, um Ihre `Dockerfile` Datei zu erstellen. Dieses Dockerfile-Beispiel entspricht dem Beispiel, das in den ECR-Image-Anweisungen in den Voraussetzungen verwendet wird.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

Um eine hinzuzufügen **`Dockerfile`** Datei zu deinem Quell-Repository

1. Öffnen Sie einen Texteditor und kopieren Sie dann die obige Docker-Datei und fügen Sie sie in eine neue Datei ein.
2. Führen Sie einen Commit und eine Push-Übertragung Ihrer `Dockerfile`-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 Dockerfile."
```

c. Führen Sie eine Push-Übertragung für das Commit durch.

```
git push
```

Stellen Sie sicher, dass Sie die Datei auf der Stammebene Ihres Repositorys platzieren.

```
/ Dockerfile
```

Schritt 2: Füge eine imagedefinitions.json-Datei zu deinem Quell-Repository hinzu

In diesem Tutorial wird die Amazon ECS-Standardbereitstellungsaktion verwendet CodePipeline , um Ihren Container in Ihrem Amazon ECS-Cluster bereitzustellen. Für die Amazon ECS-Standardbereitstellungsaktion ist eine imagedefinitions.json-Datei erforderlich, die Ihren Image-Namen und URI enthält. Weitere Informationen zur imagedefinitions.json-Datei finden Sie unter. [imagedefinitions.json-Datei für Amazon ECS-Standardbereitstellungsaktionen]

Fügen Sie diesen Beispieltext ein, um Ihre Datei zu erstellen. `imagedefinitions.json` Verwenden Sie den Namen in Ihrem Dockerfile, z. B.`hello-world`, und verwenden Sie die URI aus Ihrem Amazon ECR-Repository, in dem das Bild gespeichert ist.

```
[
  {
    "name": "hello-world",
    "imageUri": "ACCOUNT-ID.dkr.ecr.us-east-1.amazonaws.com/actions/image-repo"
  }
]
```

Um eine hinzuzufügen **`imagedefinitions.json`** Datei in dein Quell-Repository

1. Öffnen Sie einen Texteditor und kopieren Sie dann das obige Beispiel und fügen Sie es in eine neue Datei ein.
2. Führen Sie einen Commit und eine Push-Übertragung Ihrer `imagedefinitions.json`-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 imagedefinitions.json."
```

c. Führen Sie eine Push-Übertragung für das Commit durch.

```
git push
```

Stellen Sie sicher, dass Sie die Datei auf der Stammebene Ihres Repositorys platzieren.

```
/ imagedefinitions.json
```

Schritt 3: Deine Pipeline erstellen

Verwenden Sie den CodePipeline Assistenten, um Ihre Pipeline-Stufen zu erstellen und Ihr Quell-Repository zu verbinden.

So erstellen Sie Ihre Pipeline

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**MyPipeline**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

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. Fügen Sie auf der Seite Schritt 3: Quellstufe hinzufügen eine Quellstufe hinzu:

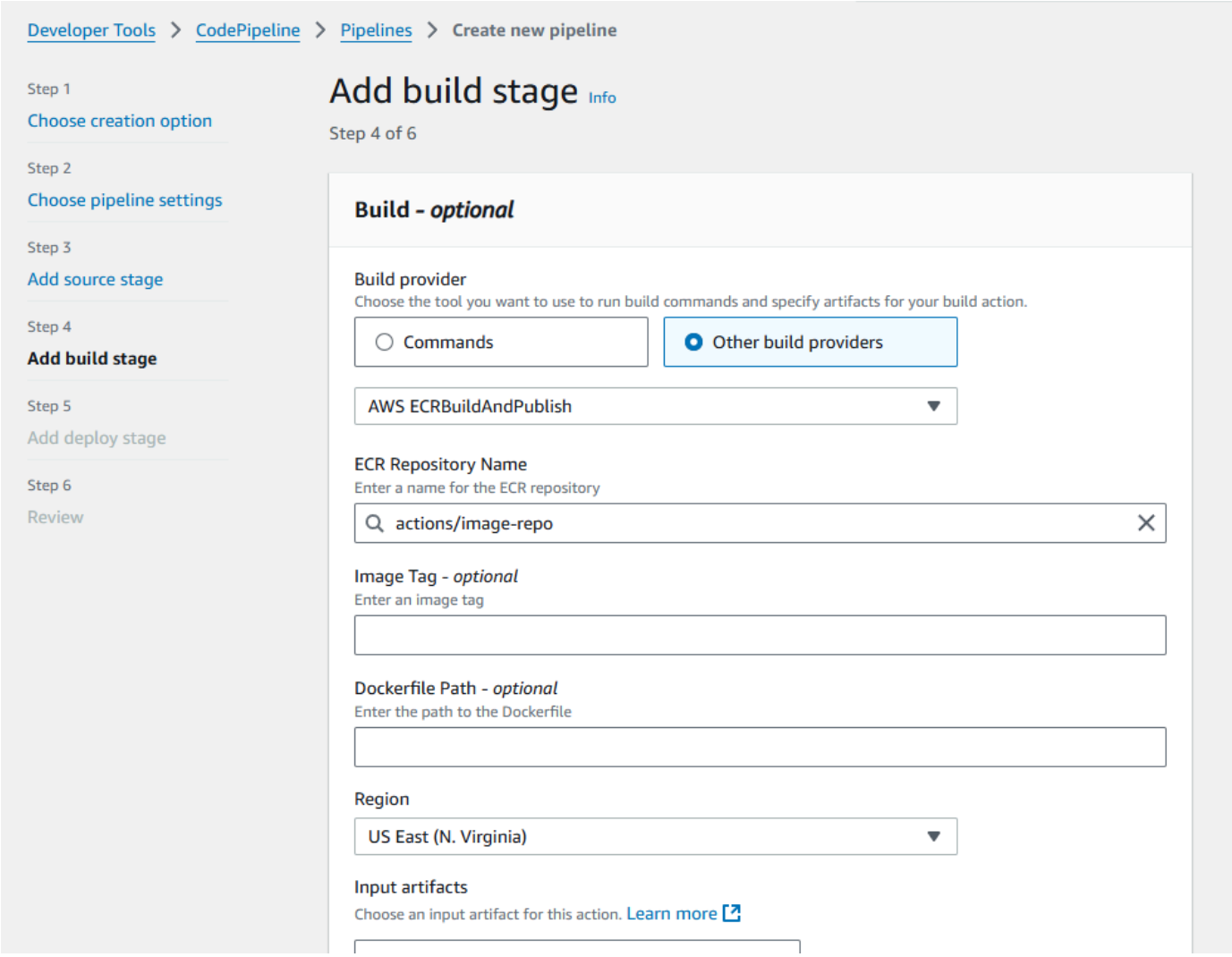
   a. Wählen Sie unter Quellanbieter die Option GitHub (über GitHub App) aus.

   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 im Feld Repository name (Repositoryname) den Namen Ihres GitHub - Repositorys aus.

   d. Wählen Sie unter Standard-Branch 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 Standard-Branch verwendet.

   Wählen Sie Weiter aus.
9. Wählen Sie auf der Seite Schritt 4: Build-Phase hinzufügen die Option Andere Build-Anbieter aus, und wählen Sie dann ECRBuildAndPublish.



a. Wählen Sie für den Namen des ECR-Repositorys Ihr Image-Repository aus.

b. Wählen Sie Weiter aus.

10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

    Wählen Sie Weiter aus.

11. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus. Im folgenden Schritt fügen Sie die ECS-Aktion hinzu.

12. Überprüfen Sie auf der Seite Schritt 7: Überprüfen Ihre Pipeline-Konfiguration und wählen Sie Create pipeline aus, um die Pipeline zu erstellen.

13. Bearbeiten Sie Ihre Pipeline, um die Amazon ECS-Bereitstellungsaktion zu Ihrer Pipeline hinzuzufügen:

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. **`Deploy`**.

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

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

e. Wählen Sie unter Action Provider die Option Amazon ECS aus. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region.

f. Wählen Sie unter Eingabeartefakte das Eingabeartefakt aus der Quellphase aus, z. B. `SourceArtifact`

g. Wählen Sie als Clustername den Amazon ECS-Cluster aus, in dem Ihr Service ausgeführt wird.

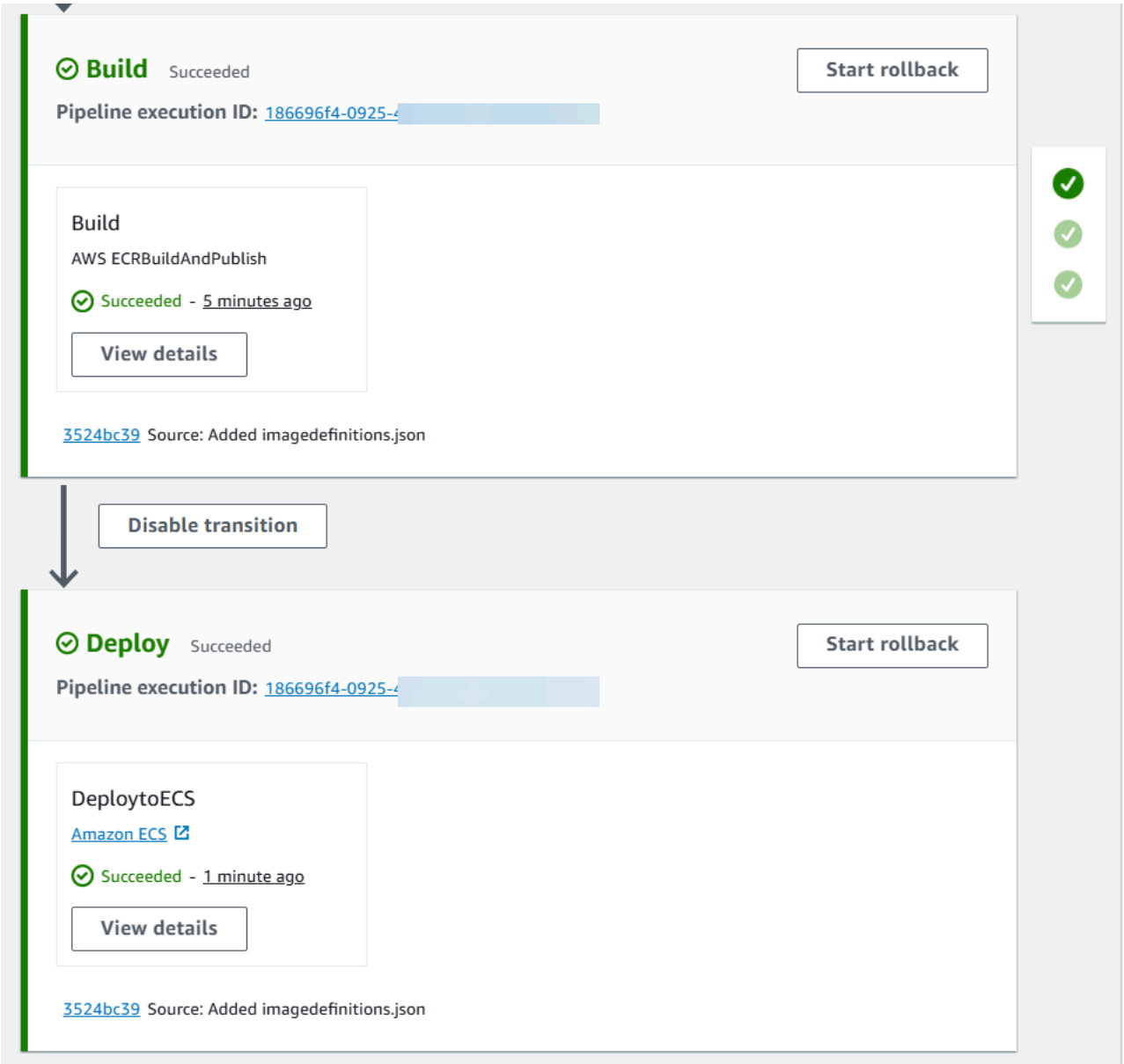
h. Wählen Sie für den Servicenamen den Service aus, der aktualisiert werden soll.

i. Wählen Sie Speichern.

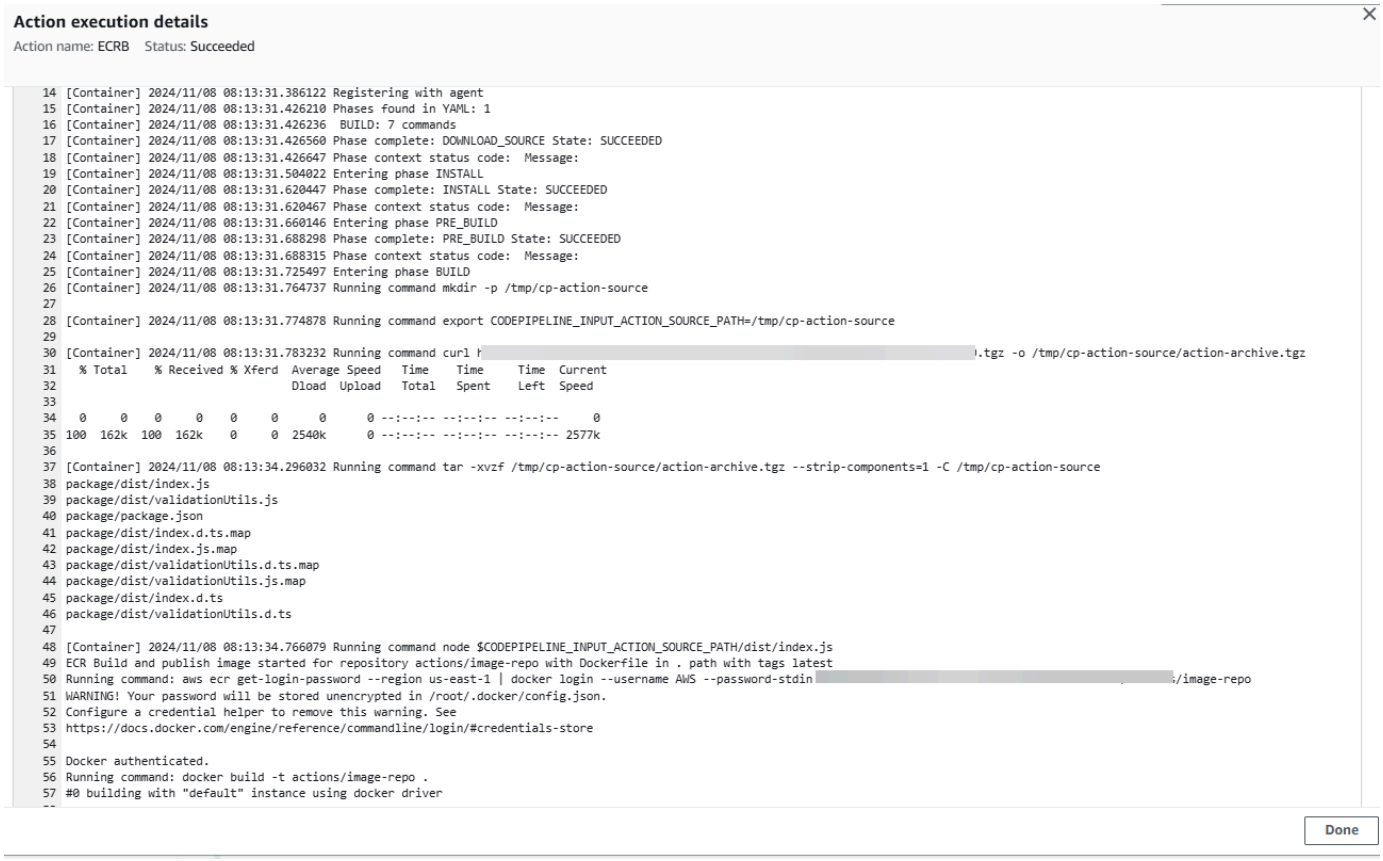
j. Wählen Sie in der Phase, die Sie bearbeiten, Fertig. Wählen Sie im AWS CodePipeline - Fenster Save (Speichern) und dann in der Warnmeldung Save (Speichern).

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

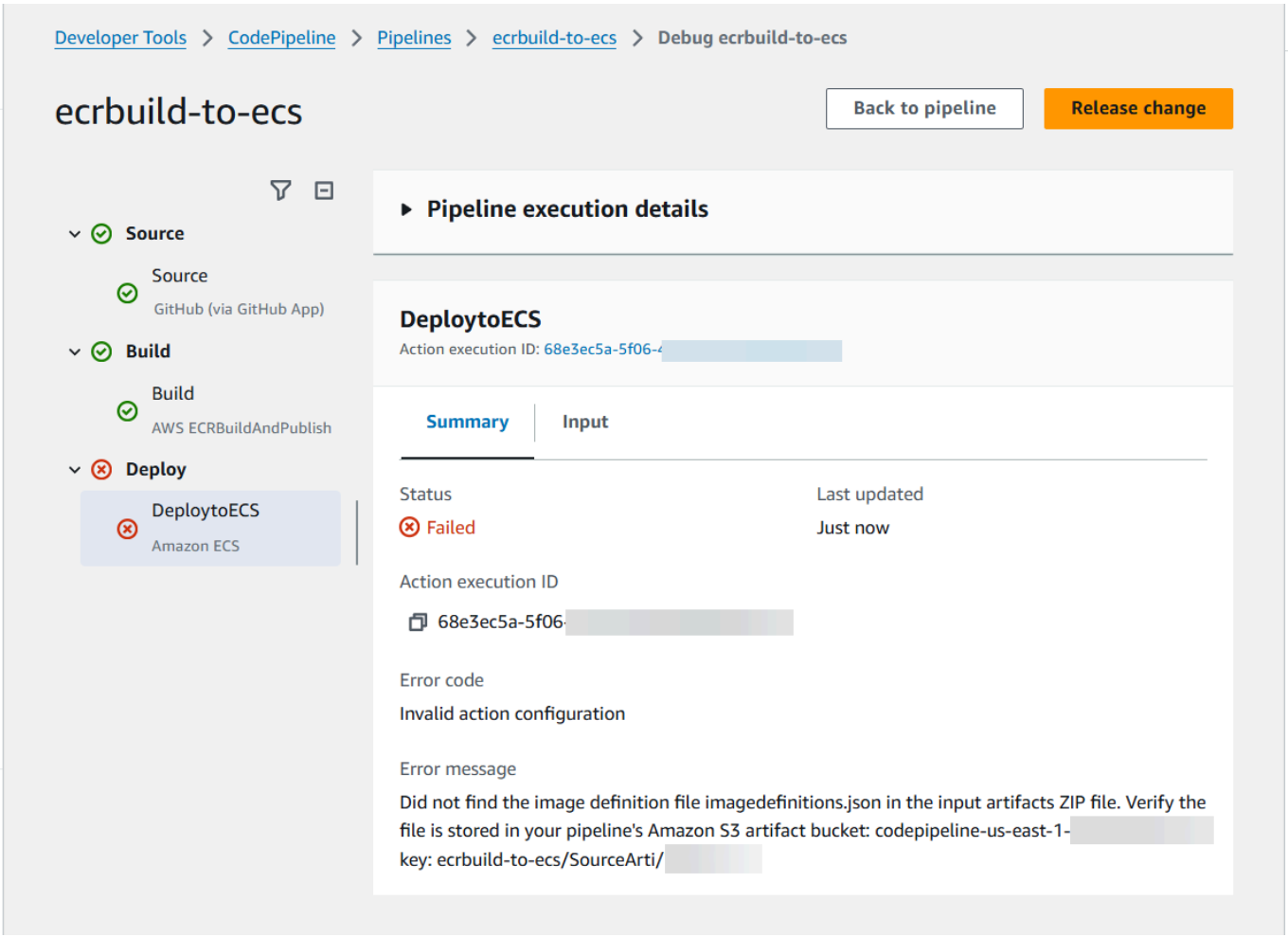
14. Nachdem die Pipeline ausgeführt wurde, sehen Sie sich die Struktur und den Status der Pipeline an.



15. Nachdem die Pipeline erfolgreich ausgeführt wurde, wählen Sie Details anzeigen, um die Protokolle der Aktion und die Ausgabe der verwalteten Rechenaktion anzuzeigen.



16. Beheben Sie alle fehlgeschlagenen Aktionen. Beispielsweise kann die ECS-Bereitstellungsaktion fehlschlagen, wenn sich die Datei imagedefinitions.json nicht im Quell-Repository befindet. Das Folgende ist ein Beispiel für die Fehlermeldung, die angezeigt wird, wenn die Datei imagedefinitions.json fehlt.



Schritt 4: Testen Ihrer Pipeline

Ihre Pipeline sollte alles enthalten, was Sie für den Betrieb einer durchgängigen nativen AWS kontinuierlichen Bereitstellung benötigen. 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 Ihre Aktion überträgt das Docker-Image auf ECR, das aufgrund Ihrer Codeänderung erstellt wurde.

Tutorial: Bereitstellung auf Amazon EKS mit CodePipeline

Dieses Tutorial hilft Ihnen beim Erstellen einer Bereitstellungsaktion CodePipeline , bei der Ihr Code in einem Cluster bereitgestellt wird, den Sie in Amazon EKS konfiguriert haben.

Die EKS-Aktion unterstützt sowohl öffentliche als auch private EKS-Cluster. Private Cluster sind der von EKS empfohlene Typ; beide Typen werden jedoch unterstützt.

> **Note**
>
> Im Rahmen der Erstellung einer Pipeline in der Konsole wird ein S3-Artefakt-Bucket von CodePipeline für Artefakte verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

> **Note**
>
> Diese Aktion verwendet CodePipeline Managed CodeBuild Compute, um Befehle in einer Build-Umgebung auszuführen. Für die Ausführung der Befehlsaktion fallen separate Gebühren in an AWS CodeBuild.

> **Note**
>
> Die Aktion „EKSBereitstellen“ ist nur für Pipelines vom Typ V2 verfügbar.

Voraussetzungen

Es müssen bestimmte Ressourcen zur Verfügung stehen, damit Sie mithilfe dieses Tutorials eine CD-Pipeline erstellen können. Zum Einstieg benötigen Sie Folgendes:

> **Note**
>
> Alle diese Ressourcen sollten in derselben AWS Region erstellt werden.

- Ein Quellcodeverwaltungs-Repository (dieses Tutorial verwendet GitHub), dem Sie eine `deployment.yaml` Beispieldatei hinzufügen werden.
- Sie müssen eine vorhandene CodePipeline Servicerolle verwenden, die Sie mit den Schritt 3: Aktualisieren Sie die CodePipeline Servicerollenrichtlinie in IAM unten aufgeführten Berechtigungen für diese Aktion aktualisieren werden. Die erforderlichen Berechtigungen hängen vom Clustertyp ab, den Sie erstellen. Weitere Informationen finden Sie unter Berechtigungen für Richtlinien zur Servicerolle.
- Ein funktionierendes Image und ein Repository-Tag, die Sie an ECR oder Ihr Image-Repository übertragen haben.

Nachdem Sie diese Voraussetzungen erfüllt haben, können Sie mit dem Tutorial fortfahren und Ihre CD-Pipeline erstellen.

Schritt 1: (Optional) Erstellen Sie einen Cluster in Amazon EKS

Sie können wählen, ob Sie einen EKS-Cluster mit einem öffentlichen oder privaten Endpunkt erstellen möchten.

In den folgenden Schritten erstellen Sie einen öffentlichen oder privaten Cluster in EKS. Dieser Schritt ist optional, wenn Sie Ihren Cluster bereits erstellt haben.

Erstellen Sie einen öffentlichen Cluster in Amazon EKS

In diesem Schritt erstellen Sie einen Cluster in EKS.

Erstellen Sie einen öffentlichen Cluster

1. Öffnen Sie die EKS-Konsole und wählen Sie dann Create cluster aus.
2. Benennen Sie Ihren Cluster im Feld Name. Wählen Sie Next.
3. Wählen Sie Erstellen aus.

Erstellen Sie einen privaten Cluster in Amazon EKS

Falls Sie sich dafür entscheiden, einen Cluster mit einem privaten Endpunkt zu erstellen, stellen Sie sicher, dass Sie nur die privaten Subnetze anhängen und dass sie über eine Internetverbindung verfügen.

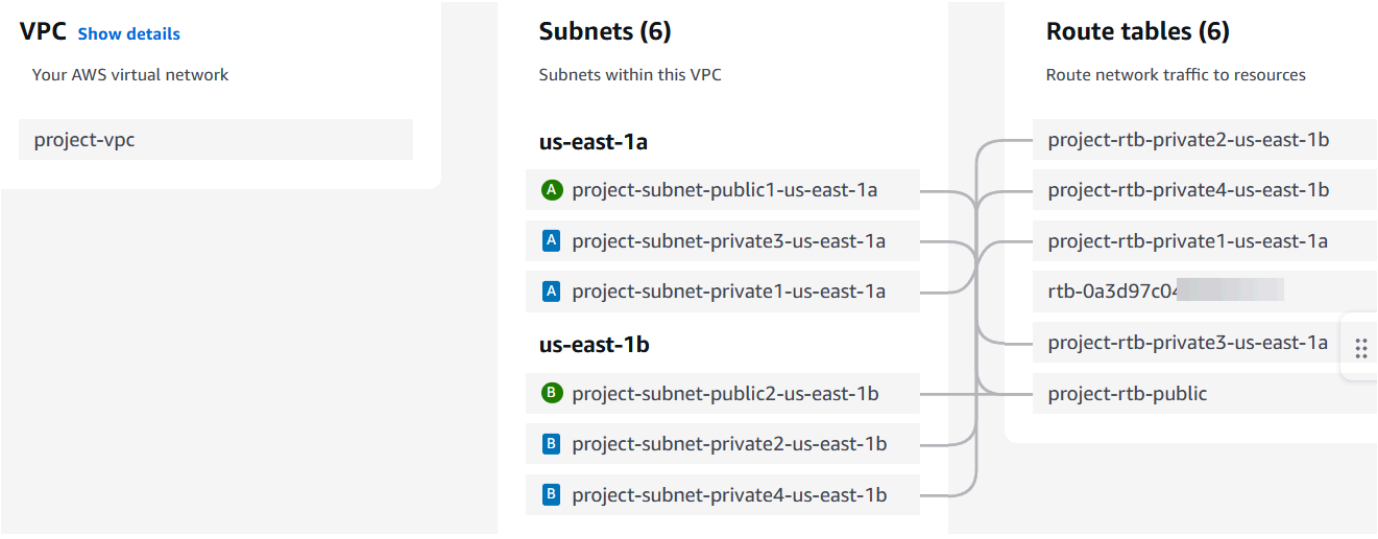
Folgen Sie den nächsten fünf Unterschritten, um einen Cluster mit einem privaten Endpunkt zu erstellen.

Erstellen Sie eine VPC in der Konsole

1. Öffnen Sie die VPC-Konsole und wählen Sie dann Create VPC aus.
2. Wählen Sie unter VPC-Einstellungen die Option VPC und mehr aus.
3. Wählen Sie, ob Sie ein öffentliches und 4 private Subnetze erstellen möchten. Wählen Sie VPC erstellen aus.
4. Wählen Sie auf der Seite „Subnetze“ die Option Privat aus.

Ermitteln Sie die privaten Subnetze in Ihrer VPC

1. Navigieren Sie zu Ihrer VPC und wählen Sie die VPC-ID aus, um die VPC-Detailseite zu öffnen.
2. Wählen Sie auf der VPC-Detailseite die Registerkarte Resource Map aus.
3. Sehen Sie sich das Diagramm an und notieren Sie sich Ihre privaten Subnetze. Die Subnetze werden mit Beschriftungen angezeigt, die auf den öffentlichen oder privaten Status hinweisen, und jedes Subnetz ist einer Routing-Tabelle zugeordnet.



Beachten Sie, dass ein privater Cluster alle privaten Subnetze enthält.

4. Erstellen Sie ein öffentliches Subnetz, um das NAT-Gateway zu hosten. Sie können nur jeweils ein Internet-Gateway nur für eine VPC anfügen.

Erstellen Sie ein NAT-Gateway im öffentlichen Subnetz

1. Erstellen Sie im öffentlichen Subnetz ein NAT-Gateway. Navigieren Sie zur VPC-Konsole und wählen Sie dann Internet-Gateways aus. Wählen Sie Create internet gateway (Internet-Gateway erstellen) aus.
2. Geben Sie im Feld Name einen Namen für Ihr Internet-Gateway ein. Wählen Sie Create internet gateway (Internet-Gateway erstellen) aus.

Aktualisieren Sie die Routing-Tabelle für das private Subnetz, um den Datenverkehr an das NAT-Gateway weiterzuleiten.

Fügen Sie das NAT-Gateway zu Ihren Routing-Tabellen für Ihre privaten Subnetze hinzu

1. Navigieren Sie zur VPC-Konsole und wählen Sie dann Subnets aus.
2. Wählen Sie für jedes private Subnetz dieses aus und wählen Sie dann auf der Detailseite die Routing-Tabelle für dieses Subnetz aus. Wählen Sie Routing-Tabelle bearbeiten aus.
3. Aktualisieren Sie die Routing-Tabelle für das private Subnetz, um den Internetverkehr zum NAT-Gateway weiterzuleiten. Wählen Sie Route hinzufügen aus. Wählen Sie das NAT-Gateway aus den Optionen aus, die Sie hinzufügen möchten. Wählen Sie das Internet-Gateway, das Sie erstellt haben.
4. Erstellen Sie für das öffentliche Subnetz eine benutzerdefinierte Routing-Tabelle. Stellen Sie sicher, dass die Network Access Control List (ACL) für Ihr öffentliches Subnetz eingehenden Datenverkehr aus dem privaten Subnetz zulässt.
5. Wählen Sie Änderungen speichern aus.

In diesem Schritt erstellen Sie einen Cluster in EKS.

Erstellen Sie einen privaten Cluster

1. Öffnen Sie die EKS-Konsole und wählen Sie dann Create cluster aus.
2. Benennen Sie Ihren Cluster im Feld Name. Wählen Sie Weiter aus.
3. Geben Sie Ihre VPC und andere Konfigurationsinformationen an. Wählen Sie Erstellen aus.

Ihr EKS-Cluster kann ein öffentlicher oder ein privater Cluster sein. Dieser Schritt gilt für Cluster, die NUR einen privaten Endpunkt haben. Stellen Sie sicher, dass Ihr Cluster privat ist.

Schritt 2: Konfigurieren Sie Ihren privaten Cluster in Amazon EKS

Dieser Schritt gilt nur, wenn Sie einen privaten Cluster erstellt haben. Dieser Schritt gilt für Cluster, die NUR einen privaten Endpunkt haben.

Konfigurieren Sie Ihren Cluster

1. Hängen Sie private Subnetze nur im EKS-Cluster auf der Registerkarte Netzwerk an. Hängen Sie die privaten Subnetze an, die im Abschnitt Ermitteln Sie die privaten Subnetze in Ihrer VPC unter erfasst sind. Schritt 1: (Optional) Erstellen Sie einen Cluster in Amazon EKS
2. Stellen Sie sicher, dass die privaten Subnetze Zugriff auf das Internet haben, da Artefakte aus dem S3-Artefakt-Bucket für Ihre Pipeline CodePipeline gespeichert und abgerufen werden.

Schritt 3: Aktualisieren Sie die CodePipeline Servicerollenrichtlinie in IAM

In diesem Schritt aktualisieren Sie eine vorhandene CodePipeline Servicerolle, z. B. mit den Berechtigungen**`cp-service-role`**, die für die Verbindung mit Ihrem Cluster erforderlich sind. CodePipeline Wenn Sie noch keine Rolle haben, erstellen Sie eine neue.

Aktualisieren Sie Ihre CodePipeline Servicerolle mit den folgenden Schritten.

So aktualisieren Sie Ihre CodePipeline Servicerollenrichtlinie

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
3. Schlagen Sie nach Ihrer CodePipeline Servicerolle, z. **`cp-service-role`** B.
4. Fügen Sie eine neue Inline-Richtlinie hinzu.
5. Geben Sie im Richtlinieneditor Folgendes ein.

   - Fügen Sie für einen öffentlichen Cluster die folgenden Berechtigungen hinzu.

     JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "EksClusterPolicy",
            "Effect": "Allow",
            "Action": "eks:DescribeCluster",
            "Resource": "arn:aws:eks:us-east-1:111122223333:cluster/my-
cluster"
        },
        {
            "Sid": "EksVpcClusterPolicy",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- Fügen Sie für einen privaten Cluster die folgenden Berechtigungen hinzu. Private Cluster benötigen, falls zutreffend, zusätzliche Berechtigungen für Ihre VPC.

  JSON

```
{
    "Version":"2012-10-17",

    "Statement": [
        {
            "Sid": "EksClusterPolicy",
            "Effect": "Allow",
            "Action": "eks:DescribeCluster",
            "Resource": "arn:aws:eks:us-east-1:111122223333:cluster/my-
cluster"
        },
        {
            "Sid": "EksVpcClusterPolicy",
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "ec2:Subnet": [
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-03ebd65daeEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0e377f6036EXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "ec2:Subnet": [
                        "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-03ebd65daeEXAMPLE",
                        "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-0e377f6036EXAMPLE",
```

```
                        "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-0db658ba1cEXAMPLE",
                        "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-0db658ba1cEXAMPLE"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteNetworkInterface",
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "ec2:Subnet": [
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-03ebd65daeEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0e377f6036EXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE"
                    ]
                }
            }
        }
    ]
}
```

6. Wählen Sie Richtlinie aktualisieren.

Schritt 4: Erstellen Sie einen Zugriffseintrag für die CodePipeline Servicerolle

In diesem Schritt erstellen Sie einen Zugriffseintrag in Ihrem Cluster, der die in Schritt 3 aktualisierte CodePipeline Servicerolle zusammen mit einer verwalteten Zugriffsrichtlinie hinzufügt.

1. Öffnen Sie die EKS-Konsole und navigieren Sie zu Ihrem Cluster.
2. Wählen Sie die Registerkarte Zugriff aus.
3. Wählen Sie unter IAM-Zugriffseinträge die Option Zugriffseintrag erstellen aus.

4. Geben Sie unter IAM-Principal-ARN die Rolle ein, die Sie gerade für die Aktion aktualisiert haben, z. B. **`cp-service-role`** Wählen Sie Weiter aus.
5. Wählen Sie auf der Seite Schritt 2: Zugriffsrichtlinie hinzufügen unter Richtlinienname die verwaltete Richtlinie für den Zugriff aus, z. B. `AmazonEKSClusterAdminPolicy` Wählen Sie Richtlinie hinzufügen aus. Wählen Sie Weiter aus.

   > **Note**
   >
   > Dies ist die Richtlinie, die die CodePipeline Aktion verwendet, um mit Kubernetes zu kommunizieren. Es hat sich bewährt, den Umfang der Berechtigungen in Ihrer Richtlinie mit den geringsten Rechten und nicht in der Verwaltungsrichtlinie einzuschränken, indem Sie stattdessen eine benutzerdefinierte Richtlinie anhängen.

6. Wählen Sie auf der Bewertungsseite die Option Erstellen aus.

Schritt 5: Erstellen Sie ein Quell-Repository und fügen Sie die Helm-Chart-Konfigurationsdateien hinzu

In diesem Schritt erstellen Sie eine für Ihre Aktion geeignete Konfigurationsdatei (Kubernetes-Manifestdateien oder Helm-Diagramm) und speichern die Konfigurationsdatei in Ihrem Quell-Repository. Verwenden Sie die entsprechende Datei für Ihre Konfiguration. Für weitere Informationen siehe https://kubernetes.io/docs/reference/kubectl/quick-reference/ oder https://helm.sh/docs/topics/charts/.

- Verwenden Sie für Kubernetes eine Manifestdatei.
- Verwenden Sie für Helm ein Helm-Diagramm.

1. Erstellen oder verwenden Sie ein vorhandenes GitHub Repository.
2. Erstellen Sie in Ihrem Repository eine neue Struktur für Ihre Helm-Diagrammdateien, wie im folgenden Beispiel gezeigt.

```
mychart
|-- Chart.yaml
|-- charts
|-- templates
|   |-- NOTES.txt
|   |-- _helpers.tpl
```

```
|   |-- deployment.yaml
|   |-- ingress.yaml
|   `-- service.yaml
`-- values.yaml
```

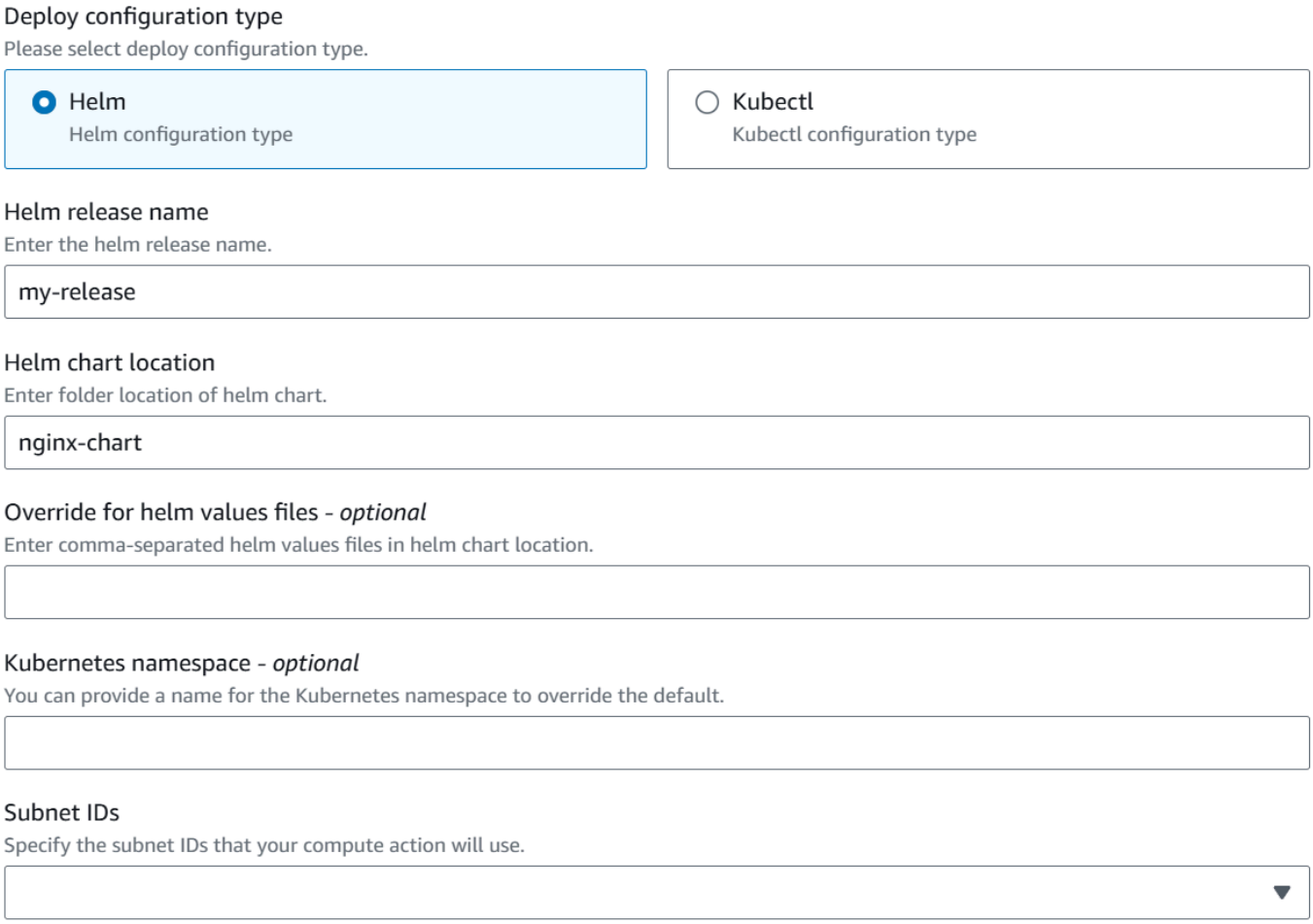
3. Fügen Sie die Datei zur Stammebene Ihres Repositorys hinzu.

Schritt 6: Deine Pipeline erstellen

Verwenden Sie den CodePipeline Assistenten, um Ihre Pipeline-Stufen zu erstellen und Ihr Quell-Repository zu verbinden.

So erstellen Sie Ihre Pipeline

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. In Schritt 2: Wählen Sie die Pipeline-Einstellungen und geben Sie im Feld Pipeline-Name den Wert ein**`MyEKSPipeline`**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
6. Wählen Sie unter Servicerolle die Servicerolle aus, die Sie in Schritt 3 aktualisiert haben.
7. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
8. Wählen Sie auf der Seite Schritt 3: Quellphase hinzufügen für Quellanbieter die Option , eine Verbindung zu Ihrem GitHub Repository herzustellen.
9. Wähle auf der Seite Schritt 4: Build Stage hinzufügen die Option Überspringen aus.
10. Wählen Sie auf der Seite Schritt 5: Bereitstellungsphase hinzufügen die Option Amazon EKS aus.

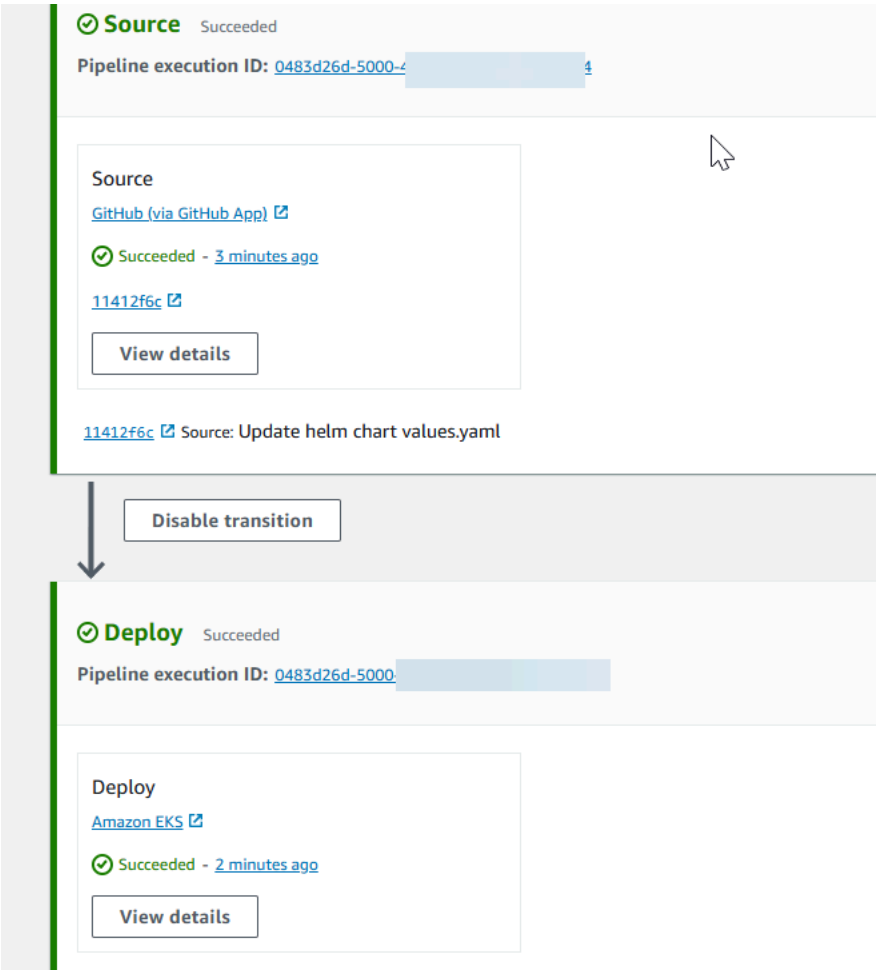


a. Wählen Sie unter Konfigurationstyp bereitstellen die Option Helm aus.

b. Geben Sie im Feld Position des Helm-Diagramms den Namen der Version ein, z. `my-release` B. Geben Sie für den Speicherort des Helm-Diagramms den Pfad für Ihre Helm-Diagrammdateien ein, z. `mychart` B.

c. Wählen Sie Weiter aus.

11. Überprüfen Sie auf der Seite Step 6: Review Ihre Pipeline-Konfiguration und wählen Sie Create pipeline, um die Pipeline zu erstellen.



12. Nachdem die Pipeline erfolgreich ausgeführt wurde, wählen Sie Details anzeigen, um die Protokolle der Aktion und die Aktionsausgabe anzuzeigen.

Tutorial: Erstellen Sie eine Pipeline, die Befehle mit Compute ausführt (Typ V2)

In diesem Tutorial konfigurieren Sie eine Pipeline, die bereitgestellte Buildbefehle mithilfe der Aktion Befehle in einer Buildphase kontinuierlich ausführt. Weitere Informationen zur Aktion „Befehle“ finden Sie unterAktionsreferenz für Befehle.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem

anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Voraussetzungen

Sie müssen bereits über Folgendes verfügen:

- Ein GitHub Repositorium. Sie können das GitHub Repository verwenden, in dem Sie es erstellt habenTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

Schritt 1: Erstellen Sie Quelldateien und übertragen Sie sie in Ihr GitHub Repository

In diesem Abschnitt erstellen Sie Ihre Beispiel-Quelldateien und übertragen sie in das Repository, das die Pipeline für Ihre Quellphase verwendet. Für dieses Beispiel erstellen und übertragen Sie Folgendes:

- Die Datei `README.txt`.

Um Quelldateien zu erstellen

1. Erstellen Sie eine Datei mit dem folgenden Text:

   ```
   Sample readme file
   ```

2. Speichern Sie die Datei als `README.txt`.

Um Dateien in dein GitHub Repository zu übertragen

1. Übertragen Sie die Dateien per Push an Ihr -Repository oder laden Sie sie 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:

   ```
   README.txt
   ```

2. So verwenden Sie die Git-Befehlszeile aus einem geklonten Repository auf Ihrem lokalen Computer:

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

   ```
   git add -A
   ```

   b. Führe den folgenden Befehl aus, um die Dateien mit einer Commit-Nachricht zu übertragen.

   ```
   git commit -m "Added source files"
   ```

   c. Führen Sie den folgenden Befehl aus, um die Dateien aus Ihrem lokalen Repository in Ihr -Repository zu übertragen:

   ```
   git push
   ```

Schritt 2: Erstellen der Pipeline

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

- Ein Quell-Stagingbereich mit einer Aktion GitHub (per GitHub App) für das Repository, in dem die Quelldateien gespeichert sind.
- Eine Build-Phase mit der Aktion „Befehle“.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS-Managementkonsole 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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**MyCommandsPipeline**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können.

Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

6. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.

> **Note**
>
> Wenn Sie eine vorhandene Servicerolle verwenden, müssen Sie die folgenden Berechtigungen für die Servicerolle hinzufügen, um die Aktion Befehle verwenden zu können. Beschränken Sie die Berechtigungen auf die Ebene der Pipeline-Ressourcen, indem Sie in der Richtlinienerklärung zur Servicerolle ressourcenbasierte Berechtigungen verwenden. Weitere Informationen finden Sie im Richtlinienbeispiel unter. Dienstrolle, Richtlinien, Berechtigungen
>
> - Protokolle: CreateLogGroup
> - Protokolle: CreateLogStream
> - Protokolle: PutLogEvents

7. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
8. Fügen Sie auf der Seite Schritt 3: Quellstufe hinzufügen eine Quellstufe hinzu:

   a. Wählen Sie unter Quellanbieter die Option GitHub (über GitHub App) 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 im Feld Repository name (Repositoryname) den Namen Ihres GitHub.com - Repositorys aus.

   d. Wählen Sie unter Standard-Branch 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 Standard-Branch verwendet. Optional können Sie auch Webhooks mit Filterung (Trigger) angeben. Weitere Informationen finden Sie unter Automatisieren Sie das Starten von Pipelines mithilfe von Triggern und Filtern.

Wählen Sie Weiter aus.

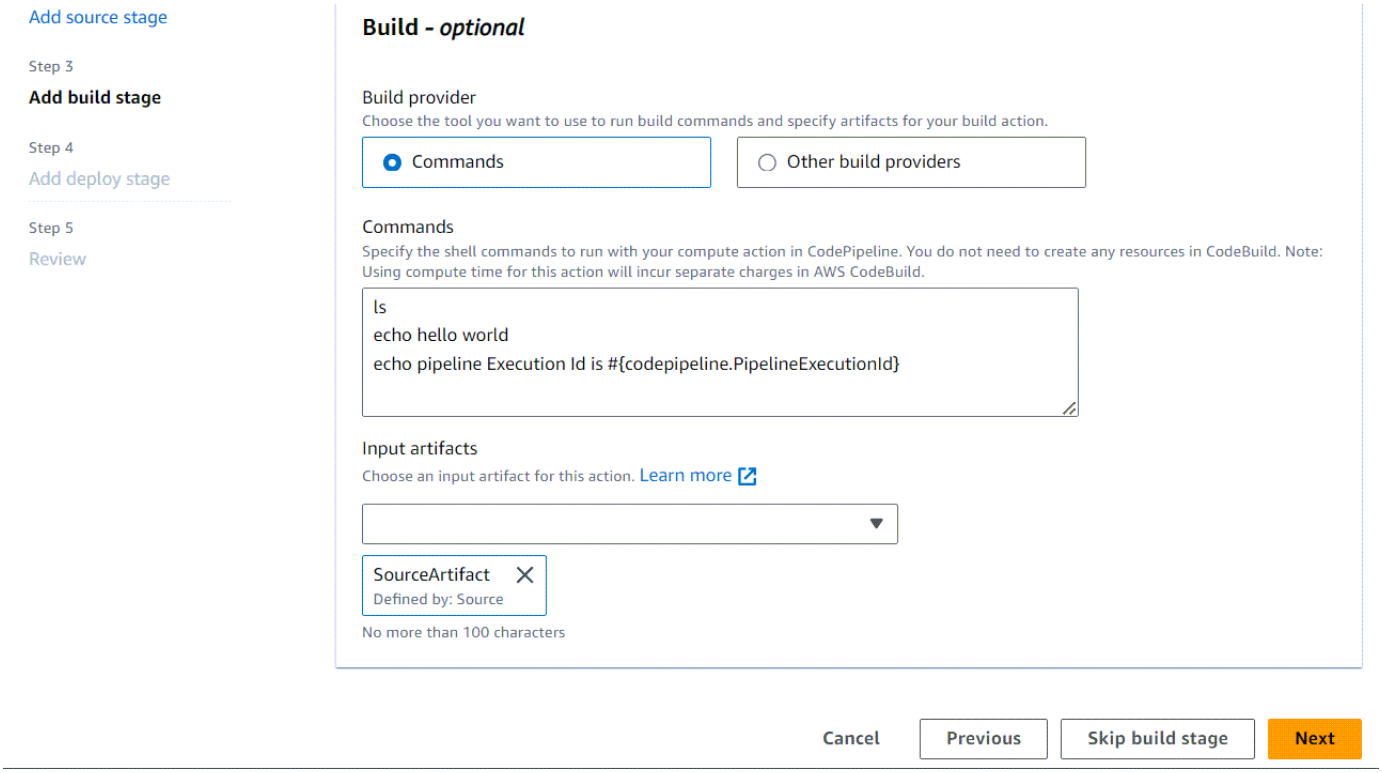
9. Wählen Sie in Schritt 4: Build-Phase hinzufügen die Option Befehle aus.

> **Note**
>
> Für die Ausführung der Aktion „Befehle“ fallen separate Gebühren an AWS CodeBuild.

Geben Sie die folgenden Befehle ein.

```
ls
echo hello world
cat README.txt
echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}
```

Wählen Sie Weiter aus.



10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

Wählen Sie Weiter aus.

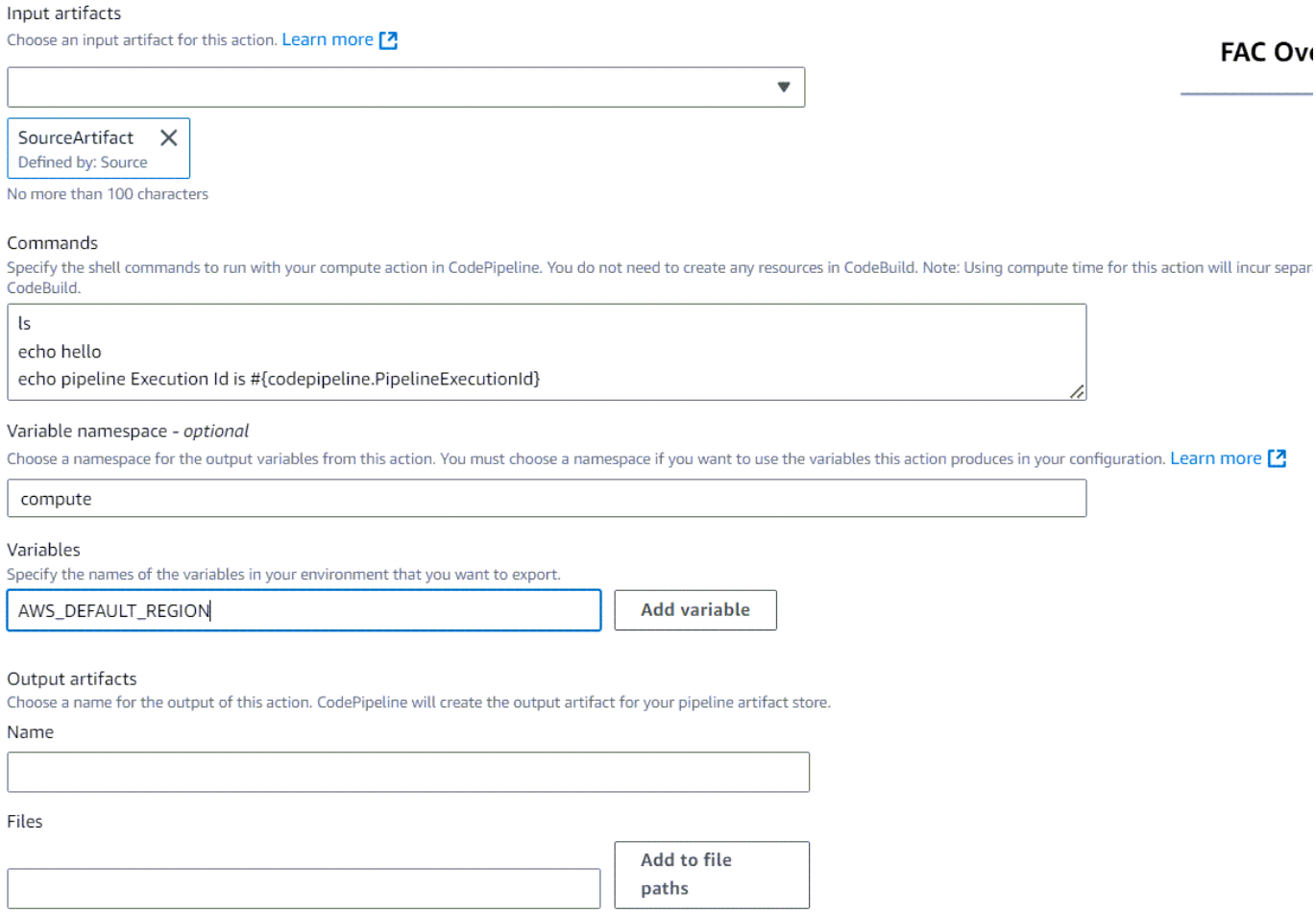
11. Wählen Sie in Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken.

    Wählen Sie Weiter aus.

12. In Schritt 7: Überprüfen, überprüfen Sie die Informationen und wählen Sie dann „Pipeline erstellen“ aus.

13. Als letzten Schritt zur Erstellung Ihrer Aktion fügen Sie der Aktion eine Umgebungsvariable hinzu, die zu einer Ausgabevariablen für die Aktion führt. Wählen Sie in der Aktion Befehle die Option Bearbeiten aus. Geben Sie auf dem Bearbeitungsbildschirm einen Variablennamespace für Ihre Aktion an, indem Sie ihn `compute` in das Feld Variablennamespace eingeben.

    Fügen Sie die CodeBuild Ausgabevariable `AWS_Default_Region` hinzu und wählen Sie dann Variable hinzufügen.



Schritt 3: Führen Sie Ihre Pipeline aus und überprüfen Sie die Build-Befehle

Geben Sie eine Änderung frei, um Ihre Pipeline auszuführen. Vergewissern Sie sich, dass die Build-Befehle ausgeführt wurden, indem Sie sich den Ausführungsverlauf, die Build-Logs und die Ausgabevariablen ansehen.

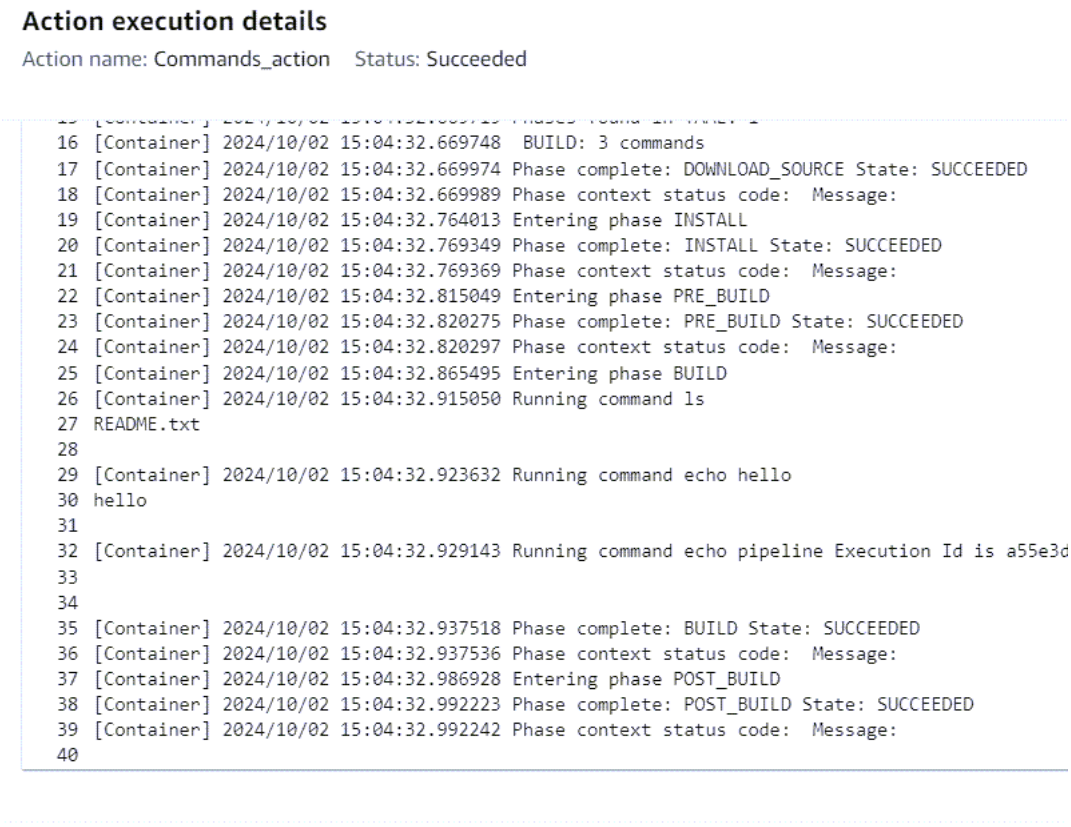
So zeigen Sie Aktionsprotokolle und Ausgabevariablen an

1. Nachdem die Pipeline erfolgreich ausgeführt wurde, können Sie die Protokolle und die Ausgabe für die Aktion einsehen.
2. Um die Ausgabevariablen für die Aktion anzuzeigen, wählen Sie Verlauf und dann Zeitleiste aus.

   Zeigen Sie die Ausgabevariable an, die der Aktion hinzugefügt wurde. Die Ausgabe für die Aktion „Befehle“ zeigt die Ausgabevariable, die in die Aktion Region aufgelöst wurde.



3. Um die Protokolle für die Aktion anzuzeigen, wählen Sie Details zur erfolgreichen Befehlsaktion anzeigen. Sehen Sie sich die Protokolle für die Aktion „Befehle“ an.



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 Hinweise zum Filtern mit Glob-Mustern finden Sie unterArbeiten mit Glob-Mustern in der Syntax.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

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) oder Europa (Zürich) nicht verfügbar. Informationen zu anderen verfügbaren Aktionen finden Sie unterProdukt- und Serviceintegrationen mit CodePipeline. Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie im Hinweis unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen.

Themen

- Voraussetzungen
- Schritt 1: Öffne CloudShell und klone dein Repository
- Schritt 2: Erstellen Sie eine Pipeline, die auf Git-Tags ausgelöst wird
- Schritt 3: Tagge deine Commits zur Veröffentlichung
- Schritt 4: Änderungen veröffentlichen und Protokolle anzeigen

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Erstellen Sie ein GitHub Repository mit Ihrem GitHub Konto.
- Halten Sie Ihre GitHub Anmeldeinformationen bereit. Wenn Sie das verwenden AWS-Managementkonsole , 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-Managementkonsole.

2. Wählen Sie in der oberen Navigationsleiste das AWS Symbol aus. Die Hauptseite der AWS-Managementkonsole Displays.
3. Wählen Sie in der oberen Navigationsleiste das AWS CloudShell Symbol aus. CloudShell öffnet sich. 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. Wähle Code und dann HTTPS. 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 die `Password` Eingabe müssen Sie ein vom Benutzer erstelltes Token anstelle Ihres Kontokennworts verwenden.

Schritt 2: Erstellen Sie eine Pipeline, die auf Git-Tags ausgelöst wird

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

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

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der CodePipeline Konsole unter an https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.

4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyGitHubTagsPipeline`**.
5. Behalten Sie unter Pipeline-Typ die Standardauswahl bei V2. Die Pipeline-Typen unterscheiden sich in ihren Eigenschaften und ihrem Preis. Weitere Informationen finden Sie unter Pipeline-Typen.
6. 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.

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 Schritt 3: Quellstufe hinzufügen eine Quellstufe hinzu:
   a. Wählen Sie unter Quellanbieter die Option GitHub (über GitHub App) 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 im Feld Repository name (Repositoryname) den Namen Ihres GitHub-Repositorys aus.

d. Wählen Sie unter Standard-Branch 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 Standard-Branch verwendet.

e. Wählen Sie unter Webhook-Ereignisse unter Filtertyp die Option Tags aus.

Geben `release*` Sie in das Feld Tags oder Muster den Wert ein.

> **Important**
>
> Pipelines, die mit Git-Tags vom Typ Trigger 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.

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 die Option /standard:5.0. aws/codebuild

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

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

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 Befehle erstellen 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. Dies kehrt zur CodePipeline Konsole zurück und erstellt 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.

10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

    Wählen Sie Weiter aus.

11. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus, und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken. Wählen Sie Weiter aus.

12. Wählen Sie auf Schritt 7: Überprüfen die Option Pipeline erstellen aus.

Schritt 3: Tagge 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 taggst du einen Commit mit dem `release-1` Tag. Jeder Commit in einem Git-Repository muss ein eindeutiges Git-Tag haben. Wenn du den Commit auswählst und ihn taggst, kannst du auf diese Weise Änderungen aus verschiedenen Branches in deine Pipeline-Bereitstellung integrieren. Beachten Sie, dass der Tag-Name release nicht für das Konzept einer Veröffentlichung in gilt GitHub.

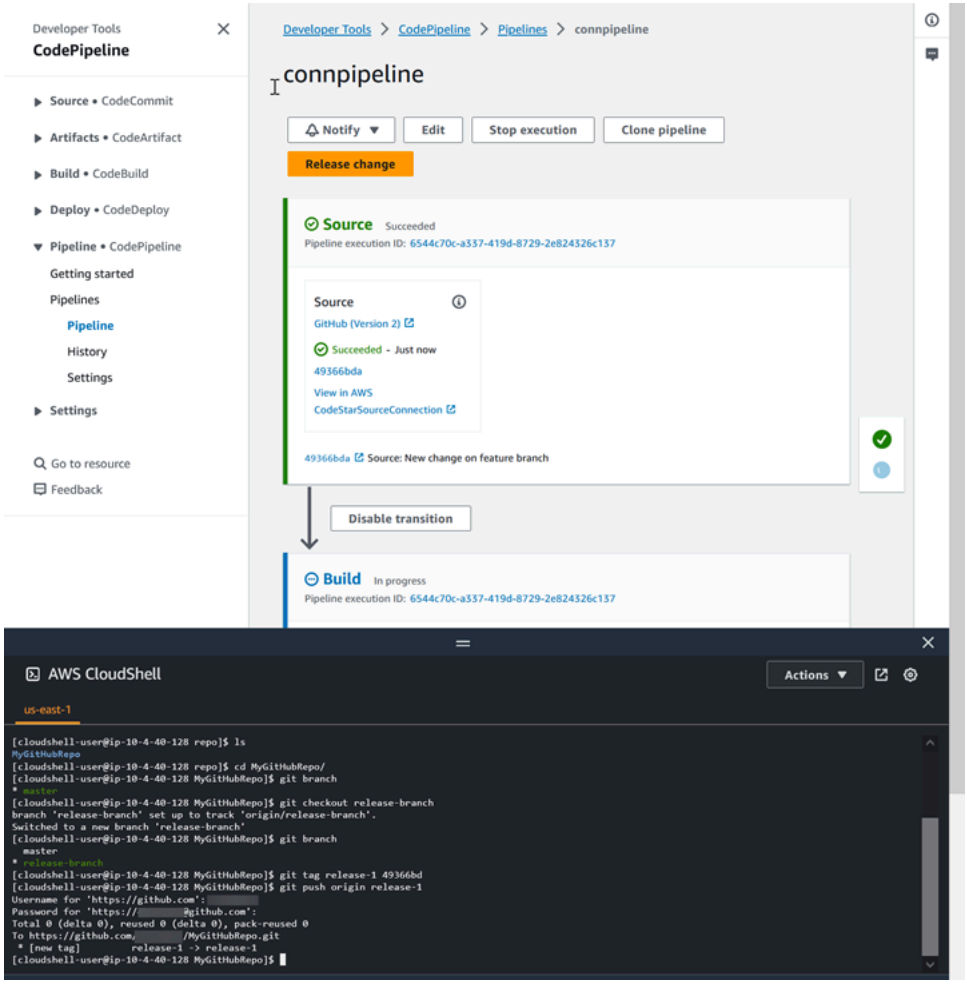
1. Verweisen Sie auf die kopierten Commit-IDs, die Sie taggen möchten. Um die Commits in jedem Branch anzuzeigen, geben Sie im CloudShell Terminal den folgenden Befehl ein, um die Commit-IDs zu erfassen, die Sie taggen möchten:

   ```
   git log
   ```

2. Gib im CloudShell Terminal den Befehl ein, um deinen Commit zu taggen und ihn an den Ursprung zu übertragen. Nachdem du deinen Commit getaggt hast, verwendest du den Befehl git push, um das Tag an den Ursprung zu übertragen. Gib im folgenden Beispiel den folgenden Befehl ein, um das `release-1` Tag für den zweiten Commit mit ID zu verwenden`49366bd`. Dieses Tag wird vom `release*` Pipeline-Tag-Filter gefiltert und startet die Pipeline.

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

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



Schritt 4: Änderungen veröffentlichen und Protokolle anzeigen

1. Nachdem die Pipeline erfolgreich ausgeführt wurde, wählen Sie in Ihrer erfolgreichen Build-Phase die Option Protokoll anzeigen aus.

   Sehen Sie sich unter Logs 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. Sehen Sie sich den Triggertyp an GitTag : Release-1.

Tutorial: Filtere nach Branch-Namen für Pull-Requests, um deine Pipeline zu starten (Typ V2)

In diesem Tutorial erstellst du eine Pipeline, die eine Verbindung zu deinem GitHub.com Repository herstellt. Dort ist die Quellaktion so konfiguriert, dass deine Pipeline mit einer Triggerkonfiguration gestartet wird, die nach Pull-Requests filtert. Wenn ein bestimmtes Pull-Request-Ereignis für einen bestimmten Branch eintritt, startet deine Pipeline. Dieses Beispiel zeigt Ihnen, wie Sie eine Pipeline erstellen, die das Filtern nach Branchnamen ermöglicht. Weitere Informationen zum Arbeiten mit Triggern finden Sie unterFügen Sie Filter für Push- und Pull-Request-Ereignistypen (CLI) hinzu. Weitere Hinweise zum Filtern mit Regex-Mustern im Glob-Format finden Sie unter. Arbeiten mit Glob-Mustern in der Syntax

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von for Artifacts verwendet. CodePipeline (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

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

Themen

- Voraussetzungen
- Schritt 1: Erstellen Sie eine Pipeline, um mit einem Pull-Request für bestimmte Branches zu beginnen
- Schritt 2: Erstelle einen Pull-Request und füge ihn zusammen GitHub.com , um mit der Ausführung deiner Pipeline zu beginnen

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

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

- Halten Sie Ihre GitHub Anmeldeinformationen bereit. Wenn Sie das verwenden AWS-Managementkonsole , um eine Verbindung herzustellen, werden Sie aufgefordert, sich mit Ihren GitHub Anmeldeinformationen anzumelden.

Schritt 1: Erstellen Sie eine Pipeline, um mit einem Pull-Request für bestimmte Branches zu beginnen

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

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

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der CodePipeline Konsole unter an https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyFilterBranchesPipeline`**.
5. Behalten Sie unter Pipeline-Typ die Standardauswahl bei V2. Die Pipeline-Typen unterscheiden sich in ihren Eigenschaften und ihrem Preis. Weitere Informationen finden Sie unter Pipeline-Typen.
6. 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.

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 Schritt 3: Quellstufe hinzufügen eine Quellstufe hinzu:

   a. Wählen Sie unter Quellanbieter die Option GitHub (über GitHub App) aus.

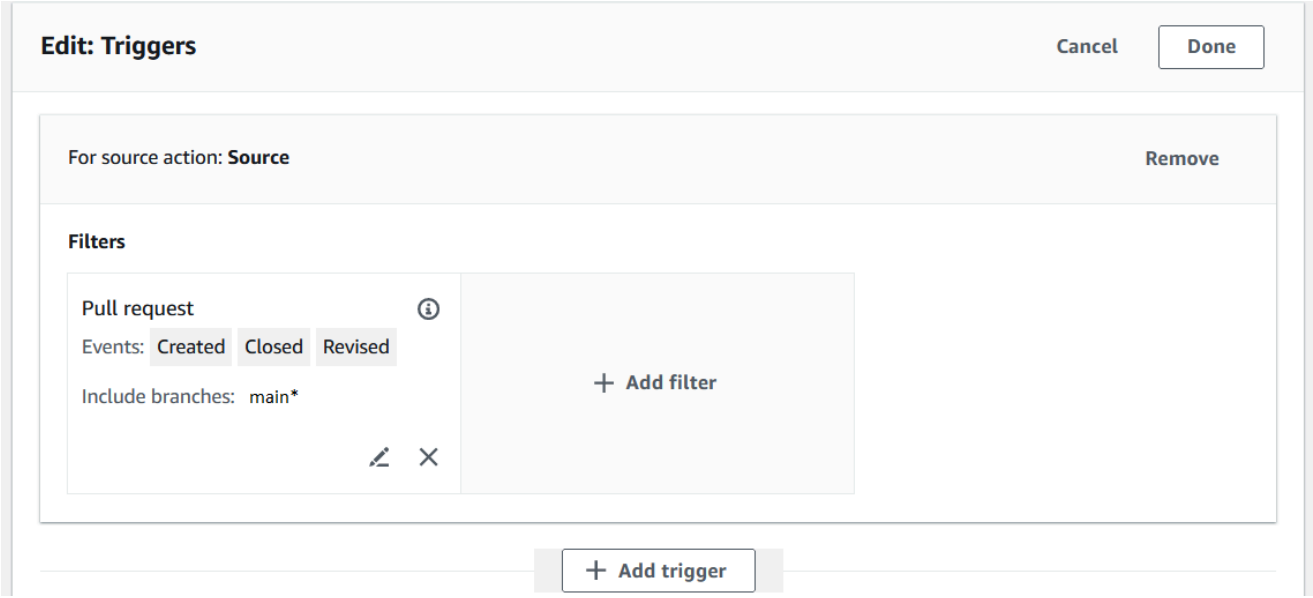
   b. Wählen Sie unter Verbindung eine bestehende Verbindung aus oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.

   c. Wählen Sie im Feld Repository name (Repositoryname) den Namen Ihres GitHub.com-Repositorys aus.

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

      Wählen Sie unter Ereignistyp die Option Pull-Request aus. Wähle alle Ereignisse unter Pull Request aus, sodass das Ereignis bei erstellten, aktualisierten oder geschlossenen Pull Requests eintritt.

      Geben Sie unter Branches in das Feld Include 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.

9. Wählen Sie in Schritt 4: Build-Phase hinzufügen unter Build-Anbieter die Option AWS CodeBuild Belassen Sie unter Region als Standardeinstellung die Pipeline-Region. Wählen oder erstellen Sie das Build-Projekt 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.
10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

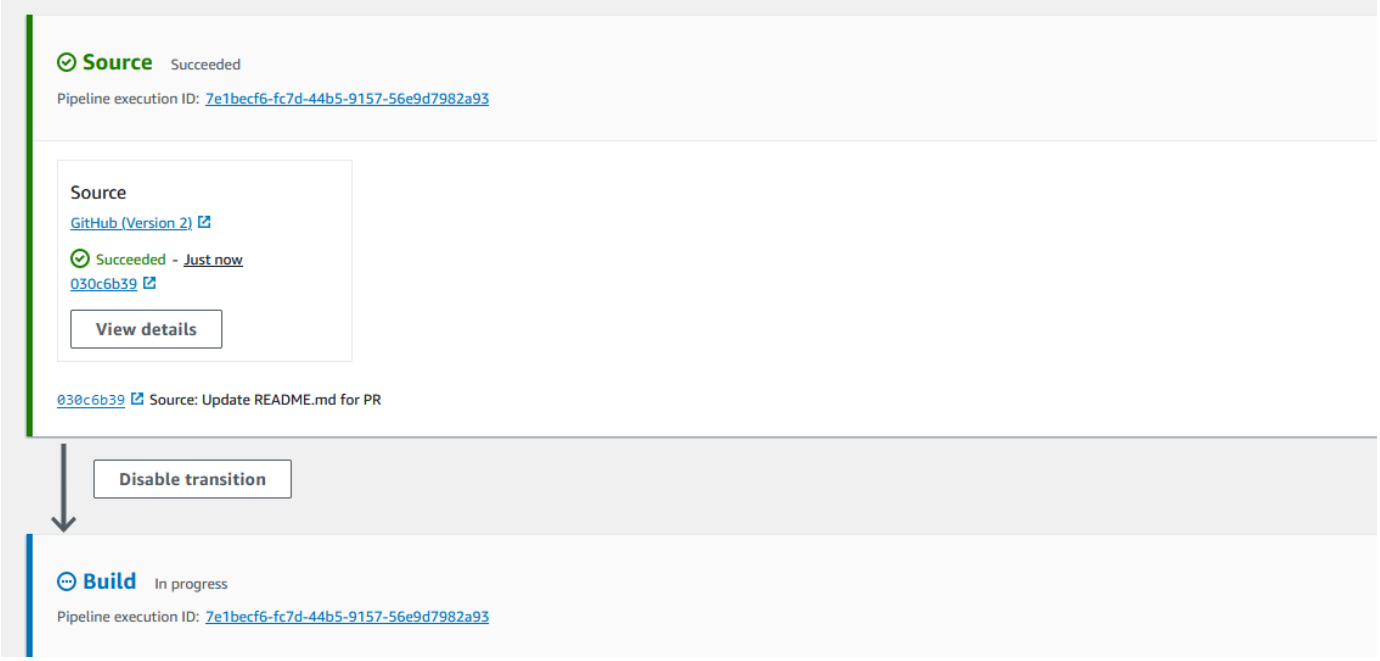
    Wählen Sie Weiter aus.
11. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus, und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken. Wählen Sie Weiter aus.
12. Wählen Sie auf Schritt 7: Überprüfen die Option Pipeline erstellen aus.

Schritt 2: Erstelle einen Pull-Request und füge ihn zusammen GitHub.com , um mit der Ausführung deiner Pipeline zu beginnen

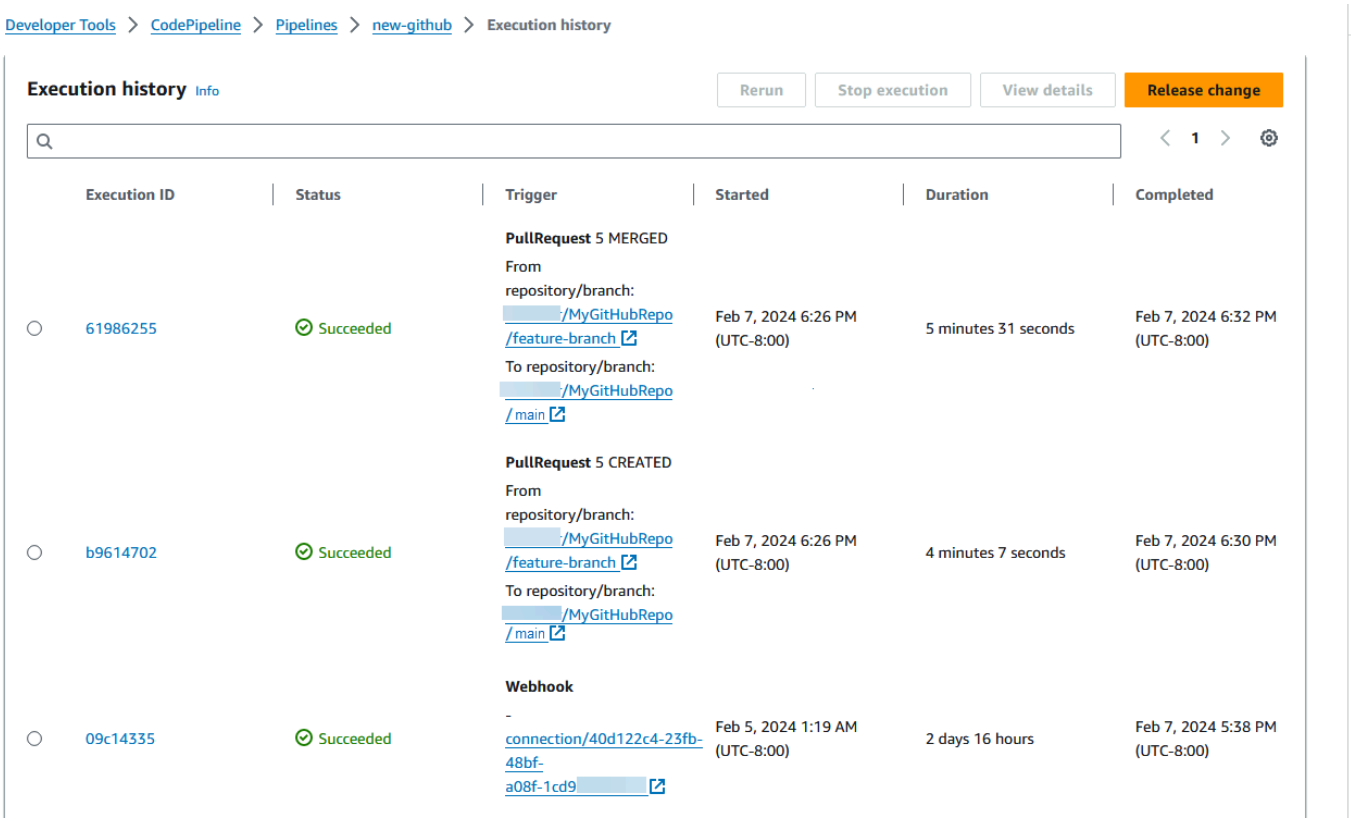
In diesem Abschnitt erstellst du einen Pull Request und führst ihn zusammen. Dadurch wird deine Pipeline mit einer Ausführung für den geöffneten Pull Request und einer Ausführung für den geschlossenen Pull Request gestartet.

Um einen Pull Request zu erstellen und deine Pipeline zu starten

1. Erstelle in einen Pull Request GitHub.com, indem du eine Änderung an einem Feature-Branch vornimmst und einen Pull Request an den `main` Branch sendest. README.md Bestätigen Sie die Änderung mit einer Nachricht wie`Update README.md for PR`.
2. Die Pipeline beginnt mit der Quellrevision und zeigt die Quellnachricht für den Pull-Request als Update README.md for PR an.



3. Wählen Sie History (Verlauf) aus. Sehen Sie sich im Verlauf der Pipeline-Ausführung die Statusereignisse CREATED und MERGED für Pull-Requests an, mit denen die Pipeline-Ausführungen gestartet wurden.



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.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Themen

- Voraussetzungen
- Schritt 1: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt
- Schritt 2: Änderungen veröffentlichen und Protokolle anzeigen

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Erstellen Sie ein Repository CodeCommit .
- 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 deinem CodeCommit Repository.
- Eine Build-Phase mit einer AWS CodeBuild Build-Aktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der CodePipeline Konsole unter an https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyVariablesPipeline`**.
5. Behalten Sie unter Pipeline-Typ die Standardauswahl bei V2. Die Pipeline-Typen unterscheiden sich in ihren Eigenschaften und ihrem Preis. Weitere Informationen finden Sie unter Pipeline-Typen.
6. 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.

7. Wählen Sie unter Variablen die Option Variable hinzufügen aus. Geben Sie unter Name `timeout` ein. Geben Sie im Feld Standard den Wert 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.

8. 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).

9. Fügen Sie auf der Seite Schritt 3: Quellstufe hinzufügen eine Quellstufe 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.

10. Fügen Sie in Schritt 4: 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 die Option aws/codebuild /standard:5.0.

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

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

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 Befehle erstellen 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. Dies kehrt zur CodePipeline Konsole zurück und erstellt 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 als angegebene Variable 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 Pipeline-Ebene `variables` und der Variablen auf Pipeline-Ebene, die in Schritt 7 für die Pipeline `timeout` erstellt wurde.

j. Wählen Sie Weiter aus.

11. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus, und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

Wählen Sie Weiter aus.

12. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus, und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken. Wählen Sie Weiter aus.
13. Wählen Sie auf Schritt 7: Überprüfen die Option Pipeline erstellen aus.

Schritt 2: Änderungen veröffentlichen und Protokolle anzeigen

1. Nachdem die Pipeline erfolgreich ausgeführt wurde, wählen Sie in Ihrer erfolgreichen Build-Phase die Option Details anzeigen aus.

   Wählen Sie auf der Detailseite den Tab Logs. Sehen Sie sich die CodeBuild Build-Ausgabe an. Die Befehle geben den Wert der eingegebenen Variablen aus.
2. Wählen Sie im linken Navigationsbereich Historie aus.

   Wählen Sie die letzte Ausführung aus und wählen Sie dann den Tab Variablen. Sehen Sie sich 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 einer Pipeline in der AWS CodePipeline Konsole zu verwenden.

In diesem Tutorial erstellen Sie eine zweistufige Pipeline, die einen versionierten S3-Quell-Bucket verwendet CodeDeploy , und veröffentlichen eine 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.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von for Artifacts verwendet. CodePipeline (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

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, betreffen 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. Du kannst regionsübergreifende Aktionen hinzufügen, wenn du deine Pipeline erstellst. 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 Sie einen S3-Quell-Bucket für Ihre Anwendung
- Schritt 2: Erstellen Sie Amazon EC2-Windows-Instances und installieren Sie den Agenten CodeDeploy
- Schritt 3: Erstellen Sie eine Anwendung in CodeDeploy
- Schritt 4: Erstellen Sie Ihre erste Pipeline in CodePipeline
- Schritt 5: Hinzufügen einer weiteren Phase zur Pipeline

- (Optional) Schritt 6: Deaktivieren und aktivieren Sie Übergänge zwischen den Stufen in CodePipeline
- Schritt 7: Bereinigen von Ressourcen

Schritt 1: Erstellen Sie einen S3-Quell-Bucket 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 unter an AWS-Managementkonsole. Öffnen Sie die S3-Konsole.
2. Wählen Sie Create Bucket (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, 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. US 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 mit durchführen möchten CodeDeploy, laden Sie die Beispielanwendung hier herunter: SampleApp _ Linux.zip.
      - Wenn Sie die Bereitstellung auf Windows Server-Instances mit 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 http://www.yaml.org YAML-Format, die von CodeDeploy zur Verwaltung einer Bereitstellung verwendet wird. Weitere Informationen zur Datei finden Sie unter AppSpec 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. Eine Datei wird für jedes der Ereignisse im Lebenszyklus einer CodeDeploy Bereitstellung wie folgt geschrieben:
        - `scripts`Ordner (nur Linux-Beispiel) — Der Ordner enthält die folgenden Shell-Skripts zum Installieren von Abhängigkeiten und zum Starten und Stoppen der Beispielanwendung für die automatische Bereitstellung: `install_dependencies`, `start_server`, und`stop_server`.
        - (Nur Windows-Beispiel) `before-install.bat` — Dies ist ein Batch-Skript für das Ereignis im Lebenszyklus der `BeforeInstall` Bereitstellung, das ausgeführt wird, um alte Dateien zu entfernen, die bei früheren Bereitstellungen dieses Beispiels geschrieben wurden, und einen Speicherort auf Ihrer Instance 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-Konsole für Ihren Bucket die Datei hoch:

   a. Klicken Sie auf Upload.

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

   c. Klicken Sie auf Upload.

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

> **Note**
>
> Dieses Tutorial enthält Beispielschritte zum Erstellen von Amazon EC2-Windows-Instances. Beispielschritte zum Erstellen von Amazon EC2-Linux-Instances finden Sie unter. Schritt 3: Erstellen Sie eine Amazon 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-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 unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
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 die Option aus. AWS-Service Wählen Sie unter Choose a use case (Anwendungsfall auswählen) die Option EC2 und dann Next: Permissions (Weiter: Berechtigungen) aus.

5. Suchen Sie nach der angegebenen Richtlinie und wählen Sie sie aus **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Suchen Sie nach der benannten 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 aus und klicken Sie oben auf der Seite auf Launch Instances.
3. Geben Sie unter Name und Tags in das Feld Name den Wert ein**`MyCodePipelineDemo`**. Dadurch werden 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 anhand der Tags die Instanzen aus, die bereitgestellt werden sollen.
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 „Für ein kostenloses Kontingent in Frage“. Sie finden es unter Schnellstart.)
5. Wählen Sie unter Instance-Typ den **`t2.micro`** Typ aus, der für das kostenlose Kontingent in Frage kommt, als Hardwarekonfiguration für Ihre Instance.
6. Wählen Sie unter Schlüsselpaar (Anmeldung) ein Schlüsselpaar aus oder erstellen Sie eines.

   Sie können auch Ohne Schlüsselpaar fortfahren 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 Auto-assign Öffentliche IP der Status Aktiviert 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 aus, wählen Sie HTTP aus, und wählen Sie 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 ein. 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 Öffentliches DNS nicht angezeigt wird, wählen Sie das Show/Hide Symbol und 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 aus, die Sie in diesem Schritt erstellen, wenn Sie Ihre Pipeline später in diesem Tutorial 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 unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
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 CodeDeploy aus 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 AWS CodeDeploy Menü Anwendungen aus.
3. Wählen Sie Create application aus.
4. Geben Sie unter Application name (Anwendungsname) `MyDemoApplication` ein.
5. Wählen Sie unter Compute Platform die Option EC2/On-premises.
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 AWS CodeDeploy über die Vertrauensstellung und die Berechtigungen verfügt, die unter Erstellen einer Servicerolle für CodeDeploy beschrieben sind. Informationen zum Erhalt des Servicerollen-ARN finden Sie unter Abrufen des Servicerollen-ARN (Konsole).
4. Wählen Sie unter Bereitstellungstyp die Option In-place.
5. Wählen Sie unter Environment configuration (Umgebungskonfiguration) die Option Amazon EC2-Instances aus. Wählen Sie im Feld Schlüssel die Option Name aus, und geben Sie im Feld Wert den Wert 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 aus 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. Vergewissern Sie sich, 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-Managementkonsole 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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyFirstPipeline`**.

   > **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.

5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
6. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:

   - Wählen Sie Neue Servicerolle, um CodePipeline 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.

7. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
8. Wählen Sie in Schritt 3: Quellstufe hinzufügen unter Quellanbieter die Option Amazon S3 aus. Geben Sie unter Bucket den Namen des S3-Buckets ein, den Sie in Schritt 1: Erstellen Sie einen S3-Quell-Bucket 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.

9. Wählen Sie in Schritt 4: Build-Phase hinzufügen die Option Build-Phase überspringen und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Skip auswählen. Wählen Sie Weiter aus.
10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

    Wählen Sie Weiter aus.
11. Wählen Sie in Schritt 6: Bereitstellungsphase hinzufügen unter Bereitstellungsanbieter die Option CodeDeploy . Das Feld Region ist standardmäßig mit dem Wert Ihrer Pipeline AWS-Region identisch. 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.
12. In Schritt 7: Überprüfen, überprüfen Sie die Informationen und wählen Sie dann Pipeline erstellen aus.

13. Die Pipeline startet die Ausführung. Sie können den Fortschritt sowie Erfolgs- und Fehlernachrichten anzeigen, während das CodePipeline Beispiel für jede Amazon EC2-Instance in der CodeDeploy Bereitstellung eine Webseite 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 Deploy (Bereitstellung)-Phase, die diese Änderungen EC2-Instances mit CodeDeploy bereitstellt.

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

Nun fügen Sie der Pipeline eine weitere Phase hinzu, um mithilfe von CodeDeploy die Bereitstellung von Staging-Servern zu Produktionsservern auszuführen. Zunächst erstellen Sie eine weitere Bereitstellungsgruppe im CodePipelineDemoApplication In CodeDeploy. Anschließend fügen Sie eine Phase hinzu, die eine Aktion enthält, die diese Bereitstellungsgruppe verwendet. Um eine weitere Stufe hinzuzufügen, verwenden Sie die CodePipeline Konsole oder die, AWS CLI um die Struktur der Pipeline in einer JSON-Datei abzurufen und manuell zu bearbeiten. Anschließend führen Sie 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-Instances wie zuvor. Dies erfolgt lediglich zu Demonstrationszwecken. Es ist absichtlich so konzipiert, dass es Ihnen nicht zeigt, wie Fehler angezeigt werden. CodePipeline

So erstellen Sie eine zweite Bereitstellungsgruppe in 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 Servicerolle dieselbe CodeDeploy Servicerolle aus, die Sie für die Erstbereitstellung verwendet haben (nicht die CodePipeline Servicerolle).

6. Wählen Sie unter Bereitstellungstyp die Option aus In-place.
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 Stufe hinzuzufügen.

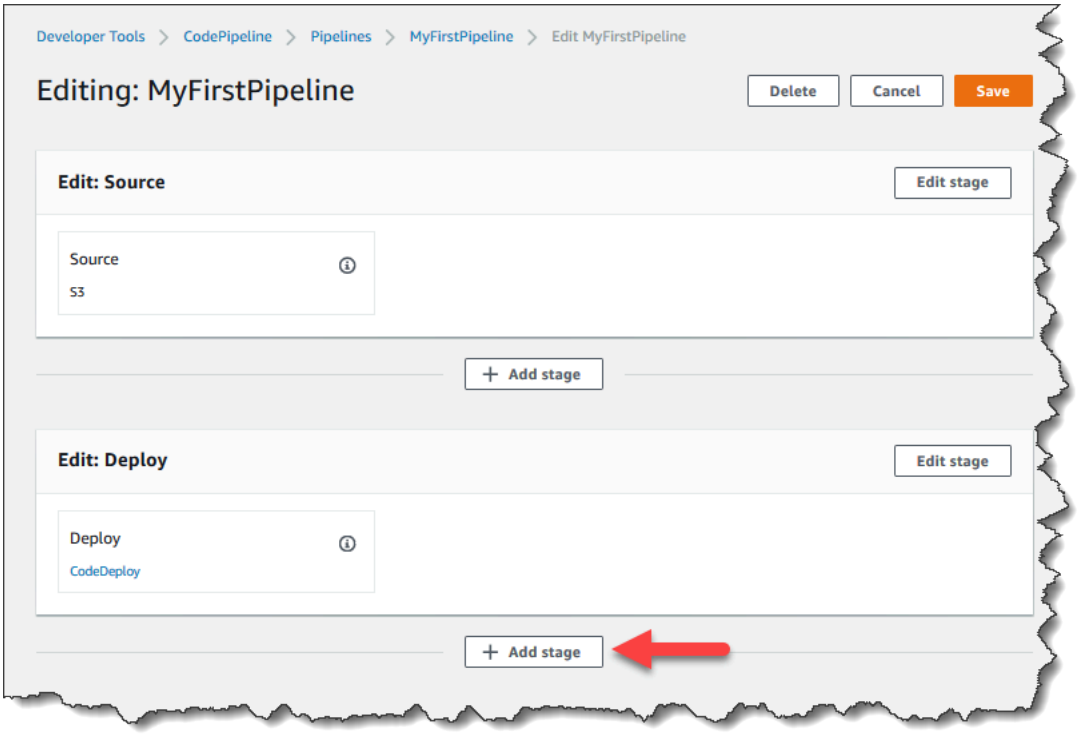
Themen

- Erstellen einer dritten Phase (Konsole)
- Erstellen einer dritten Phase (CLI)

Erstellen einer dritten Phase (Konsole)

Sie können die CodePipeline Konsole verwenden, um eine neue Stufe 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-Managementkonsole 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, den Sie zuvor 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 im Aktionsanbieter unter Deploy die Option aus CodeDeploy.
8. Wählen Sie im CodeDeploy Abschnitt unter 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 Änderung veröffentlichen 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, können Sie alternativ von einem Terminal auf Ihrem lokalen Linux-, macOS- oder Unix-Computer oder von einer Befehlszeile auf Ihrem lokalen Windows-Computer aus den start-pipeline-execution Befehl ausführen und dabei den Namen der Pipeline angeben. 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 die Option, die Ansichtsseite MyFirstPipeline zu öffnen.

    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. Sie sehen, dass die Bereitstellung in den ersten beiden Phasen wie zuvor erfolgreich war, aber in der Produktionsphase wird angezeigt, dass die Deploy-Second-Deployment Aktion fehlgeschlagen ist.

12. Wählen Sie in der Deploy-Second-Deployment Aktion Details aus. 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)

Die Verwendung der AWS CLI zum Hinzufügen einer Phase zu Ihrer Pipeline ist zwar komplexer als die Verwendung 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 gerade erstellten Pipeline anzuzeigen. 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": "amzn-s3-demo-bucket",
           },
           "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 Deploy-Phase 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 MyFirstPipeline aus 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. Obwohl die Bereitstellung in den ersten beiden Phasen wie zuvor erfolgreich ist, zeigt die Produktionsphase, dass die Deploy-Second-Deployment Aktion fehlgeschlagen ist.

9. Wählen Sie in der Deploy-Second-Deployment Aktion Details aus, 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.

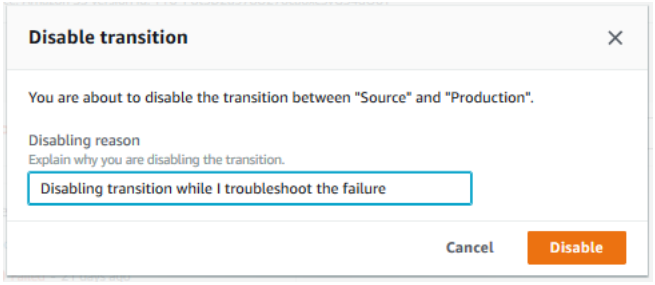
(Optional) Schritt 6: Deaktivieren und aktivieren Sie Übergänge zwischen den Stufen in CodePipeline

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.

So deaktivieren und aktivieren Sie Übergänge zwischen Stufen in einer CodePipeline Pipeline

1. Öffnen Sie die CodePipeline Konsole und wählen Sie MyFirstPipeline aus 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 diese dritte Phase erfolgreich sein soll, bearbeiten Sie die CodePipelineProductionFleet Bereitstellungsgruppe, bevor Sie den Übergang aktivieren, und geben Sie eine andere Gruppe 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. Beispielsweise können Sie die CodeDeploy Anwendung und die Bereitstellung wiederverwenden. Sie können eine Build-Aktion bei 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-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 Löschen einer Pipeline in AWS CodePipeline.
2. Folgen Sie den Anweisungen unter So bereinigen Sie CodeDeploy Ressourcen (Konsole), um Ihre Ressourcen zu bereinigen.
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: Eine einfache Pipeline (CodeCommit Repository) erstellen

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

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von for Artifacts verwendet. CodePipeline (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Die Pipeline hat zwei Phasen:

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

Der einfachste Einstieg AWS CodePipeline ist die Verwendung des Assistenten zum Erstellen einer Pipeline in der CodePipeline Konsole.

> **Note**
>
> Bevor Sie beginnen, stellen Sie sicher, dass Sie Ihren Git-Client so eingerichtet haben, dass Sie damit arbeiten können CodeCommit. Anweisungen finden Sie unter Einrichten für CodeCommit.

Schritt 1: Erstelle ein CodeCommit Repository

Zuerst 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 den 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 aus.

> **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 mit Ihrem Remote-CodeCommit-Repository herzustellen.

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

In diesem Schritt lädst du Code für eine Beispielanwendung herunter, die für einen CodeDeploy Beispiel-Walkthrough erstellt wurde, und fügst ihn deinem 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 den Schritten in diesem Tutorial für Linux-Instances folgen möchten.

      - Wenn Sie die Bereitstellung auf Amazon Linux-Instances mit durchführen möchten CodeDeploy, laden Sie die Beispielanwendung hier herunter: SampleApp _ Linux.zip.

- Wenn Sie die Bereitstellung auf Windows Server-Instances mit 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 http://www.yaml.org YAML-Format, die von CodeDeploy zur Verwaltung einer Bereitstellung verwendet wird. Weitere Informationen zur Datei finden Sie unter AppSpec 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. Eine Datei wird für jedes der Ereignisse im Lebenszyklus einer CodeDeploy Bereitstellung wie folgt geschrieben:
  - `scripts`Ordner (nur Linux-Beispiel) — Der Ordner enthält die folgenden Shell-Skripts zum Installieren von Abhängigkeiten und zum Starten und Stoppen der Beispielanwendung für die automatische Bereitstellung: `install_dependencies`, `start_server`, und`stop_server`.
  - (Nur Windows-Beispiel) `before-install.bat` — Dies ist ein Batch-Skript für das Ereignis im Lebenszyklus der `BeforeInstall` Bereitstellung, das ausgeführt wird, um alte Dateien zu entfernen, die bei früheren Bereitstellungen dieses Beispiels geschrieben wurden, und einen Speicherort auf Ihrer Instance 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. Verwende eine der folgenden Methoden, um Dateien in dein Repository hochzuladen.

   a. Um die CodeCommit Konsole zum Hochladen deiner Dateien zu verwenden:

      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 aus Ihrem lokalen Repository in Ihr CodeCommit-Repository zu übertragen:

```
git push
```

4. Die Dateien, die du heruntergeladen und zu deinem lokalen Repo hinzugefügt hast, wurden jetzt zum `main` Branch in deinem CodeCommit `MyDemoRepo` Repository hinzugefügt und sind bereit, in eine Pipeline aufgenommen zu werden.

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

In diesem Schritt erstellen Sie die Amazon EC2-Instance, auf 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 unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
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 die Option aus. AWS-Service Wählen Sie unter Wählen Sie einen Anwendungsfall 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 benannten **`AmazonEC2RoleforAWSCodeDeploy`** Richtlinie und wählen Sie sie aus.
6. Suchen Sie nach der benannten 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 aus und klicken Sie oben auf der Seite auf Launch Instances.
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 anhand der Tags die Instanzen aus, die bereitgestellt werden sollen.
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) beschrieben und trägt die Bezeichnung „Kostenloses Kontingent berechtigt“.)
5. Wählen Sie unter Instance-Typ den **`t2.micro`** Typ aus, der für das kostenlose Kontingent in Frage kommt, als Hardwarekonfiguration für Ihre Instance.
6. Wählen Sie unter Schlüsselpaar (Anmeldung) ein Schlüsselpaar aus oder erstellen Sie eines.

Sie können auch Ohne Schlüsselpaar fortfahren 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 Auto-assign Öffentliche IP der Status Aktiviert lautet.

   Wählen Sie für die erstellte Sicherheitsgruppe 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 ein. 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 Öffentliches DNS nicht angezeigt wird, wählen Sie das Show/Hide Symbol und 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 Bereitstellungen der Beispielanwendung auf Ihrer Amazon 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 unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
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 CodeDeploy aus 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 die Option „Anwendungen“.
3. Wählen Sie Create application aus.
4. Geben Sie unter Application name (Anwendungsname) **`MyDemoApplication`** ein.
5. Wählen Sie unter Compute Platform die Option EC2/On-premises.
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 Bereitstellungstyp die Option aus In-place.
5. Wählen Sie unter Environment configuration (Umgebungskonfiguration) die Option Amazon EC2-Instances aus. Geben Sie in das Feld Schlüssel ein**`Name`**. Geben Sie im Feld Wert den Namen ein, den Sie zum Markieren der Instanz verwendet haben (z. B.**`MyCodePipelineDemo`**).

6. Wählen Sie unter Agentenkonfiguration mit AWS Systems Manager die Option Jetzt aus 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-Managementkonsole 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 auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) aus, oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyFirstPipeline`**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in Eigenschaften und Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
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 in Schritt 3: Quellstufe hinzufügen unter Quellanbieter die Option aus CodeCommit. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, in dem Sie erstellt habenSchritt 1: Erstelle 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 muss.

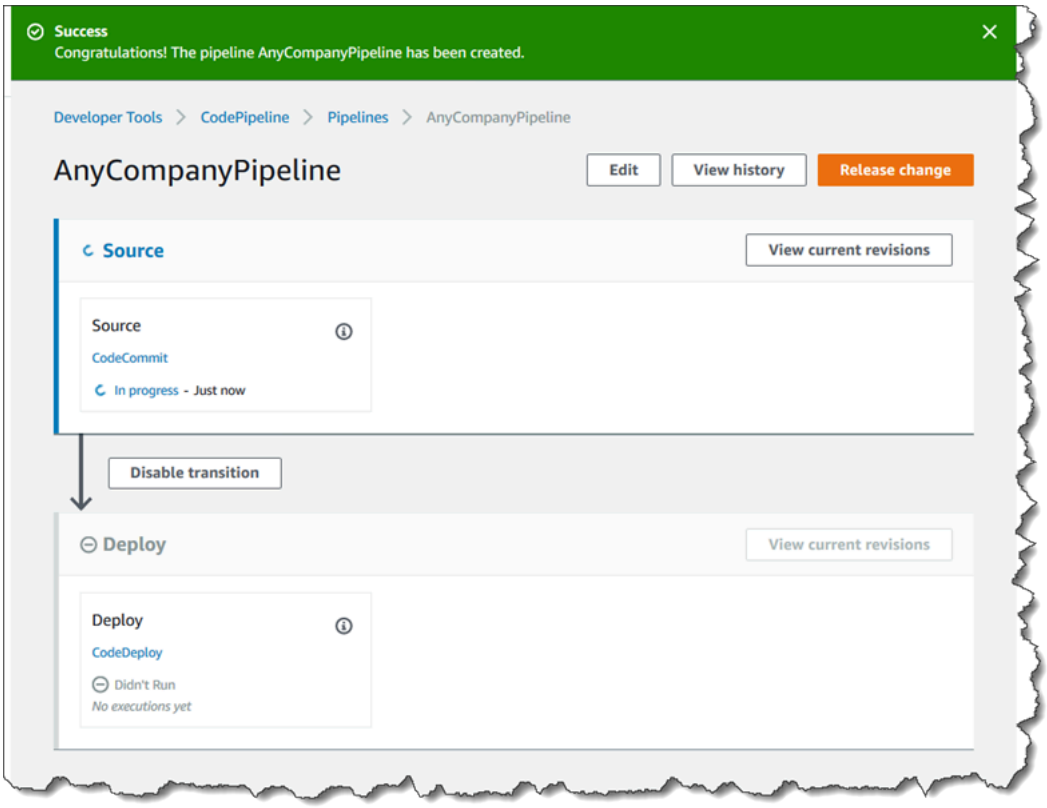
   Lassen Sie die Standardwerte in Change detection options unverändert. Auf diese Weise können CodePipeline Sie Amazon CloudWatch Events verwenden, um Änderungen in Ihrem Quell-Repository zu erkennen.

   Wählen Sie Weiter aus.
9. Wählen Sie in Schritt 4: Build-Phase hinzufügen die Option Build-Phase überspringen und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Skip wählen. 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 in diesem Schritt CodeBuild konfigurieren.
10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

    Wählen Sie Weiter aus.
11. Wählen Sie in Schritt 6: Bereitstellungsphase hinzufügen unter Bereitstellungsanbieter die Option 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.
12. In Schritt 7: Überprüfen, überprüfen Sie die Informationen und wählen Sie dann Create pipeline aus.

13. 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 den Fortschritt sowie Erfolgs- und Fehlernachrichten einsehen, während das CodePipeline Beispiel die Webseite für die Amazon EC2-Instance in der Bereitstellung bereitstellt. CodeDeploy



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 „Erfolgreich“ für den Pipeline-Status angezeigt wird, wählen Sie im Statusbereich der Bereitstellungsphase die Option CodeDeploy. Dadurch wird die CodeDeploy Konsole geöffnet. Wenn Erfolgreich nicht angezeigt wird, finden Sie weitere Informationen unterProblembehebung 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 ausgeführt wird, sobald der Code in Ihrem CodeCommit-Repository geändert wird. 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 Projektarchiv, 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. Beispielsweise können Sie 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-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 Löschen einer Pipeline in AWS CodePipeline.
2. Um Ihre CodeDeploy Ressourcen zu bereinigen, folgen Sie den Anweisungen unter Ressourcen zur Bereinigung der Bereitstellung.
3. Folgen Sie den Anweisungen unter Löschen eines CodeCommit Repositorys, um das CodeCommit Repository zu löschen.

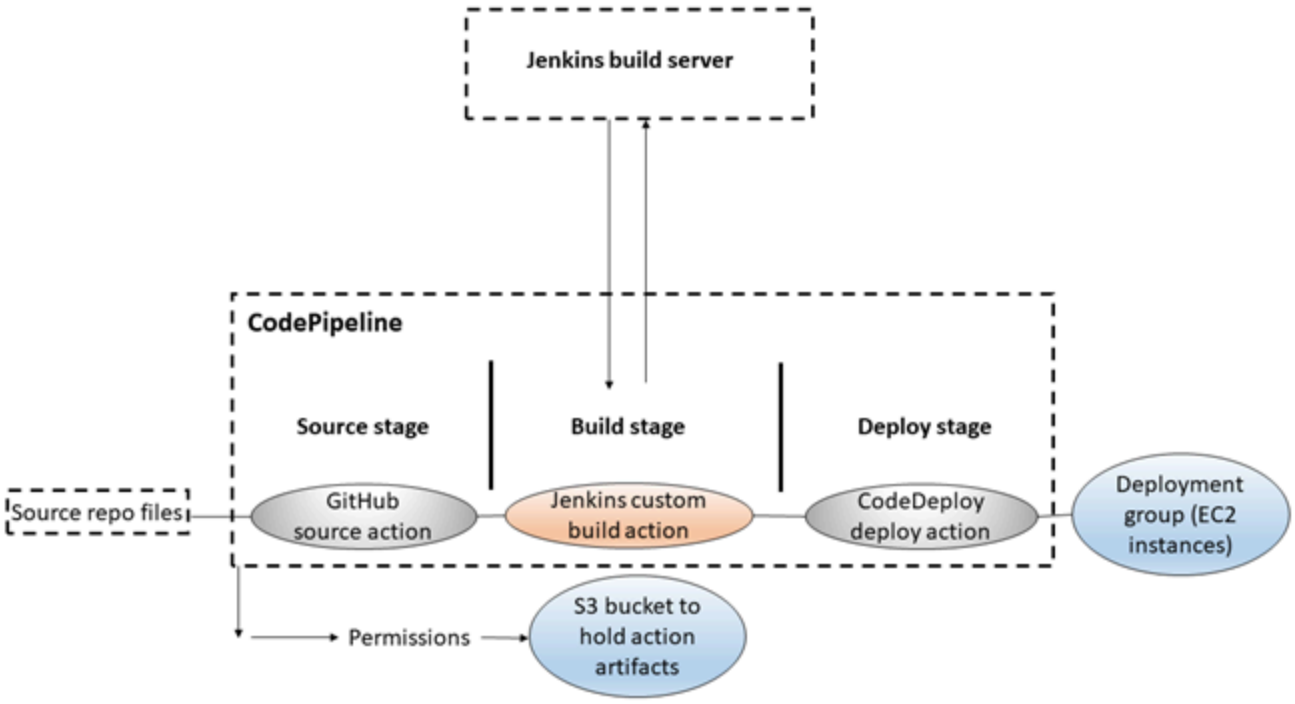
Schritt 8: Weitere Informationen

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 Hilfe von 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: Eine einfache Pipeline (CodeCommit Repository) erstellen 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 zum Bereitstellen 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 in diesem Verfahren zu Ihrer Pipeline hinzufügen, betreffen 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. Du kannst regionsübergreifende Aktionen hinzufügen, wenn du deine Pipeline erstellst. 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.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

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

Themen

- Schritt 1: Erfüllen der Voraussetzungen
- Schritt 2: Erstellen Sie eine Pipeline in CodePipeline
- Schritt 3: Hinzufügen einer weiteren Phase zur Pipeline
- Schritt 4: Bereinigen von Ressourcen

Schritt 1: Erfüllen der Voraussetzungen

Für die Integration mit Jenkins AWS CodePipeline müssen Sie das CodePipeline Plugin für Jenkins auf jeder Jenkins-Instanz installieren, mit der Sie es verwenden möchten. CodePipeline Sie sollten auch einen dedizierten IAM-Benutzer oder eine eigene 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-Instance-Rolle 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 den Schritten unter folgen. 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
- Erstellen Sie eine IAM-Rolle, die Sie für die Jenkins-Integration verwenden möchten
- Installieren und konfigurieren Sie Jenkins und das Plugin für Jenkins CodePipeline

Kopieren oder klonen Sie das Beispiel in ein Repository GitHub

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

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, wählen Sie codepipeline-jenkins-aws-codedeploy_linux.zip.
   - Wenn Sie Ihr Beispiel auf Windows Server-Instances bereitstellen, wählen Sie CodePipeline-Jenkins-AWSCodeDeploy _ Windows.zip.
2. Wählen Sie im Repository Fork aus, um das Beispiel-Repo in ein Repo in Ihrem GitHub Konto zu klonen. Weitere Informationen finden Sie in der GitHub-Dokumentation.

Erstellen Sie eine IAM-Rolle, die Sie für die Jenkins-Integration verwenden möchten

Als bewährte Methode sollten Sie erwägen, eine EC2-Instance zum Hosten Ihres Jenkins-Servers zu starten und der Instance mithilfe einer IAM-Rolle die für die Interaktion mit ihr erforderlichen Berechtigungen zu gewähren. CodePipeline

1. Melden Sie sich bei der an AWS-Managementkonsole 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 Seite Review (Überprüfen) in Role name (Rollenname) den Namen der Rolle ein, die speziell für die Jenkins-Integration erstellt werden soll (z. B. *JenkinsAccess*). Wählen Sie anschließend Create role (Rolle erstellen) aus.

Beim Erstellen der EC2-Instance, auf der Sie Jenkins installieren werden, müssen Sie in Step 3: Configure Instance Details (Schritt 3: Instance-Details konfigurieren) die Instance-Rolle wählen (z. B. *JenkinsAccess*).

Weitere Informationen zu Instance-Rollen und Amazon EC2 finden Sie unter IAM-Rollen für Amazon EC2, Verwenden von IAM-Rollen zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon 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-Instance, in der Sie Jenkins installieren werden. Stellen Sie in Step 3: Configure Instance Details (Schritt 3: Instance-Details konfigurieren) sicher, dass Sie die Instance-Rolle auswählen, die Sie erstellt haben (z. B. *JenkinsAccess*). Weitere Informationen zum Erstellen von EC2-Instances finden Sie unter Starten einer Amazon EC2-Instance im Amazon EC2-Benutzerhandbuch.

   > **Note**
   >
   > Wenn Sie bereits über Jenkins-Ressourcen verfügen, die Sie verwenden möchten, können Sie dies tun, aber Sie müssen einen speziellen IAM-Benutzer erstellen, die `AWSCodePipelineCustomActionAccess` verwaltete Richtlinie auf diesen Benutzer anwenden und dann die Zugangsdaten für diesen Benutzer in 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 wählen Sie Jetzt herunterladen und nach dem Neustart installieren.
6. Wählen Sie auf der Plugins/Upgrades Seite Installation die Option Jenkins neu starten, wenn die Installation abgeschlossen ist und keine Jobs ausgeführt werden.
7. Wählen Sie Back to Dashboard aus.
8. Klicken Sie auf der Hauptseite auf New Item.
9. Geben Sie im Feld Artikelname einen Namen für das Jenkins-Projekt ein (z. B.). *MyDemoProject* Wählen Sie Freestyle project und OK aus.

   > **Note**
   >
   > Stellen Sie sicher, dass der Name Ihres Projekts die Anforderungen für CodePipeline erfüllt. 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 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, so wie er in CodePipeline (z. B.) erscheinen wird. *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 den Namenskonventionen für Aktionen in CodePipeline entsprechen. 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:

    ```
    * * * * *
    ```

Dies 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 dann Publisher aus AWS CodePipeline . 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-Managementkonsole 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 habenus-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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
5. Geben Sie auf der Seite „Schritt 2: Pipeline-Einstellungen auswählen“ im Feld Pipeline-Name den Namen für Ihre Pipeline ein.
6. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
7. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
8. Belassen Sie die Advanced settings (Erweiterte Einstellungen) auf den Standardwerten und wählen Sie Next (Weiter) aus.
9. Wählen Sie auf der Seite Schritt 3: Quellphase hinzufügen unter Quellanbieter die Option aus GitHub.
10. Wählen Sie unter Verbindung eine vorhandene Verbindung aus oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unterGitHub Verbindungen.
11. Wählen Sie in Schritt 4: Build-Phase hinzufügen die Option Jenkins hinzufügen aus. Geben Sie im Feld Anbietername den Namen der Aktion ein, die Sie im CodePipeline Plugin für Jenkins angegeben haben (zum Beispiel). *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 im Feld Projektname den Namen des Projekts ein, das Sie in Jenkins erstellt haben, z. B.*MyDemoProject*, und wählen Sie dann Weiter.
12. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

    Wählen Sie Weiter aus.
13. Verwenden Sie in Schritt 6: Bereitstellungsphase hinzufügen die CodeDeploy Anwendung und die Bereitstellungsgruppe, in der Sie erstellt haben, erneutTutorial: 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.

14. In Schritt 7: Überprüfen, überprüfen Sie die Informationen und wählen Sie dann Pipeline erstellen aus.
15. Die Pipeline wird automatisch gestartet und führt das Beispiel über die Pipeline aus. Während die Pipeline das Haml-Beispiel im HTML-Format erstellt und als Webseite für jede Amazon EC2-Instance in der Bereitstellung bereitstellt, können Sie sich den Fortschritt sowie Erfolgs- und Fehlernachrichten anzeigen lassen. 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
- Erstellen eines Jenkins-Projekts zum Testen der Bereitstellung
- Erstellen einer vierten Phase

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 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-Instance installiert ist oder es auf einer EC2-Instance installiert ist, auf der ein Windows-Betriebssystem ausgeführt wird, füllen Sie die Felder gemäß Ihren Proxy-Host- und Port-Einstellungen aus und geben Sie die Anmeldeinformationen des IAM-Benutzers oder

der IAM-Rolle ein, den 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 unter 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:

```
* * * * *
```

Das 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 Execute Shell. 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 den Befehl Batch 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 dann Publisher aus AWS CodePipeline . 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-Managementkonsole 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, den Sie zuvor 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 ein*MyJenkinsTest-Action*. 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 klicken Sie dann auf Fertig.

> **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 Sample durch die überarbeitete Pipeline laufen zu lassen, wählen Sie auf der Seite mit den Pipeline-Details die Option Änderung veröffentlichen 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 die Pipeline nicht weiter verwenden möchten CodePipeline, löschen Sie die Pipeline, dann die CodeDeploy Anwendung und die zugehörigen Amazon EC2-Instances und schließlich den Amazon S3-Bucket, der zum Speichern von Artefakten verwendet wird. Sie sollten auch überlegen, ob Sie andere Ressourcen, wie z. B. das GitHub Repository, löschen möchten, 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 aus, um die von Ihnen erstellte Pipeline zu löschen. 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 Bereinigen, 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-Bucket, der zum Speichern von Artefakten für Ihre Pipeline verwendet wird. 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 Event-Regel ein, um E-Mail-Benachrichtigungen über 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 in den Phasen oder Aktionen in Ihren Pipelines ändert. Weitere Informationen zur Verwendung von CloudWatch Ereignissen zum Einrichten von Benachrichtigungen für Änderungen des Pipelinestatus finden Sie unter. Überwachung von CodePipeline Ereignissen

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 Events-Regel eine Input-Transformer-Methode 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-Benachrichtigung und die CloudWatch Events-Regel, darauf, dass die Ressourcen in derselben AWS Region wie Ihre Pipeline erstellt werden.

Themen

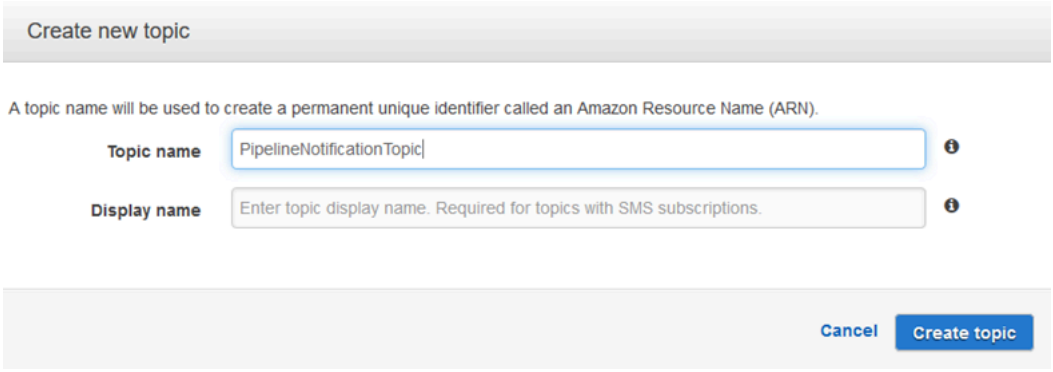
- Schritt 1: Richten Sie eine E-Mail-Benachrichtigung mit Amazon SNS ein
- Schritt 2: Erstellen einer Regel und Hinzufügen des SNS-Themas als Ziel
- Schritt 3: Bereinigen von Ressourcen

Schritt 1: Richten Sie eine E-Mail-Benachrichtigung mit Amazon SNS ein

Amazon SNS koordiniert die Verwendung von Themen zur Übermittlung 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-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 Ereignisse, um Benachrichtigungen zu diesem Thema über Amazon SNS zu senden. Erstellen Sie ein Thema wie folgt:

1. Öffnen Sie die Amazon-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 im Amazon SNS-Entwicklerhandbuch unter Thema erstellen.

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

1. Wählen Sie in der Amazon SNS-Konsole in der Themenliste das Kontrollkästchen neben Ihrem neuen Thema aus. 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. Wählen Sie Create Subscription.
6. Amazon SNS sendet eine Abonnement-Bestätigungs-E-Mail 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 Abonnieren eines Themas im Amazon SNS-Entwicklerhandbuch.

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 die Verwendung einer Regel zum Senden von Build-Benachrichtigungen zu beenden, 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 aus und klicken Sie dann auf Aktionen, Löschen.

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 zum Bereinigen der SNS-Benachrichtigung und zum Löschen der Amazon CloudWatch Events-Regel finden Sie unter Bereinigen (Abmelden von einem Amazon SNS-Thema) und als 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**
>
> Im Rahmen der Erstellung einer Pipeline in der Konsole wird ein S3-Artefakt-Bucket von CodePipeline for artefacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

> **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.

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 benötigen sie für die Erstellung Ihrer Pipeline in 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-Managementkonsole und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   b. 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).

   c. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.

   d. Geben Sie auf der Seite Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Namen für Ihre Pipeline ein.

   e. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und im Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

   f. 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 Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu.

```
{
     "Effect": "Allow",
     "Action": [
        "devicefarm:ListProjects",
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
     ],
     "Resource": "*"
}
```

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

h. Wählen Sie auf der Seite Schritt 3: Quellstufe hinzufügen unter Quellanbieter die Option GitHub (via GitHub App) aus.

i. 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.

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

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

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

4. Fügen Sie in Schritt 4: Build-Phase hinzufügen eine Build-Phase hinzu:

a. Wählen Sie unter Buildanbieter die Option Andere Buildanbieter und dann aus AWS CodeBuild. 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 aus:5.0aws/codebuild/standard.

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 in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

   Wählen Sie Weiter aus.

6. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken. Wählen Sie Weiter aus.

7. Wählen Sie auf Schritt 7: Überprüfen die Option Pipeline erstellen aus. Sie sollten ein Diagramm sehen, das Ihre Pipeline-Quell- und -Build-Stufen anzeigt.

8. 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 ProjectIdunter Ihre Device Farm Farm-Projekt-ID ein. 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 den 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 AppTypeSie Android ein.

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.

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**
>
> 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**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

> **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.

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 benötigen sie für die Erstellung Ihrer Pipeline in 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 Sie einen S3-Quell-Bucket 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-Managementkonsole und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   b. 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).

   c. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.

   d. Geben Sie auf der Seite Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Namen für Ihre Pipeline ein.

   e. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und im Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

   f. 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 Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu.
>
> ```
> {
>      "Effect": "Allow",
>      "Action": [
>         "devicefarm:ListProjects",
>         "devicefarm:ListDevicePools",
>         "devicefarm:GetRun",
>         "devicefarm:GetUpload",
>         "devicefarm:CreateUpload",
>         "devicefarm:ScheduleRun"
>      ],
>      "Resource": "*"
> }
> ```

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

h. Wählen Sie auf der Seite Schritt 3: Quellstufe hinzufügen unter Quellanbieter die Option Amazon S3 aus.

i. 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.

j. Wählen Sie Weiter aus.

4. Erstellen Sie in Schritt 4: Build-Phase hinzufügen eine Platzhalter-Build-Phase 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.

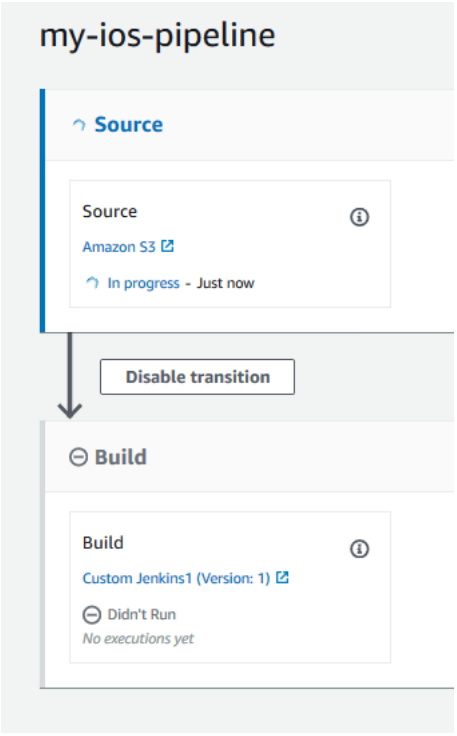
e. Wählen Sie Weiter aus.

f. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

   Wählen Sie Weiter aus.

g. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken.

h. Wählen Sie auf Schritt 7: Überprüfen die Option 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 den 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.

   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**
   >
   > 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 im AWS CodePipeline - Fenster Save (Speichern) und dann in der Warnmeldung Save (Speichern).

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 im Service Catalog bereitgestellt wird

Mit Service Catalog können Sie Produkte auf der Grundlage von AWS CloudFormation Vorlagen erstellen und bereitstellen.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Dieses Tutorial zeigt Ihnen, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre Produktvorlage in Service Catalog bereitzustellen und Änderungen zu übertragen, die Sie in Ihrem Quell-Repository vorgenommen haben (das bereits in GitHub oder Amazon S3 erstellt wurde). CodeCommit

> **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 in Service Catalog und dann eine Pipeline in AWS CodePipeline. Dieses Tutorial bietet zwei Optionen für die Einrichtung der Bereitstellungskonfiguration:

- Erstellen Sie ein Produkt in 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 eine separate Konfigurationsdatei). Siehe Option 1: Bereitstellung im Service Catalog ohne Konfigurationsdatei.

  > **Note**
  >
  > Die Vorlagendatei kann im Format YAML oder JSON erstellt werden.

- Erstellen Sie ein Produkt in 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: Mithilfe einer Konfigurationsdatei im Service Catalog bereitstellen.

Option 1: Bereitstellung im Service Catalog ohne Konfigurationsdatei

In diesem Beispiel laden Sie die AWS CloudFormation Beispielvorlagendatei für einen S3-Bucket hoch und erstellen dann Ihr Produkt in 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"
    }
  }
}
```

Mit dieser Vorlage können AWS CloudFormation Sie einen S3-Bucket erstellen, 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, gehen 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 benötigen sie für die Erstellung Ihrer Pipeline in 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-Managementkonsole und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   b. 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).

c. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.

d. Geben Sie in Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name einen Namen für Ihre Pipeline ein.

e. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.

f. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.

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

2. Gehen Sie wie folgt vor, um auf der Seite „Schritt 3: Quellstufe hinzufügen“ eine Quellstufe 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 in Schritt 4: Build-Phase hinzufügen die Option Build-Phase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

4. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

   Wählen Sie Weiter aus.

5. Gehen Sie in Schritt 6: Bereitstellungsphase hinzufügen wie folgt vor:

   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 Option CloudFormation Vorlage aus.

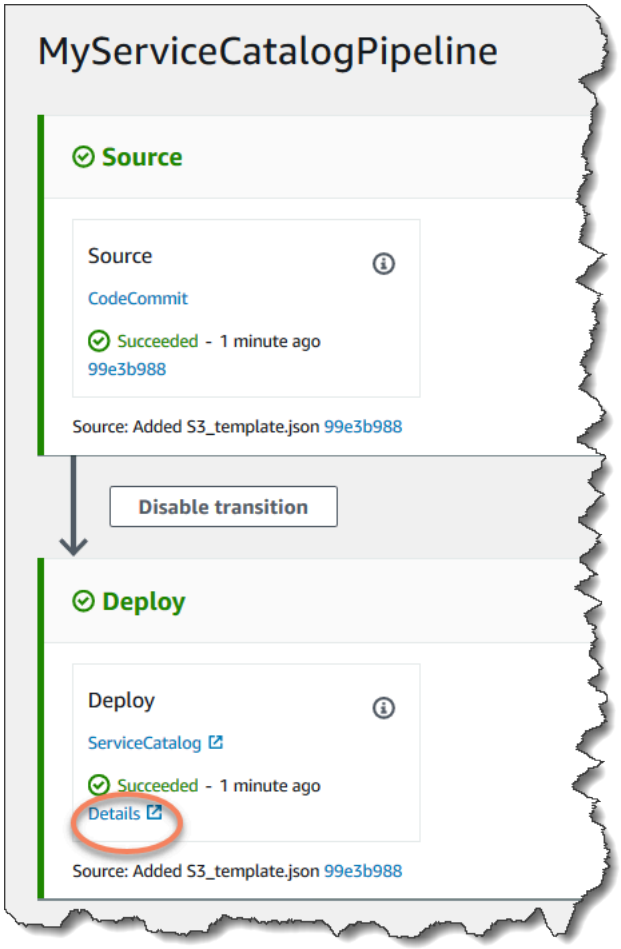
f. Geben Sie im Feld Name der Produktversion den Namen der Produktversion ein, die Sie im Service Catalog angegeben haben. Wenn die Vorlagenänderung in einer neuen Produktversion bereitgestellt werden soll, geben Sie einen Namen für die Produktversion ein, der noch nicht für eine vorherige Produktversion im selben Produkt verwendet wurde.

g. Wählen Sie für Input artifact (Eingabeartefakt) das Quell-Eingabeartefakt.

h. Wählen Sie Weiter aus.

6. In Schritt 7: Überprüfen, überprüfen Sie Ihre Pipeline-Einstellungen und wählen Sie dann Erstellen aus.

7. Wenn Ihre Pipeline in der Bereitstellungsphase erfolgreich ausgeführt wird, wählen Sie Details. Dadurch wird Ihr Produkt im Servicekatalog geöffnet.



8. Wählen Sie den Namen Ihrer Version in den Produktinformationen, um die Produktvorlage zu öffnen. Zeigen Sie die Vorlagenbereitstellung an.

Schritt 4: Übernehmen Sie eine Änderung und verifizieren Sie Ihr Produkt im Service Catalog

1. Sehen Sie sich Ihre Pipeline in der CodePipeline Konsole an und wählen Sie auf Ihrer Quellstufe 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: Mithilfe einer Konfigurationsdatei im Service Catalog bereitstellen

In diesem Beispiel laden Sie die AWS CloudFormation Beispielvorlagendatei für einen S3-Bucket hoch und erstellen dann Ihr Produkt in 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"
    }
  }
}
```

Mit dieser Vorlage können AWS CloudFormation Sie einen S3-Bucket erstellen, 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-Bereitstellung zu definieren parameters/preferences. 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, gehen 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 benötigen sie für die Erstellung Ihrer Pipeline in 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-Managementkonsole 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).

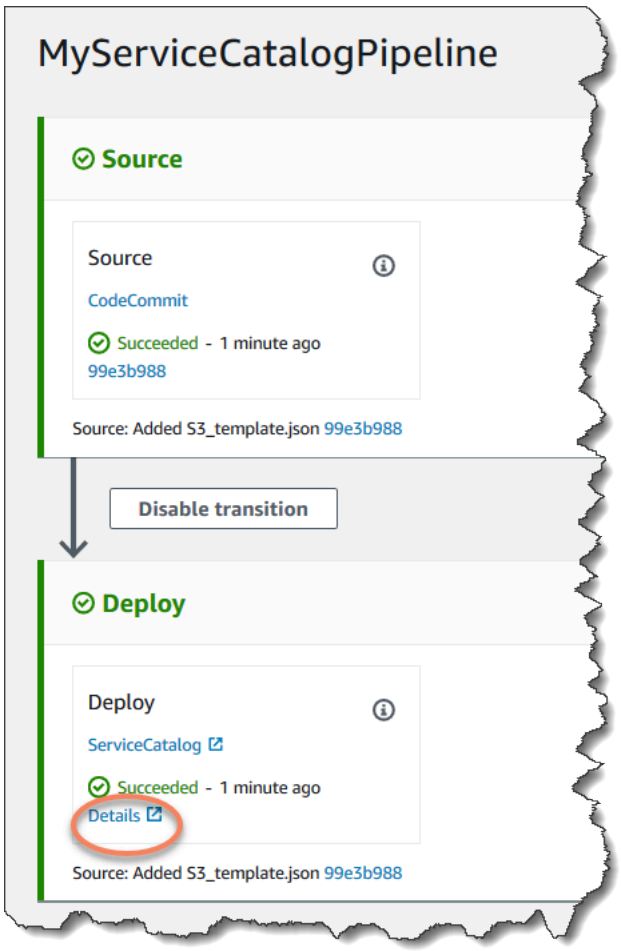
   c. Fügen Sie in das Feld Produkt-ID die Produkt-ID ein, die Sie aus der Service Catalog-Konsole kopiert haben.

   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 Instances bereitstellt. AWS CloudFormation Die Beispielvorlage erstellt eine Pipeline, die in AWS CodePipeline angezeigt werden kann. Die Pipeline erkennt das Eintreffen einer gespeicherten Änderung über Amazon CloudWatch Events.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Themen

- Beispiel 1: Erstellen Sie eine AWS CodeCommit Pipeline mit AWS CloudFormation
- Beispiel 2: Erstellen Sie eine Amazon S3-Pipeline mit AWS CloudFormation

Beispiel 1: Erstellen Sie eine AWS CodeCommit Pipeline mit AWS CloudFormation

Diese exemplarische Vorgehensweise zeigt Ihnen, wie Sie mithilfe 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 Ihre Amazon Events-Regel umfasst. CloudWatch Nachdem Sie Ihren Ressourcenstapel erstellt haben AWS CloudFormation, können Sie Ihre Pipeline in der Konsole anzeigen. AWS CodePipeline Die Pipeline ist eine zweistufige Pipeline mit einer CodeCommit Quellphase und einer CodeDeploy Bereitstellungsphase.

Voraussetzungen:

Sie müssen die folgenden Ressourcen erstellt haben, um sie mit der AWS CloudFormation Beispielvorlage verwenden zu können:

- Sie müssen ein Quellrepository erstellt haben. Sie können das AWS CodeCommit Repository verwenden, in dem Sie es erstellt habenTutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen.
- Sie müssen eine CodeDeploy Anwendung und eine Bereitstellungsgruppe erstellt haben. Sie können die in Tutorial: Eine einfache Pipeline (CodeCommit Repository) erstellenerstellten CodeDeploy-Ressourcen verwenden.
- Wählen Sie einen dieser Links, um die AWS CloudFormation Beispielvorlagendatei zum Erstellen einer Pipeline herunterzuladen: YAML | JSON samples/codepipeline-codecommit-events-json.zip

  Entpacken Sie die Datei und speichern Sie sie auf Ihrem lokalen Computer.
- Laden Sie die SampleApp _ Linux.zip Beispielanwendungsdatei 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: Füge Beispielcode zu deinem CodeCommit Repository 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 in BetaFleet den Namen Ihrer CodeDeploy Bereitstellungsgruppe ein.

   c. Geben Sie in BranchName den Repository-Branch ein, den Sie verwenden möchten.

d. Geben Sie in RepositoryName 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 Capabilities die Option Ich bestätige, dass AWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden, und wählen Sie dann Create stack 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, der die Pipeline erstellt, 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-Managementkonsole 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 anzuzeigen, suchen Sie auf der Registerkarte Ressourcen nach der Spalte Logische ID für Ihren Stack in 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-Name) 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-Pipeline mit AWS CloudFormation

Diese exemplarische Vorgehensweise zeigt Ihnen, wie Sie mithilfe der AWS CloudFormation Konsole eine Infrastruktur erstellen, die eine Pipeline umfasst, die mit einem Amazon 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 wie Ihre Amazon CloudWatch Events-Regel und Ihren Amazon Events-Trail umfasst. CloudTrail Nachdem Sie Ihren Ressourcenstapel erstellt haben AWS CloudFormation, können Sie Ihre Pipeline in der Konsole anzeigen. AWS CodePipeline Die Pipeline ist eine zweistufige Pipeline mit einer Amazon S3-Quellphase und einer CodeDeploy Bereitstellungsphase.

Voraussetzungen:

Sie müssen über die folgenden Ressourcen verfügen, um die AWS CloudFormation Beispielvorlage verwenden zu können:

- Sie müssen die Amazon EC2-Instances erstellt haben, auf denen Sie den CodeDeploy Agenten installiert haben. Sie müssen eine CodeDeploy Anwendung und eine Bereitstellungsgruppe erstellt haben. Verwenden Sie Amazon EC2 und die CodeDeploy Ressourcen, in Tutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen denen Sie erstellt haben.
- Wählen Sie die folgenden Links, um die AWS CloudFormation Beispielvorlagendateien für die Erstellung einer Pipeline mit einer Amazon S3-Quelle herunterzuladen:
  - Laden Sie die Beispielvorlage für Ihre Pipeline herunter: YAML | JSON
  - Laden Sie die Beispielvorlage für Ihren CloudTrail Bucket und Ihren Trail herunter: YAML | JSON samples/codepipeline-s3-cloudtrail-json.zip
  - 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. 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 in BetaFleet den Namen Ihrer CodeDeploy Bereitstellungsgruppe ein. Sie können den `DemoFleet`-Standardnamen ersetzen.

   c. Geben SourceObjectKey Sie 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 Capabilities die Option Ich bestätige, dass AWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden, und wählen Sie dann Create stack 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, der die Pipeline erstellt, 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-Pipeline erstellen AWS CloudFormation zu können:

```
{
     "Effect": "Allow",
```

```
    "Action": [
       "events:PutRule",
       "events:PutEvents",
       "events:PutTargets",
       "events:DeleteRule",
       "events:RemoveTargets",
       "events:DescribeRule"
    ],
    "Resource": "resource_ARN"
}
```

7. Sehen Sie sich CloudFormation auf der Registerkarte Ressourcen für Ihren Stack die Ressourcen an, die für Ihren Stack erstellt wurden.

> **Note**
>
> Um die erstellte Pipeline anzuzeigen, suchen Sie unter dem Tab Ressourcen für Ihren Stack nach der Spalte Logische ID in 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-Name) 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 CloudFormation neu erstellt wird. Öffnen Sie die Amazon S3-Konsole und suchen Sie nach Ihrem `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-Managementkonsole 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.
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 von 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:

   - SourceObjectKeyÜbernehmen 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-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-Managementkonsole 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: Erstellen Sie eine Pipeline, die Variablen von verwendet AWS CloudFormation Aktionen zur Bereitstellung

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, der auf der von der AWS CloudFormation Aktion erzeugten `StackName` Variablen basiert. Nach einer manuellen Genehmigung führen Sie den Änderungssatz aus und erstellen dann eine Stack-Löschaktion, die den Stack basierend auf der `StackName`-Variablen löscht.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Themen

- Voraussetzungen: Erstellen Sie eine AWS CloudFormation Servicerolle und ein CodeCommit Repository
- Schritt 1: Laden Sie das Beispiel herunter, bearbeiten Sie es und laden Sie es hoch AWS CloudFormation Vorlage
- Schritt 2: Erstellen der Pipeline
- Schritt 3: Fügen Sie eine hinzu CloudFormation Bereitstellungsaktion zum Erstellen des Änderungssatzes
- 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 Stapel zu löschen

Voraussetzungen: Erstellen Sie eine AWS CloudFormation Servicerolle und ein CodeCommit Repository

Sie müssen bereits über Folgendes verfügen:

- Ein CodeCommit Repositorium. Sie können das AWS CodeCommit Repository verwenden, in dem Sie es erstellt habenTutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen.
- In diesem Beispiel wird ein Amazon 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 das Beispiel herunter, bearbeiten Sie es und laden Sie es hoch AWS CloudFormation Vorlage

Laden Sie die AWS CloudFormation Beispielvorlagendatei herunter und laden Sie sie in Ihr CodeCommit Repository hoch.

1. Navigiere zur Beispielvorlage für deine Region. Verwenden Sie beispielsweise die Tabelle unter, https://docs.aws.amazon.com/documentdb/latest/developerguide/quick_start_cfn.html#quick_start_cfn-launch_stack um die Region auszuwählen und die Vorlage herunterzuladen. Laden Sie 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 verwendest du die CodeCommit Konsole, um deine 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 Quellphase mit einer CodeCommit Aktion, bei der das Quellartefakt Ihre Vorlagendatei ist.
- Eine Bereitstellungsphase mit einer 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 Variablenreferenz.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS-Managementkonsole 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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyCFNDeployPipeline`**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
6. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:
   - Wählen Sie Neue Servicerolle, 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.

7. Unter Artifact store (Artefaktspeicher):

   a. Wählen Sie Standardspeicherort, um den Standard-Artefaktspeicher, wie z. B. den Amazon 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 haben, z. B. einen Amazon S3-Artefakt-Bucket, in derselben Region wie Ihre Pipeline.

   > **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 erhalten Sie unter Eingabe- und Ausgabe-Artefakte und CodePipeline Referenz zur Pipeline-Struktur.

   Wählen Sie Weiter aus.

8. In Schritt 3: Quellstufe 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 erstellt habenSchritt 1: Erstelle 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 muss.

   Wählen Sie Weiter aus.

9. Wählen Sie in Schritt 4: Build-Phase hinzufügen die Option Build-Phase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Skip auswählen.

   Wählen Sie Weiter aus.

10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

    Wählen Sie Weiter aus.

11. In Schritt 6: Bereitstellungsphase 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 in Schritt 7: Überprüfen die Option Pipeline erstellen aus. Sie sollten ein Diagramm sehen, das die Phasen der Pipeline zeigt. 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 hinzu CloudFormation Bereitstellungsaktion zum Erstellen des Änderungssatzes

Erstellen Sie eine nächste Aktion in Ihrer Pipeline, die es ermöglicht CloudFormation , den Änderungssatz vor der manuellen Genehmigungsaktion 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, 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 im Feld Aktionsname den Wert ein Change_Set. Wählen Sie im Feld Aktionsanbieter die Option aus AWS CloudFormation .

   b. Wählen Sie im Feld 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 Ressourcenänderungssatz überprüfen, 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, die es CloudFormation ermöglicht, den Änderungssatz nach der manuellen Genehmigungsaktion auszuführen.

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 im Feld Aktionsname den Wert ein Execute_Change_Set. Wählen Sie im Feld Aktionsanbieter die Option aus AWS CloudFormation.

b. Wählen Sie im Feld 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 Stapel zu löschen

Erstellen Sie eine letzte Aktion in Ihrer Pipeline, die es CloudFormation ermöglicht, den Stacknamen aus der Variablen in der Ausgabedatei abzurufen und den Stack zu löschen.

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 unter Aktionsname die Option aus 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, durchgängige CD-Pipeline (Continuous Deployment) mit Amazon ECS mit zu erstellen CodePipeline.

Important

Im Rahmen der Erstellung einer Pipeline in der Konsole wird ein S3-Artefakt-Bucket von CodePipeline für Artefakte verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Note

Dieses Tutorial bezieht sich auf die Amazon ECS-Standardbereitstellungsaktion für. CodePipeline Ein Tutorial, in dem Amazon ECS für CodeDeploy blue/green Bereitstellungsaktionen verwendet wird CodePipeline, finden Sie unterTutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-CodeDeploy.

Note

Dieses Tutorial bezieht sich auf die Amazon ECS-Standardbereitstellungsaktion CodePipeline mit einer Quellaktion. Ein Tutorial, das die Amazon ECS-Standardbereitstellungsaktion zusammen mit der ECRBuildAndPublish Build-Aktion verwendet, CodePipeline um Ihr Image

zu übertragen, finden Sie unterTutorial: Erstellen Sie ein Docker-Image und übertragen Sie es auf Amazon ECR mit CodePipeline (Typ V2).

Voraussetzungen

Es müssen bestimmte Ressourcen zur Verfügung stehen, damit Sie mithilfe dieses Tutorials eine CD-Pipeline erstellen können. Zum Einstieg benötigen Sie Folgendes:

> **Note**
>
> Alle diese Ressourcen sollten in derselben AWS Region erstellt werden.

- Ein Quellcodeverwaltungs-Repository (dieses Tutorial verwendet CodeCommit) mit Ihrem Dockerfile und Ihrer Anwendungsquelle. Weitere Informationen finden Sie im AWS CodeCommit Benutzerhandbuch unter Erstellen eines CodeCommit Repositorys.
- Ein Docker-Image-Repository (dieses Tutorial verwendet Amazon ECR), das ein Image enthält, das Sie aus Ihrer Dockerfile- und Anwendungsquelle erstellt haben. Weitere Informationen finden Sie unter Erstellen eines Repositorys und Übertragen eines Images im Amazon Elastic Container Registry Benutzerhandbuch.
- 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 Version 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 Überarbeitungen.

  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 und 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

Dieses Tutorial dient CodeBuild dazu, Ihr Docker-Image zu erstellen und das Image an Amazon ECR zu übertragen. Fügen Sie eine `buildspec.yml`-Datei zu Ihrem Quellcode-Repository hinzu, um CodeBuild mitzuteilen, wie zu verfahren ist. Die nachfolgende Beispiel-Buildspezifikation bewirkt Folgendes:

- Pre-build Stufe:
  - 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.
- Post-build Stufe:
  - Führen Sie die Push-Übertragung des Image zu Ihrem ECR-Repository mit beiden Tags durch.
  - Schreiben Sie eine Datei mit dem Namen `imagedefinitions.json` im Build-Stammverzeichnis, 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 Image 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 Beispiel-Aufgabendefinition geschrieben, die in Voraussetzungen bereitgestellt wurde und vom Amazon ECS-Service für dieses Tutorial verwendet wurde. 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.

Um einen hinzuzufügen **`buildspec.yml`** Datei zu deinem Quell-Repository

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 der Willkommensseite eine Einführungsseite angezeigt. Wählen Sie Get Started Now.
3. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie in Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Namen für Ihre Pipeline ein. Bei diesem Tutorial lautet der Pipeline-Name hello-world.
5. Behalten Sie unter Pipeline-Typ die Standardauswahl bei V2. Die Pipeline-Typen unterscheiden sich in ihren Eigenschaften und ihrem Preis. Weitere Informationen finden Sie unter Pipeline-Typen. Wählen Sie Weiter aus.
6. Wählen Sie auf der Seite Schritt 3: Quellstufe hinzufügen für Quellanbieter die Option AWS CodeCommit.

   a. Wählen Sie für Repository name (Repository-Name) den Namen des CodeCommit-Repositorys aus, das als Quellspeicherort für Ihre Pipeline verwendet werden soll.

b. Wählen Sie für Branch name (Zweigname) den zu verwendenden Zweig und dann Next (Weiter) aus.

7. Wählen Sie auf der Seite Schritt 4: Build-Phase hinzufügen für Build-Anbieter die Option AWS CodeBuild 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 Bild 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. Deaktivieren Sie CloudWatch Protokolle. Möglicherweise müssen Sie „Erweitert“ erweitern.

   i. Wählen Sie Weiter zu CodePipeline.

   j. Wählen Sie Weiter aus.

   > **Note**
   >
   > Der Assistent erstellt eine CodeBuild Servicerolle für Ihr Build-Projekt mit dem Namen codebuild- ***`build-project-name`*** -service-role. Notieren Sie sich diesen Rollennamen, wenn Sie ihm später Amazon ECR-Berechtigungen hinzufügen.

8. Wählen Sie auf der Seite Schritt 5: Bereitstellungsphase hinzufügen für 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.

9. Überprüfen Sie auf der Seite Step 6: Review Ihre Pipeline-Konfiguration und wählen Sie Create pipeline, 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 Build-Phase fehlschlägt. Der nächste Abschnitt fügt die Berechtigungen für die Build-Phase hinzu.

Schritt 3: Amazon ECR-Berechtigungen zur CodeBuild Rolle hinzufügen

Der CodePipeline Assistent erstellte eine IAM-Rolle für das CodeBuild Build-Projekt mit dem Namen codebuild- -service-role. ***build-project-name*** Für dieses Tutorial lautet der Name codebuild-hello-world-service-role. 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.

Um Amazon ECR-Berechtigungen zur Rolle hinzuzufügen CodeBuild

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im linken Navigationsbereich Roles aus.
3. Geben Sie in das Suchfeld codebuild- ein und wählen Sie die Rolle aus, die CodePipeline vom Assistenten erstellt wurde. In diesem Tutorial lautet der Name der Rolle code-build-hello-world-service-role.
4. Wählen Sie auf der Seite Summary (Zusammenfassung) die Option Attach policy (Richtlinie anfügen) aus.
5. Wählen Sie das Feld links neben der AmazonEC2ContainerRegistryPowerUser Richtlinie aus und wählen Sie Richtlinie anhängen aus.

Schritt 4: Testen Ihrer Pipeline

Ihre Pipeline sollte alles enthalten, was Sie für eine durchgängige native AWS kontinuierliche Bereitstellung benötigen. 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 anhand Ihrer Codeänderung erstellt wurde.

Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-CodeDeploy

In diesem Tutorial konfigurieren Sie eine Pipeline, AWS CodePipeline die Containeranwendungen mithilfe einer blue/green Bereitstellung bereitstellt, die Docker-Images unterstützt. In einer blue/green Bereitstellung können Sie die neue Version Ihrer Anwendung zusammen mit der alten Version starten und die neue Version testen, bevor Sie den Datenverkehr umleiten. Sie können auch den Bereitstellungsprozess überwachen und bei Problemen schnell ein Rollback durchführen.

Important

Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Note

Für dieses Tutorial muss das Amazon ECS Maßnahmen zur CodeDeploy blue/green Bereitstellung ergreifen. CodePipeline Ein Tutorial, das die Amazon ECS-Standardbereitstellungsaktion in verwendet CodePipeline, finden Sie unterTutorial: Amazon ECS-Standardbereitstellung mit CodePipeline.

Die fertige Pipeline erkennt Änderungen an Ihrem Image, das in einem Image-Repository wie Amazon ECR gespeichert ist. Es wird verwendet, CodeDeploy um den Datenverkehr an einen Amazon ECS-Cluster und 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 blue/green Bereitstellung verwendet werden, finden Sie unter Tutorial: Bereitstellen eines Amazon ECS-Service.

Die Pipeline ist auch so konfiguriert, dass sie einen Quellspeicherort verwendet, z. B. den Ort CodeCommit, an dem Ihre Amazon ECS-Aufgabendefinition gespeichert ist. In diesem Tutorial konfigurieren Sie jede dieser AWS Ressourcen und erstellen dann Ihre Pipeline mit Stufen, die Aktionen für jede Ressource enthalten.

Ihre Continuous Delivery-Pipeline erstellt und implementiert automatisch Container-Images, 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, Ihren Container-Namen, Ihren Amazon ECS-Servicenamen und Ihre 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 Produktionsdatenverkehr CodeDeploy 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 Detail.json Image-Datei für Amazon blue/green ECS-Bereitstellungsaktionen.

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
- Schritt 1: Image erstellen und in ein Amazon ECR-Repository übertragen
- Schritt 2: Erstellen Sie die Aufgabendefinition und die AppSpec Quelldateien und übertragen Sie sie in ein Repository CodeCommit
- Schritt 3: Erstellen Ihres Application Load Balancer und der Zielgruppen
- Schritt 4: Erstellen Sie Ihren Amazon ECS-Cluster und -Service
- Schritt 5: Erstellen Sie Ihre CodeDeploy Anwendung und Bereitstellungsgruppe (ECS-Rechenplattform)
- Schritt 6: Erstellen Ihrer Pipeline
- Schritt 7: Vornehmen einer Änderung an Ihrer Pipeline und Überprüfen der Bereitstellung

Voraussetzungen

Sie müssen die folgenden Ressourcen erstellt haben:

- Ein CodeCommit Repository. Sie können das AWS CodeCommit Repository verwenden, in dem Sie es erstellt habenTutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen.
- Starten Sie eine Amazon 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` Image 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. Übertragen Sie das Bild mit dem aus dem vorherigen Schritt auf Amazon ECR. `repositoryUri`

```
docker push 111122223333.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
```

Schritt 2: Erstellen Sie die Aufgabendefinition und die AppSpec Quelldateien und übertragen Sie sie in ein Repository CodeCommit

In diesem Abschnitt erstellen Sie eine JSON-Datei mit einer Aufgabendefinition und registrieren sie bei Amazon ECS. Anschließend erstellen Sie eine AppSpec Datei für CodeDeploy und verwenden Ihren Git-Client, 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 zur 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 dein CodeCommit Repository zu übertragen

1. Pushen Sie die Dateien oder laden Sie sie in Ihr CodeCommit Repository hoch. Diese Dateien sind das Quellartefakt, das vom Assistenten Create Pipeline (Pipeline erstellen) 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. Führe den folgenden Befehl aus, um die Dateien von deinem lokalen Repo in dein CodeCommit Repository zu übertragen:

```
git push
```

b. So verwendest du die CodeCommit Konsole, um deine 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 später mit Ihrem Load Balancer erstellen, 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 an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole 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.

Um einen Amazon EC2 Application Load Balancer zu erstellen

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon EC2-Konsole 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-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 aus.

> **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 (an einen Amazon ECS-Cluster und nicht an EC2-Instances). Um Ihren Amazon ECS-Service zu erstellen, müssen Sie die Subnetznamen, die Sicherheitsgruppe und den Zielgruppenwert 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 Cluster-Vorlage „Nur für Netzwerke“, die AWS Fargate verwendet, und wählen Sie dann Weiter.
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 aus.

Um einen Amazon ECS-Service zu erstellen

Verwenden Sie das 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 das `taskDefinition` Feld eine Aufgabendefinition in Amazon ECS registrieren, weisen Sie ihr eine Familie zu. 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-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 Verhalten bei der Verkehrsumleitung zu verweisen.

Um eine Anwendung zu erstellen CodeDeploy

1. Öffnen Sie die CodeDeploy Konsole und wählen Sie Create application.
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 unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
   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 die Option aus AWS-Service. Wählen Sie unter Wählen Sie einen Anwendungsfall aus CodeDeploy. Wählen Sie unter Wählen Sie Ihren Anwendungsfall aus 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 unter Umgebungskonfiguration Ihren Amazon ECS-Clusternamen und Servicenamen aus.
5. Wählen Sie unter Load Balancers den Namen des Load Balancers aus, der den Datenverkehr zu Ihrem Amazon ECS-Service weiterleitet.
6. Wählen Sie unter Production Listener Port den Port und das Protokoll für den Listener aus, der den Produktionsdatenverkehr zu Ihrem 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 festzulegen, wie lange nach einer erfolgreichen Bereitstellung der Datenverkehr an Ihre aktualisierte 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-Managementkonsole 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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyImagePipeline`**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
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 in Schritt 3: Quellstufe hinzufügen unter Quellanbieter die Option aus AWS CodeCommit. Wählen Sie in Repository name (Repository-Name) den Namen des CodeCommit-Repositorys aus, das Sie in Schritt 1: Erstelle ein CodeCommit Repository erstellt haben. Wählen Sie in Branch name den Namen des Branch aus, der das neueste Code-Update enthält.

   Wählen Sie Weiter aus.
9. Wählen Sie in Schritt 4: Build-Phase hinzufügen die Option Build-Phase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Skip auswählen. Wählen Sie Weiter aus.
10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

    Wählen Sie Weiter aus.
11. In Schritt 6: Bereitstellungsphase hinzufügen:

    a. Wählen Sie unter Bereitstellungsanbieter die Option Amazon ECS (Blue/Green) 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 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.

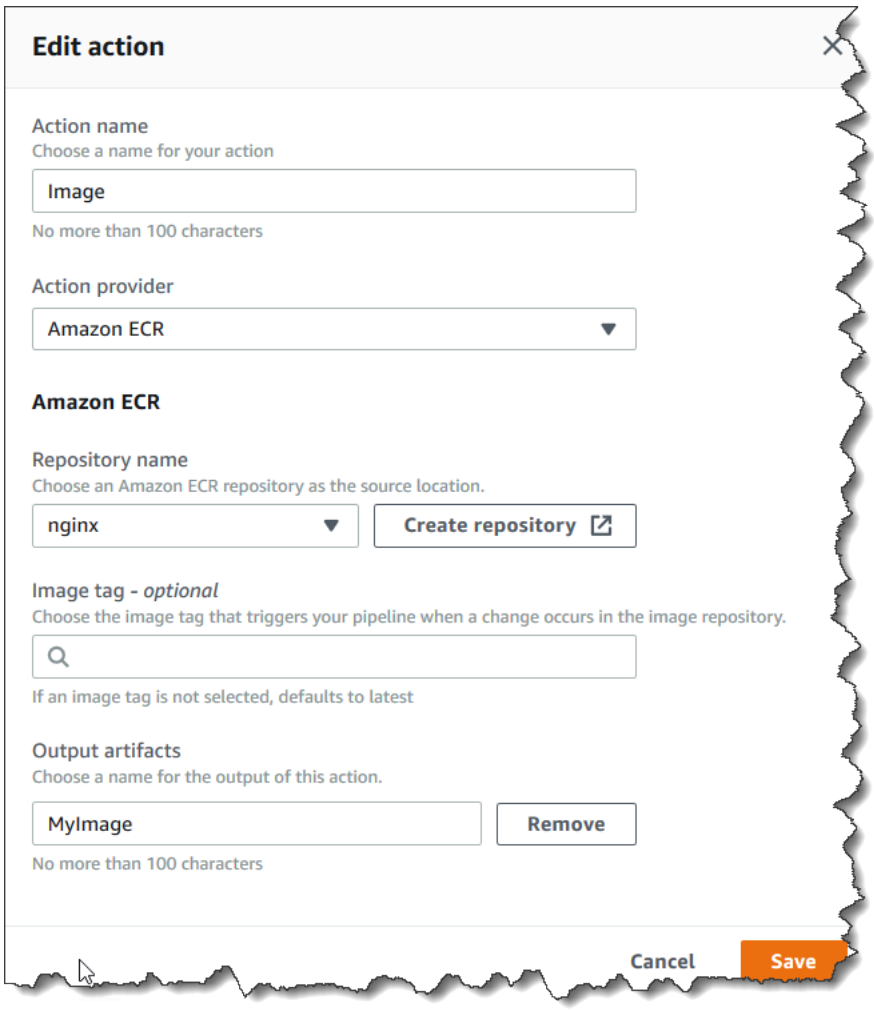
d. Wählen Sie Weiter aus.

12. In Schritt 7: Überprüfen, überprüfen Sie die Informationen und wählen Sie dann Create pipeline aus.

Um Ihrer Pipeline eine Amazon ECR-Quellaktion hinzuzufügen

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 parallele 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 muss.

So senden Sie Ihre Quellartefakte an die Bereitstellungsaktion

1. Wählen Sie in Ihrer Bereitstellungsphase die Option Bearbeiten aus und klicken Sie auf das Symbol, um die Amazon ECS (Blue/Green) -Aktion 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 in der Aufgabendefinition die Option aus, und überprüfen Sie dann SourceArtifact, ob die Eingabe erfolgt **`taskdef.json`** ist.
4. Wählen Sie im AWS CodeDeploy AppSpec Feld Datei die Option aus SourceArtifact, und überprüfen Sie dann, ob **`appspec.yaml`** der Wert eingegeben wurde.
5. Wählen Sie unter Bild der Aufgabendefinition dynamisch aktualisieren im Feld Eingabeartefakt mit Bild-URI die Option aus MyImage, 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 die Option Pipeline-Änderung speichern aus, und klicken Sie dann auf Änderung speichern. 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 anzusehen CodeDeploy und den Fortschritt der Verkehrsverlagerung zu verfolgen.

   > **Note**
   >
   > Möglicherweise wird ein Bereitstellungsschritt mit einer optionalen Wartezeit angezeigt. CodeDeploy Wartet nach einer erfolgreichen Bereitstellung standardmäßig eine Stunde, 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.

Important

Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

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.

Voraussetzungen

Sie müssen bereits über Folgendes verfügen:

- Ein CodeCommit Repository. Sie können das AWS CodeCommit Repository verwenden, in dem Sie es erstellt habenTutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen.
- 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.

- Verwenden Sie die Schritte unter generate-lwa-tokens, um ein Sicherheitsprofil zu erstellen.
- Notieren Sie sich die Client-ID und den Clientschlüssel, nachdem Sie das Profil erstellt haben.
- Stellen Sie sicher, dass Sie den Wert „Zulässige Rücksendung“ URLs wie in den Anweisungen angegeben eingeben. Sie URLs ermöglichen es dem ASK CLI-Befehl, Aktualisierungstoken-Anfragen umzuleiten.

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-Managementkonsole 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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
5. Geben Sie in Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Wert ein**`MyAlexaPipeline`**.
6. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in Eigenschaften und Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
7. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
8. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
9. Wählen Sie in Schritt 3: Quellstufe hinzufügen im Feld Quellanbieter die Option AWS CodeCommit. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, in dem Sie es erstellt habenSchritt 1: Erstelle ein CodeCommit Repository. 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.

10. Wählen Sie in Schritt 4: Build-Phase hinzufügen die Option Build-Phase überspringen und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken.

    Wählen Sie Weiter aus.

11. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

    Wählen Sie Weiter aus.

12. Gehen Sie in Schritt 6: Bereitstellungsphase hinzufügen wie folgt vor:

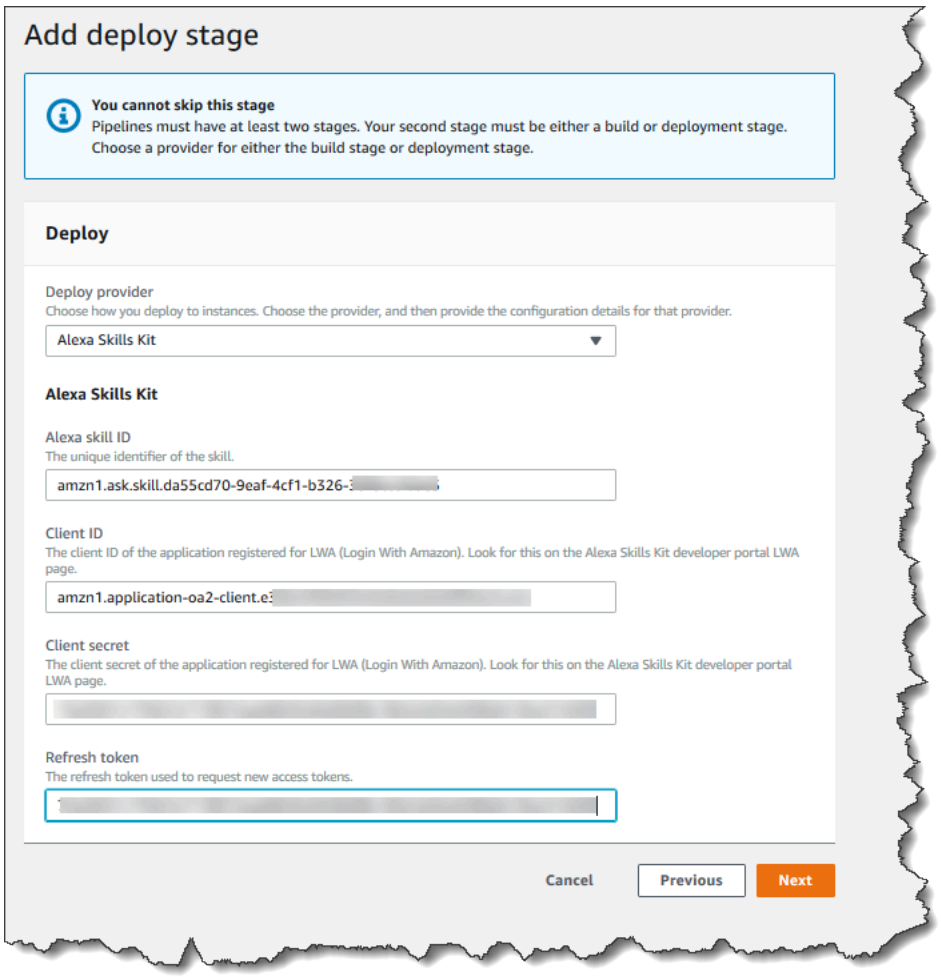
    a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Alexa Skills Kit.

    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.

13. Überprüfen Sie in Schritt 7: Überprüfen Sie die Informationen und wählen Sie dann Pipeline erstellen 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.

Important

Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

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: Eine einfache Pipeline (CodeCommit Repository) erstellen.
- 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 stellen das Quellartefakt dar, das vom Assistenten Create Pipeline (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-Managementkonsole 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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie in Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Wert ein**`MyS3DeployPipeline`**.

5. Wählen Sie unter Pipeline-Typ die Option V2 aus. Weitere Informationen finden Sie unter Pipeline-Typen. Wählen Sie Weiter aus.
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 in Schritt 3: Quellstufe hinzufügen im Feld Quellanbieter die Option AWS CodeCommit. Wählen Sie in Repository name (Repository-Name) den Namen des CodeCommit-Repositorys aus, das Sie in Schritt 1: Erstelle ein CodeCommit Repository 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.
9. Wählen Sie in Schritt 4: Build-Phase hinzufügen die Option Build-Phase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

   Wählen Sie Weiter aus.
10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

    Wählen Sie Weiter aus.
11. Gehen Sie in Schritt 6: Bereitstellungsphase hinzufügen wie folgt vor:

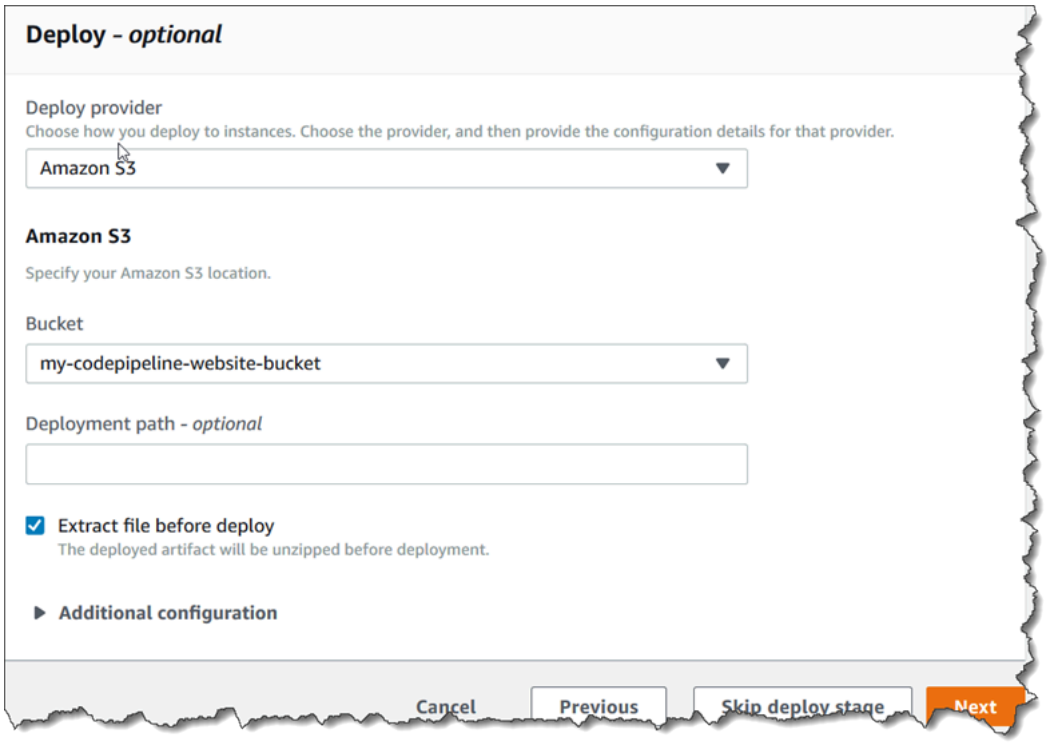
    a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Amazon S3.

    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.

12. Überprüfen Sie in Schritt 7: Überprüfen Sie die Informationen und wählen Sie dann Pipeline erstellen aus.

13. 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
```

14. 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 Upload. Diese Dateien stellen das Quellartefakt dar, das vom Assistenten Create Pipeline (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-Managementkonsole 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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie in Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Wert ein**`MyS3DeployPipeline`**.
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 3: Quellstufe hinzufügen unter Quellanbieter die Option Amazon S3 aus. 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.
8. In Schritt 4: Build-Phase 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 aws/codebuild/ standard Sie für die Runtime-Version aus:1.0.
   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.

9. In Schritt 5: Bereitstellungsphase hinzufügen:

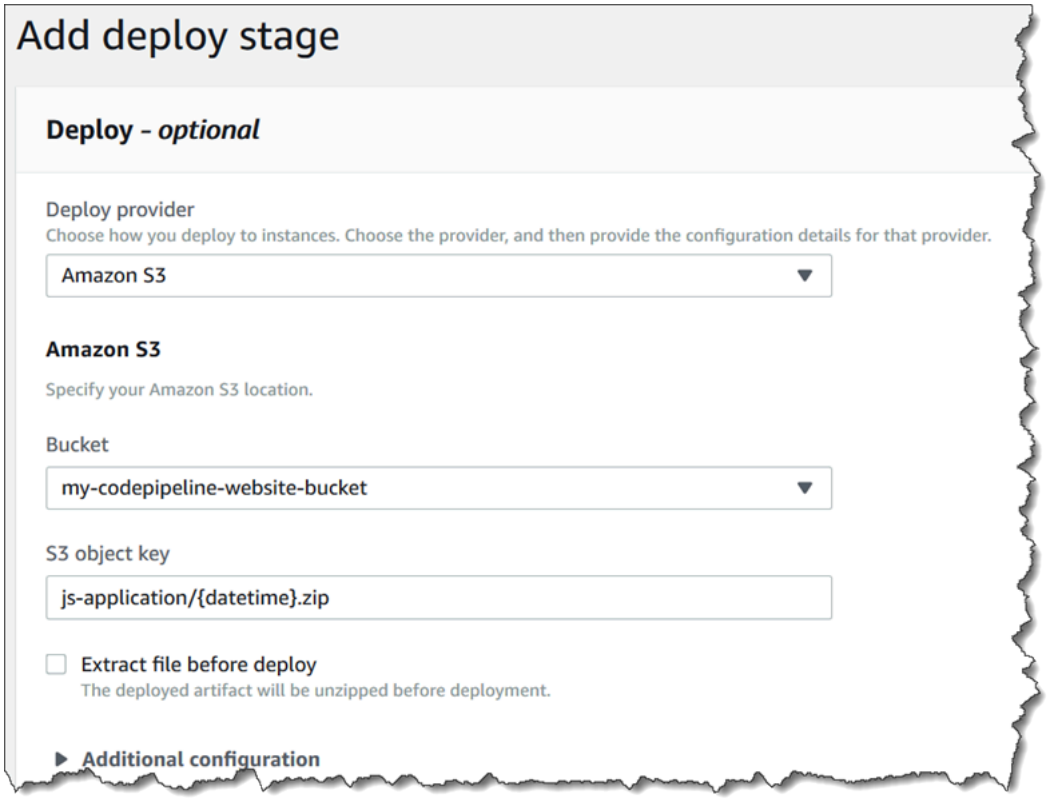
a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Amazon S3.

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.

10. Prüfen Sie in Step 6: Review die Informationen und wählen Sie dann Create pipeline aus.

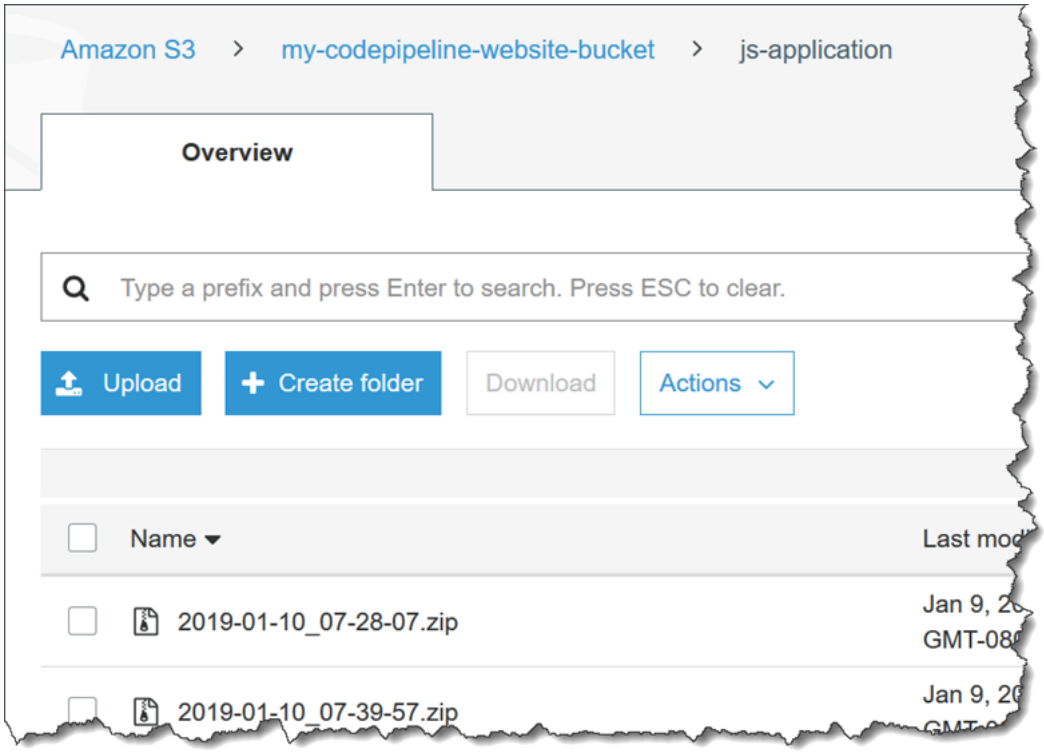
11. Nachdem Ihre Pipeline erfolgreich ausgeführt wurde, sehen Sie sich Ihren Bucket in der Amazon S3 S3-Konsole an. Vergewissern Sie sich, dass Ihre bereitgestellte ZIP-Datei in Ihrem Ziel-Bucket unter dem Ordner `js-application` angezeigt wird. Die in der JavaScript ZIP-Datei enthaltene Datei sollte 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 veröffentlicht AWS Serverless Application Repository

Sie können AWS CodePipeline es verwenden, um Ihre AWS SAM serverlose Anwendung kontinuierlich an die bereitzustellen. AWS Serverless Application Repository

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline für Artefakte verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

Dieses Tutorial zeigt, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre serverlose Anwendung zu erstellen, auf der gehostet wird, GitHub und sie automatisch auf der zu veröffentlichen. AWS Serverless Application Repository 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. *`template.yml`* Ersetzen Sie sie 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 dort zu erstellen AWS-Region , wo Sie Ihre serverlose Anwendung veröffentlichen möchten.

1. Melden Sie sich bei der an AWS-Managementkonsole 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 auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) aus, oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
4. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
5. Wählen Sie Create pipeline (Pipeline erstellen) aus. Geben Sie auf der Seite Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Namen für Ihre Pipeline ein.
6. Wählen Sie unter Pipeline-Typ die Option V2 aus. Weitere Informationen finden Sie unter Pipeline-Typen. Wählen Sie Weiter aus.
7. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
8. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
9. Wählen Sie auf der Seite Schritt 3: Quellstufe hinzufügen im Feld Quellanbieter die Option aus GitHub.
10. 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.
11. Wählen Sie unter Repository Ihr GitHub Quell-Repository aus.
12. Wählen Sie unter Branch Ihren GitHub Branch aus.
13. Behalten Sie die übrigen Standardeinstellungen für die Quellaktion bei. Wählen Sie Weiter aus.
14. Fügen Sie auf der Seite Schritt 4: 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.

15. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

    Wählen Sie Weiter aus.

16. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken. Wählen Sie Weiter aus.

17. Wählen Sie auf Schritt 7: Überprüfen die Option Pipeline erstellen aus. Sie sollten ein Diagramm sehen, das die Phasen zeigt.

18. 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. *`bucketname`*Ersetzen Sie es 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 CloudFormation Stack, um die 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 beim Erstellen der neuen Veröffentlichungsaktion in 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 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: Bereitstellung von Lambda-Funktionen mit CodePipeline

Dieses Tutorial hilft Ihnen, eine Bereitstellungsaktion zu erstellen CodePipeline , die Ihren Code für Ihre Funktion bereitstellt, die Sie in Lambda konfiguriert haben. Sie erstellen eine Lambda-Beispielfunktion, in der Sie einen Alias und eine Version erstellen, die komprimierte Lambda-Funktion zu Ihrem Quellspeicherort hinzufügen und die Lambda-Aktion in Ihrer Pipeline ausführen.

> **Note**
>
> Im Rahmen der Erstellung einer Pipeline in der Konsole wird ein S3-Artefakt-Bucket von for artefacts verwendet. CodePipeline (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

> **Note**
>
> Die Lambda Bereitstellungsaktion ist nur für Pipelines vom Typ V2 verfügbar.

Voraussetzungen

Es müssen bestimmte Ressourcen zur Verfügung stehen, damit Sie mithilfe dieses Tutorials eine CD-Pipeline erstellen können. Zum Einstieg benötigen Sie Folgendes:

> **Note**
>
> Alle diese Ressourcen sollten in derselben AWS Region erstellt werden.

- Ein Quellcodeverwaltungs-Repository wie GitHub, oder ein S3-Quell-Bucket (dieses Tutorial verwendet S3), in dem Sie eine `.zip` Datei speichern, die Sie für Ihre Lambda-Funktion erstellen.
- Sie müssen eine vorhandene CodePipeline Servicerolle verwenden, die mit den Berechtigungen für diese Aktion aktualisiert wurde. Informationen zum Aktualisieren Ihrer Servicerolle finden Sie unterBerechtigungen für die Lambda-Bereitstellungsaktion (Richtlinie) für die Aktion „Lambda-Bereitstellung“.

Nachdem Sie diese Voraussetzungen erfüllt haben, können Sie mit dem Tutorial fortfahren und Ihre CD-Pipeline erstellen.

Schritt 1: Erstellen Sie Ihre Lambda-Beispielfunktion

In diesem Schritt erstellen Sie die Lambda-Funktion, die Sie bereitstellen werden.

Um Ihre Lambda-Funktion zu erstellen

1. Rufen Sie die Lambda-Konsole auf und folgen Sie den Schritten im folgenden Tutorial, um eine Lambda-Beispielfunktion zu erstellen: Link.
2. Wählen Sie in der oberen Navigationsleiste Erstellen und dann oben auf der Seite die Option Von vorne beginnen aus.
3. Geben Sie unter Name **`MyLambdaFunction`** ein.
4. Veröffentlichen Sie eine neue Version. Dies ist die Version, auf die der Alias verweisen wird.

a. Wählen Sie Ihre Funktion aus.
b. Wählen Sie das Drop-down-Menü Aktionen aus.
c. Wählen Sie Publish new version (Neue Version veröffentlichen) aus.
d. (Optional) Zur Beschreibung unter Beschreibung hinzufügen.
e. Wählen Sie Publish.

5. Erstellen Sie einen Alias für Ihre Funktion, z. `aliasV1` B.
6. Stellen Sie sicher, dass der Alias auf die Version verweist, die Sie gerade erstellt haben (z. B. 1).

> **Note**
>
> Wenn Sie $LATEST wählen, können Sie keine Funktionen zur Verkehrsverlagerung verwenden, da Lambda $LATEST nicht für einen Alias unterstützt, der auf mehr als eine Version verweist.

Schritt 2: Laden Sie die Funktionsdatei in Ihr Repository hoch

Laden Sie die Funktion herunter und speichern Sie sie als Zip-Datei. Laden Sie die komprimierte Datei mithilfe der folgenden Schritte in Ihren S3-Bucket hoch.

So fügen Sie eine **`.zip`**-Datei zu Ihrem Quell-Repository hinzu

1. Öffnen Sie Ihren S3-Bucket.
2. Klicken Sie auf Upload.
3. Laden Sie die Zip-Datei mit Ihrer `sample_lambda_source.zip` Datei in Ihren Quell-Bucket hoch.

   Notieren Sie sich den Pfad.

   ```
   object key
   ```

Schritt 3: Erstellen Sie Ihre Pipeline

Verwenden Sie den CodePipeline Assistenten, um Ihre Pipeline-Stufen zu erstellen und Ihr Quell-Repository zu verbinden.

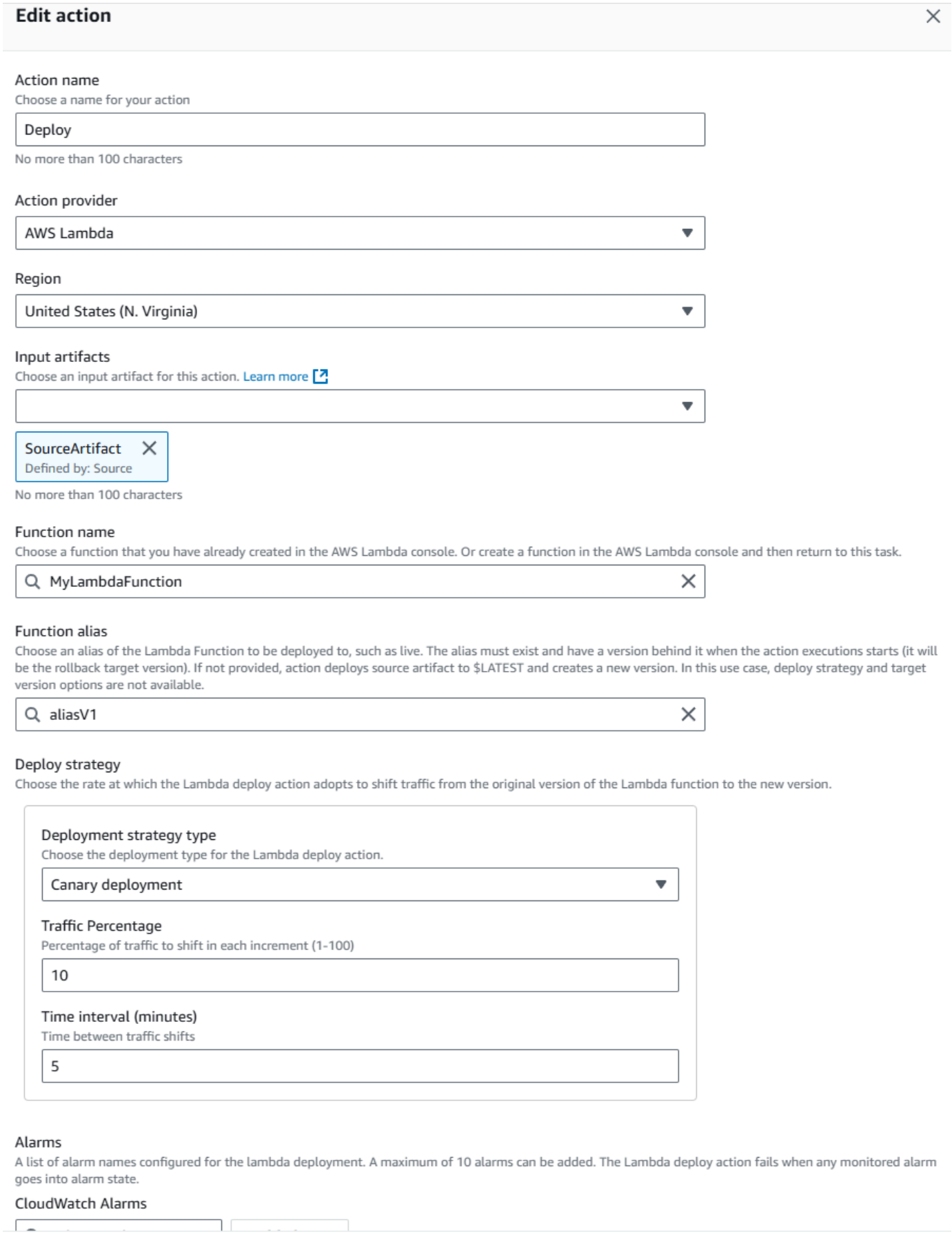
So erstellen Sie Ihre Pipeline

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie in Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Wert ein**`MyPipeline`**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in Eigenschaften und Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
6. Wählen Sie unter Servicerolle die Option Bestehende Servicerolle verwenden und wählen Sie dann die CodePipeline Servicerolle aus, die mit den erforderlichen Berechtigungen für diese Aktion aktualisiert wurde. Informationen zum Konfigurieren Ihrer CodePipeline Servicerolle für diese Aktion finden Sie unterBerechtigungen für die Lambda-Bereitstellungsaktion (Richtlinie) für die Aktion „Lambda-Bereitstellung“.
7. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
8. Fügen Sie auf der Seite Schritt 3: Quellstufe hinzufügen eine Quellstufe hinzu:

   a. Wählen Sie unter Source provider (Quellanbieter) die Option Amazon S3 aus.

   b. Fügen Sie im Feld Objektschlüssel den Namen Ihrer ZIP-Datei einschließlich der Dateierweiterung hinzu, z. B. `sample_lambda_source.zip`

   Wählen Sie Weiter aus.
9. Wählen Sie auf der Seite Schritt 4: Buildphase hinzufügen die Option Überspringen aus.
10. Wählen Sie auf der Seite Schritt 5: Testphase hinzufügen die Option Überspringen aus.
11. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Lambda aus.

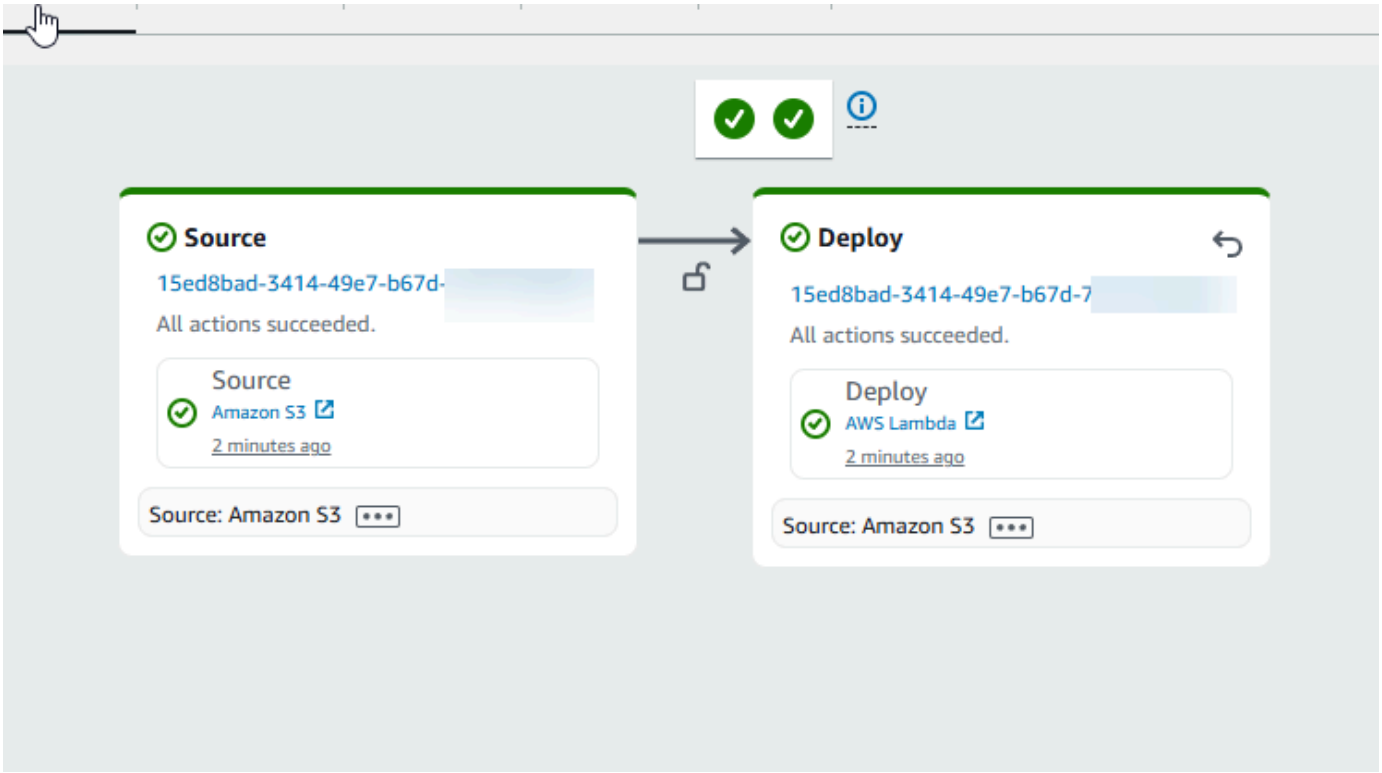

Edit action
Action name
Choose a name for your action
Deploy
No more than 100 characters
Action provider
AWS Lambda
Region
United States (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
Defined by: Source
No more than 100 characters
Function name
Choose a function that you have already created in the AWS Lambda console. Or create a function in the AWS Lambda console and then return to this task.
MyLambdaFunction
Function alias
Choose an alias of the Lambda Function to be deployed to, such as live. The alias must exist and have a version behind it when the action executions starts (it will be the rollback target version). If not provided, action deploys source artifact to $LATEST and creates a new version. In this use case, deploy strategy and target version options are not available.
aliasV1
Deploy strategy
Choose the rate at which the Lambda deploy action adopts to shift traffic from the original version of the Lambda function to the new version.
Deployment strategy type
Choose the deployment type for the Lambda deploy action.
Canary deployment
Traffic Percentage
Percentage of traffic to shift in each increment (1-100)
10
Time interval (minutes)
Time between traffic shifts
5
Alarms
A list of alarm names configured for the lambda deployment. A maximum of 10 alarms can be added. The Lambda deploy action fails when any monitored alarm goes into alarm state.
CloudWatch Alarms
Cancel
Done

a. Fügen Sie Ihren Funktionsnamen und Alias hinzu.

b. Wählen Sie Ihre Bereitstellungsstrategie.

c. Wählen Sie Weiter aus.

12. Überprüfen Sie auf der Seite Schritt 7: Überprüfen Ihre Pipeline-Konfiguration und wählen Sie Pipeline erstellen aus, um die Pipeline zu erstellen.



Schritt 4: Testen Ihrer Pipeline

Ihre Pipeline sollte alles enthalten, 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 Ihre Aktion wird für Ihre Lambda-Funktion bereitgestellt.

Weitere Informationen

Die Lambda-Bereitstellungsaktion ermöglicht zwei Bereitstellungsmethoden. Eine Methode ist die alleinige Verkehrsverlagerung ohne ein Eingabeartefakt aus der Quellaktion. Die andere Methode besteht darin, den Funktionscode mithilfe eines Eingabeartefakts aus der Quellaktion zu aktualisieren und dann eine neue Version auf der Grundlage des aktualisierten Codes zu veröffentlichen. Bei der zweiten Methode, wenn der Alias angegeben wird, CodePipeline wird der Traffic ebenfalls verlagert. Dieses Tutorial zur Lambda-Deploy-Aktion zeigt, wie Sie Ihre Funktion mithilfe eines Quellartefakts aktualisieren.

Weitere Informationen zur Aktion finden Sie auf der Aktionsreferenzseite unter. AWS Lambda Aktionsreferenz bereitstellen

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. Variablenreferenz

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

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
- Schritt 1: Erstellen einer Lambda-Funktion
- Schritt 2: Fügen Sie Ihrer Pipeline eine Lambda-Aufrufaktion und eine manuelle Genehmigungsaktion hinzu

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Sie können die Pipeline mit der Quelle erstellen oder verwenden. CodeCommit Tutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen
- Bearbeiten Sie Ihre vorhandene Pipeline, sodass 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.

Um eine Lambda-Funktion und eine Ausführungsrolle zu erstellen

1. Melden Sie sich bei der an AWS-Managementkonsole 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 Ihrer Pipeline später im Tutorial konfigurieren, aber Sie fügen den Code unter der Annahme hinzu, dass die Variable übergeben wird.

   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 putJobSuccess = async (message) => {
        const params = {
            jobId: jobId,
            outputVariables: {
                testRunId: Math.floor(Math.random() * 1000).toString(),
                dateTime: Date(Date.now()).toString(),
                region: lambdaRegion
            }
        };
```

   Fügen Sie in Ihrer neuen Funktion auf der Registerkarte Code den folgenden Beispielcode unter ein`index.mjs`.

```
import { CodePipeline } from '@aws-sdk/client-codepipeline';

export const handler = async (event, context) => {
    const codepipeline = new CodePipeline({});

    // Retrieve the Job ID from the Lambda action
    const jobId = event["CodePipeline.job"].id;

    // Retrieve UserParameters
    const params =
 event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;

    // The region from where the lambda function is being executed
    const lambdaRegion = process.env.AWS_REGION;
```

```
    // Notify CodePipeline of a successful job
    const putJobSuccess = async (message) => {
        const params = {
            jobId: jobId,
            outputVariables: {
                testRunId: Math.floor(Math.random() * 1000).toString(),
                dateTime: Date(Date.now()).toString(),
                region: lambdaRegion
            }
        };

        try {
            await codepipeline.putJobSuccessResult(params);
            return message;
        } catch (err) {
            throw err;
        }
    };

    // Notify CodePipeline of a failed job
    const putJobFailure = async (message) => {
        const params = {
            jobId: jobId,
            failureDetails: {
                message: JSON.stringify(message),
                type: 'JobFailed',
                externalExecutionId: context.invokeid
            }
        };

        try {
            await codepipeline.putJobFailureResult(params);
            throw message;
        } catch (err) {
            throw err;
        }
    };

    try {
        console.log("Testing commit - " + params);

        // Your tests here
```

```
        // Succeed the job
        return await putJobSuccess("Tests passed.");
    } catch (ex) {
        // If any of the assertions failed then fail the job
        return await putJobFailure(ex);
    }
};
```

7. Erlauben Sie der Funktion, automatisch zu speichern.
8. Kopieren Sie den Amazon-Ressourcennamen (ARN), der im Feld Function ARN oben auf dem Bildschirm enthalten ist.
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: Erstellen Sie die Lambda-Funktion .

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.

Um der Pipeline eine Lambda-Aktion und eine manuelle Genehmigungsaktion hinzuzufügen

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 Aktionsgruppe hinzufügen, um eine Aktion hinzuzufügen. Geben Sie unter Action name (Aktionsname) den Namen der Aufrufaktion ein, z. B. **`Test_Commit`**.

c. Wählen Sie unter Aktionsanbieter die Option AWS Lambda.

d. Wählen Sie unter Input artifacts (Eingabeartefakte) den Namen des Ausgabeartefakts Ihrer Quellaktion aus, z. B. `SourceArtifact`.

e. Fügen Sie FunctionName unter den ARN der Lambda-Funktion hinzu, die Sie erstellt haben.

f. Fügen Sie unter Variable namespace (Variablennamespace) den Namespace-Namen hinzu, z. B. **`TestVariables`**.

g. Fügen Sie im Feld Ausgabeartefakte den Namen des Ausgabeartefakts hinzu, z. B. **`LambdaArtifact`**

h. Wählen Sie Fertig 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 `SourceVariables` Standard-Namespace. 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.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

In diesem Tutorial führen Sie folgende Aufgaben aus:

- Erstellen Sie eine 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
- Schritt 1: Erstellen des Beispielzustandsautomaten
- Schritt 2: Fügen Sie Ihrer Pipeline eine Aktion zum Aufrufen von Step Functions hinzu

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: Eine einfache Pipeline (CodeCommit Repository) erstellen 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 Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu.

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-Managementkonsole 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 Fertig 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.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem

anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

Themen

- Voraussetzungen
- Schritt 1: Erstellen Sie Ihre Ressourcen AWS AppConfig
- Schritt 2: Laden Sie Dateien in Ihren S3-Quell-Bucket hoch
- 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 Sie einen S3-Quell-Bucket 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-Managementkonsole.
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-Managementkonsole 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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie in Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Wert ein**`MyAppConfigPipeline`**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in Eigenschaften und Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
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 in Schritt 3: Quellstufe hinzufügen unter Quellanbieter die Option Amazon S3 aus. Wählen Sie unter 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.

9. Wählen Sie in Schritt 4: Buildphase hinzufügen die Option Buildphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken.

   Wählen Sie Weiter aus.

10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

    Wählen Sie Weiter aus.

11. Gehen Sie in Schritt 6: Bereitstellungsphase hinzufügen wie folgt vor:

    a. Wählen Sie unter Anbieter bereitstellen die Option aus AWS AppConfig.

    b. Wählen Sie unter Anwendung den Namen der Anwendung aus, in der Sie sie erstellt haben AWS AppConfig. 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.

12. Überprüfen Sie in Schritt 7: Überprüfen die Informationen und wählen Sie dann Pipeline erstellen 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.

> **Note**
>
> Die hier beschriebene Option zum vollständigen Klonen bezieht sich auf die Angabe, ob Repository-Metadaten geklont werden CodePipeline sollen. Diese können nur von CodeBuild Befehlen verwendet werden. Um ein GitHub Benutzerzugriffstoken zur Verwendung mit CodeBuild Projekten zu verwenden, folgen Sie den hier aufgeführten Schritten, um den AWS Connector für die GitHub App zu installieren, und lassen Sie dann das Feld App-Installation leer. AWS CodeConnections verwendet das Benutzerzugriffstoken für die Verbindung.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

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 ausführt, 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) oder AWS GovCloud (US-West) nicht verfügbar. Weitere verfügbare 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 und GitLab.com GitLab selbstverwaltete Aktionen.

Themen

- Voraussetzungen
- Schritt 1: Erstellen Sie eine README-Datei
- Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt
- Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um Verbindungen zu verwenden
- Schritt 4: Repository-Befehle in der Build-Ausgabe anzeigen

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-Managementkonsole , 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` 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 bei der CodePipeline Konsole an unter https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie in Schritt 2: Pipeline-Einstellungen auswählen im Feld Pipeline-Name den Wert ein**`MyGitHubPipeline`**.
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 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.

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 Schritt 3: Quellstufe hinzufügen eine Quellstufe hinzu:

   a. Wählen Sie unter Quellanbieter die Option GitHub (über GitHub App) 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.

      Sie installieren eine App für alle Verbindungen mit einem bestimmten Anbieter. Wenn Sie den AWS Connector für die GitHub App bereits installiert haben, wählen Sie ihn aus und überspringen Sie diesen Schritt.

      > **Note**
      >
      > Wenn Sie ein Benutzerzugriffstoken erstellen möchten, stellen Sie sicher, dass Sie den AWS Connector für die GitHub App bereits installiert haben, und lassen Sie dann das Feld App-Installation leer. AWS CodeConnections verwendet das Benutzerzugriffstoken für die Verbindung. Weitere Informationen finden Sie unter Zugriff auf Ihren Quellanbieter in CodeBuild.

   c. Wählen Sie im Feld Repository name (Repositoryname) den Namen Ihres GitHub-Repositorys aus.

   d. Wählen Sie unter Branchname 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.

9. Fügen Sie in Schritt 4: 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 aws/codebuild /standard:5.0.

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.

> **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.

10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen wählen.

    Wählen Sie Weiter aus.

11. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken. Wählen Sie Weiter aus.

12. Wählen Sie auf Schritt 7: Überprüfen die Option 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 `codestar-connections: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, GitHub Enterprise Server oder hinzu GitHub GitLab.com

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 zum vollständigen 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 zum vollständigen Klonen 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. Auch wenn Ihre Pipeline kontoübergreifende Aktionen enthalten kann, müssen sich die CodeCommit Aktion und die CodeBuild Aktion im selben Konto befinden, damit die vollständige Clone-Option erfolgreich ist.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Themen

- Voraussetzungen
- Schritt 1: Erstellen Sie eine README-Datei
- Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt
- Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um das Repository zu klonen
- Schritt 4: Repository-Befehle in der Build-Ausgabe anzeigen

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 eine README-Datei zu Ihrem Quell-Repository hinzuzufügen. Die README-Datei enthält ein Beispiel für eine Quelldatei, die von der CodeBuild Downstream-Aktion gelesen werden kann.

Um eine README-Datei hinzuzufügen

1. Logge dich in dein Repository ein und wähle dein Repository aus.
2. Um eine neue Datei zu erstellen, wähle Datei hinzufügen > Datei erstellen. Benennen Sie die Datei `README.md` 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 bei der CodePipeline Konsole unter an https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.

4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyCodeCommitPipeline`**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
6. 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 für Ihre Servicerollenrichtlinie verfügen. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline Servicerolle.
7. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie Weiter aus.
8. Gehen Sie auf der Seite Schritt 3: 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. Wähle unter Filialname deinen Filialnamen aus.

   d. Stellen Sie sicher, dass die Option Starten der Pipeline bei Änderung des Quellcodes ausgewählt ist.

   e. Wählen Sie unter Ausgabe-Artefaktformat 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.
9. Gehen Sie in Schritt 4: Build-Phase 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 die Option aws/codebuild /standard:5.0.

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

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

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 es wird ein CodeBuild Projekt erstellt, 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.

10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

    Wählen Sie Weiter aus.

11. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus, und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken. Wählen Sie Weiter aus.

12. Wählen Sie auf Schritt 7: Überprüfen die Option Pipeline erstellen aus.

Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um das Repository zu klonen

Die erste Ausführung der Pipeline schlägt fehl, da Sie die CodeBuild Servicerolle mit den Berechtigungen aktualisieren müssen, Daten aus Ihrem Repository abzurufen.

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 anzusehen

1. Wenn Ihre Servicerolle erfolgreich aktualisiert wurde, wählen Sie in der CodeBuild Phase „Fehlgeschlagen“ die Option „Erneut versuchen“.
2. Nachdem die Pipeline erfolgreich ausgeführt wurde, wählen Sie in Ihrer erfolgreichen Build-Phase die Option Details anzeigen aus.

   Wählen Sie auf der Detailseite den Tab Logs. 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 führen es aus`git describe --all`.

Tutorial: Erstellen Sie eine Pipeline mit AWS CloudFormation StackSets Aktionen zur Bereitstellung

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 Instances, in denen das Stack-Set bereitgestellt wird.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Es gibt zwei Möglichkeiten, Berechtigungen für ein Stack-Set zu verwalten: selbstverwaltete und AWS-verwaltete IAM-Rollen. Dieses Tutorial enthält Beispiele für selbstverwaltete Berechtigungen.

Um es am effektivsten nutzen StackSets zu können CodePipeline, sollten Sie ein klares Verständnis der dahinter stehenden Konzepte CloudFormation StackSets und ihrer Funktionsweise haben. Die StackSets Konzepte finden Sie im AWS CloudFormation Benutzerhandbuch.

Themen

- Voraussetzungen
- Schritt 1: Laden Sie das Beispiel hoch AWS CloudFormation Vorlage und Parameterdatei
- Schritt 2: Erstellen der Pipeline
- Schritt 3: Erste Bereitstellung anzeigen
- Schritt 4: Fügen Sie eine CloudFormationStackInstances Aktion hinzu
- Schritt 5: Sehen Sie sich die Stack-Set-Ressourcen für Ihre Bereitstellung an
- Schritt 6: Aktualisieren Sie Ihr Stack-Set

Voraussetzungen

Für Stack-Set-Operationen verwenden Sie zwei verschiedene Konten: ein Verwaltungskonto 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 das Beispiel hoch AWS CloudFormation Vorlage und Parameterdatei

Erstellen Sie einen Quell-Bucket für Ihre Stack-Set-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 an AWS-Managementkonsole und öffnen Sie die Amazon S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Create Bucket (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 Create Bucket (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 das zu erstellen AWS CloudFormation Vorlagendatei

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, Validierung und globale Ereignisse 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 zum Erstellen einer Pipeline für Ihre Bereitstellungsaktion in CodePipeline erstellt wurde.

Schritt 2: Erstellen der Pipeline

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

- Eine Quellphase mit einer S3-Quellaktion, bei der das Quellartefakt Ihre Vorlagendatei und alle unterstützenden Quelldateien sind.
- Eine Bereitstellungsphase mit einer CloudFormation Stack-Set-Bereitstellungsaktion, die das Stack-Set erstellt.
- Eine Bereitstellungsphase mit einer CloudFormation Stack-Instances Deployment-Aktion, 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-Managementkonsole 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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyStackSetsPipeline`**.
5. Wählen Sie für die Zwecke dieses Tutorials unter Pipeline-Typ die Option V1 aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und ihrem Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
6. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline das Erstellen einer Servicerolle in IAM zu ermöglichen.
7. Belassen Sie im Artifact Store die Standardeinstellungen.

> **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 erhalten Sie unter Eingabe- und Ausgabe-Artefakte und CodePipeline Referenz zur Pipeline-Struktur.

Wählen Sie Weiter aus.

8. Wählen Sie auf der Seite „Schritt 3: Quellphase hinzufügen“ unter Quellanbieter die Option Amazon S3 aus.
9. Geben Sie unter Bucket den S3-Quell-Bucket ein, den Sie für dieses Tutorial erstellt haben, z. `BucketName` B. Geben Sie unter S3-Objektschlüssel den Dateipfad und den Dateinamen für Ihre ZIP-Datei ein, z. `MyFiles.zip` B.
10. Wählen Sie Weiter aus.
11. Wählen Sie in Schritt 4: Build-Phase hinzufügen die Option Build-Phase überspringen und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Skip wählen.

    Wählen Sie Weiter aus.
12. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

    Wählen Sie Weiter aus.
13. In Schritt 6: Bereitstellungsphase hinzufügen:
    a. Wählen Sie unter Deploy-Anbieter die Option AWS CloudFormation Stack Set aus.
    b. Geben Sie im Feld Stack-Set-Name 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 für das Quellartefakt. `SourceArtifact`

```
SourceArtifact::template.yml
```

d. Geben Sie im Feld Einsatzziele 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 für das Quellartefakt. `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 für das Quellartefakt. `SourceArtifact`

```
SourceArtifact::parameters.txt
```

Um die Parameter als literale Eingabe 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 in Schritt 7: Überprüfen die Option 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, dass Ihr Stack-Set durch die Bereitstellung erfolgreich erstellt wurde, können Sie die zweite Aktion zu Ihrer Bereitstellungsphase hinzufügen.

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 View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
3. Wählen Sie die CloudFormation Aktion für die CloudFormationStackSet Aktion in Ihrer Pipeline aus. Die Vorlage, die Ressourcen und Ereignisse für Ihr Stack-Set werden in der 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, die es ermöglicht, die verbleibenden Stack-Instances CloudFormation StackSets zu erstellen.

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 Bearbeiten aus.
4. Wählen Sie unter der AWS CloudFormation Stack Set-Bereitstellungsaktion 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 Action Provider die Option AWS CloudFormation Stack Instances aus.
   c. Wählen Sie unter Input Artifacts die Option aus SourceArtifact.
   d. Geben Sie im Feld Stack-Set-Name 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 und den Dateipfad des Artefakts ein, in den Sie Ihre Kontodatei hochgeladen haben. Geben Sie beispielsweise Folgendes ein und verwenden Sie dabei den Standardnamen für das Quellartefakt. `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 mit zwei Aktionen in der Bereitstellungsphase angezeigt.

Schritt 5: Sehen Sie sich die Stack-Set-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 aus und klicken Sie dann auf Ansicht. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
3. Wählen Sie die 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 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: Aktualisieren Sie Ihr Stack-Set

Führen Sie ein Update für Ihr Stack-Set durch und stellen Sie das Update auf Instances bereit. In diesem Beispiel nehmen Sie auch eine Änderung an den Bereitstellungszielen vor, die Sie für die Aktualisierung festlegen möchten. Die Instanzen, die nicht Teil des Updates sind, erhalten den Status „Veraltet“.

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 Phase „Bereitstellen“ die Option Bearbeiten aus.
3. Wählen Sie, ob Sie die AWS CloudFormation Stack Set-Aktion in Ihrer Pipeline bearbeiten möchten. Überschreiben Sie im Feld Beschreibung die vorhandene 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 Deployment-Ziel 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 CloudFormation. Wählen Sie den StackSet Info-Tab. Vergewissern Sie sich, dass unter StackSet Beschreibung die neue Beschreibung angezeigt wird.
7. Wählen Sie den Tab Stack-Instances. Stellen Sie unter Status sicher, dass der Status für die Stack-Instances in us-east-2 lautet. `OUTDATED`

Tutorial: Erstellen Sie eine Variablenprüfregel für eine Pipeline als Eingabebedingung

In diesem Tutorial konfigurieren Sie eine Pipeline, die kontinuierlich Dateien bereitstellt und dabei GitHub als Quellaktions-Provider in Ihrer Quellphase dient. Die fertige Pipeline erkennt Änderungen, wenn Sie in Ihrem Quell-Repository eine Änderung an den Quelldateien vornehmen. Die Pipeline wird ausgeführt und vergleicht dann die Ausgabevariablen mit dem Namen des Quell-Repositorys und des Branch-Namens, die in der Bedingung für den Eintritt in die Build-Phase angegeben sind.

Important

Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto als dem Konto für Ihre Pipeline befindet, stellen Sie sicher, dass der S3-Artefakt-Bucket dem Konto gehört AWS-Konten , das sicher und zuverlässig ist.

Important

Viele der Aktionen, die Sie in diesem Verfahren zu Ihrer Pipeline hinzufügen, betreffen 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. Du kannst regionsübergreifende Aktionen hinzufügen, wenn du deine Pipeline erstellst. 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.

In diesem Beispiel wird die Beispielpipeline mit einer Quellaktion GitHub (Version2) und einer CodeBuild Build-Aktion verwendet, bei der die Eingabebedingung für die Build-Phase nach Variablen sucht.

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Erstellen Sie ein GitHub Repository mit Ihrem GitHub Konto.
- Halten Sie Ihre GitHub Anmeldeinformationen bereit. Wenn Sie das verwenden AWS-Managementkonsole , um eine Verbindung herzustellen, werden Sie aufgefordert, sich mit Ihren GitHub Anmeldeinformationen anzumelden.
- Eine Verbindung zu deinem Repository, die du GitHub (per GitHub App) als Quellaktion für deine Pipeline einrichten kannst. Informationen zum Herstellen einer Verbindung zu deinem GitHub Repository findest du unterGitHub Verbindungen.

Schritt 1: Erstelle eine Beispiel-Quelldatei und füge sie deinem GitHub Repository hinzu

In diesem Abschnitt erstellst du deine Beispiel-Quelldatei und fügst sie dem Repository hinzu, das die Pipeline für deine Quellphase verwendet. Für dieses Beispiel erstellen Sie Folgendes und fügen es hinzu:

- Die Datei `README.md`.

Nachdem Sie Ihr GitHub Repository erstellt haben, gehen Sie wie folgt vor, um Ihre README-Datei hinzuzufügen.

1. Logge dich in dein GitHub Repository ein und wähle dein Repository aus.
2. Um eine neue Datei zu erstellen, wähle Datei hinzufügen und dann Neue Datei erstellen. Benennen Sie die Datei `README.md` und fügen Sie den folgenden Text hinzu.

   ```
   This is a GitHub repository!
   ```

3. Wählen Sie Commit changes (Änderungen übernehmen) aus. Fügen Sie für dieses Tutorial eine Commit-Nachricht hinzu, die das großgeschriebene Wort „Update“ enthält, wie im folgenden Beispiel dargestellt:

   ```
   Update to source files
   ```

> **Note**
>
> Bei der Regelüberprüfung für Zeichenketten wird zwischen Groß- und Kleinschreibung unterschieden.

Die Datei `README.md` muss sich im Stammverzeichnis Ihres Repositorys befinden.

Schritt 2: Erstellen der Pipeline

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

- Eine Quellstufe mit einer Verbindung zu deinem GitHub Repository und deiner Aktion.
- Eine CodeBuild Build-Phase, in der für die Phase die Bedingung On Entry für die Variablenprüfungsregel konfiguriert ist.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der CodePipeline Konsole unter an https://console.aws.amazon.com/codepipeline/.
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. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.
4. Geben Sie auf Schritt 2: Wählen Sie die Pipeline-Einstellungen aus und geben Sie im Feld Pipeline-Name den Wert ein**`MyVarCheckPipeline`**.
5. CodePipeline bietet Pipelines vom Typ V1 und V2, die sich in ihren Eigenschaften und ihrem Preis unterscheiden. Der Typ V2 ist der einzige Typ, den Sie in der Konsole auswählen können. Weitere Informationen finden Sie unter Pipeline-Typen. Informationen zur Preisgestaltung für finden Sie CodePipeline unter Preisgestaltung.
6. 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.

7. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert.

   Wählen Sie Weiter aus.

8. Fügen Sie auf der Seite Schritt 3: Quellstufe hinzufügen eine Quellstufe hinzu:

   a. Wählen Sie unter Quellanbieter die Option GitHub (über GitHub App) 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 im Feld Repository name (Repositoryname) den Namen Ihres GitHub -Repositorys aus.

   d. Wählen Sie unter Branchname den Repository-Branch aus, den Sie verwenden möchten.

   e. Vergewissern Sie sich, dass die Option Kein Trigger ausgewählt ist.

   Wählen Sie Weiter aus.

9. Fügen Sie in Schritt 4: 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 die Option aws/codebuild /standard:5.0.

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

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

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 Befehle erstellen 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. Dies kehrt zur CodePipeline Konsole zurück und erstellt 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.

10. Wählen Sie in Schritt 5: Testphase hinzufügen die Option Testphase überspringen aus und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Überspringen auswählen.

    Wählen Sie Weiter aus.

11. Wählen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen die Option Bereitstellungsphase überspringen aus, und akzeptieren Sie dann die Warnmeldung, indem Sie erneut auf Überspringen klicken. Wählen Sie Weiter aus.

12. Wählen Sie auf Schritt 7: Überprüfen die Option Pipeline erstellen aus.

Schritt 2: Bearbeiten Sie die Build-Phase, um die Bedingung und die Regel hinzuzufügen

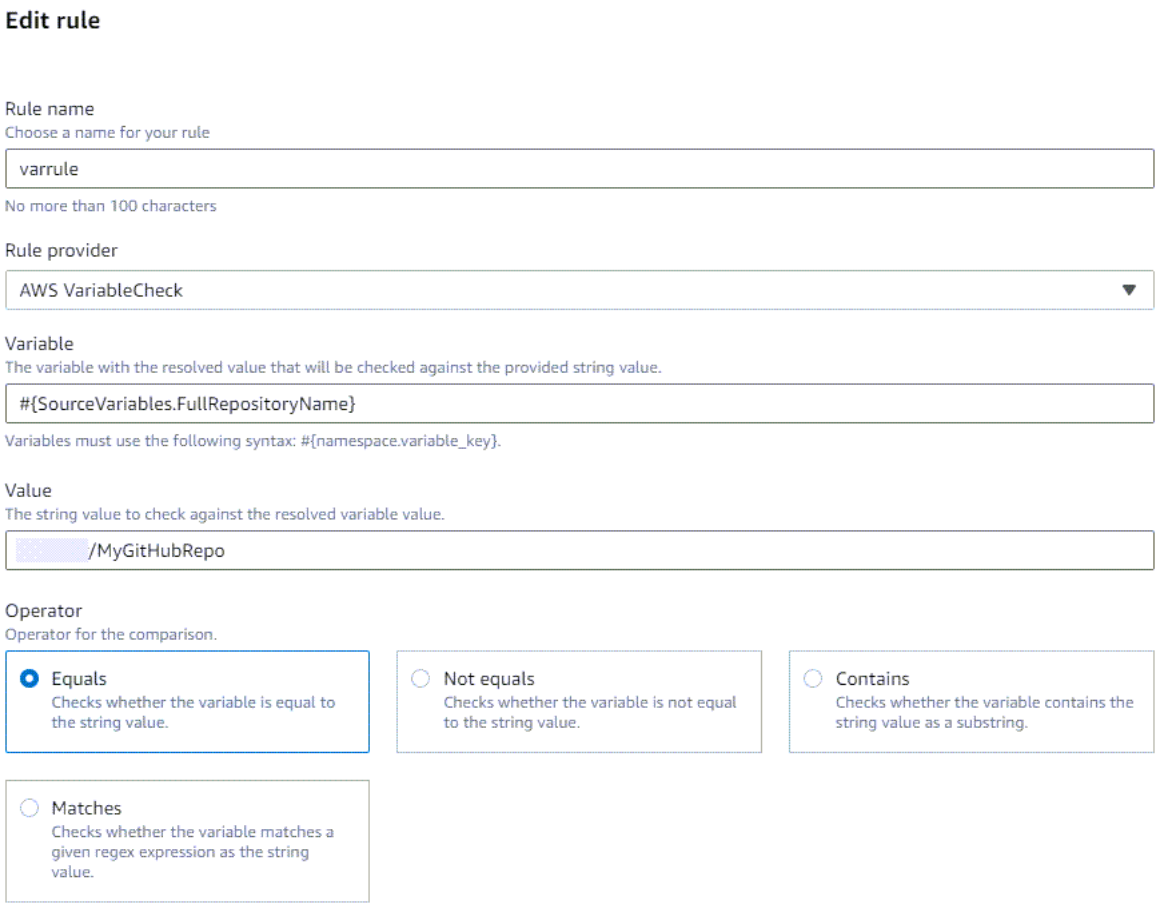
In diesem Schritt bearbeiten Sie die Phase, um eine On-Entry-Bedingung für die Variablenprüfungsregel hinzuzufügen.

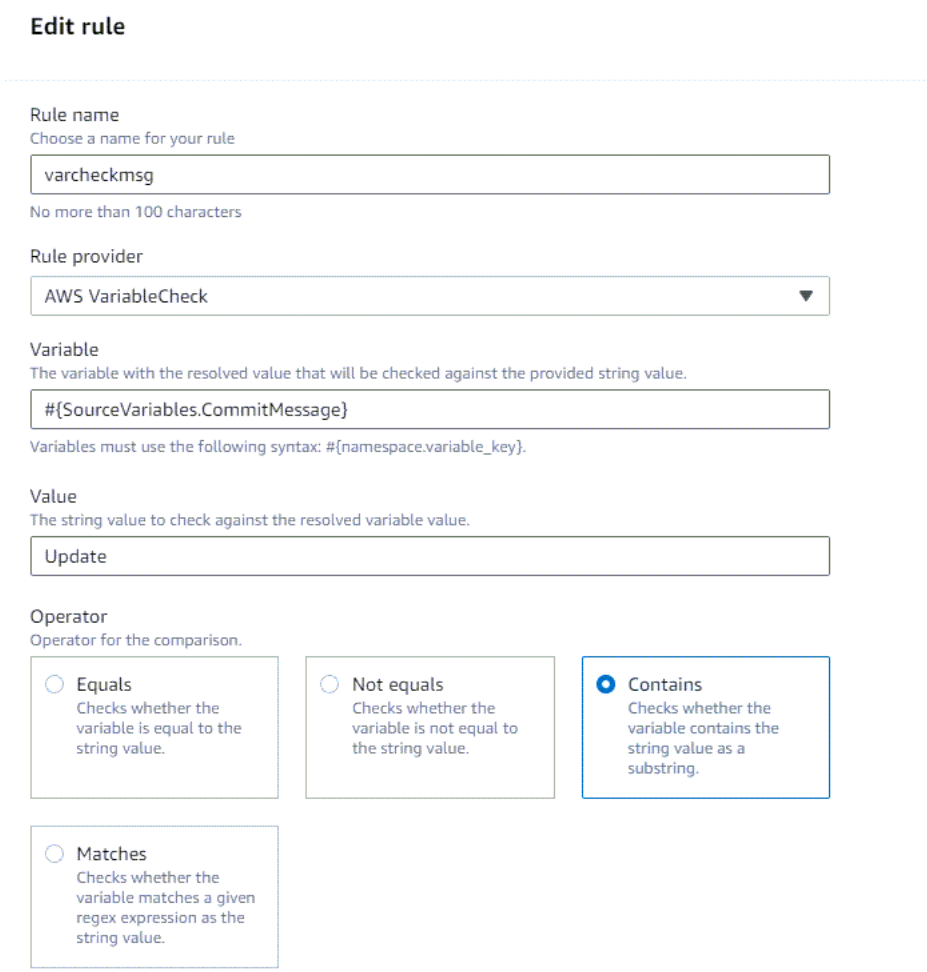
1. Wählen Sie Ihre Pipeline aus, und klicken Sie dann auf Bearbeiten. Wählen Sie, ob Sie in der Build-Phase eine Eingaberegel hinzufügen möchten.

   Wählen Sie unter Regelanbieter die Option VariableCheck.

2. Geben Sie im Feld Variable die Variable oder Variablen ein, die Sie überprüfen möchten. Geben Sie im Feld Wert den Zeichenkettenwert ein, der mit der aufgelösten Variablen

verglichen werden soll. In den folgenden Beispielbildschirmen wird eine Regel für eine Gleichheitsüberprüfung und eine weitere Regel für eine Prüfung auf „enthält“ erstellt.





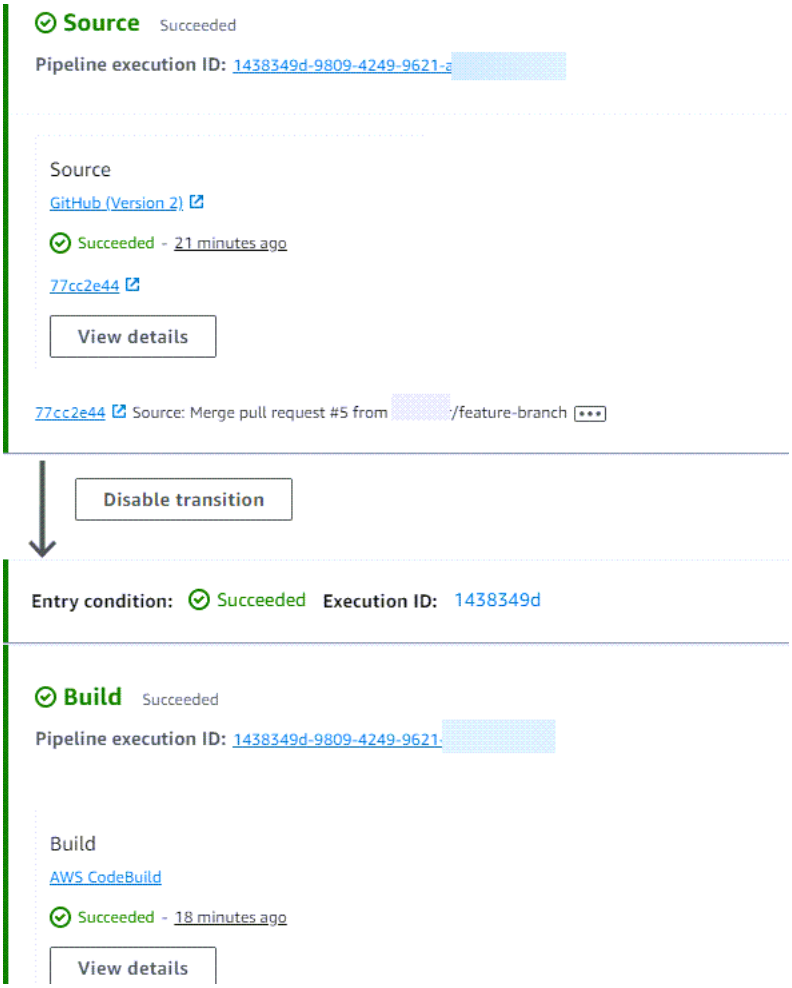
3. Wählen Sie Speichern.

   Wählen Sie Fertig aus.

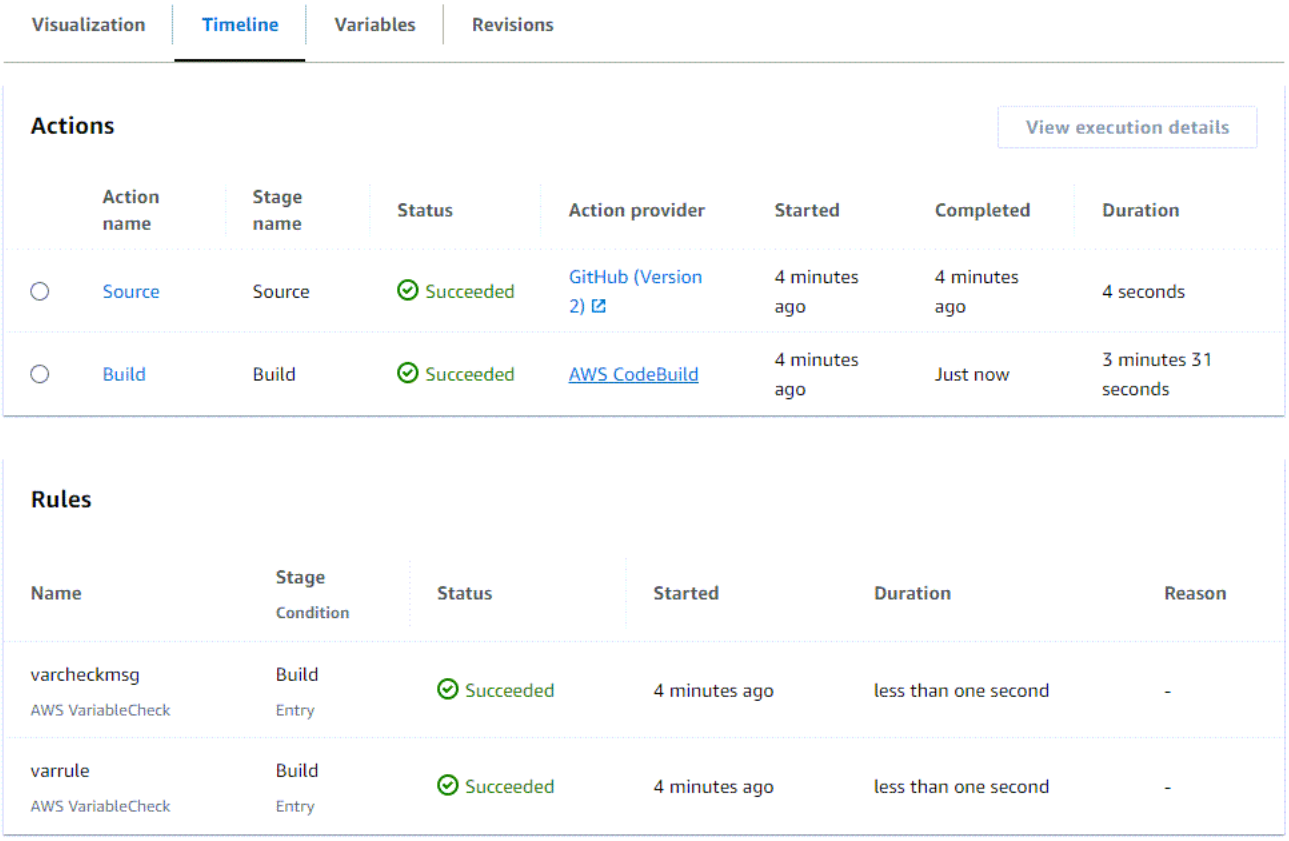
Schritt 3: Führen Sie die Pipeline aus und sehen Sie sich die aufgelösten Variablen an

In diesem Schritt sehen Sie sich die aufgelösten Werte und Ergebnisse der Variablenprüfungsregel an.

1. Zeigen Sie den aufgelösten Lauf an, nachdem die Regelüberprüfung erfolgreich war, wie im folgenden Beispiel gezeigt.



2. Sehen Sie sich die Variableninformationen auf der Registerkarte Zeitleiste an.


Visualization
Timeline
Variables
Revisions
Actions
View execution details
Action name
Stage name
Status
Action provider
Started
Completed
Duration
Source
Source
Succeeded
GitHub (Version 2)
4 minutes ago
4 minutes ago
4 seconds
Build
Build
Succeeded
AWS CodeBuild
4 minutes ago
Just now
3 minutes 31 seconds
Rules
Name
Stage
Condition
Status
Started
Duration
Reason
varcheckmsg
AWS VariableCheck
Build
Entry
Succeeded
4 minutes ago
less than one second
-
varrule
AWS VariableCheck
Build
Entry
Succeeded
4 minutes ago
less than one second
-

CodePipeline Anwendungsfälle

In den folgenden Abschnitten werden Anwendungsfälle für beschrieben CodePipeline.

Themen

- Anwendungsfälle für CodePipeline

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 CodePipeline die Automatisierung Ihrer Codeveröffentlichungen mithilfe verschiedener Produktintegrationen. 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 S3, AWS CodeCommit, und AWS CodeDeploy
- Verwendung CodePipeline mit Drittanbieter-Aktionsanbietern (GitHub und Jenkins)
- Wird verwendet, CodePipeline um Code zu kompilieren, zu erstellen und zu testen mit CodeBuild
- Verwendung CodePipeline mit Amazon ECS für die kontinuierliche Bereitstellung containerbasierter Anwendungen in der Cloud
- Verwendung CodePipeline mit Elastic Beanstalk für die kontinuierliche Bereitstellung von Webanwendungen in der Cloud
- Verwenden Sie mit CodePipeline AWS Lambda für die kontinuierliche Bereitstellung von Lambda-based und serverlose Anwendungen
- Verwenden Sie mit CodePipeline CloudFormation Vorlagen für die kontinuierliche Bereitstellung in der Cloud

Verwendung CodePipeline mit Amazon S3 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: Eine einfache Pipeline (CodeCommit Repository) erstellen 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 (GitHub und 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.

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 im 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 mit CodePipeline AWS Lambda für die kontinuierliche Bereitstellung von Lambda-based und serverlose 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 mit CodePipeline CloudFormation Vorlagen für die kontinuierliche Bereitstellung in der Cloud

Sie können CloudFormation mit CodePipeline für kontinuierliche Bereitstellung und Automatisierung verwenden. Weitere Informationen finden Sie unter Continuous Delivery with CodePipeline. CloudFormation wird auch verwendet, um die Vorlagen für Pipelines zu erstellen, die in AWS CodeStar erstellt wurden.

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
- Subnets
- 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)
- Europa (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 Sie einen VPC-Endpunkt für CodePipeline

Sie können die Amazon VPC-Konsole verwenden, um die Datei com.amazonaws zu erstellen. ***region***.codepipeline VPC-Endpunkt. In der Konsole *region* befindet sich 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 Ihr Netzwerk ACLs den Verkehr zulässt.
4. Stellen Sie sicher, dass Ihre Sicherheitsgruppen den Datenverkehr zulassen.
5. Stellen Sie sicher, dass die Routentabelle für private Subnetze auf das virtuelle 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

Definieren Sie CI/CD Pipelines mit Phasen und Aktionen

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 Sie eine benutzerdefinierte Aktion und fügen Sie sie 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 Tutorials, falls Sie die zum Erstellen einer Pipeline verwenden möchten AWS CLI ,. Eine Pipeline, Phasen und Aktionen erstellen Eine Liste der Pipelines finden Sie unter Pipelines und Details anzeigen in CodePipeline.

Themen

- Eine Pipeline, Phasen und Aktionen erstellen
- Eine Pipeline bearbeiten in CodePipeline
- Pipelines und Details anzeigen in CodePipeline
- Löschen Sie eine Pipeline in CodePipeline
- Erstellen Sie eine Pipeline CodePipeline , in der Ressourcen von einer anderen verwendet werden AWS Konto

- Migrieren Sie Polling-Pipelines, um ereignisbasierte Änderungserkennung zu verwenden
- Erstellen Sie die CodePipeline Servicerolle
- Taggen von Ressourcen
- Markieren Sie eine Pipeline in CodePipeline
- Erstellen einer Benachrichtigungsregel

Eine Pipeline, Phasen und Aktionen erstellen

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.

> **Important**
>
> Im Rahmen der Erstellung einer Pipeline wird ein vom Kunden bereitgestellter S3-Artefakt-Bucket von CodePipeline for Artifacts verwendet. (Dies unterscheidet sich von dem Bucket, der für eine S3-Quellaktion verwendet wird.) Wenn sich der S3-Artefakt-Bucket in einem anderen Konto befindet als das Konto für Ihre Pipeline, stellen Sie sicher, dass der S3-Artefakt-Bucket denjenigen gehört AWS-Konten , die sicher und zuverlässig sind.

Sie können Ihrer Pipeline Aktionen hinzufügen, die sich von Ihrer Pipeline AWS-Region unterscheiden. 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 Sie eine benutzerdefinierte Pipeline (Konsole)
- Erstellen einer Pipeline (CLI)
- Erstellen Sie eine Pipeline aus statischen Vorlagen

Erstellen Sie eine benutzerdefinierte Pipeline (Konsole)

Um eine benutzerdefinierte Pipeline in der Konsole zu erstellen, müssen Sie den Speicherort der Quelldatei und Informationen zu den Anbietern 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 (Quelle, 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-Managementkonsole 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 Erste Schritte.

3. Wählen Sie auf der Seite Schritt 1: Erstellungsoption auswählen unter Erstellungsoptionen die Option Benutzerdefinierte Pipeline erstellen aus. Wählen Sie Weiter aus.

4. Geben Sie auf der Seite Schritt 2: Pipeline-Einstellungen auswählen im Feld 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.

5. 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.

6. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:

   - Wählen Sie Neue Servicerolle aus, um eine neue Servicerolle in IAM erstellen CodePipeline zu können.
   - 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 Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu.

Weitere Informationen über die Service-Rolle und ihre Richtlinienanweisung finden Sie unter Verwalten Sie die CodePipeline Servicerolle.

7. (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 unterVariablenreferenz. 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.

8. (Optional) Erweitern Sie den Abschnitt Advanced Settings (Erweiterte Einstellungen).
9. 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 erhalten Sie unter Eingabe- und Ausgabe-Artefakte und CodePipeline Referenz zur Pipeline-Struktur.

10. 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.

11. Wählen Sie Weiter aus.

Schritt 2: Erstellen Sie eine Quellstufe

1. Wählen Sie auf der Seite Schritt 3: Quellcode hinzufügen im Feld Quellanbieter den Typ des Repositorys aus, in dem Ihr Quellcode gespeichert ist, und geben Sie die erforderlichen Optionen an. Die zusätzlichen Felder werden je nach ausgewähltem Quellanbieter wie folgt angezeigt.

    - Für Bitbucket Cloud, GitHub (über GitHub App), GitHub Enterprise Server oder GitLab selbstverwaltet: GitLab.com

        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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen. 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 nachgeschalteten Aktionen verwendet werden.

     Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschrieben. Problembehebung 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.

- Für 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 nachgeschalteten Aktionen verwendet werden.

      Wenn Sie diese Option wählen, müssen Sie Ihrer CodeBuild Servicerolle die `codecommit:GitPull` entsprechende Berechtigung hinzufügen, wie unter beschrieben. Fügen Sie CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen hinzu Sie müssen außerdem die `codecommit:GetRepository` Berechtigungen zu Ihrer CodePipeline Servicerolle hinzufügen, wie unter beschriebenFügen Sie der CodePipeline Servicerolle Berechtigungen hinzu. 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 Bereitstellung ECS-to-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 Detail.json Image-Datei für Amazon blue/green ECS-Bereitstellungsaktionen.

> **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.

2. Um die Phase für die automatische Wiederholung zu konfigurieren, wählen Sie Automatische Wiederholung bei Stufenausfall aktivieren. Weitere Informationen zur automatischen Wiederholung finden Sie unter. Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler
3. Wählen Sie Weiter aus.

Schritt 4: Erstellen Sie eine Buildphase

Dieser Schritt ist optional, wenn Sie die Erstellung einer Bereitstellungsstufe planen.

1. Führen Sie auf der Seite Schritt 4: Buildphase hinzufügen einen der folgenden Schritte aus und wählen Sie dann Weiter aus:

- Wählen Sie „Erstellungsphase überspringen“, wenn Sie eine Test- oder Bereitstellungsphase erstellen möchten.
- Um die Aktion Befehle für Ihre Build-Phase auszuwählen, wählen Sie Befehle.

  > **Note**
  >
  > Für die Ausführung der Aktion „Befehle“ fallen separate Gebühren in an AWS CodeBuild. Wenn Sie Build-Befehle als Teil einer CodeBuild Aktion einfügen möchten, fahren Sie fort und wählen Sie Andere Buildanbieter und dann aus CodeBuild.

  Geben Sie im Feld Befehle die Shell-Befehle für Ihre Aktion ein. Weitere Informationen zur Aktion Befehle finden Sie unterAktionsreferenz für Befehle.
- Um andere Build-Anbieter auszuwählen CodeBuild, wie z. B., wählen Sie Andere Anbieter. 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. Befolgen Sie den Anweisungen unter Erstellen einer Pipeline, die CodeBuild verwendet im CodeBuild-Benutzerhandbuch.

  Unter Buildspezifikationen ist die CodeBuild Buildspec-Datei optional, und Sie können stattdessen Befehle eingeben. Geben Sie im Feld Build-Befehle einfügen die Shell-Befehle für Ihre Aktion ein. Weitere Informationen zu Überlegungen zur Verwendung von Build-Befehlen finden Sie unterAktionsreferenz für Befehle. Wählen Sie Buildspec-Datei verwenden, wenn Sie Befehle in anderen Phasen ausführen möchten oder wenn Sie eine lange Befehlsliste haben.

Um Ihrer Build-Aktion Umgebungsvariablen hinzuzufügen, wählen Sie unter CodeBuild 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.

2. Um die Phase für die automatische Wiederholung zu konfigurieren, wählen Sie Automatische Wiederholung bei Stufenausfall aktivieren. Weitere Informationen zur automatischen Wiederholung finden Sie unter. Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler
3. Wählen Sie Weiter aus.

Schritt 5: Erstellen Sie eine Testphase

Dieser Schritt ist optional, wenn Sie eine Build- oder Bereitstellungsphase erstellen möchten.

1. Führen Sie auf der Seite Schritt 5: Testphase hinzufügen einen der folgenden Schritte aus und wählen Sie dann Weiter aus:

   - Wählen Sie Testphase überspringen, wenn Sie eine Build- oder Bereitstellungsphase erstellen möchten.
   - Wählen Sie unter Testanbieter den Testaktionsanbieter aus und füllen Sie die entsprechenden Felder aus.
2. Wählen Sie Weiter aus.

Schritt 6: Erstellen Sie eine Bereitstellungsphase

Dieser Schritt ist optional, falls Sie bereits eine Build-Stufe erstellt haben.

1. Führen Sie auf der Seite Schritt 6: Bereitstellungsphase hinzufügen einen der folgenden Schritte aus, und wählen Sie dann Weiter aus:

   - Wählen Sie „Bereitstellungsphase überspringen“, wenn Sie in den vorherigen Schritten eine Build- oder Testphase erstellt haben.

     > **Note**
     >
     > Diese Option wird nicht angezeigt, wenn Sie die Build- oder Testphase 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 eine Anwendung, eine Bereitstellungsgruppe oder beides auch in der CodeDeploy-Konsole 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 ausführliches Beispiel für die Verwendung einer einfachen Pipeline CodePipeline als Quelle für Code, 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.

- 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, 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 Standarddateiname für die Datei lautet imagedefinitions.json. Wenn Sie einen anderen Dateinamen verwenden, müssen Sie diesen bereitstellen, wenn Sie die Pipeline-Bereitstellungsphase erstellen. Weitere Informationen finden Sie unter imagedefinitions.json-Datei für Amazon ECS-Standardbereitstellungsaktionen.

Wählen Sie Weiter aus.

> **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 zum Bereitstellen von Container-basierten Anwendungen mit Ihrer Pipeline finden Sie unter Tutorial: Kontinuierliche Bereitstellung mit CodePipeline.

- Amazon ECS (Blue/Green)

  Geben Sie die CodeDeploy Anwendungs- und Bereitstellungsgruppe, die Amazon ECS-Aufgabendefinition und die AppSpec Dateiinformationen ein, und klicken Sie dann auf Weiter.

  > **Note**
  >
  > Die Amazon ECS (Blue/Green) -Aktion erfordert eine Detail.json Bilddatei 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 Detail.json Image-Datei für Amazon blue/green ECS-Bereitstellungsaktionen.

  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 Bereitstellung ECS-to-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 im 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 es nicht ausgewählt ist, müssen Sie einen Wert in den 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.

2. Um die Phase für die automatische Wiederholung zu konfigurieren, wählen Sie Automatische Wiederholung bei Stufenausfall aktivieren. Weitere Informationen zur automatischen Wiederholung finden Sie unter. Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler
3. Um die Phase für automatisches Rollback zu konfigurieren, wählen Sie Automatisches Rollback bei Stufenausfall konfigurieren. Weitere Informationen zum automatischen Rollback finden Sie unter. Konfigurieren Sie eine Phase für das automatische Rollback
4. Klicken Sie auf Nächster Schritt.

Schritt 7: Überprüfen Sie die Pipeline

- Überprüfen Sie auf der Seite Schritt 7: Überprüfen Ihre Pipeline-Konfiguration und wählen Sie dann Pipeline erstellen, um die Pipeline zu erstellen, oder Zurück, um zurückzugehen und Ihre Auswahl zu bearbeiten. 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 unter Variablenreferenz.
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": "amzn-s3-demo-source-bucket",
                            "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. Taggen 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. 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, 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 beschriebenErstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (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 beschriebenVerbindung zu Amazon S3-Quellaktionen herstellen, die EventBridge und verwenden AWS CloudTrail.

Erstellen Sie eine Pipeline aus statischen Vorlagen

Sie können in der Konsole eine Pipeline erstellen, die eine Vorlage verwendet, um eine Pipeline mit von Ihnen festgelegtem Quellcode und Eigenschaften zu konfigurieren. Sie müssen den Speicherort der Quelldatei und Informationen zu den Quellanbietern angeben, die Sie für Ihre Aktionen verwenden werden. Sie können eine Quellaktion für Amazon ECR oder ein von unterstütztes Drittanbieter-Repository angeben AWS CodeConnections, z. B. GitHub

Die Vorlage erstellt einen Stack in CloudFormation für Ihre Pipeline, der die folgenden Ressourcen enthält:

- Eine Pipeline wird mit dem Pipeline-Typ V2 erstellt. 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.
- Eine Servicerolle wird für Ihre Pipeline erstellt und in der Vorlage referenziert.
- Ein Artefaktspeicher wird mithilfe des Standardartefaktspeichers, z. B. des als Standard festgelegten S3-Artefakt-Buckets, für Ihre Pipeline in der Pipeline erstellt, die AWS-Region Sie für Ihre Pipeline ausgewählt haben.

Eine Sammlung von Open-Source-Startvorlagen, die für den Assistenten zur Erstellung statischer Vorlagen verwendet werden, finden Sie im Repository unter. https://github.com/aws/codepipeline-starter-templates

Wenn Sie statische Vorlagen verwenden, um eine Pipeline zu erstellen, wird die Pipeline-Struktur in jeder Vorlage entsprechend den Anforderungen des Anwendungsfalls konfiguriert. In diesem Verfahren CloudFormation wird beispielsweise die Vorlage für eine Bereitstellung als Beispiel verwendet. Die Vorlage generiert eine Pipeline DeployToCloudFormationServicemit dem Namen der folgenden Struktur:

- Eine Erstellungsphase, die eine Quellaktion mit einer Konfiguration enthält, die Sie im Assistenten angeben.
- Eine Bereitstellungsphase mit einer Bereitstellungsaktion und einem zugehörigen Ressourcenstapel CloudFormation.

Wenn Sie eine statische Vorlage verwenden, um eine Pipeline zu erstellen, CodePipeline erstellt 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: Greifen Sie auf die Konsole zu

1. Melden Sie sich bei der an AWS-Managementkonsole 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 Erste Schritte.

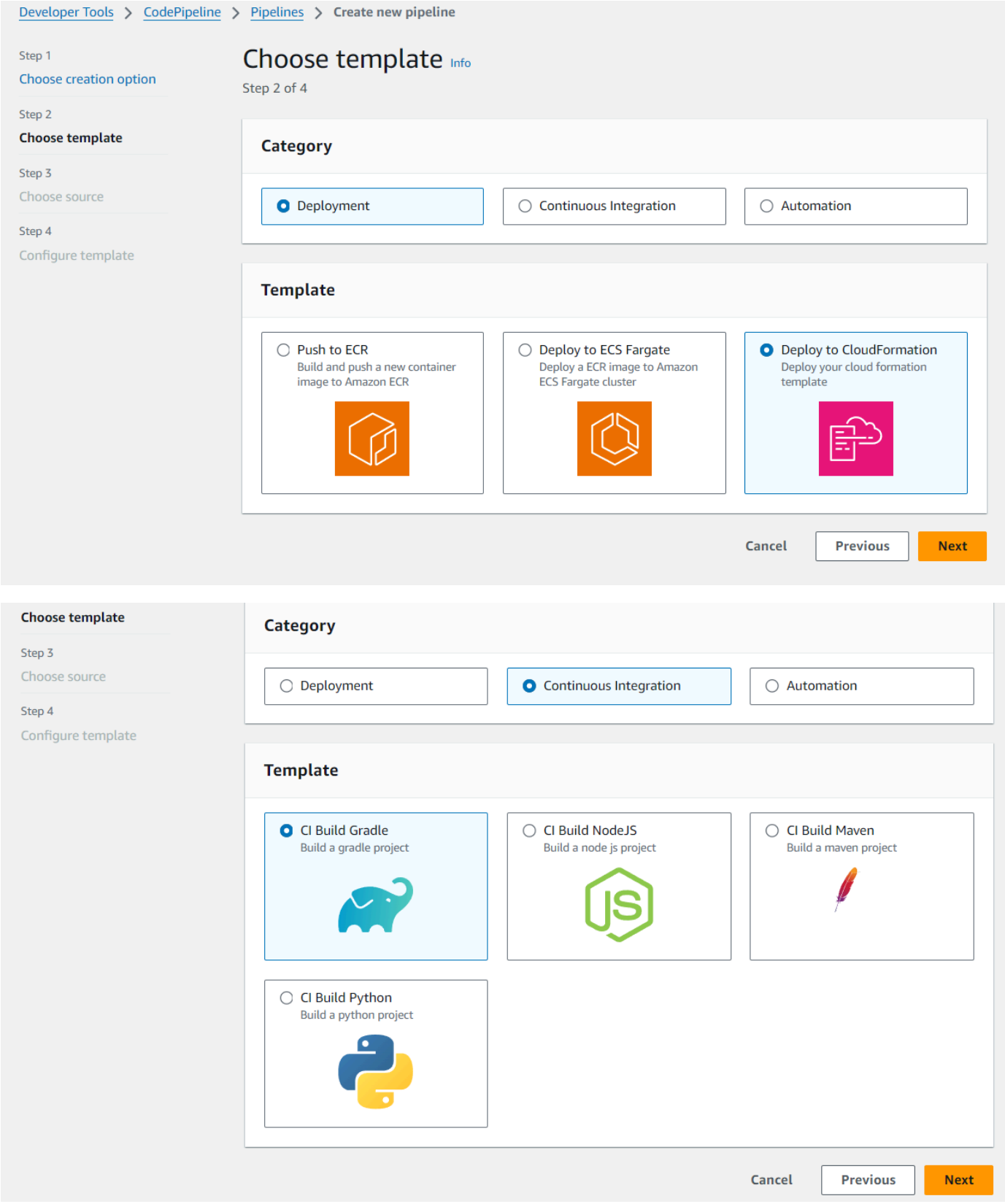
Schritt 2: Vorlage wählen

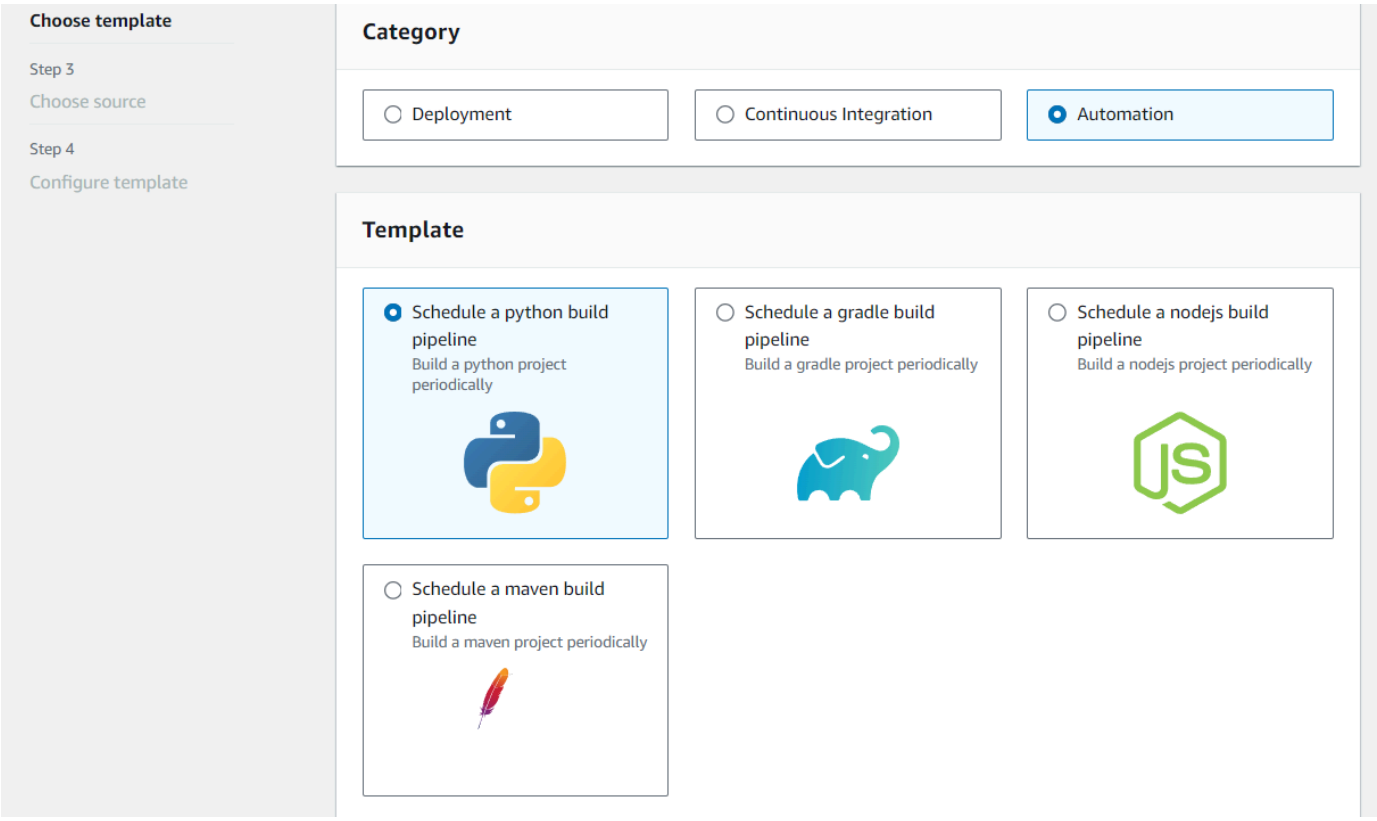
Wählen Sie eine Vorlage, um eine Pipeline mit einer Bereitstellungsphase, Automatisierung oder einer CI-Pipeline zu erstellen.

1. Führen Sie auf der Seite Schritt 1: Erstellungsoption auswählen eine der folgenden Aktionen aus und wählen Sie dann Weiter aus:

   - Wählen Sie Bereitstellung, wenn Sie eine Bereitstellungsphase erstellen möchten. Sehen Sie sich die Optionen für Vorlagen an, die in ECR oder CloudFormation bereitgestellt werden. Wählen Sie für dieses Beispiel Deployment und anschließend Deployment to CloudFormation aus.
   - Wählen Sie Kontinuierliche Integration, wenn Sie eine CI-Pipeline erstellen möchten. Sehen Sie sich die Optionen für CI-Pipelines an, z. B. das Erstellen für Gradle.
   - Wählen Sie Automatisierung, wenn Sie eine automatisierte Pipeline erstellen möchten. Sehen Sie sich die Optionen für die Automatisierung an, z. B. die Planung eines Python-Builds.

2.


Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose template
Step 3
Choose source
Step 4
Configure template
Choose template Info
Step 2 of 4
Category
Deployment
Continuous Integration
Automation
Template
Push to ECR
Build and push a new container image to Amazon ECR
Deploy to ECS Fargate
Deploy a ECR image to Amazon ECS Fargate cluster
Deploy to CloudFormation
Deploy your cloud formation template
Cancel
Previous
Next
Choose template
Step 3
Choose source
Step 4
Configure template
Category
Deployment
Continuous Integration
Automation
Template
CI Build Gradle
Build a gradle project
CI Build NodeJS
Build a node js project
CI Build Maven
Build a maven project
CI Build Python
Build a python project
Cancel
Previous
Next



Schritt 3: Wählen Sie die Quelle

- Wählen Sie auf der Seite Schritt 3: Quelle auswählen unter Quellanbieter den Anbieter für das Repository aus, in dem Ihr Quellcode gespeichert ist, geben Sie die erforderlichen Optionen an und klicken Sie dann auf Nächster Schritt.

  - Für Bitbucket Cloud GitHub (über GitHub App), GitHub Enterprise Server oder GitLab selbstverwaltet: GitLab.com

    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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen. 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 nachgeschalteten Aktionen verwendet werden.

     Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschrieben. Problembehebung 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 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.

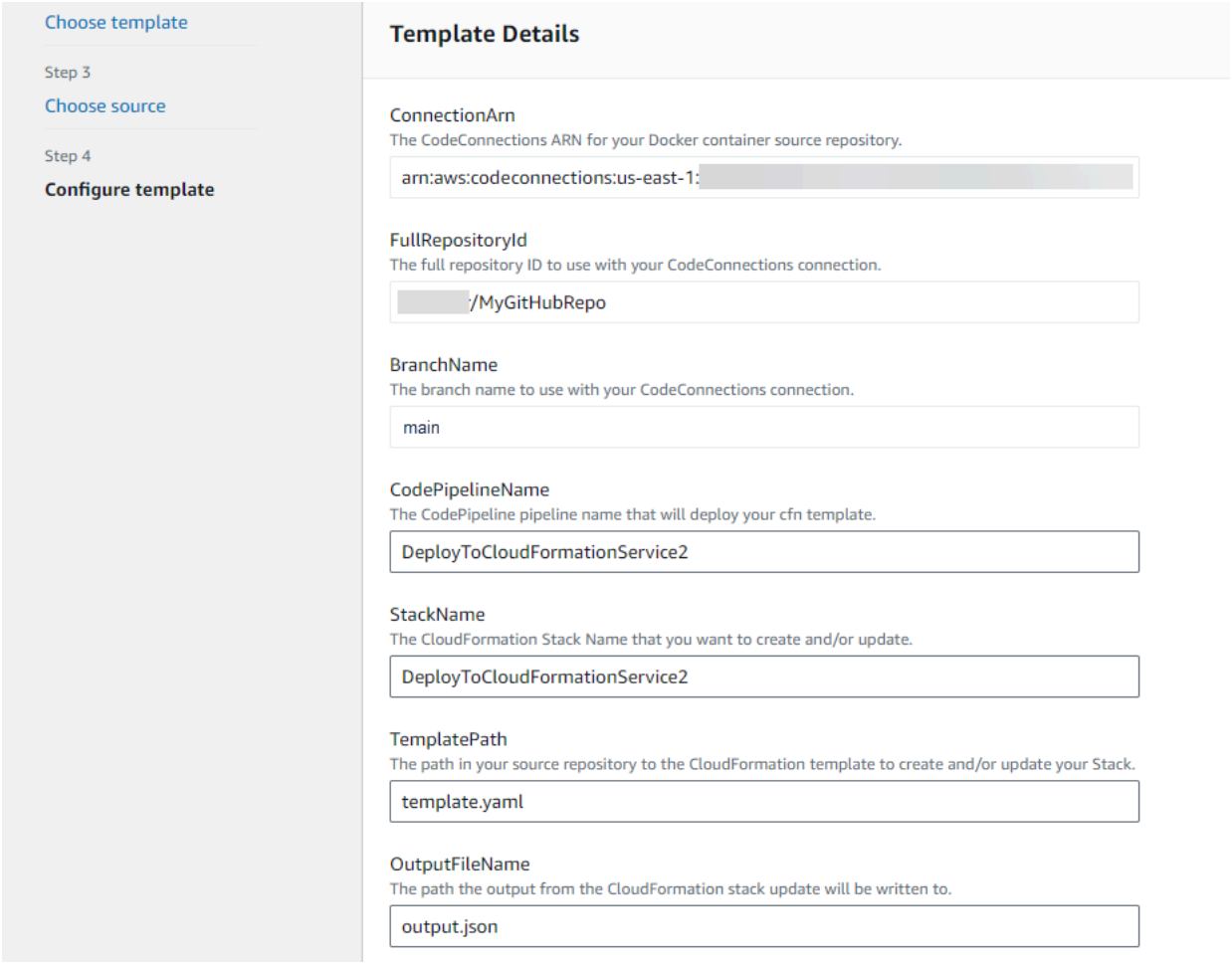
  Wenn Sie eine Amazon ECR-Quellstufe in Ihre Pipeline aufnehmen, generiert die Quellaktion eine `imageDetail.json` Datei als Ausgabeartefakt, wenn Sie eine Änderung festschreiben. Informationen zur `imageDetail.json`-Datei finden Sie unter Detail.json Image-Datei für Amazon blue/green ECS-Bereitstellungsaktionen.

> **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 4: Vorlage konfigurieren

Für dieses Beispiel CloudFormation wurde die Bereitstellung für ausgewählt. Fügen Sie in diesem Schritt die Konfiguration für Ihre Vorlage hinzu.



1. Geben Sie in Schritt 4: Vorlage konfigurieren im Feld Stackname einen Namen für Ihre Pipeline ein.
2. Bearbeiten Sie die Platzhalter-IAM-Richtlinie für die Berechtigungen, die für Ihre Vorlage gelten.
3. Wählen Sie Pipeline aus Vorlage erstellen
4. Es wird eine Meldung angezeigt, die besagt, dass Ihre Pipeline-Ressourcen erstellt werden.

Schritt 5: Sehen Sie sich die Pipeline an

- Nachdem Sie Ihre Pipeline erstellt haben, können Sie sie in der CodePipeline Konsole anzeigen und den Stack darin anzeigen CloudFormation. 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.

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 unterErstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (CLI). Weitere Informationen zur Verwendung der CLI zum Erstellen oder Aktualisieren einer Amazon S3-Pipeline finden Sie unterErstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (CLI).

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 bei der an AWS-Managementkonsole und öffnen Sie die CodePipeline Konsole. 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. 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.

   Weitere Informationen zu Variablen auf Pipeline-Ebene finden Sie unterVariablenreferenz. 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 Auslöser 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 (über GitHub App), GitHub Enterprise Server, GitLab .com oder GitLab zur Selbstverwaltung, wie 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.

Beispiele zum Hinzufügen von Aktionsanbietern mithilfe der Standard-Felder für die einzelnen Anbieter finden Sie unter Erstellen Sie eine benutzerdefinierte Pipeline (Konsole).

Informationen zum Hinzufügen CodeBuild als Build- oder Testaktion zu einer Phase finden Sie unter Verwenden CodePipeline CodeBuild zum Testen von Code und Ausführen von 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 Phase wird nach der ersten genannten Bereitstellungsphase ausgeführt. *`Staging`*

   > **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 beschriebenErstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (CLI).

- Für eine Amazon S3 S3-Quelle müssen Sie die Regel und den AWS CloudTrail Trail für CloudWatch Ereignisse erstellen, wie unter beschriebenVerbindung zu Amazon S3-Quellaktionen herstellen, die EventBridge und verwenden 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)
- Aktionsdetails in einer Pipeline (Konsole) anzeigen
- Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen
- Anzeigen von der Pipeline-Details und des Verlaufs (CLI)
- Regelergebnisse für Stufenbedingungen im Ausführungsverlauf anzeigen

Pipelines anzeigen (Konsole)

Sie können den Status, die Übergänge und die Artefaktaktualisierungen für eine Pipeline anzeigen.

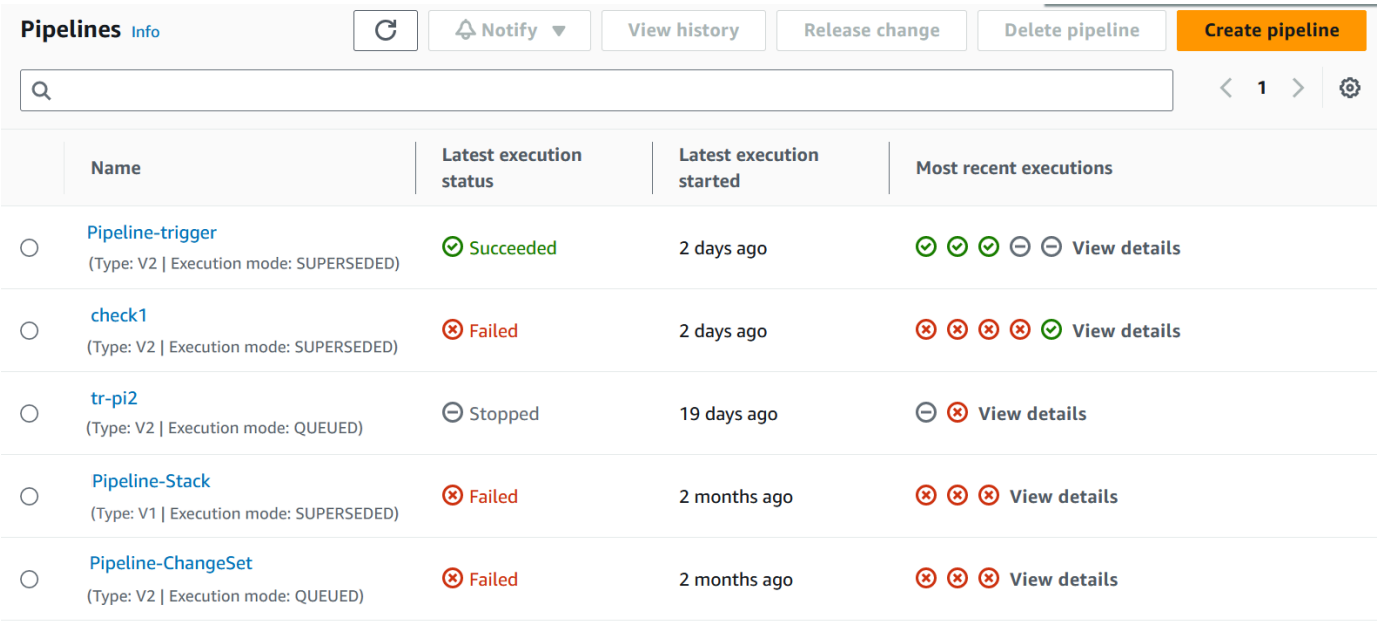
> **Note**
>
> Nach einer Stunde wird die detaillierte Anzeige der Pipeline in Ihrem Browser nicht mehr automatisch aktualisiert. Um aktuelle Informationen anzuzeigen, müssen Sie die Seite aktualisieren.

Um Pipelines anzuzeigen

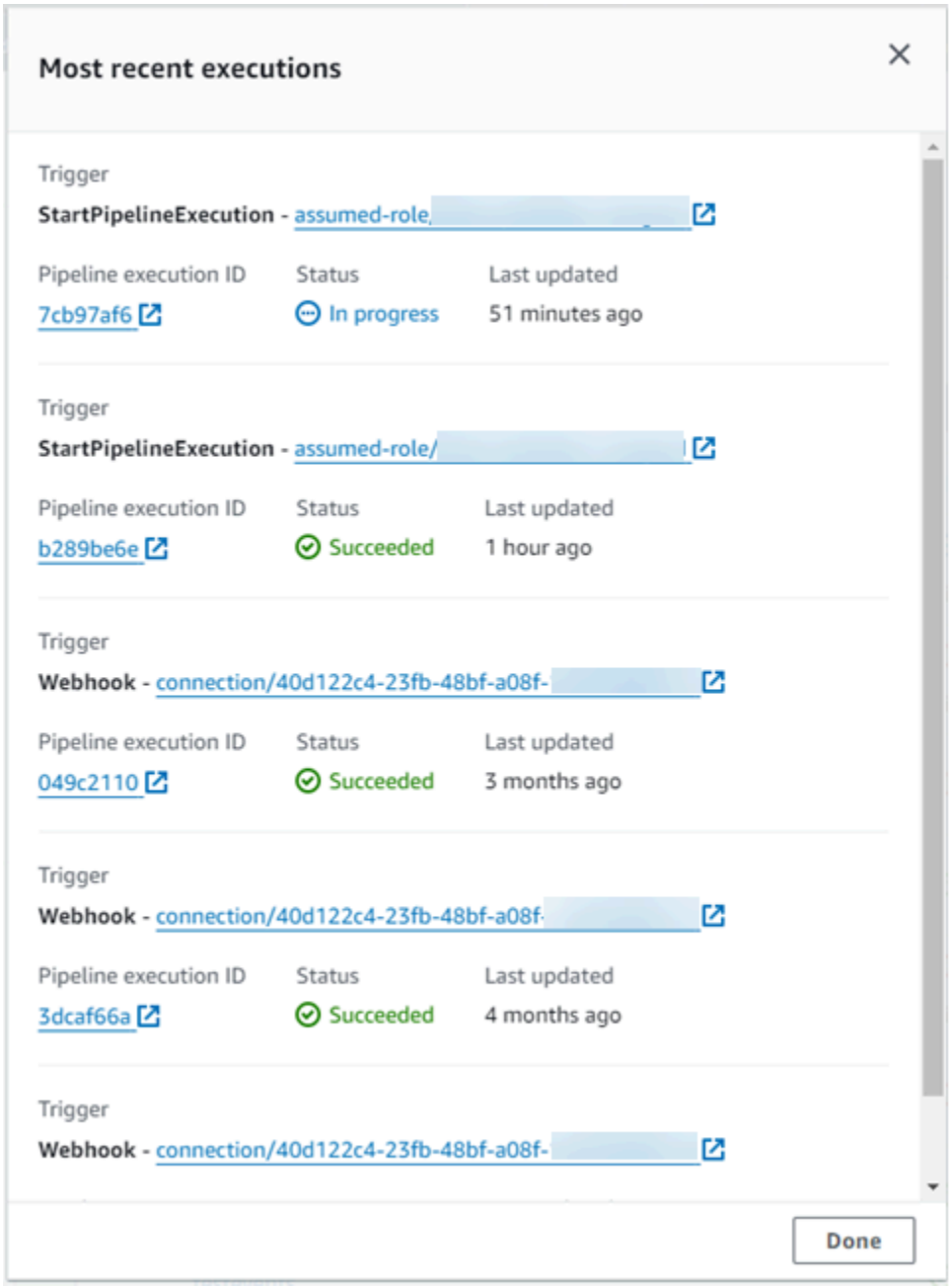
1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon. com/codesuite/codepipeline/home.

   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.

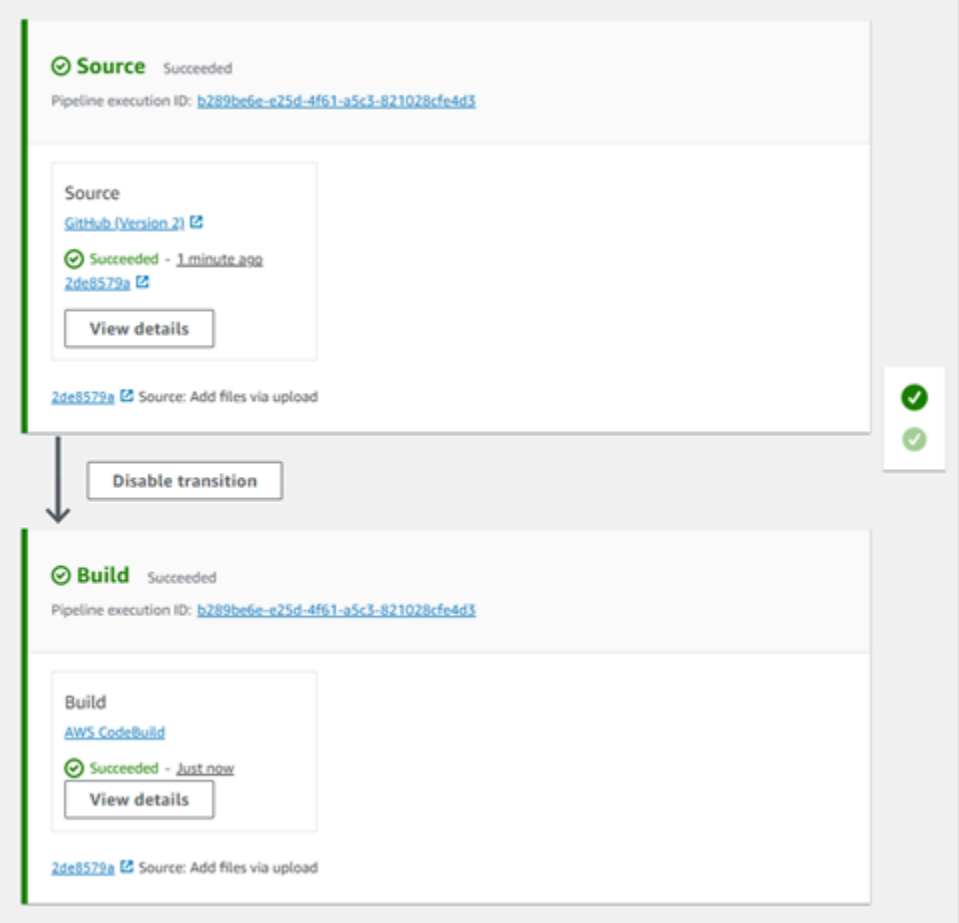


Um Details zu den neuesten Ausführungen für die Pipeline anzuzeigen, wählen Sie View History (Verlauf anzeigen). Für frühere Ausführungen können Sie Revisionsdetails anzeigen, die mit Quellartefakten verknüpft sind, wie Ausführung IDs, Status, Start- und Endzeit, Dauer sowie IDs Commit und Meldungen.

> **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.

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 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.

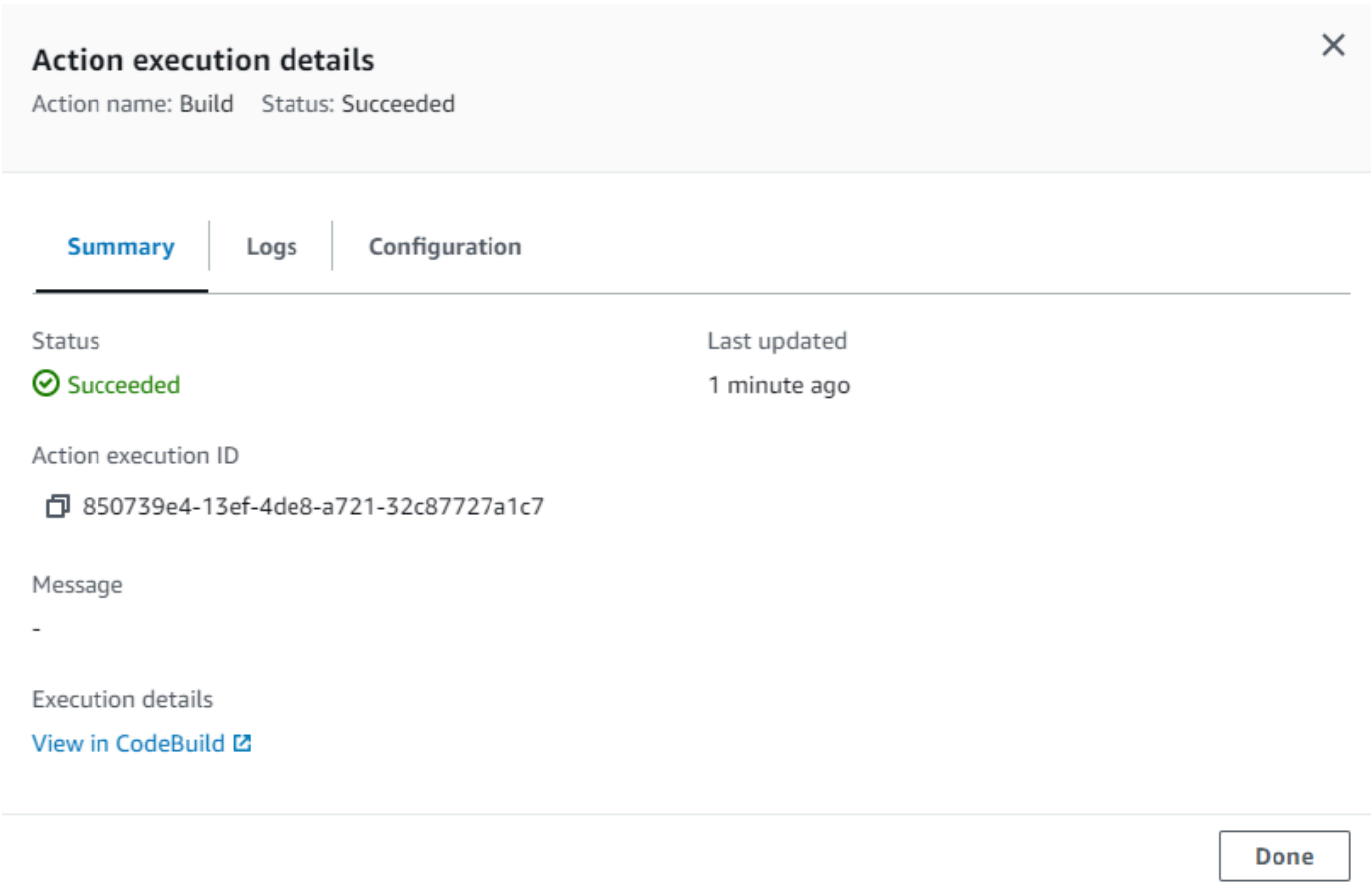
> **Note**
>
> Nach einer Stunde wird die detaillierte Anzeige der Pipeline in Ihrem Browser nicht mehr automatisch aktualisiert. Um aktuelle Informationen anzuzeigen, müssen Sie die Seite aktualisieren.

Um Aktionsdetails in einer Pipeline anzuzeigen

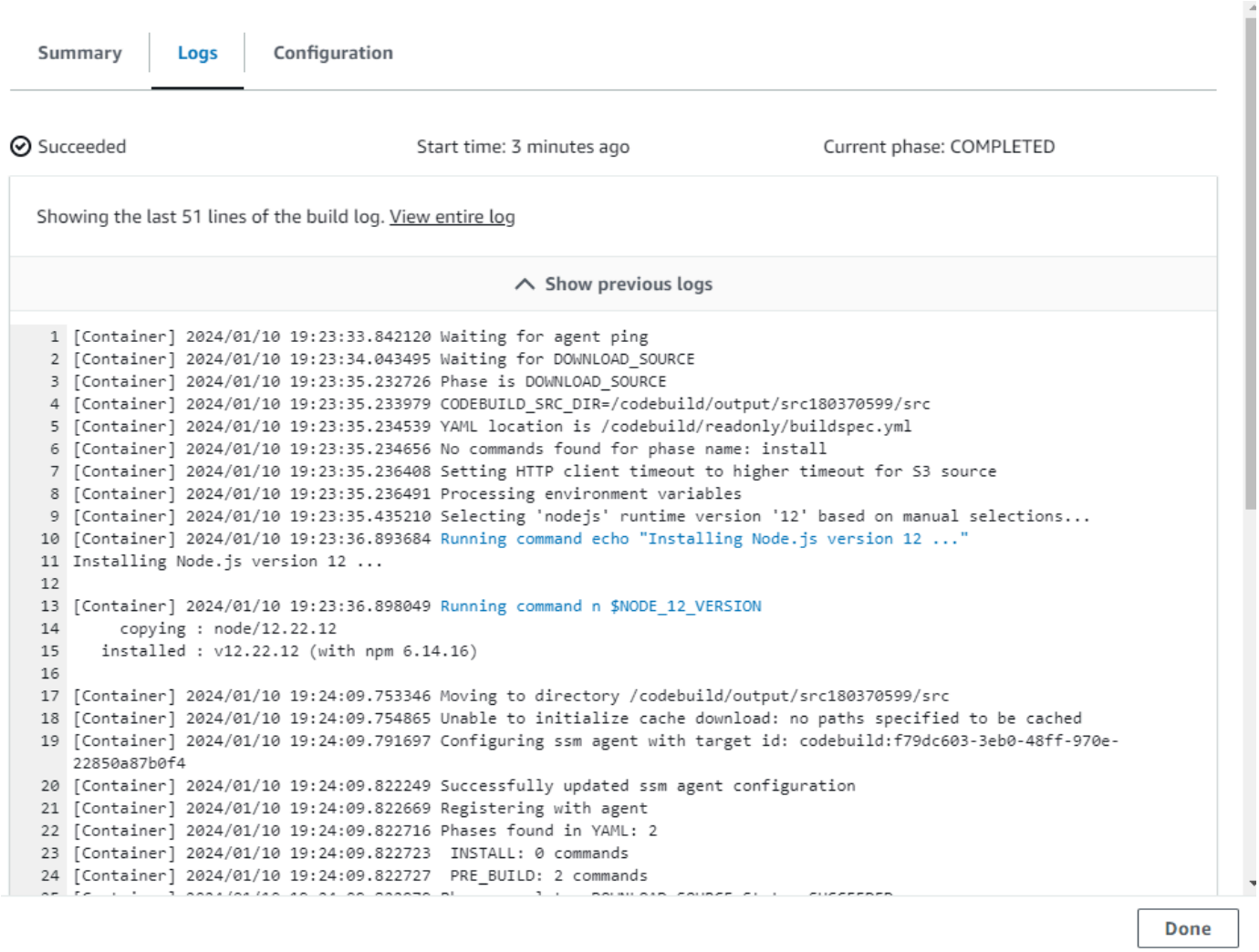
1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon. com/codesuite/codepipeline/home.

   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.

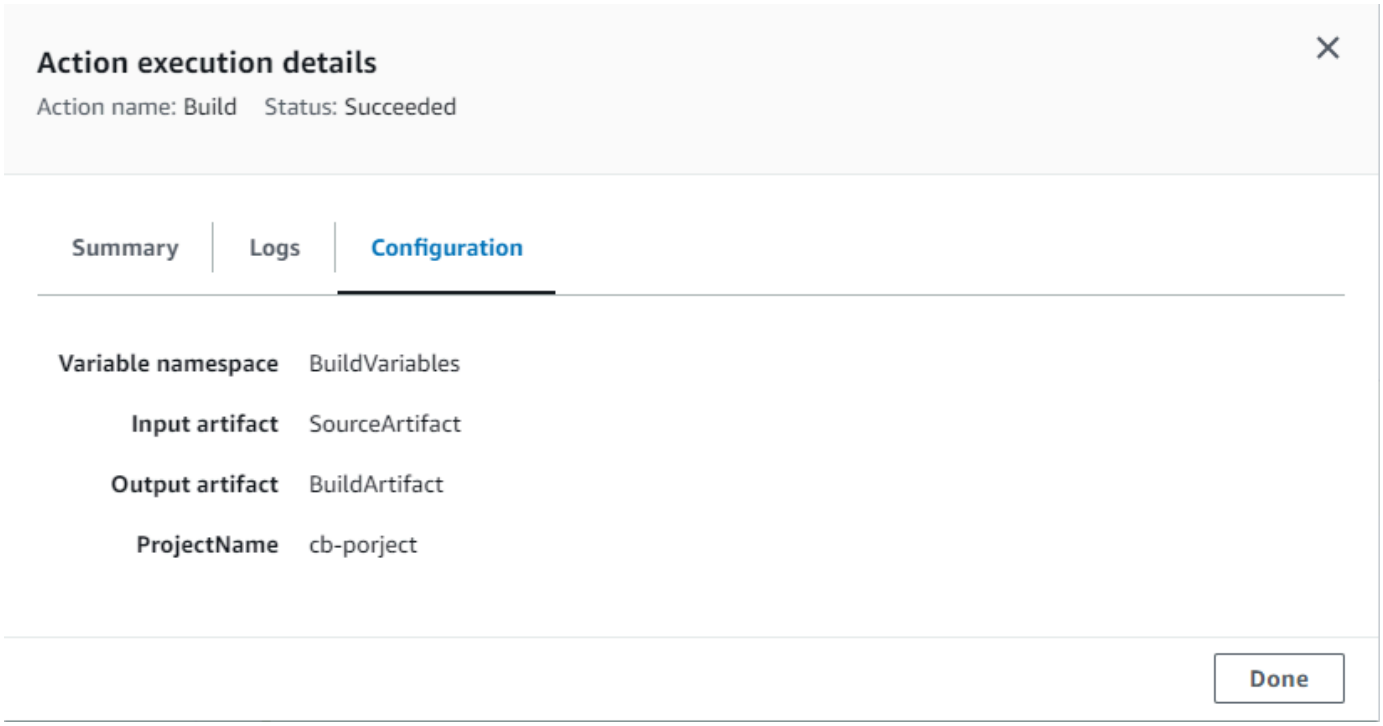
   > **Note**
   >
   > Die Registerkarte Protokolle ist für CloudFormation Aktionen CodeBuild und Aktionen verfügbar.
3. Um die Aktionsübersicht für eine Aktion in einer Phase einer Pipeline anzuzeigen, wählen Sie Details zur Aktion anzeigen und dann die Registerkarte Zusammenfassung aus.



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-Managementkonsole 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 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 account 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. Wenn Sie beispielsweise Details zu einer Pipeline mit dem Namen *MyFirstPipeline* anzeigen möchten, geben Sie Folgendes ein:

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

   Dieser Befehl gibt die Struktur der Pipeline zurück.

Regelergebnisse für Stufenbedingungen im Ausführungsverlauf anzeigen

Sie können die Regelergebnisse für eine Ausführung anzeigen, bei der eine Stufenbedingung eine Regel ausgeführt hat und ein Ergebnis für die Phase ausgelöst hat, z. B. ein Rollback oder ein Fehler.

Gültige Statuswerte für Bedingungen und Regeln lauten wie folgt: `InProgress | Failed | Errored | Succeeded | Cancelled | Abandoned | Overridden`

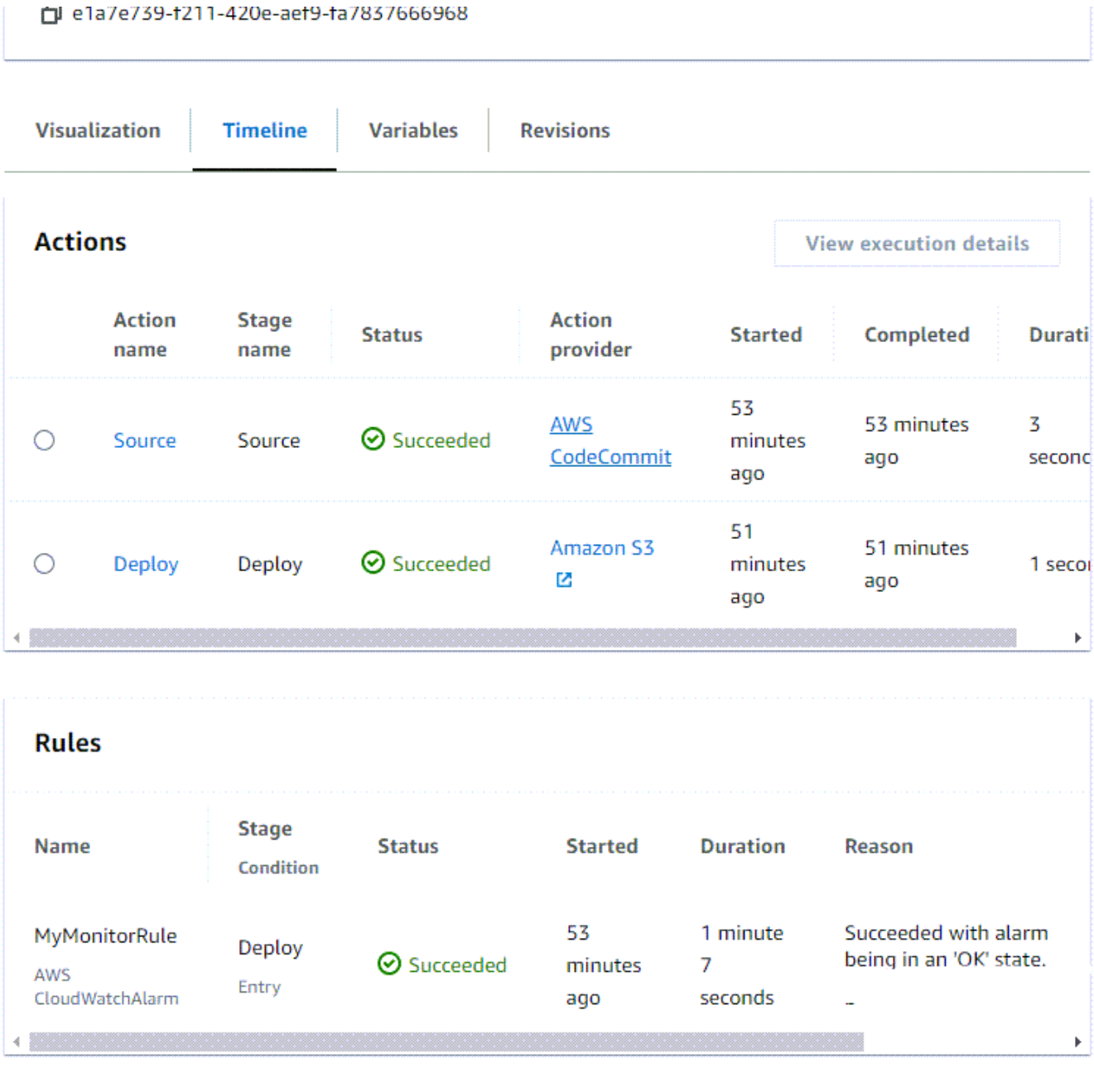
Regelergebnisse für Phasenbedingungen im Ausführungsverlauf anzeigen (Konsole)

Sie können die Konsole verwenden, um die Regelergebnisse für eine Ausführung anzuzeigen, bei der eine Stufenbedingung eine Regel ausgeführt und ein Ergebnis für die Phase ausgelöst hat.

Regelergebnisse für Phasenbedingungen anzeigen (Konsole)

1. Melden Sie sich bei der an AWS-Managementkonsole 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 Verlauf und dann die Ausführung aus. Wählen Sie auf der Seite „Verlauf“ die Registerkarte „Zeitleiste“. Sehen Sie sich unter Regeln die Regelergebnisse für die Ausführung an.



Regelergebnisse für Phasenbedingungen mit **`list-rule-executions`** (CLI) anzeigen

Sie können die CLI verwenden, um die Regelergebnisse für eine Ausführung anzuzeigen, bei der eine Stufenbedingung eine Regel ausgeführt und ein Ergebnis für die Phase ausgelöst hat.

- Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) und verwenden Sie die AWS CLI , um den list-rule-executions Befehl für eine Pipeline mit dem Namen auszuführen*`MyPipeline`*:

```
aws codepipeline list-rule-executions --pipeline-name MyFirstPipeline
```

Dieser Befehl gibt eine Liste aller abgeschlossenen Regelausführungen zurück, die mit der Pipeline verknüpft sind.

Das folgende Beispiel zeigt die zurückgegebenen Daten für eine Pipeline mit einer Stufenbedingung, in der die Regel benannt *MyMonitorRule* ist.

```
{
    "ruleExecutionDetails": [
        {
            "pipelineExecutionId": "e1a7e739-f211-420e-aef9-fa7837666968",
            "ruleExecutionId": "3aafc0c7-0e1c-44f1-b357-d1b16a28e483",
            "pipelineVersion": 9,
            "stageName": "Deploy",
            "ruleName": "MyMonitorRule",
            "startTime": "2024-07-29T15:55:01.271000+00:00",
            "lastUpdateTime": "2024-07-29T15:56:08.682000+00:00",
            "status": "Succeeded",
            "input": {
                "ruleTypeId": {
                    "category": "Rule",
                    "owner": "AWS",
                    "provider": "CloudWatchAlarm",
                    "version": "1"
                },
                "configuration": {
                    "AlarmName": "CWAlarm",
                    "WaitTime": "1"
                },
                "resolvedConfiguration": {
                    "AlarmName": "CWAlarm",
                    "WaitTime": "1"
                },
                "region": "us-east-1",
                "inputArtifacts": []
            },
            "output": {
                "executionResult": {
                    "externalExecutionSummary": "Succeeded with alarm 'CWAlarm'
 being i
```

```
n an 'OK' state."
                }
            }
        }
```

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)
- Löschen einer Pipeline (CLI)

Löschen einer Pipeline (Konsole)

So löschen Sie eine Pipeline

1. Melden Sie sich bei der an AWS-Managementkonsole 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 eine Pipeline mit dem Namen *MyFirstPipeline* zu löschen, gehen Sie wie folgt vor:

   ```
   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 einer anderen verwendet werden AWS Konto

Sie möchten möglicherweise eine Pipeline erstellen, die Ressourcen verwendet, die von einem anderen AWS -Konto erstellt oder verwaltet werden. Beispielsweise möchten Sie ein Konto für Ihre Pipeline und ein anderes Konto 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 auf demselben 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.
> - Cross-account Aktionen werden für die folgenden Aktionstypen nicht unterstützt:
>   - Jenkins-Build-Aktionen

Für dieses Beispiel müssen Sie einen zu verwendenden AWS Key Management Service (AWS KMS) -Schlüssel 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. Bei kontoübergreifenden Aktionen können Sie nur die Schlüssel-ID oder den Schlüssel-ARN verwenden, um den Schlüssel zu identifizieren. Cross-account Bei Aktionen wird die Rolle

des anderen Kontos (AccountB) verwendet. Wenn Sie also die Schlüssel-ID angeben, wird der Schlüssel des anderen Kontos (accounB) verwendet.

In dieser exemplarischen Vorgehensweise und ihren Beispielen *AccountA* wird das Konto dargestellt, das ursprünglich zum Erstellen der Pipeline verwendet wurde. Es hat Zugriff auf den Amazon S3-Bucket, der zum Speichern von Pipeline-Artefakten verwendet wird, sowie auf die von AWS CodePipeline verwendete Servicerolle. *AccountB*ist das Konto, das ursprünglich zur Erstellung der CodeDeploy Anwendung, Bereitstellungsgruppe und Servicerolle verwendet wurde, die von verwendet wurden CodeDeploy.

*AccountA*Um eine Pipeline für die Verwendung der von erstellten CodeDeploy Anwendung zu bearbeiten*AccountB*, *AccountA* müssen Sie:

- Fordern Sie den ARN oder die Konto-ID von an *AccountB* (in dieser exemplarischen Vorgehensweise lautet die *AccountB* ID*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 die Berechtigungen zur Verwendung dieses Schlüssels (*CodePipeline_Service_Role*) und*AccountB*.
- Erstellen Sie eine Amazon S3-Bucket-Richtlinie, die *AccountB* Zugriff auf den Amazon S3-Bucket gewährt (z. B.*codepipeline-us-east-2-1234567890*).
- Erstellen Sie eine Richtlinie, die es ermöglicht, eine von konfigurierte Rolle *AccountA* zu übernehmen*AccountB*, und hängen Sie diese Richtlinie an die Servicerolle an (*CodePipeline_Service_Role*).
- Bearbeiten Sie die Pipeline so, dass sie den vom Kunden verwalteten AWS KMS Schlüssel anstelle des Standardschlüssels verwendet.

*AccountB*Um einer Pipeline, die in erstellt wurde, den Zugriff auf ihre Ressourcen zu ermöglichen*AccountA*, *AccountB* müssen Sie:

- Fordern Sie den ARN oder die Konto-ID von an *AccountA* (in dieser exemplarischen Vorgehensweise lautet die *AccountA* ID*012ID_ACCOUNT_A*).
- Erstellen Sie eine Richtlinie, die auf die Amazon EC2-Instance-Rolle angewendet wird CodeDeploy , für die der Zugriff auf den Amazon S3-Bucket () *codepipeline-us-east-2-1234567890* konfiguriert ist.

- Erstellen Sie eine Richtlinie, die auf die Amazon EC2-Instance-Rolle angewendet wird CodeDeploy , für die konfiguriert wurde und die den Zugriff auf den vom AWS KMS Kunden verwalteten Schlüssel ermöglicht, der zum Verschlüsseln der Pipeline-Artefakte verwendet wird. *AccountA*
- Konfigurieren und fügen Sie eine IAM-Rolle (*CrossAccount_Role*) mit einer Vertrauensbeziehungsrichtlinie hinzu, die es der CodePipeline Servicerolle ermöglicht, die Rolle *AccountA* zu übernehmen.
- Erstellen Sie eine Richtlinie, die den Zugriff auf die für die Pipeline benötigten Bereitstellungsressourcen ermöglicht, und fügen Sie sie an*CrossAccount_Role*.
- Erstellen Sie eine Richtlinie, die den Zugriff auf den Amazon S3-Bucket (*codepipeline-us-east-2-1234567890*) ermöglicht, und hängen Sie ihn an*CrossAccount_Role*.

Themen

- Voraussetzung: Erstellen Sie eine AWS KMS Verschlüsselungsschlüssel
- Schritt 1: Einrichten von Kontorichtlinien und Rollen
- Schritt 2: Bearbeiten der Pipeline

Voraussetzung: Erstellen Sie eine AWS KMS Verschlüsselungsschlüssel

Customer-managed Schlüssel sind spezifisch für eine Region, genau wie alle AWS KMS Schlüssel. 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, müssen Sie AWS KMS

1. Melden Sie sich bei with AWS-Managementkonsole an *AccountA* 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 AWS Konto hinzufügen aus. Geben Sie die Konto-ID ein*AccountB*, um den ARN zu vervollständigen, und wählen Sie dann Weiter.
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 von beiden Seiten*AccountA*. *AccountB*

Themen

- Konfigurieren Sie Richtlinien und Rollen in dem Konto, das die Pipeline erstellen wird (Konto A)
- Konfigurieren Sie Richtlinien und Rollen in dem Konto, dem das gehört AWS Ressource (Konto B)

Konfigurieren Sie Richtlinien und Rollen in dem Konto, das die Pipeline erstellen wird (***Konto A***)

Um eine Pipeline zu erstellen, die CodeDeploy Ressourcen verwendet, die mit einem anderen AWS Konto verknüpft sind, *AccountA* müssen Richtlinien sowohl für den Amazon S3-Bucket, in dem Artefakte gespeichert werden, als auch für die Servicerolle für CodePipeline konfiguriert werden.

Um eine Richtlinie für den Amazon S3-Bucket zu erstellen, die Zugriff auf AccountB (Konsole) gewährt

1. Melden Sie sich bei AWS-Managementkonsole with an *AccountA* und öffnen Sie die Amazon S3-Konsole unter https://console.aws.amazon.com/s3/.

2. Wählen Sie in der Liste der Amazon S3-Buckets den Amazon S3-Bucket aus, in dem Artefakte für Ihre Pipelines gespeichert werden. Dieser Bucket hat einen Namen. Dabei *region* handelt es sich um die AWS Region`codepipeline-`*region*`-`*1234567EXAMPLE*, in der Sie die Pipeline erstellt haben, und *1234567EXAMPLE* es handelt sich um eine zehnstellige Zufallszahl, die sicherstellt, dass der Bucket-Name eindeutig ist (z. B.). *codepipeline-us-east-2-1234567890*
3. Wählen Sie auf der Detailseite für den Amazon S3-Bucket die Option Properties aus.
4. Erweitern Sie im Eigenschaftenbereich das Feld Permissions und wählen Sie Add bucket policy aus.

   > **Note**
   >
   > Wenn Ihrem Amazon 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 IAM-Richtlinien.

5. Erstellen Sie im Bucket Policy Editor-Fenster eine Richtlinie, die den *AccountB* Zugriff auf die Pipeline-Artefakte ermöglicht und *AccountB* die Möglichkeit bietet, Ausgabeartefakte hinzuzufügen, wenn sie durch eine Aktion, z. B. eine benutzerdefinierte Quelle oder eine Build-Aktion, erstellt werden.
6. Wählen Sie Save und schließen Sie dann den Richtlinien-Editor.
7. Wählen Sie Speichern, um die Berechtigungen für den Amazon S3-Bucket zu speichern.

Um eine Richtlinie für die Servicerolle für CodePipeline (Konsole) zu erstellen

1. Melden Sie sich bei with AWS-Managementkonsole an *AccountA* und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen.
3. Wählen Sie in der Liste der Rollen unter Role Name (Rollenname) den Namen der Service-Rolle für CodePipeline aus.
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 die Übernahme der Rolle *AccountB* zu ermöglichen. Im folgenden Beispiel *012ID_ACCOUNT_B* ist das ARN für*AccountB*:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": [
            "arn:aws:iam::111122223333: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 das gehört AWS Ressource (*Konto B*)

Wenn Sie eine Anwendungs-, Bereitstellungs- und Bereitstellungsgruppe in erstellen CodeDeploy, erstellen Sie auch eine Amazon 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 erstellte Pipeline *AccountA* die in erstellten CodeDeploy Ressourcen verwenden kann*AccountB*, müssen Sie:

- Konfigurieren Sie eine Richtlinie für die Instance-Rolle, die ihr den Zugriff auf den Amazon S3-Bucket ermöglicht, in dem Pipeline-Artefakte gespeichert sind.
- Erstellen Sie eine zweite Rolle, die für den kontoübergreifenden Zugriff *AccountB* konfiguriert ist.

  Diese zweite Rolle muss nicht nur Zugriff auf den Amazon S3-Bucket haben*AccountA*, sie muss auch eine Richtlinie enthalten, die den Zugriff auf die CodeDeploy Ressourcen ermöglicht, und eine Vertrauensbeziehungsrichtlinie, die es der CodePipeline Servicerolle ermöglicht, die Rolle *AccountA* zu übernehmen.

> **Note**
>
> Diese Richtlinien beziehen sich speziell auf 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 ihren Ressourcenbedarf zugeschnitten sind.

Um eine Richtlinie für die Amazon EC2-Instance-Rolle zu erstellen, die für CodeDeploy (Konsole) konfiguriert ist

1. Melden Sie sich bei AWS-Managementkonsole with an *AccountB* und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/
2. Wählen Sie im Navigationsbereich Rollen.
3. Wählen Sie in der Rollenliste unter Rollenname den Namen der Servicerolle aus, die als Amazon 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-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-Bucket zu gewähren, der von *AccountA* zum Speichern von Artefakten für Pipelines verwendet wird (in diesem Beispiel): *codepipeline-us-east-2-1234567890*

   JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "s3:Get*"
       ],
       "Resource": [
         "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

```
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ]
    }
  ]
}
```

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, in der festgelegt ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** wird AWS KMS , wo der ARN des vom Kunden verwalteten Schlüssels angezeigt wird, der erstellt *AccountA* und so konfiguriert ist, dass er verwendet werden kann*AccountB*:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt",
                "kms:ReEncrypt*",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:us-
east-1:111122223333: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 Sie für den kontoübergreifenden Zugriff verwenden können, und konfigurieren Sie sie so, dass die CodePipeline Servicerolle in die Rolle übernehmen *AccountA* kann. Diese Rolle muss Richtlinien enthalten, die den Zugriff auf die CodeDeploy Ressourcen und den Amazon S3-Bucket ermöglichen, in dem Artefakte gespeichert werden. *AccountA*

So konfigurieren Sie die kontoübergreifende Rolle in IAM

1. Melden Sie sich bei AWS-Managementkonsole with an *AccountB* und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam
2. Wählen Sie im Navigationsbereich Rollen aus. Wählen Sie Create role (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, in dem die Pipeline erstellt werden soll CodePipeline (*AccountA*), und wählen Sie dann Weiter: Berechtigungen aus.

> **Important**
>
> In diesem Schritt wird die Vertrauensbeziehungsrichtlinie zwischen *AccountB* und erstellt*AccountA*. Dadurch wird jedoch Root-Zugriff auf das Konto gewährt, und es wird CodePipeline empfohlen, den Bereich auf die CodePipeline Service-Rolle in zu reduzieren. *AccountA* Folgen Sie Schritt 16, um die Berechtigungen einzuschränken.

4. Wählen AmazonS3ReadOnlyAccess Sie unter Richtlinien zum Anhängen von Berechtigungen die Option 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 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 ermöglichen:

   JSON

```
{
  "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-Bucket abrufen und Ausgabeartefakte in den Amazon S3-Bucket einfügen kann*AccountA*:

    JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "s3:GetObject*",
         "s3:PutObject",
         "s3:PutObjectAcl"
       ],
       "Resource": [
         "arn:aws:s3:::amzn-s3-demo-bucket/*"
       ]
     }
   ]
}
```

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 AmazonS3ReadOnlyAccess in der Liste der Richtlinien unter Richtlinienname und wählen Sie das Löschsymbol (X) neben der Richtlinie aus. Wählen Sie nach Aufforderung Detach aus.
16. Wählen Sie die Registerkarte Vertrauensbeziehung und dann Vertrauensrichtlinie bearbeiten aus. Wählen Sie in der linken Spalte die Option Prinzipal hinzufügen aus. Wählen Sie als Principal-Typ die Option IAM-Rollen aus, und geben Sie dann den ARN für die CodePipeline Servicerolle unter an*AccountA*. `arn:aws:iam::Account_A:root`Aus der Liste für AWS Principals entfernen, 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 einem anderen AWS Konto zugeordnet sind. Sie können die Konsole jedoch verwenden, um die allgemeine Struktur der Pipeline zu erstellen, und dann die verwenden, AWS CLI um die Pipeline 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 einer anderen zugeordnet sind AWS Konto (AWS CLI)

1. Führen Sie an einem Terminal (Linux, macOS oder Unix) oder einer Befehlszeile (Windows) den get-pipeline Befehl in der Pipeline aus, zu der Sie Ressourcen hinzufügen möchten. Kopieren Sie die Ausgabe des Befehls in eine JSON-Datei. Für eine Pipeline mit dem Namen MyFirstPipeline geben Sie beispielsweise Folgendes ein:

   ```
   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. Dabei *codepipeline-us-east-2-1234567890* handelt es sich um den Namen des Amazon S3-Buckets, der zum Speichern von Artefakten für die Pipeline verwendet wird, und ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** um den ARN des kundenverwalteten Schlüssels, 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 zugehörigen CodeDeploy Ressourcen zu verwenden*AccountB*, einschließlich der `roleArn` Werte für die von Ihnen erstellte kontoübergreifende Rolle (). *CrossAccount_Role*

   Das folgende Beispiel zeigt JSON, das eine Bereitstellungsaktion mit dem Namen *ExternalDeploy* hinzufügt. Es verwendet die CodeDeploy Ressourcen, die *AccountB* in einer Phase mit dem Namen erstellt wurden*Staging*. Im folgenden Beispiel *AccountB* lautet der ARN für*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 einer anderen zugeordnet sind AWS Konto

1. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer start-pipeline-execution Befehlszeile (Windows) aus und geben Sie dabei den Namen der Pipeline an, ähnlich dem Folgenden:

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

Weitere Informationen finden Sie unter Manuelles Starten einer Pipeline.

2. Melden Sie sich bei with AWS-Managementkonsole an *AccountA* 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 die Aktion anzuzeigen, während Sie angemeldet sind*AccountA*. Melden Sie sich ab und melden Sie sich dann mit an*AccountB*, um die Bereitstellungsdetails einzusehen CodeDeploy.

Migrieren Sie Polling-Pipelines, um ereignisbasierte Änderungserkennung zu verwenden

AWS CodePipeline unterstützt eine vollständige, durchgängige kontinuierliche Bereitstellung. Dazu gehört auch, dass Ihre Pipeline bei jeder Codeänderung gestartet wird. Es gibt zwei unterstützte Methoden, um Ihre Pipeline bei einer Codeänderung zu starten: ereignisbasierte Änderungserkennung und Polling. Wir empfehlen die ereignisbasierte Änderungserkennung für Pipelines.

Verwenden Sie die hier aufgeführten Verfahren, um Ihre Polling-Pipelines auf die ereignisbasierte Änderungserkennungsmethode für Ihre Pipeline zu migrieren (zu aktualisieren).

Die empfohlene Methode zur ereignisbasierten Änderungserkennung für Pipelines wird von der Pipeline-Quelle bestimmt, z. B. CodeCommit In diesem Fall müsste die Polling-Pipeline beispielsweise zur ereignisbasierten Änderungserkennung mit migrieren. EventBridge

Wie migriert man Polling-Pipelines

Um Polling-Pipelines zu migrieren, ermitteln Sie Ihre Polling-Pipelines und legen Sie dann die empfohlene Methode zur ereignisbasierten Änderungserkennung fest:

- Verwenden Sie die Schritte unter, um Ihre Polling-Pipelines Polling-Pipelines in Ihrem Konto anzeigen zu ermitteln.
- Suchen Sie in der Tabelle Ihren Pipeline-Quelltyp und wählen Sie dann das Verfahren mit der Implementierung aus, das Sie für die Migration Ihrer Polling-Pipeline verwenden möchten. Jeder Abschnitt enthält mehrere Methoden für die Migration, z. B. die Verwendung der CLI oder CloudFormation.

Migrieren von Pipelines zur empfohlenen Methode zur Änderungserkennung		
Quelle der Pipeline	Empfohlene ereignisbasierte Erkennungsmethode	Verfahren zur Migration
AWS CodeCommit	EventBridge (empfohlen).	Siehe Migrieren Sie Polling-Pipelines mit einer Quelle CodeCommit.
Amazon S3	EventBridge und Bucket für Ereignisbenachrichtigungen aktiviert (empfohlen).	Siehe Migrieren Sie Polling-Pipelines mit einer S3-Quelle, die für Ereignisse aktiviert ist.
Amazon S3	EventBridge und ein AWS CloudTrail Trail.	Siehe Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail.
GitHub (per GitHub App)	Verbindungen (empfohlen)	Siehe Migrieren Sie Polling-Pipelines für eine GitHub Quellaktion (über die OAuth-App) zu Verbindungen.
GitHub (über die OAuth-App)	Webhooks	Siehe Migrieren Sie Polling-Pipelines für eine GitHub Quellaktion (über die OAuth-App) zu Webhooks.

> **Important**
>
> Für entsprechende Aktualisierungen der Pipeline-Aktionskonfiguration, z. B. Pipelines mit einer Aktion GitHub (über die OAuth-App), müssen Sie den `PollForSourceChanges` Parameter in der Konfiguration Ihrer Quellaktion 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 Polling zu konfigurieren, indem Sie beispielsweise eine Regel konfigurieren und den Parameter weglassen. EventBridge `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 Polling konfiguriert sind. Dies sind die Pipelines, die zur ereignisbasierten Änderungserkennung migriert werden müssen.

Polling-Pipelines in Ihrem Konto anzeigen (Skript)

Gehen Sie wie folgt vor, um mithilfe eines Skripts 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 zu erstellen PollingPipelinesExtractor.sh.

     ```
     vi PollingPipelinesExtractor.sh
     ```

   - Um ein Python-Skript zu verwenden, führen Sie den folgenden Befehl aus, um ein neues Python-Skript mit dem Namen zu erstellen PollingPipelinesExtractor.py.

     ```
     vi PollingPipelinesExtractor.py
     ```

2. Kopieren Sie den folgenden Code und fügen Sie ihn in das PollingPipelinesExtractor Skript 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 angegebene Skript auszuführen PollingPipelinesExtractor.sh. 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 auszuführen. PollingPipelinesExtractor.py 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 Polling-Quelle auf die empfohlene ereignisbasierte Methode zur Erkennung von Änderungen.

> **Note**
>
> Ihre Polling-Pipelines werden durch die Aktionskonfiguration der Pipeline für den Parameter bestimmt. `PollForSourceChanges` Wenn in der Pipeline-Quellkonfiguration der `PollForSourceChanges` Parameter weggelassen wird, wird Ihr Repository CodePipeline standardmäßig nach Quelländerungen abgefragt. Dieses Verhalten ist dasselbe, als ob `PollForSourceChanges` „enthalten“ und auf „true“ gesetzt wäre. Weitere Informationen finden Sie in den Konfigurationsparametern für die Quellaktion Ihrer Pipeline, z. B. in den Konfigurationsparametern für Amazon S3-Quellaktionen unterReferenz zur Amazon S3-Quellaktion.

Beachten Sie, dass dieses Skript auch eine CSV-Datei generiert, die die Liste der Polling-Pipelines in Ihrem Konto enthält, und die CSV-Datei im aktuellen Arbeitsordner speichert.

Migrieren Sie Polling-Pipelines mit einer Quelle CodeCommit

Sie können Ihre Polling-Pipeline migrieren, um Änderungen in Ihrem CodeCommit Quell-Repository oder Ihrem Amazon S3-Quell-Bucket zu erkennen. EventBridge

CodeCommit— Bei einer Pipeline mit einer CodeCommit Quelle ändern Sie die Pipeline so, dass die Änderungserkennung automatisch erfolgt EventBridge. Wählen Sie aus den folgenden Methoden, um die Migration zu implementieren:

- Konsole: Migrieren Sie Polling-Pipelines (CodeCommit oder Amazon S3-Quelle) (Konsole)
- CLI: Migrieren Sie Polling-Pipelines (CodeCommit Quelle) (CLI)
- CloudFormation: Migrieren Sie Polling-Pipelines (CodeCommit Quelle) (CloudFormation Vorlage)

Migrieren Sie Polling-Pipelines (CodeCommit oder Amazon S3-Quelle) (Konsole)

Sie können die CodePipeline Konsole verwenden, um Ihre Pipeline zu aktualisieren, damit Sie EventBridge Änderungen in Ihrem CodeCommit Quell-Repository oder Ihrem Amazon S3-Quell-Bucket erkennen können.

> **Note**
>
> Wenn Sie die Konsole verwenden, um eine Pipeline zu bearbeiten, die über ein CodeCommit Quell-Repository oder einen Amazon S3-Quell-Bucket verfügt, werden die Regel und die IAM-Rolle für Sie erstellt. Wenn Sie die verwenden, AWS CLI um die Pipeline zu 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 Eine Pipeline, Phasen und Aktionen erstellen.

So bearbeiten Sie die Quellphase einer Pipeline

1. Melden Sie sich bei der an AWS-Managementkonsole 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 bei Änderungen automatisch zu starten (empfohlen).

Eine Meldung mit der EventBridge Regel, die für diese Pipeline erstellt werden soll, wird angezeigt. Wählen Sie Aktualisieren aus.

Wenn Sie eine Pipeline aktualisieren, die über eine Amazon S3-Quelle verfügt, wird die folgende Meldung angezeigt. Wählen Sie Aktualisieren aus.

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 die verwenden AWS CLI , um eine Änderung an der in der Quellphase der Pipeline angegebenen Quelle festzuschreiben.

Migrieren Sie Polling-Pipelines (CodeCommit Quelle) (CLI)

Gehen Sie wie folgt vor, um eine Pipeline zu bearbeiten, die Polling (regelmäßige Überprüfungen) verwendet, um eine EventBridge Regel zum Starten der Pipeline zu verwenden. Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Eine Pipeline, Phasen und Aktionen erstellen.

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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

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 EventBridge Servicerolle zu übernehmen. Geben Sie der Vertrauensrichtlinie den Namen `trustpolicyforEB.json`.

      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`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333: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 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 es CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die folgenden Parameter ein:

   - 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
   ```

4. (Optional) Um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Bild-ID zu konfigurieren, verwenden Sie den folgenden JSON-Code in Ihrem CLI-Befehl. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

- `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
- `COMMIT_ID`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
- In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "pipeline-ARN",
            "InputTransformer": {
                "sourceRevisions": {
                    "actionName": "Source",
                    "revisionType": "COMMIT_ID",
                    "revisionValue": "<revisionValue>"
                },
                "variables": [
                    {
                        "name": "Branch_Name",
                        "value": "value"
                    }
                ]
            }
        }
    ]
}
```

Migrieren Sie Polling-Pipelines (CodeCommit Quelle) (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 CloudFormation es zum Erstellen und Verwalten Ihrer Pipelines 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

```
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 Pipeline zu aktualisieren CloudFormation Vorlage und EventBridge Regel erstellen

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` CloudFormation RessourceResources, 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? Das Hinzufügen der `AWS::IAM::Role` Ressource ermöglicht CloudFormation das Erstellen von Berechtigungen für EventBridge. Diese Ressource wird Ihrem 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` CloudFormation Ressource`Resources`, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis, das Push-Änderungen an deinem Repository überwacht. Wenn eine Statusänderung des Repositorys 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 CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem 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. (Optional) Verwenden Sie den folgenden YAML-Snippet, um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Bild-ID zu konfigurieren. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

   - `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
   - `COMMIT_ID`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
   - In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`
   - Die Ausgabevariablen für `BranchName` und `Value` sind angegeben.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
```

```
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer, und öffnen Sie dann die 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 die Vorlage hoch und zeigen Sie dann die in CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
7. Wählen Sie 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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

- 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 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 Polling-Pipelines mit einer S3-Quelle, die für Ereignisse aktiviert ist

Bei einer Pipeline mit einer Amazon S3-Quelle ändern Sie die Pipeline so, dass die Änderungserkennung durch EventBridge und mit einem Quell-Bucket, der für Ereignisbenachrichtigungen aktiviert ist, automatisiert wird. Dies ist die empfohlene Methode, wenn Sie die CLI verwenden oder CloudFormation Ihre Pipeline migrieren.

> **Note**
>
> Dazu gehört auch die Verwendung eines Buckets, der für Ereignisbenachrichtigungen aktiviert ist, sodass Sie keinen separaten CloudTrail Trail erstellen müssen. Wenn Sie die Konsole verwenden, werden eine Event-Regel und ein CloudTrail Trail für Sie eingerichtet. Diese Schritte 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 einem CloudTrail S3-Trail (CLI)
- CloudFormation: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail (CloudFormation Vorlage)

Migrieren Sie Polling-Pipelines mit einer für Ereignisse aktivierten S3-Quelle (CLI)

Gehen Sie wie folgt vor, um eine Pipeline, die Polling (regelmäßige Überprüfungen) verwendet, zu bearbeiten und stattdessen ein Ereignis zu verwenden. EventBridge Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Eine Pipeline, Phasen und Aktionen erstellen.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und erstellen dann die folgenden Ressourcen:

- EventBridge Regel für Ereignisse
- IAM-Rolle, damit das EventBridge Ereignis Ihre Pipeline starten kann

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Erteilen Sie Berechtigungen EventBridge zur Verwendung CodePipeline zum Aufrufen der Regel. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel zum Erstellen der Vertrauensrichtlinie, mit der EventBridge die Servicerolle übernehmen kann. Geben Sie ihr den Namen `trustpolicyforEB.json`.

      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 EventBridge erstellt.

```
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`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333: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\":
[\"amzn-s3-demo-source-bucket\"]}}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-
for-MyRule"
```

3. Um es CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die Parameter und an`--rule`. `--targets`

   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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

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 `amzn-s3-demo-source-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": "amzn-s3-demo-source-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 für Ereignisse aktivierten S3-Quelle (CloudFormation Vorlage)

Dieses Verfahren gilt für eine Pipeline, bei der für den Quell-Bucket Ereignisse aktiviert sind.

Gehen Sie wie folgt vor, um Ihre Pipeline mit einer Amazon S3-Quelle vom Polling 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 Ihrer Vorlage die folgenden Ressourcen hinzu:

- EventBridge Regel und IAM-Rolle, damit dieses Ereignis Ihre Pipeline starten kann.

Wenn Sie es verwenden CloudFormation , um Ihre Pipelines zu erstellen und zu verwalten, 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 Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` CloudFormation RessourceResources, um die IAM-Rolle zu konfigurieren, die es Ihrem Event ermöglicht, Ihre Pipeline zu starten. 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? Das Hinzufügen einer `AWS::IAM::Role` Ressource ermöglicht CloudFormation das Erstellen von Berechtigungen für EventBridge. Diese Ressource wird Ihrem 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` CloudFormation Ressource, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erstellt ein Ereignis, das die Erstellung oder Löschung von Objekten in Ihrem Amazon 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 Hinzufügen der `AWS::Events::Rule` Ressource kann CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem 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 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 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 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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

- 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. CloudFormation

> **Note**
>
> Hören Sie hier nicht auf. Obwohl Ihre Pipeline erstellt wurde, müssen Sie eine zweite CloudFormation Vorlage für Ihre Amazon 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

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "appconfig:StartDeployment",
                "appconfig:StopDeployment",
                "appconfig:GetDeployment"
            ],
            "Resource": [
                "arn:aws:appconfig:*:111122223333:application/
[[Application]]",
```

```
                "arn:aws:appconfig:*:111122223333:application/[[Application]]/
*",
                "arn:aws:appconfig:*:111122223333:deploymentstrategy/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail

Ändern Sie bei einer Pipeline mit einer Amazon S3-Quelle die Pipeline so, dass die Änderungserkennung automatisch erfolgt. EventBridge Wählen Sie aus den folgenden Methoden, um die Migration zu implementieren:

- Konsole: Migrieren Sie Polling-Pipelines (CodeCommit oder Amazon S3-Quelle) (Konsole)
- CLI: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (CLI)
- CloudFormation: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail (CloudFormation Vorlage)

Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (CLI)

Gehen Sie wie folgt vor, um eine Pipeline, die Polling (regelmäßige Überprüfungen) verwendet, zu bearbeiten und stattdessen ein Ereignis zu verwenden. EventBridge Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Eine Pipeline, Phasen und Aktionen erstellen.

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 eine zu erstellen AWS CloudTrail Protokollierung verfolgen und aktivieren

Um den zum Erstellen eines Trails AWS CLI zu verwenden, 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 Einen Trail mit der AWS Befehlszeilenschnittstelle erstellen.

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 `amzn-s3-demo-source-bucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket
   ```

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? Mit diesem Befehl werden Ereignisse gefiltert.

Beispiel:

Im folgenden Beispielbefehl werden `--trail-name` und `--event-selectors` verwendet, um die Verwaltung von Datenereignissen für einen Quell-Bucket und einen Präfix namens `amzn-s3-demo-source-bucket/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:::amzn-s3-
demo-source-bucket/myFolder/file.zip"] }] }]'
```

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Erteilen Sie Berechtigungen EventBridge zur 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, die es ermöglicht, die EventBridge Servicerolle zu übernehmen. Geben Sie ihr den Namen `trustpolicyforEB.json`.

      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 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`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333: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\":[\"amzn-s3-
demo-source-bucket\"],\"key\":[\"my-key\"]}}}
 --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Um es 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
   ```

4. (Optional) Verwenden Sie den folgenden JSON-Code in Ihrem CLI-Befehl, um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Bild-ID zu konfigurieren. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

   - `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
   - `S3_OBJECT_VERSION_ID`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
   - In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`

   ```
   {
       "Rule": "my-rule",
       "Targets": [
           {
               "Id": "MyTargetId",
               "Arn": "ARN",
               "InputTransformer": {
                   "InputPathsMap": {
                       "revisionValue": "$.detail.object.version-id"
                   },
   ```

```
                "InputTemplate": {
                    "sourceRevisions": {
                        "actionName": "Source",
                        "revisionType": "S3_OBJECT_VERSION_ID",
                        "revisionValue": "<revisionValue>"
                    }
                }
            }
        }
    ]
}
```

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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

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 `amzn-s3-demo-source-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": "amzn-s3-demo-source-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 S3-Trail CloudTrail (CloudFormation Vorlage)

Gehen Sie wie folgt vor, um Ihre Pipeline mit einer Amazon S3-Quelle vom Polling 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 Ihrer Vorlage die folgenden Ressourcen hinzu:

- EventBridge erfordert, dass alle Amazon 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 Protokollieren von Datenereignissen für Pfade und Protokollieren von Verwaltungsereignissen für Pfade.
- EventBridge Regel und IAM-Rolle, damit dieses Ereignis Ihre Pipeline starten kann.

Wenn Sie es verwenden CloudFormation , um Ihre Pipelines zu erstellen und zu verwalten, 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 Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` CloudFormation RessourceResources, um die IAM-Rolle zu konfigurieren, die es Ihrem Event ermöglicht, Ihre Pipeline zu starten. 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? Das Hinzufügen einer `AWS::IAM::Role` Ressource ermöglicht CloudFormation das Erstellen von Berechtigungen für EventBridge. Diese Ressource wird Ihrem 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` CloudFormation Ressource, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erstellt ein Ereignis`CopyObject`, `PutObject` das Ihren `CompleteMultipartUpload` Amazon 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 Hinzufügen der `AWS::Events::Rule` Ressource kann CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem 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. (Optional) Verwenden Sie den folgenden YAML-Snippet, um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Bild-ID zu konfigurieren. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

   - `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
   - `S3_OBJECT_VERSION_ID`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
   - In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    InputPathsMap:
      revisionValue: "$.detail.object.version-id"
    InputTemplate:
      sourceRevisions:
        actionName: Source
        revisionType: S3_OBJECT_VERSION_ID
        revisionValue: '<revisionValue>'
```

5. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die CloudFormation Konsole.
6. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
7. Laden Sie Ihre aktualisierte Vorlage hoch und zeigen Sie dann die in CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
8. Wählen Sie 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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

- 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-Pipeline zu erstellen

- Verwenden Sie in einer separaten Vorlage unter `Resources` die `AWS::CloudTrail::Trail` CloudFormation RessourcenAWS::S3::Bucket,AWS::S3::BucketPolicy, und, um eine einfache Bucket-Definition und einen Pfad für bereitzustellen CloudTrail.

  Warum nehme ich diese Änderung vor? Da das aktuelle Limit von fünf Trails pro Konto gilt, muss der CloudTrail Trail separat erstellt und verwaltet werden. (Siehe Grenzwerte unter AWS CloudTrail.) Sie können jedoch viele Amazon S3-Buckets in einen einzigen Trail aufnehmen, sodass Sie den Trail einmal erstellen und dann bei Bedarf Amazon 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 verwenden CloudFormation , wird Ihre Pipeline ausgelöst, wenn Dateien in Ihrem Repository erstellt oder aktualisiert werden.

> **Note**
>
> Hören Sie hier nicht auf. Obwohl Ihre Pipeline erstellt wurde, müssen Sie eine zweite CloudFormation Vorlage für Ihre Amazon 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 GitHub Quellaktion (über die OAuth-App) zu Verbindungen

Sie können eine Quellaktion GitHub (über die OAuth-App) migrieren, um Verbindungen für Ihr externes Repository zu verwenden. Dies ist die empfohlene Methode zur Erkennung von Änderungen für Pipelines mit einer Quellaktion GitHub (über die OAuth-App).

Für eine Pipeline mit einer Quellaktion GitHub (über die OAuth-App) empfehlen wir, die Pipeline so zu ändern, dass eine Aktion GitHub (über die GitHub App) verwendet wird, sodass die

Änderungserkennung automatisiert erfolgt. AWS CodeConnections Weitere Informationen zum Arbeiten mit Verbindungen finden Sie unter. GitHub 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 (über die OAuth-App)

Verwenden Sie die Pipeline-Bearbeitungsseite, um Ihre Aktion GitHub (über die OAuth-App) durch eine Aktion GitHub (über GitHub die App) zu ersetzen.

Um Ihre Aktion GitHub (über die OAuth-App) zu ersetzen

1. Loggen Sie sich in die CodePipeline Konsole ein.
2. Wählen Sie Ihre Pipeline aus und klicken Sie auf Bearbeiten. Wählen Sie auf Ihrer Quellstufe Stufe bearbeiten aus. Es wird eine Meldung angezeigt, in der empfohlen wird, Ihre Aktion zu aktualisieren.
3. Wählen Sie unter Aktionsanbieter die Option GitHub (über GitHub App) aus.
4. Führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie unter Verbindung die Option Verbinden mit aus, falls Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben GitHub. Fahren Sie mit Schritt 2 fort: Erstellen Sie eine Verbindung zu GitHub.
   - Wenn Sie bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Verbindung aus. Fahren Sie mit Schritt 3: Speichern Sie die Quellaktion für Ihre Verbindung fort.

Schritt 2: Erstellen Sie eine Verbindung zu GitHub

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die GitHub Seite Verbinden mit angezeigt.

Um eine Verbindung herzustellen zu GitHub

1. Unter GitHub Verbindungseinstellungen wird Ihr Verbindungsname im Feld Verbindungsname angezeigt.

Wählen Sie unter GitHub Apps eine App-Installation aus oder wählen Sie Eine 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, um fortzufahren.
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 Verbinden mit 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 Folgendes ein, `owner-name/repository-name` wie in diesem Beispiel gezeigt:
>
> ```
> my-account/my-repository
> ```

2. Wählen Sie im Ausgabe-Artefaktformat 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 gezeigtFügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub Enterprise Server oder hinzu GitHub GitLab.com. Ein Tutorial, das Ihnen zeigt, wie Sie die Option „Vollständiger Klon“ verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

3. Unter Ausgabeartefakte können Sie den Namen des Ausgabe-Artefakts für diese Aktion beibehalten, z. B. `SourceArtifact` Wählen Sie Fertig, um die Seite „Aktion 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.

Erstellen Sie eine Verbindung 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 oder hergestellt haben CloudFormation, bearbeiten Sie die Verbindung mithilfe der Konsole, um ihren Status zu ermittelnAVAILABLE.

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 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 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 GitHub Quellaktion (über die OAuth-App) zu Webhooks

Sie können Ihre Pipeline migrieren, um mithilfe von Webhooks Änderungen in Ihrem Quell-Repository zu erkennen. GitHub Diese Migration zu Webhooks ist nur für die Aktion GitHub (über die OAuth-App) vorgesehen.

- Konsole: Migrieren Sie Polling-Pipelines zu Webhooks (über die OAuth-AppGitHub) (Quellaktionen) (Konsole)

- CLI: Migrieren Sie Polling-Pipelines zu Webhooks GitHub (über die OAuth-App) (Quellaktionen) (CLI)
- CloudFormation: Aktualisieren Sie die Pipelines für Push-Ereignisse GitHub (über die OAuth-App) (Quellaktionen) (CloudFormation Vorlage)

> **Important**
>
> Verwenden Sie beim Erstellen von CodePipeline Webhooks nicht Ihre eigenen Anmeldeinformationen und verwenden Sie nicht dasselbe geheime Token für mehrere Webhooks wieder. Generieren Sie für optimale Sicherheit ein eindeutiges geheimes Token für jeden Webhook, den Sie erstellen. Das geheime Token ist eine beliebige Zeichenfolge, die Sie angeben und mit der die GitHub Webhook-Payloads berechnet und signiert werden, an die gesendeten Webhook-Payloads CodePipeline, um die Integrität und Authentizität der Webhook-Payloads zu schützen. Die Verwendung Ihrer eigenen Anmeldeinformationen oder die Wiederverwendung desselben Tokens in mehreren Webhooks kann zu Sicherheitslücken führen.

Migrieren Sie Polling-Pipelines zu Webhooks (über die OAuth-AppGitHub) (Quellaktionen) (Konsole)

Für die Quellaktion GitHub (über die OAuth-App) können Sie die CodePipeline Konsole verwenden, um Ihre Pipeline so zu aktualisieren, dass Webhooks verwendet werden, um Änderungen in Ihrem Quell-Repository zu erkennen. GitHub

Folgen Sie diesen Schritten, um eine Pipeline zu bearbeiten, die stattdessen Polling (regelmäßige Überprüfungen) verwendet. EventBridge Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Eine Pipeline, Phasen und Aktionen erstellen.

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-Managementkonsole 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 erfolgt (empfohlen).

   Es wird eine Meldung angezeigt, die darauf hinweist, dass ein Webhook CodePipeline erstellt AWS CodePipeline wird GitHub , um Änderungen an der Quelle zu erkennen: erstellt einen Webhook für Sie. Sie können sich in den folgenden Optionen abmelden. Wählen Sie Aktualisieren aus. Zusätzlich zum Webhook wird Folgendes CodePipeline erstellt:

   - Ein Geheimnis, das zufällig generiert wird und verwendet wird, um die Verbindung zu autorisieren. 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 die verwenden AWS CLI , um eine Änderung an der in der Quellphase der Pipeline angegebenen Quelle festzuschreiben.

Migrieren Sie Polling-Pipelines zu Webhooks GitHub (über die OAuth-App) (Quellaktionen) (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 Eine Pipeline, Phasen und Aktionen erstellen.

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 CloudFormation eine Pipeline erstellen und einen Webhook hinzufügen, müssen Sie regelmäßige Überprüfungen 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 CloudFormation Pipeline, dass die `PollForSourceChanges` Standardeinstellung true ist und in der Ausgabe der Pipeline-Struktur nicht angezeigt wird. Weitere Hinweise zu PollForSourceChanges Standardwerten finden Sie unter. Gültige Einstellungen für den `PollForSourceChanges` Parameter

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 Hinweise zum Taggen von Ressourcen in finden Sie CodePipeline unter. Taggen 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
   ```

Um den Parameter Ihrer Pipeline zu PollForSourceChanges 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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

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.

Aktualisieren Sie die Pipelines für Push-Ereignisse GitHub (über die OAuth-App) (Quellaktionen) (CloudFormation Vorlage)

Folgen Sie diesen Schritten, um Ihre Pipeline (mit einer GitHub Quelle) von regelmäßigen Überprüfungen (Polling) auf ereignisbasierte 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 Ihrer Vorlage eine GitHub Webhook-Ressource hinzu.

Wenn Sie es verwenden CloudFormation , um Ihre Pipelines zu erstellen und zu verwalten, enthält 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 Option verwenden, AWS Secrets Manager um Ihre Anmeldeinformationen zu speichern. 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 CloudFormation Vorlage, um Ihre Anmeldeinformationen zu speichern.

Wenn Sie die CLI verwenden oder CloudFormation eine Pipeline erstellen und einen Webhook hinzufügen, müssen Sie regelmäßige Überprüfungen 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 CloudFormation Pipeline, dass die `PollForSourceChanges` Standardeinstellung true ist und in der Ausgabe der Pipeline-Struktur nicht angezeigt wird. Weitere Hinweise zu PollForSourceChanges Standardwerten finden Sie unter. Gültige Einstellungen für den `PollForSourceChanges` Parameter

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` 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 dem zugeordnete 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-Wert 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 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 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 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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

- 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 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 CloudFormation Beispielvorlage mit der angegebenen Servicerolle und Richtlinie finden Sie in den Tutorials unterTutorial: Erstellen Sie eine Pipeline, die Variablen von verwendet AWS CloudFormation Aktionen zur Bereitstellung.

Die Servicerolle ist keine AWS verwaltete Rolle, sondern wurde ursprünglich für die Pipeline-Erstellung erstellt. Wenn der Servicerollenrichtlinie dann 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 Verwalten Sie die CodePipeline Servicerolle.

Erstellen Sie die CodePipeline Servicerolle (Konsole)

Wenn Sie die Konsole verwenden, um eine Pipeline zu erstellen, erstellen Sie die CodePipeline Servicerolle mit dem Assistenten zur Pipeline-Erstellung.

1. Melden Sie sich bei der an AWS-Managementkonsole 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 um die Erstellung einer neuen Servicerolle in IAM zu ermöglichen.
3. Führen Sie die Pipeline-Erstellung durch. Ihre Pipeline-Servicerolle kann in Ihrer Liste der IAM-Rollen eingesehen werden. 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 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.

> **Note**
>
> Wir empfehlen, dass Sie nur administrativen Benutzern das Erstellen von Servicerollen gestatten. 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 JSON-Datei für die Rollenrichtlinie einfügen. In diesem Beispiel wird VIM verwendet.

   ```
   vim TrustPolicy.json
   ```

2. Fügen Sie den folgenden JSON-Code in die Datei ein.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "codepipeline.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

Um die Datei zu speichern und zu beenden, geben Sie den folgenden VIM-Befehl ein:

```
: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 JSON-Datei für die Rollenrichtlinie einfügen. In diesem Beispiel wird VIM verwendet.

   ```
   vim RolePolicy.json
   ```

4. Fügen Sie die JSON-Richtlinie in die Datei ein. Verwenden Sie die Mindestrichtlinie für Servicerollen, wie unter beschriebenCodePipeline Richtlinie für die Servicerolle. Fügen Sie Ihrer Servicerolle außerdem die entsprechenden Berechtigungen hinzu, die auf den Aktionen basieren, die Sie verwenden möchten. Eine Liste der Aktionen und einen Link zu den für jede Aktion erforderlichen Servicerollenberechtigungen finden Sie unterFügen Sie der CodePipeline Servicerolle Berechtigungen hinzu.

   Stellen Sie sicher, dass Sie den Geltungsbereich der Berechtigungen so weit wie möglich einschränken, indem Sie den Geltungsbereich auf die Ressourcenebene im `Resource` Feld reduzieren.

   Um die Datei zu speichern und zu beenden, geben Sie den folgenden VIM-Befehl ein:

   ```
   :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 zuzuweisen. 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 benannte Rolle ein`MyRole`:

```
aws iam get-role --role-name MyRole
```

8. Verwenden Sie den ARN der Servicerolle, wenn Sie Ihre Pipeline mit der AWS CLI oder erstellen CloudFormation.

Taggen von Ressourcen

Ein Tag ist eine benutzerdefinierte Attributbezeichnung, die Sie einer AWS Ressource AWS zuweisen oder diese 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 Stichwörtern können Sie Ihre AWS Ressourcen identifizieren und organisieren. Viele AWS-Services unterstützen das Tagging, sodass Sie Ressourcen aus verschiedenen Diensten dasselbe Tag zuweisen können, um anzuzeigen, dass die Ressourcen miteinander verwandt sind. Beispielsweise können Sie einer Pipeline dasselbe Tag zuweisen, das Sie einem Amazon S3-Quell-Bucket zuweisen.

Tipps zur Verwendung von Tags finden Sie unter AWS -Tagging-Strategien im AWS -Antworten-Blog.

Sie können in CodePipeline die folgenden Ressourcentypen mit Tags markieren:

- 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.

Zusätzlich zum Identifizieren, Organisieren und Nachverfolgen Ihrer Ressource mithilfe von Tags können Sie Tags in IAM-Richtlinien verwenden, um zu kontrollieren, wer Ihre Ressource anzeigen und mit ihr interagieren kann. Beispiele für Tag-basierte Zugriffsrichtlinien finden Sie unter Verwenden von Tags zur Steuerung des Zugriffs auf Ressourcen CodePipeline.

Markieren Sie eine Pipeline in CodePipeline

Tags sind Schlüssel-Wert-Paare, die Ressourcen zugeordnet AWS sind. Sie können Tags auf Ihre Pipelines in CodePipeline anwenden. Informationen zur Kennzeichnung von CodePipeline Ressourcen, zu Anwendungsfällen, Einschränkungen für Tagschlüssel und -werte sowie zu unterstützten Ressourcentypen finden Sie unter. Taggen 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)
- Markieren von Pipelines (CLI)

Markieren von Pipelines (Konsole)

Sie können Ressourcen über die Konsole oder CLI mit Tags markieren. Pipelines sind die einzigen CodePipeline-Ressourcen, die über die Konsole oder die CLI verwaltet werden können.

Themen

- Hinzufügen von Tags zu einer Pipeline (Konsole)

- Anzeigen von Tags für eine Pipeline (Konsole)
- Bearbeiten von Tags für eine Pipeline (Konsole)
- Entfernen von Tags aus einer Pipeline (Konsole)

Hinzufügen von Tags zu einer Pipeline (Konsole)

Sie können die Konsole verwenden, um Tags zu einer vorhandenen Pipeline hinzuzufügen.

1. Melden Sie sich unter http://console.aws.amazon bei der an AWS-Managementkonsole und öffnen Sie die CodePipeline Konsole. 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 der an AWS-Managementkonsole 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 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 der an AWS-Managementkonsole 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 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 der an AWS-Managementkonsole 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 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 Eine Pipeline, Phasen und Aktionen erstellen.

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 von *Test* und einem Tag-Schlüssel *IscontainerBased* mit dem Tag-Wert von*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 benannt ist *Project* und*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-Managementkonsole 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 für die Verwendung mit Benachrichtigungen konfiguriert haben, wählen Sie unter Zieltyp auswählen entweder Amazon Q Developer in Chat-Anwendungen (Slack) oder SNS-Thema aus. Wählen Sie unter Ziel auswählen den Namen des Clients (für einen in Amazon Q Developer konfigurierten Slack-Client in Chat-Anwendungen) 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 Benachrichtigungsregel keinen Amazon Q Developer in Chat-Anwendungen Client erstellen. Wenn Sie Amazon Q Developer in Chat-Anwendungen (Slack) wählen, wird Ihnen eine Schaltfläche angezeigt, mit der Sie in Chat-Anwendungen einen Client in Amazon Q Developer konfigurieren können. Wenn Sie diese Option wählen, wird die Amazon Q Developer in Chat-Anwendungskonsole geöffnet. Weitere Informationen finden Sie unter Konfigurieren von Integrationen zwischen Benachrichtigungen und Amazon Q Developer in Chat-Anwendungen.
> - 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 und Amazon Q Developer in Chat-Anwendungen einrichten, um Benachrichtigungen an Amazon Chime-Chatrooms oder Slack-Kanäle zu senden. Weitere Informationen finden Sie unter Konfiguration der Integration zwischen Benachrichtigungen und Amazon Q Developer in Chat-Anwendungen.

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 heißt die Datei *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 mit dem Namen *codestar-notifications-MyNotificationTopic* when Pipeline-Ausführungen gesendet:

```
{
    "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"
}
```

Quellaktionen und Methoden zur Erkennung von Änderungen

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 mit CloudTrail Ressourcen	Diese Quellaktion verwendet eine Ereignisregel und zusätzliche CloudTrail Ressourcen. Wenn Sie die CLI verwenden oder CloudFormation diese Aktion erstellen, erstellen und verwalten Sie auch diese Ressourcen.	Siehe Eine Pipeline, Phasen und Aktionen erstellen und Verbindung zu Amazon S3-Quellaktionen herstellen, die EventBridge und verwenden AWS CloudTrail	
Amazon S3 ohne CloudTrail Ressourcen	Diese Quellaktion verwendet einen Bucket, der für Ereignisse mit einer Ereignisregel aktiviert ist, ohne dass zusätzliche CloudTrai	Siehe Eine Pipeline, Phasen und Aktionen erstellen und Herstellen einer Verbindung zu Amazon S3-Quellaktionen	

Quelle	Verwendet zusätzliche Ressourcen?	Schritte	
	l Ressourcen erforderl ich sind. Wenn Sie die CLI verwenden oder CloudFormation diese Aktion erstellen, erstellen und verwalten Sie auch diese Ressourcen.	mit einer für Ereignisse aktivierten Quelle	
Bitbucket Cloud	Diese Quellaktion verwendet eine Verbindun gsressource.	Siehe Bitbucket Cloud-Ver bindungen	
AWS CodeCommit	Amazon EventBridge (empfohlen). Dies ist die Standardeinstellung für eine Pipeline mit einer CodeCommit -Quelle, die in der Konsole erstellt oder bearbeitet wurde.	Siehe Eine Pipeline, Phasen und Aktionen erstellen und CodeCommi t Quellaktionen und EventBridge	
Amazon ECR	Amazon EventBridge. Dies wird vom Assistent en für Pipelines mit einer Amazon ECR-Quelle erstellt, die in der Konsole erstellt oder bearbeitet wurde.	Siehe Eine Pipeline, Phasen und Aktionen erstellen und Amazon ECR-Quellaktionen und Ressourcen EventBridge.	
GitHub oder Enterprise Cloud GitHub	Diese Quellaktion verwendet eine Verbindun gsressource.	Siehe GitHub Verbindun gen	

Quelle	Verwendet zusätzliche Ressourcen?	Schritte	
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 selbstverwaltet	Diese Quellaktion verwendet eine Verbindungsressource und eine Hostressource.	Siehe Verbindungen für GitLab Selbstverwaltung	

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.

Wenn Sie die Änderungserkennung für eine Quellaktion, die Verbindungen verwendet, deaktivieren möchten, finden Sie weitere Informationen unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen.

Stellen Sie mithilfe von Quellaktionen eine Verbindung zu Erstanbietern her

Sie können die AWS CodePipeline Konsole oder die verwenden AWS CLI , um eine Verbindung zu Quellaktionsanbietern wie CodeCommit oder S3 herzustellen.

> **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

- Amazon ECR-Quellaktionen und Ressourcen EventBridge
- Herstellen einer Verbindung zu Amazon S3-Quellaktionen mit einer für Ereignisse aktivierten Quelle
- Verbindung zu Amazon S3-Quellaktionen herstellen, die EventBridge und verwenden AWS CloudTrail
- CodeCommit Quellaktionen und EventBridge

Amazon ECR-Quellaktionen und Ressourcen EventBridge

Um eine Amazon ECR-Quellaktion hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Verwenden Sie den Assistenten zum Erstellen einer Pipeline in der CodePipeline Konsole (Erstellen Sie eine benutzerdefinierte Pipeline (Konsole)) oder auf der Aktionsseite „Bearbeiten“, um die Amazon ECR-Provider-Option 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 das ECR Beispiel einer Aktionskonfiguration unterReferenz zur Amazon ECR-Quellaktion, um Ihre Aktion zu erstellen, wie unter gezeigtErstellen einer Pipeline (CLI).

- Bei der Methode zur Erkennung von Änderungen wird die Pipeline standardmäßig gestartet, indem die Quelle abgefragt wird. Sie sollten regelmäßige Überprüfungen deaktivieren und die Regel zur Erkennung von Änderungen manuell erstellen. Verwenden Sie eine der folgenden Methoden: Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (Konsole)Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (CLI), oderErstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (CloudFormation Vorlage).

Themen

- Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (Konsole)
- Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (CLI)
- Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (CloudFormation Vorlage)

Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (Konsole)

Um eine EventBridge Regel zur Verwendung in CodePipeline Operationen zu erstellen (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 Regel erstellen aus , und wählen Sie dann unter Eventquelle 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 Beispielereignismuster in das Fenster „Ereignisquelle“ für ein `eb-test` Repository mit dem 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-Ereignisse 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 nach 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 account pipeline-name*
> Pipeline-Beispiel-ARN:
> `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

8. Erstellen Sie eine IAM-Servicerolle, 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 zum Starten Ihrer Pipeline-Ausführungen gewährt.
- Wählen Sie Bestehende Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihnen EventBridge Berechtigungen zum Starten Ihrer Pipeline-Ausführungen erteilt.

9. (Optional) Um Quellüberschreibungen mit einer bestimmten Bild-ID anzugeben, verwenden Sie den Eingangstransformator, um die Daten als JSON-Parameter zu übergeben. Sie können den Eingangstransformator auch verwenden, um Pipeline-Variablen zu übergeben.

   - Erweitern Sie Additional settings (Zusätzliche Einstellungen).

     Wählen Sie unter Zieleingang konfigurieren die Option Eingangstransformator konfigurieren aus.

     Wählen Sie im Dialogfenster die Option Eigenen eingeben. Geben Sie in das Feld Eingabepfad die folgenden Schlüssel-Wert-Paare ein.

     ```
     {"revisionValue": "$.detail.image-digest"}
     ```

   - Geben Sie in das Feld Vorlage die folgenden Schlüssel-Wert-Paare ein.

     ```
     {
         "sourceRevisions": [
             {
                 "actionName": "Source",
                 "revisionType": "IMAGE_DIGEST",
                 "revisionValue": "<revisionValue>"
             }
         ],
          "variables": [
             {
                 "name": "Variable_Name",
                 "value": "Variable_Value"
             }
         ]
     }
     ```

   - Wählen Sie Bestätigen aus.

10. Überprüfen Sie die eingerichteten Regeln, um sicherzustellen, dass sie Ihren Anforderungen entsprechen.
11. Wählen Sie Details konfigurieren.
12. 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.
13. Wenn Sie mit der Regel zufrieden sind, klicken Sie auf Create rule (Regel erstellen).

Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (CLI)

Rufen Sie den Befehl put-rule mit folgenden Angaben auf:

- Einen Namen, der die von Ihnen erstellte Regel eindeutig bezeichnet. Dieser Name muss in allen Pipelines, die Sie für Ihr AWS Konto erstellen, eindeutig sein. CodePipeline
- 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 zum Erstellen der Vertrauensrichtlinie, mit der EventBridge die Servicerolle übernehmen kann. Geben Sie der Vertrauensrichtlinie den Namen `trustpolicyforEB.json`.

      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`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333: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 ausgeführt 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-Ereignisse 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 ein:

   - 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
```

4. (Optional) Um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Bild-ID zu konfigurieren, verwenden Sie den folgenden JSON-Code in Ihrem CLI-Befehl. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

   - `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
   - `IMAGE_DIGEST`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
   - In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                    "revisionValue": "$.detail.image-digest"
                },
                "InputTemplate": {
                    "sourceRevisions": [
                        {
                            "actionName": "Source",
                            "revisionType": "IMAGE_DIGEST",
                            "revisionValue": "<revisionValue>"
                        }
                    ]
                }
            }
        }
    ]
}
```

Erstellen Sie eine EventBridge Regel für eine Amazon ECR-Quelle (CloudFormation Vorlage)

Um eine Regel CloudFormation zu erstellen, verwenden Sie das Vorlagen-Snippet wie hier gezeigt.

Um Ihre Pipeline zu aktualisieren CloudFormation Vorlage und EventBridge Regel erstellen

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` 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 von 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` CloudFormation RessourceResources, um eine EventBridge Regel für die Amazon ECR-Quelle hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis, das die Commits in Ihrem Repository überwacht. Wenn eine Statusänderung des Repositorys 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 ausgeführt 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-Ereignisse EventBridge und/oder Amazon Elastic Container Registry Events.

3. (Optional) Verwenden Sie den folgenden YAML-Snippet, um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Image-ID zu konfigurieren. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

   - `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
   - `IMAGE_DIGEST`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
   - In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: ARN
  InputTransformer:
    InputPathsMap:
      revisionValue: "$.detail.image-digest"
    InputTemplate:
      sourceRevisions:
        actionName: Source
        revisionType: IMAGE_DIGEST
        revisionValue: '<revisionValue>'
```

4. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die 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 die Vorlage hoch und zeigen Sie dann die in CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
7. Wählen Sie Ausführen.

Herstellen einer Verbindung zu Amazon S3-Quellaktionen mit einer für Ereignisse aktivierten Quelle

Die Anweisungen in diesem Abschnitt enthalten die Schritte zum Erstellen der S3-Quellaktion, für die Sie keine AWS CloudTrail Ressourcen erstellen oder verwalten müssen.

> **Important**
>
> Das Verfahren zum Erstellen dieser Aktion ohne AWS CloudTrail Ressourcen ist in der Konsole nicht verfügbar. Informationen zur Verwendung der CLI finden Sie hier oder unterMigrieren Sie Polling-Pipelines mit einer S3-Quelle, die für Ereignisse aktiviert ist.

Bei einer Pipeline mit einer Amazon S3-Quelle ändern Sie die Pipeline so, dass die Änderungserkennung durch 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 CloudFormation möchten.

> **Note**
>
> Dazu gehört auch die Verwendung eines Buckets, der für Ereignisbenachrichtigungen aktiviert ist, sodass Sie keinen separaten CloudTrail Trail erstellen müssen. Wenn Sie die Konsole verwenden, werden eine Event-Regel und ein CloudTrail Trail für Sie eingerichtet. Diese Schritte 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 einem CloudTrail S3-Trail (CLI)
- CloudFormation: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail (CloudFormation Vorlage)

Erstellen Sie Pipelines mit einer für Ereignisse aktivierten S3-Quelle (CLI)

Gehen Sie wie folgt vor, um eine Pipeline mit einer S3-Quelle zu erstellen, die ein Ereignis EventBridge zur Änderungserkennung verwendet. Die vollständigen Schritte zum Erstellen einer Pipeline mit der CLI finden Sie unterEine Pipeline, Phasen und Aktionen erstellen.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und erstellen dann die folgenden Ressourcen:

- EventBridge Regel für Ereignisse
- IAM-Rolle, damit das EventBridge Ereignis Ihre Pipeline starten kann

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Erteilen Sie Berechtigungen EventBridge zur Verwendung CodePipeline zum Aufrufen der Regel. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel zum Erstellen der Vertrauensrichtlinie, mit der EventBridge die Servicerolle übernehmen kann. Geben Sie ihr den Namen `trustpolicyforEB.json`.

      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 EventBridge erstellt.

```
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`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333: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\":
[\"amzn-s3-demo-source-bucket\"]}}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-
for-MyRule"
```

3. Um es CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die Parameter und an`--rule`. `--targets`

   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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

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 `amzn-s3-demo-source-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": "amzn-s3-demo-source-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.

Erstellen Sie Pipelines mit einer für Ereignisse aktivierten S3-Quelle (CloudFormation Vorlage)

Dieses Verfahren gilt für eine Pipeline, bei der für den Quell-Bucket Ereignisse aktiviert sind.

Verwenden Sie diese Schritte, um eine Pipeline mit einer Amazon S3-Quelle für die ereignisbasierte Änderungserkennung zu erstellen.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und fügen dann Ihrer Vorlage die folgenden Ressourcen hinzu:

- EventBridge Regel und IAM-Rolle, damit dieses Ereignis Ihre Pipeline starten kann.

Wenn Sie es verwenden CloudFormation , um Ihre Pipelines zu erstellen und zu verwalten, 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 Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` CloudFormation RessourceResources, um die IAM-Rolle zu konfigurieren, die es Ihrem Event ermöglicht, Ihre Pipeline zu starten. 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? Das Hinzufügen einer `AWS::IAM::Role` Ressource ermöglicht CloudFormation das Erstellen von Berechtigungen für EventBridge. Diese Ressource wird Ihrem CloudFormation Stapel 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` CloudFormation Ressource, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erstellt ein Ereignis, das die Erstellung oder Löschung von Objekten in Ihrem Amazon 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 Hinzufügen der `AWS::Events::Rule` Ressource kann CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem 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 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 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 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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

- 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. CloudFormation

> **Note**
>
> Hören Sie hier nicht auf. Obwohl Ihre Pipeline erstellt wurde, müssen Sie eine zweite CloudFormation Vorlage für Ihre Amazon 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

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "appconfig:StartDeployment",
                "appconfig:StopDeployment",
                "appconfig:GetDeployment"
            ],
            "Resource": [
                "arn:aws:appconfig:*:111122223333:application/
[[Application]]",
                "arn:aws:appconfig:*:111122223333:application/[[Application]]/
*",
                "arn:aws:appconfig:*:111122223333:deploymentstrategy/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Verbindung zu Amazon S3-Quellaktionen herstellen, die EventBridge und verwenden AWS CloudTrail

Die Anweisungen in diesem Abschnitt enthalten die Schritte zum Erstellen der S3-Quellaktion, die AWS CloudTrail Ressourcen verwendet, die Sie erstellen und verwalten müssen. Um die S3-Quellaktion zu verwenden, für EventBridge die keine zusätzlichen AWS CloudTrail Ressourcen erforderlich sind, verwenden Sie die CLI-Anweisungen unterMigrieren Sie Polling-Pipelines mit einer S3-Quelle, die für Ereignisse aktiviert ist.

> **Important**
>
> Dieses Verfahren enthält die Schritte zum Erstellen der S3-Quellaktion, die AWS CloudTrail Ressourcen verwendet, die Sie erstellen und verwalten müssen. Das Verfahren zum Erstellen dieser Aktion ohne AWS CloudTrail Ressourcen ist in der Konsole nicht verfügbar. Informationen zur Verwendung der CLI finden Sie unterMigrieren Sie Polling-Pipelines mit einer S3-Quelle, die für Ereignisse aktiviert ist.

Um eine Amazon S3-Quellaktion hinzuzufügen CodePipeline, wählen Sie eine der folgenden Optionen:

- Verwenden Sie den Assistenten zum Erstellen einer Pipeline in der CodePipeline Konsole (Erstellen Sie eine benutzerdefinierte Pipeline (Konsole)) oder auf der Seite „Aktion bearbeiten“, um die S3-Provider-Option 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 das S3 Beispiel einer Aktionskonfiguration unterReferenz zur Amazon S3-Quellaktion, um Ihre Aktion zu erstellen, wie unter gezeigtErstellen einer Pipeline (CLI).
  - Bei der Methode zur Erkennung von Änderungen wird die Pipeline standardmäßig gestartet, indem die Quelle abgefragt wird. Sie sollten die regelmäßigen Überprüfungen deaktivieren und die Regel zur Erkennung von Änderungen und das Protokoll manuell erstellen. Verwenden Sie eine der folgenden Methoden: Erstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (Konsole)Erstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (CLI), oderErstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (CloudFormation Vorlage).

AWS CloudTrail ist ein Service, der Ereignisse in Ihrem Amazon S3-Quell-Bucket protokolliert und filtert. Der Trail sendet die gefilterten Quelländerungen an die EventBridge Regel. Die EventBridge Regel erkennt die Quelländerung und startet dann Ihre Pipeline.

Voraussetzungen:

- Wenn Sie keinen Trail erstellen, verwenden Sie einen vorhandenen AWS CloudTrail Trail, um Ereignisse in Ihrem Amazon 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-Bucket zu übermitteln. 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-Bucket erstellen, AWS CloudTrail hängt die erforderlichen Berechtigungen für Sie an einen Bucket an. Weitere Informationen finden Sie in der Amazon S3-Bucket-Richtlinie für CloudTrail.

Erstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (Konsole)

Bevor Sie eine Regel 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.

So erstellen Sie einen Trail

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 Ressourcenbesitzer (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-Log-Bucket und -Ordner einen Amazon 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 Management-Ereignisse die Option Write aus. Der Trail zeichnet die Amazon S3-API-Aktivität 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-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 Beispielereignistyp 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 Event-Muster ein. Stellen Sie sicher, dass Sie den Bucket-Namen und den S3-Objektschlüssel (oder Schlüsselnamen) hinzufügen, der das Objekt im Bucket eindeutig als identifiziert`requestParameters`. In diesem Beispiel wird eine Regel für einen Bucket mit dem Namen `amzn-s3-demo-source-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": [
                "amzn-s3-demo-source-bucket"
            ],
            "key": [
                "my-files.zip"
            ]
        }
    }
}
```

9. Wählen Sie Weiter aus.
10. Wählen Sie unter Zieltypen die Option AWS Service aus.
11. Wählen Sie unter Ziel auswählen die Option CodePipeline. Geben Sie im Feld Pipeline-ARN den Pipeline-ARN für die Pipeline ein, die nach 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 account pipeline-name*
Pipeline-Beispiel-ARN:
arn: aws: codepipeline: us-east- 2:80398 EXAMPLE:MyFirstPipeline

12. So erstellen oder spezifizieren Sie eine IAM-Servicerolle, 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 zum Starten Ihrer Pipeline-Ausführungen gewährt.
    - Wählen Sie Bestehende Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihnen EventBridge Berechtigungen zum Starten Ihrer Pipeline-Ausführungen erteilt.

13. (Optional) Um Quellüberschreibungen mit einer bestimmten Bild-ID anzugeben, verwenden Sie den Eingangstransformator, um die Daten als JSON-Parameter zu übergeben. Sie können den Eingangstransformator auch verwenden, um Pipeline-Variablen zu übergeben.

    - Erweitern Sie Additional settings (Zusätzliche Einstellungen).

      Wählen Sie unter Zieleingang konfigurieren die Option Eingangstransformator konfigurieren aus.

      Wählen Sie im Dialogfenster die Option Eigenen eingeben. Geben Sie in das Feld Eingabepfad die folgenden Schlüssel-Wert-Paare ein.

      ```
      {"revisionValue": "$.detail.object.version-id"}
      ```

    - Geben Sie in das Feld Vorlage die folgenden Schlüssel-Wert-Paare ein.

      ```
                                  {
          "sourceRevisions": [
              {
                  "actionName": "Source",
                  "revisionType": "S3_OBJECT_VERSION_ID",
                  "revisionValue": "<revisionValue>"
              }
          ],
           "variables": [
      ```

```
        {
            "name": "Variable_Name",
            "value": "Variable_Value"
        }
    ]
}
```

- Wählen Sie Bestätigen aus.

14. Wählen Sie Weiter aus.
15. Wählen Sie auf der Seite „Tags“ die Option Weiter aus.
16. Überprüfen Sie auf der Seite Überprüfen und erstellen die Regelkonfiguration. Wenn Sie mit der Regel zufrieden sind, klicken Sie auf Create rule (Regel erstellen).

Erstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (CLI)

Um eine zu erstellen AWS CloudTrail Protokollierung verfolgen und aktivieren

Um den zum Erstellen eines Trails AWS CLI zu verwenden, 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 Einen Trail mit der AWS Befehlszeilenschnittstelle erstellen.

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 `amzn-s3-demo-source-bucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket
   ```

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? Mit diesem Befehl werden Ereignisse gefiltert.

   Beispiel:

   Im folgenden Beispielbefehl werden `--trail-name` und `--event-selectors` verwendet, um die Verwaltung von Datenereignissen für einen Quell-Bucket und einen Präfix namens `amzn-s3-demo-source-bucket/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:::amzn-s3-
demo-source-bucket/myFolder/file.zip"] }] }]'
```

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Erteilen Sie Berechtigungen EventBridge zur 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, die es ermöglicht, die EventBridge Servicerolle zu übernehmen. Geben Sie ihr den Namen `trustpolicyforEB.json`.

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 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`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
```

```
                "arn:aws:codepipeline:us-
west-2:111122223333: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\":[\"amzn-s3-
demo-source-bucket\"],\"key\":[\"my-key\"]}}}
 --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Um es 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
```

4. (Optional) Verwenden Sie den folgenden JSON-Code in Ihrem CLI-Befehl, um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Bild-ID zu konfigurieren. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

- `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
- `S3_OBJECT_VERSION_ID`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
- In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                    "revisionValue": "$.detail.object.version-id"
                },
                "InputTemplate": {
                    "sourceRevisions": {
                        "actionName": "Source",
                        "revisionType": "S3_OBJECT_VERSION_ID",
                        "revisionValue": "<revisionValue>"
                    }
                }
            }
        }
    ]
}
```

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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

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 `amzn-s3-demo-source-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": "amzn-s3-demo-source-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.

Erstellen Sie eine EventBridge Regel für eine Amazon S3-Quelle (CloudFormation Vorlage)

Um eine Regel CloudFormation zu erstellen, aktualisieren Sie Ihre Vorlage wie hier gezeigt.

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` CloudFormation Ressource`Resources`, um die IAM-Rolle zu konfigurieren, die es Ihrem Event ermöglicht, Ihre Pipeline zu starten. 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? Das Hinzufügen von `AWS::IAM::Role` Ressourcen CloudFormation ermöglicht das Erstellen von Berechtigungen für EventBridge. Diese Ressource wird Ihrem 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` CloudFormation Ressource, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erstellt ein Ereignis`CopyObject`, `PutObject` das Ihren `CompleteMultipartUpload` Amazon 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 Hinzufügen der `AWS::Events::Rule` Ressource kann CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem 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. (Optional) Verwenden Sie den folgenden YAML-Snippet, um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Bild-ID zu konfigurieren. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

   - `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
   - `S3_OBJECT_VERSION_ID`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
   - In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    InputPathsMap:
      revisionValue: "$.detail.object.version-id"
    InputTemplate:
      sourceRevisions:
        actionName: Source
        revisionType: S3_OBJECT_VERSION_ID
        revisionValue: '<revisionValue>'
```

5. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die CloudFormation Konsole.

6. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
7. Laden Sie Ihre aktualisierte Vorlage hoch und zeigen Sie dann die in CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
8. Wählen Sie 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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

- 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-Pipeline zu erstellen

- Verwenden Sie in einer separaten Vorlage unter `Resources` die `AWS::CloudTrail::Trail` CloudFormation RessourcenAWS::S3::Bucket,AWS::S3::BucketPolicy, und, um eine einfache Bucket-Definition und einen Pfad für bereitzustellen CloudTrail.

Warum nehme ich diese Änderung vor? Da das aktuelle Limit von fünf Trails pro Konto gilt, muss der CloudTrail Trail separat erstellt und verwaltet werden. (Siehe Grenzwerte unter AWS CloudTrail.) Sie können jedoch viele Amazon S3-Buckets in einen einzigen Trail aufnehmen, sodass Sie den Trail einmal erstellen und dann bei Bedarf Amazon 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
      }
    }
  }
}

...
```

CodeCommit Quellaktionen und EventBridge

Um eine CodeCommit Quellaktion hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Verwenden Sie den Assistenten zum Erstellen einer Pipeline in der CodePipeline Konsole (Erstellen Sie eine benutzerdefinierte Pipeline (Konsole)) oder auf der Seite „Aktion bearbeiten“, um die CodeCommit Provider-Option auszuwählen. Die Konsole erstellt eine EventBridge Regel, die Ihre Pipeline startet, wenn sich die Quelle ändert.
- Verwenden Sie den AWS CLI , um die Aktionskonfiguration für die `CodeCommit` Aktion hinzuzufügen und zusätzliche Ressourcen wie folgt zu erstellen:
  - Verwenden Sie das `CodeCommit` Beispiel einer Aktionskonfiguration unterCodeCommit Quellaktionsreferenz, um Ihre Aktion zu erstellen, wie unter gezeigtErstellen einer Pipeline (CLI).
  - Bei der Methode zur Erkennung von Änderungen wird die Pipeline standardmäßig gestartet, indem die Quelle abgefragt wird. Sie sollten regelmäßige Überprüfungen 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)Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (CLI), oderErstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (CloudFormation Vorlage).

Themen

- Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (Konsole)
- Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (CLI)
- Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (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 Operationen 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 Beispielereignistyp 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 Fenster Event (Ereignis) für das `MyTestRepo`-Repository MyTestRepo mit einer Verzweigung mit dem Namen `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 nach 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 account pipeline-name*
> Pipeline-Beispiel-ARN:
> `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

11. So erstellen oder spezifizieren Sie eine IAM-Servicerolle, 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 zum Starten Ihrer Pipeline-Ausführungen gewährt.
- Wählen Sie Bestehende Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihnen EventBridge Berechtigungen zum Starten Ihrer Pipeline-Ausführungen erteilt.

12. (Optional) Um Quellüberschreibungen mit einer bestimmten Bild-ID anzugeben, verwenden Sie den Eingangstransformator, um die Daten als JSON-Parameter zu übergeben. Sie können den Eingangstransformator auch verwenden, um Pipeline-Variablen zu übergeben.

    - Erweitern Sie Additional settings (Zusätzliche Einstellungen).

      Wählen Sie unter Zieleingang konfigurieren die Option Eingangstransformator konfigurieren aus.

      Wählen Sie im Dialogfenster die Option Eigenen eingeben. Geben Sie in das Feld Eingabepfad die folgenden Schlüssel-Wert-Paare ein.

      ```
      {"revisionValue": "$.detail.image-digest"}
      ```

    - Geben Sie in das Feld Vorlage die folgenden Schlüssel-Wert-Paare ein.

      ```
      {
          "sourceRevisions": [
              {
                  "actionName": "Source",
                  "revisionType": "COMMIT_ID",
                  "revisionValue": "<revisionValue>"
              }
          ],
          "variables": [
              {
                  "name": "Branch_Name",
                  "value": "value"
              }
          ]
      }
      ```

    - Wählen Sie Bestätigen aus.

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).

Erstellen Sie eine EventBridge Regel für eine CodeCommit Quelle (CLI)

Rufen Sie den Befehl put-rule mit folgenden Angaben auf:

- Einen Namen, der die von Ihnen erstellte Regel eindeutig bezeichnet. Dieser Name muss in allen Pipelines, die Sie für Ihr AWS Konto erstellen CodePipeline , 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 EventBridge Servicerolle zu übernehmen. Geben Sie der Vertrauensrichtlinie den Namen `trustpolicyforEB.json`.

      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`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333: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 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 es CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die folgenden Parameter ein:

   - 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
   ```

4. (Optional) Um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Bild-ID zu konfigurieren, verwenden Sie den folgenden JSON-Code in Ihrem CLI-Befehl. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

   - `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
   - `COMMIT_ID`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
   - In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "pipeline-ARN",
            "InputTransformer": {
                "sourceRevisions": {
                    "actionName": "Source",
                    "revisionType": "COMMIT_ID",
                    "revisionValue": "<revisionValue>"
                },
                "variables": [
                    {
                        "name": "Branch_Name",
                        "value": "value"
                    }
                ]
            }
        }
    ]
}
```

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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

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 (CloudFormation Vorlage)

Um eine Regel CloudFormation zu erstellen, aktualisieren Sie Ihre Vorlage wie hier gezeigt.

Um Ihre Pipeline zu aktualisieren CloudFormation Vorlage und EventBridge Regel erstellen

1. Verwenden Sie in der Vorlage unter die `AWS::IAM::Role` 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? Das Hinzufügen der `AWS::IAM::Role` Ressource ermöglicht CloudFormation das Erstellen von Berechtigungen für EventBridge. Diese Ressource wird Ihrem 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` CloudFormation Ressource`Resources`, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis, das Push-Änderungen an deinem Repository überwacht. Wenn

eine Statusänderung des Repositorys 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 CloudFormation das Ereignis erstellt werden. Diese Ressource wird Ihrem 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. (Optional) Verwenden Sie den folgenden YAML-Snippet, um einen Eingangstransformator mit Quellüberschreibungen für eine bestimmte Bild-ID zu konfigurieren. Im folgenden Beispiel wird eine Überschreibung konfiguriert, bei der:

   - `Source`In diesem Beispiel ist der `actionName` dynamische Wert, der bei der Pipelineerstellung definiert und nicht aus dem Quellereignis abgeleitet wurde.
   - `COMMIT_ID`In diesem Beispiel ist der `revisionType` dynamische Wert, der bei der Pipelineerstellung definiert und nicht vom Quellereignis abgeleitet wurde.
   - In diesem Beispiel wird **<** *`revisionValue`* **>** von der Quellereignisvariablen abgeleitet. `revisionValue`
   - Die Ausgabevariablen für `BranchName` und `Value` sind angegeben.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer, und öffnen Sie dann die 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 die Vorlage hoch und zeigen Sie dann die in CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
7. Wählen Sie 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 Gültige Einstellungen für den `PollForSourceChanges` Parameter.

- 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
    }
  ]
},
```

Fügen Sie externe Quellanbieter zu Pipelines hinzu mit AWS CodeConnections

Sie können die AWS CodePipeline Konsole oder die verwenden, AWS CLI um Ihre Pipeline mit Repositorys von Drittanbietern zu verbinden.

> **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

- DevOps Azure-Verbindungen
- Bitbucket Cloud-Verbindungen
- GitHub Verbindungen
- GitHub Enterprise Server-Verbindungen
- GitLab.com Verbindungen
- Verbindungen für GitLab Selbstverwaltung
- Verwenden Sie eine Verbindung, die mit einem anderen Konto geteilt wird

DevOps Azure-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**
>
> Anstatt eine bestehende Verbindung in Ihrem Konto zu erstellen oder zu verwenden, können Sie eine gemeinsame Verbindung zwischen anderen verwenden AWS-Konto. Siehe Verwenden Sie eine Verbindung, die mit einem anderen Konto geteilt wird.

> **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 und GitLab.com GitLab selbstverwaltete Aktionen.

Um eine DevOps Azure-Quellaktion hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Verwenden Sie die CodePipeline Konsolenseite „Pipeline erstellen“ oder „Aktion bearbeiten“, um die DevOpsAzure-Anbieteroption auszuwählen. Informationen Stelle eine Verbindung zu Bitbucket Cloud (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 `Azure DevOps` Anbieter wie folgt hinzuzufügen:
  - Informationen zum Erstellen Ihrer Verbindungsressourcen finden Sie unter Verbindung zu Azure DevOps (CLI) herstellen So erstellen Sie eine Verbindungsressource mit der CLI.
  - Verwenden Sie die `CreateSourceConnection` Beispiel-Aktionskonfiguration inCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com 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 ein Konto beim Anbieter des Drittanbieter-Repositorys, z. B. Azure, erstellt haben DevOps
- Sie müssen bereits ein Code-Repository eines Drittanbieters erstellt haben, z. B. ein DevOps Azure-Repository.

> **Note**
>
> DevOps Azure-Verbindungen bieten nur Zugriff auf Repositorys, die dem DevOps Azure-Konto gehören, mit dem die Verbindung hergestellt wurde.
> Um die Verbindung zu installieren, benötigen Sie Administratorberechtigungen für das Azure-Konto.

Themen

- Stellen Sie eine Verbindung zu Azure DevOps (Konsole) her
- Verbindung zu Azure DevOps (CLI) herstellen

Stellen Sie eine Verbindung zu Azure DevOps (Konsole) her

Gehen Sie wie folgt vor, um mithilfe der CodePipeline Konsole eine Verbindungsaktion für Ihr DevOps Azure-Repository hinzuzufügen.

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 zu vervollständigen, und wählen Sie Weiter. Wählen Sie auf der Quellseite unter Quellanbieter die Option Azure aus DevOps.
   - 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 Azure aus DevOps.

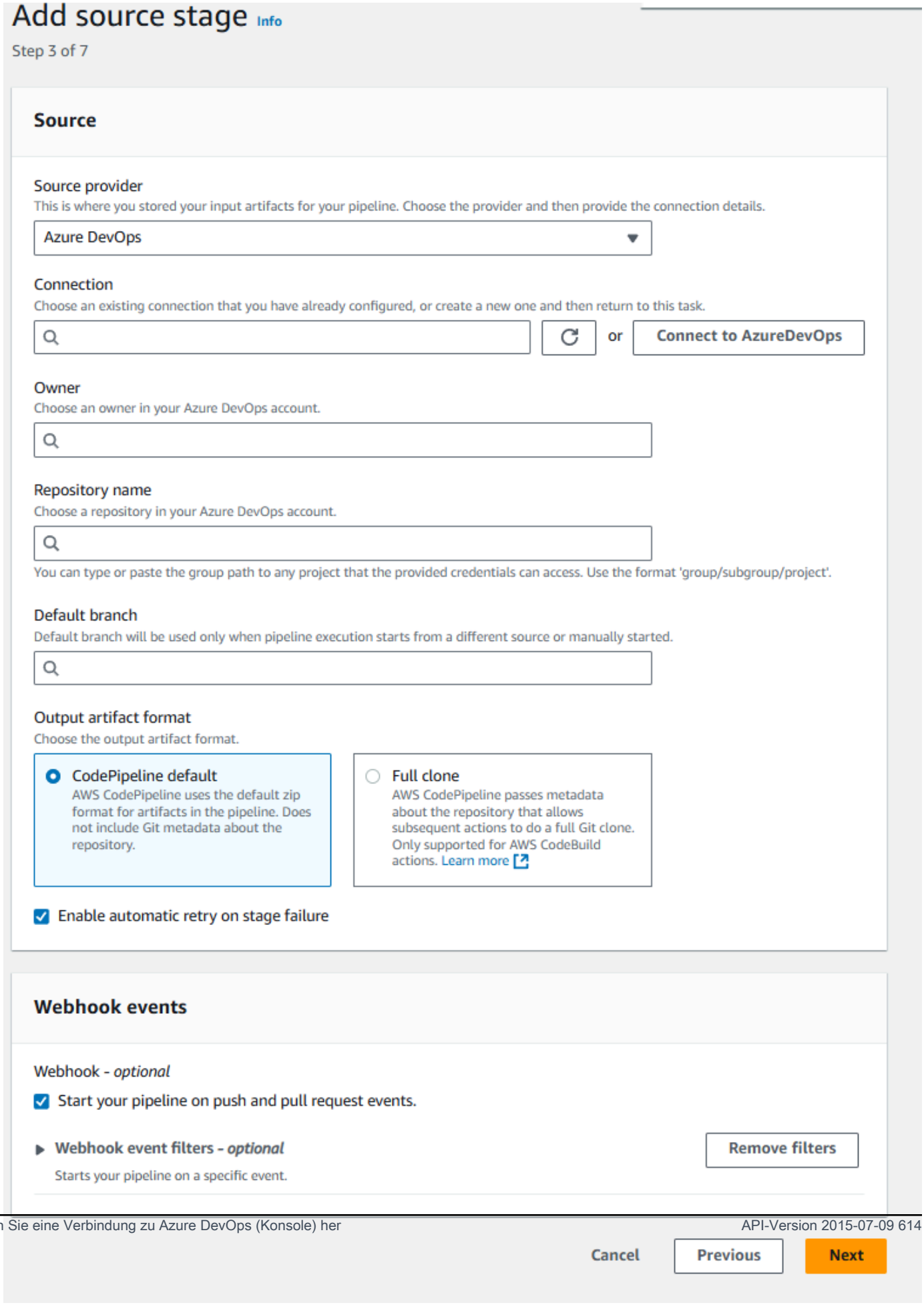
3. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie unter Verbindung noch keine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie Connect to Azure aus DevOps. Fahren Sie mit Schritt 2 fort: Verbindung zu Azure herstellen DevOps.
   - 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 zu Azure her DevOps

Um eine Verbindung zu Azure herzustellen DevOps

1. Um eine Verbindung zu einem DevOps Azure-Repository herzustellen, wählen Sie unter Anbieter auswählen die Option Azure aus DevOps. Geben Sie unter Connection name (Verbindungsname) den Namen für die Verbindung ein, die Sie erstellen möchten. Wählen Sie Connect to Azure DevOps und fahren Sie mit Schritt 2 fort.


Add source stage Info
Step 3 of 7
Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
Azure DevOps
Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.
or
Connect to AzureDevOps
Owner
Choose an owner in your Azure DevOps account.
Repository name
Choose a repository in your Azure DevOps account.
You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.
Default branch
Default branch will be used only when pipeline execution starts from a different source or manually started.
Output artifact format
Choose the output artifact format.
CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.
Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions. Learn more
Enable automatic retry on stage failure
Webhook events
Webhook - optional
Start your pipeline on push and pull request events.
Webhook event filters - optional
Starts your pipeline on a specific event.
Remove filters
Cancel
Previous
Next

2. Auf der DevOps Einstellungsseite Connect to Azure wird Ihr Verbindungsname angezeigt.

   Wählen Sie unter DevOps Azure-Apps eine App-Installation aus oder wählen Sie Neue App installieren, um eine zu erstellen.

   > **Note**
   >
   > Sie installieren die App nur einmal für jedes DevOps Azure-Konto. Wenn Sie die Connector-App bereits installiert haben, wählen Sie sie aus und fahren Sie mit dem letzten Schritt in diesem Abschnitt fort.

3. Wenn die Anmeldeseite für Microsoft angezeigt wird, melden Sie sich mit Ihren Anmeldeinformationen an und klicken Sie dann auf Fortfahren.
4. Auf der App-Installationsseite wird eine Meldung angezeigt, dass die Connector-App versucht, eine Verbindung zu Ihrem DevOps Azure-Konto herzustellen.

   Wählen Sie Grant access (Zugriff gewähren).
5. Auf der Verbindungsseite wird die Verbindungs-ID für Ihre neue Installation angezeigt. Wählen Sie Connect aus. Die erstellte Verbindung wird in der Verbindungsliste angezeigt.

Schritt 3: Speichern Sie Ihre DevOps Azure-Quellaktion

Verwenden Sie diese Schritte auf dem Assistenten oder auf der Seite 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 AWS CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen.
3. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

- Um Ausgabeartefakte aus der DevOps Azure-Aktion mit der Standardmethode zu speichern, wählen Sie CodePipeline Standard. 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.

  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, GitHub Enterprise Server oder hinzu GitHub GitLab.com.

4. Wählen Sie im Assistenten Weiter oder auf der Aktionsseite Bearbeiten die Option Speichern aus.

Verbindung zu Azure DevOps (CLI) herstellen

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 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 `AzureDevOps` und der angegebene Verbindungsname `MyConnection`.

```
aws codeconnections create-connection --provider-type AzureDevOps --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. 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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen. 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"
                            ]
                        }
                    }
                ]
            }
        }
    ]
```

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**
>
> Anstatt eine bestehende Verbindung in Ihrem Konto zu erstellen oder zu verwenden, können Sie eine gemeinsame Verbindung zwischen anderen verwenden AWS-Konto. Siehe Verwenden Sie eine Verbindung, die mit einem anderen Konto geteilt wird.

> **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 und GitLab.com 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 und GitLab.com 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:

- 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
- Verbindung zu Bitbucket Cloud (CLI) herstellen

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

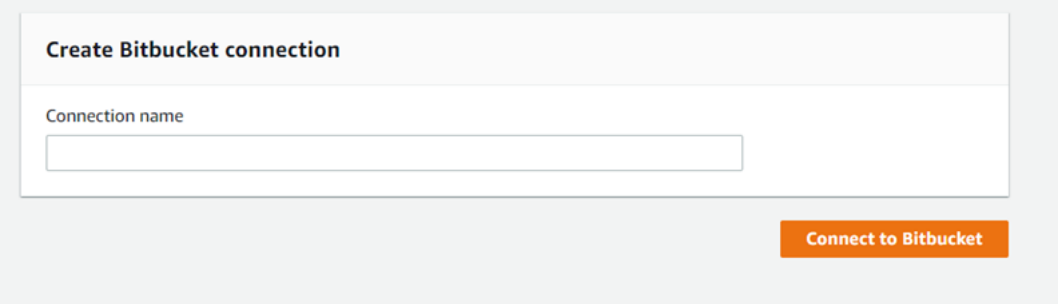
1. Melden Sie sich bei der CodePipeline Konsole an.
2. Wählen Sie eine der folgenden Optionen aus.
   - Wählen Sie, ob Sie eine Pipeline erstellen möchten. Folgen Sie den Schritten unter Pipeline erstellen, um den ersten Bildschirm zu vervollständigen, und wählen Sie Weiter. Wähle auf der Seite „Quelle“ unter „Quellanbieter“ die Option Bitbucket aus.
   - Wähle, ob du eine bestehende Pipeline bearbeiten möchtest. 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. 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

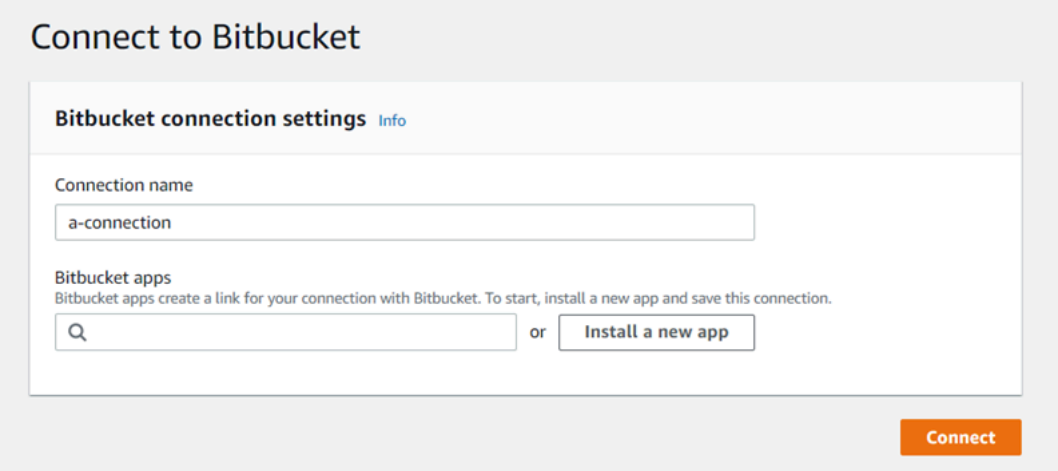
1. Gib auf der Einstellungsseite „Mit Bitbucket Connect“ deinen Verbindungsnamen ein und wähle Mit Bitbucket verbinden aus.



   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.

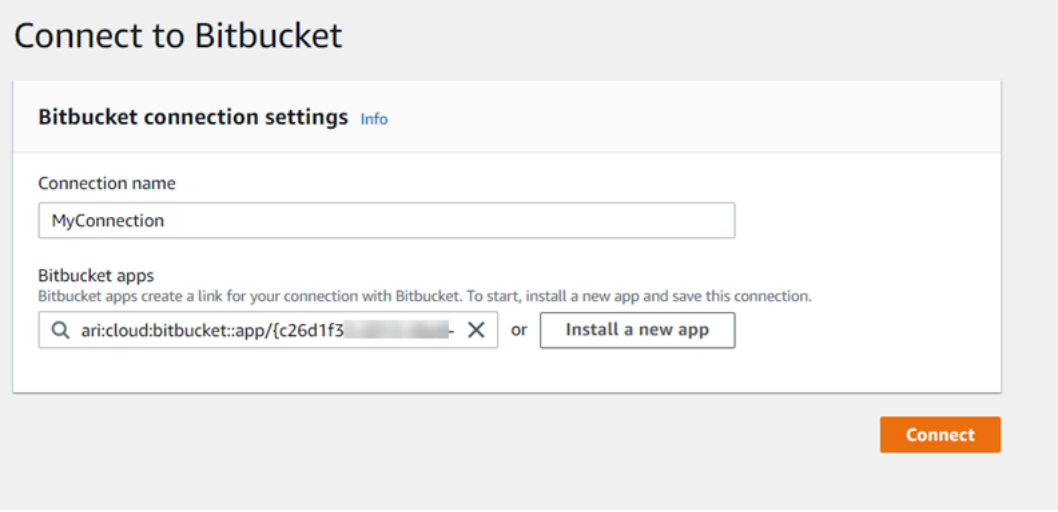
> **Note**
>
> Du installierst die App nur einmal für jeden Bitbucket Cloud-Workspace oder Account. Wenn du die Bitbucket-App bereits installiert hast, wähle sie aus und fahre mit Schritt 4 fort.



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 AWS CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen.
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, GitHub Enterprise Server oder hinzu GitHub GitLab.com.
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 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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen. 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"
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
     ```

GitHub Verbindungen

Sie verwenden Verbindungen, um Konfigurationen zu autorisieren und einzurichten, die Ihren Drittanbieter mit Ihren AWS Ressourcen verknüpfen.

> **Note**
>
> Anstatt eine bestehende Verbindung in Ihrem Konto zu erstellen oder zu verwenden, können Sie eine gemeinsame Verbindung zwischen anderen AWS-Konto verwenden. Siehe Verwenden Sie eine Verbindung, die mit einem anderen Konto geteilt wird.

> **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 und GitLab.com 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 in der CodePipeline Konsole den Assistenten „Pipeline erstellen“ oder die Aktionsseite „Aktion bearbeiten“, um die Anbieteroption GitHub (über GitHub App) auszuwählen. Informationen Stellen Sie eine Verbindung zu GitHub (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 zum Klonen von Metadaten 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. Siehe Verbindung herstellen.

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. Detaillierte Anweisungen finden Sie unter Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu.

> **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
- Verbindung herstellen zu GitHub (CLI)

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.

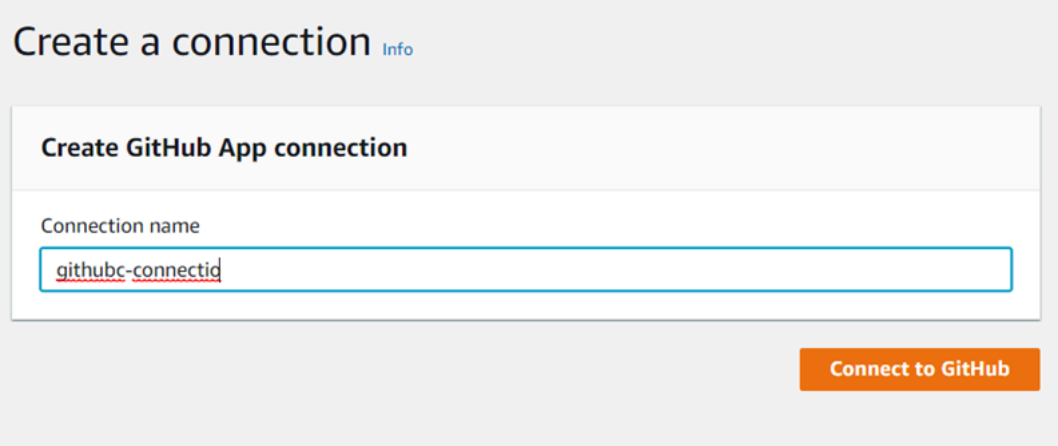
> **Note**
>
> In diesen Schritten können Sie unter Repository-Zugriff bestimmte Repositorys auswählen. Alle Repositorys, die nicht ausgewählt sind, sind für nicht zugänglich oder sichtbar. CodePipeline

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 abzuschließen, und wählen Sie Weiter. Wählen Sie auf der Quellseite unter Quellanbieter die Option GitHub (via GitHub App) 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 (über GitHub App) 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 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.

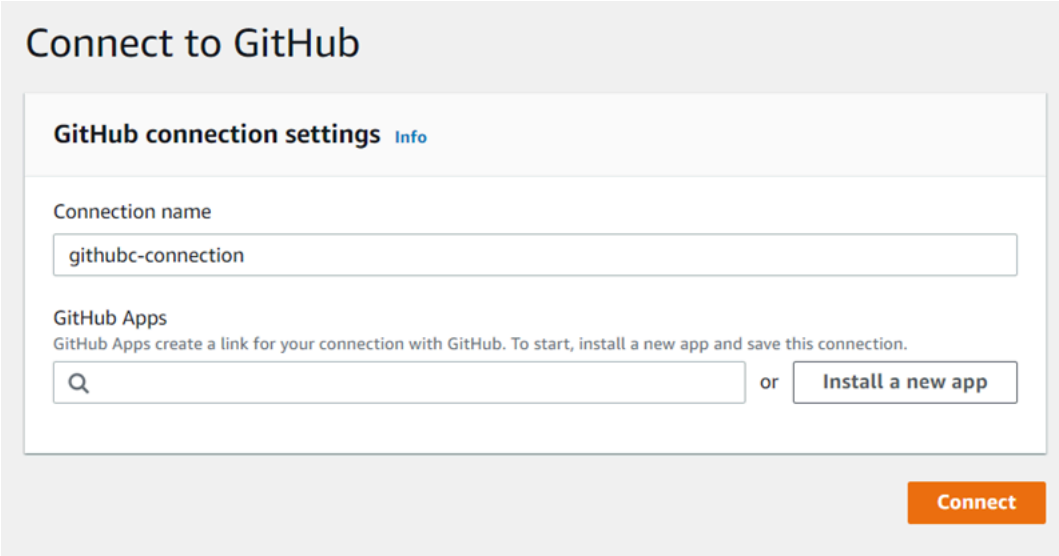
Schritt 2: Stellen Sie eine Verbindung her zu GitHub

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.

   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.

> **Note**
>
> Wenn Sie ein Benutzerzugriffstoken erstellen möchten, stellen Sie sicher, dass Sie den AWS Connector für die GitHub App bereits installiert haben, und lassen Sie dann das Feld App-Installation leer. AWS CodeConnections verwendet das Benutzerzugriffstoken für die Verbindung.

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 AWS CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen.

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, GitHub Enterprise Server oder hinzu GitHub GitLab.com. 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.

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 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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen. 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**
>
> Anstatt eine bestehende Verbindung in Ihrem Konto zu erstellen oder zu verwenden, können Sie eine gemeinsame Verbindung zwischen anderen verwenden AWS-Konto. Siehe Verwenden Sie eine Verbindung, die mit einem anderen Konto geteilt wird.

> **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 und GitLab.com 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 inCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com 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:

- 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
- Erstellen Sie einen Host und eine Verbindung zu GitHub Enterprise Server (CLI)

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.

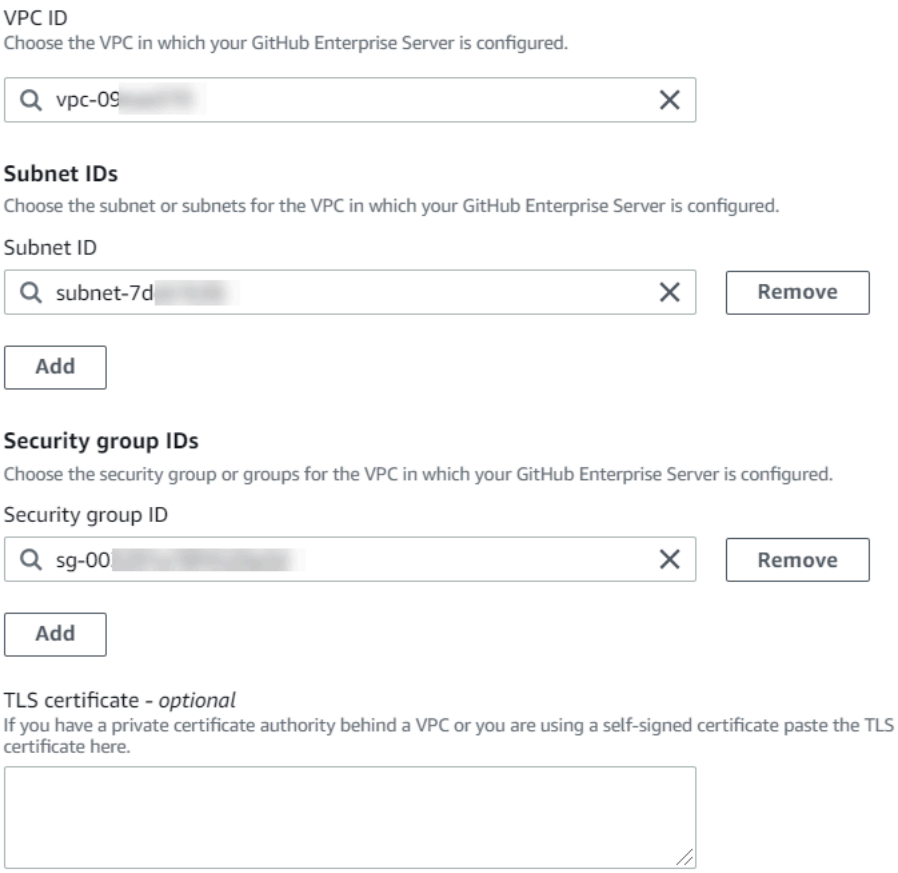
b. Wählen Sie unter Subnetz-ID (Subnetz-ID) die Option Add (Hinzufügen) aus. Wählen Sie im Feld die Subnetz-ID aus, die Sie für Ihren Host verwenden möchten. Sie können bis zu 10 Subnetze wählen.

   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 AWS CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen.
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.

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 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`.

So erstellen Sie einen Host

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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen. 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**
>
> Anstatt eine bestehende Verbindung in Ihrem Konto zu erstellen oder zu verwenden, können Sie eine gemeinsame Verbindung zwischen anderen verwenden AWS-Konto. Siehe Verwenden Sie eine Verbindung, die mit einem anderen Konto geteilt wird.

> **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 und GitLab.com GitLab selbstverwaltete Aktionen.

Um eine GitLab.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 herstellen zu GitLab.com (CLI) So erstellen Sie eine Verbindungsressource mit der CLI.

- Verwenden Sie die `CreateSourceConnection` Beispiel-Aktionskonfiguration inCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com 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 in autorisieren GitLab.com, 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 erstellt haben. GitLab.com

  > **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
- Verbindung herstellen zu GitLab.com (CLI)

Stelle eine Verbindung zu GitLab.com (Konsole) her

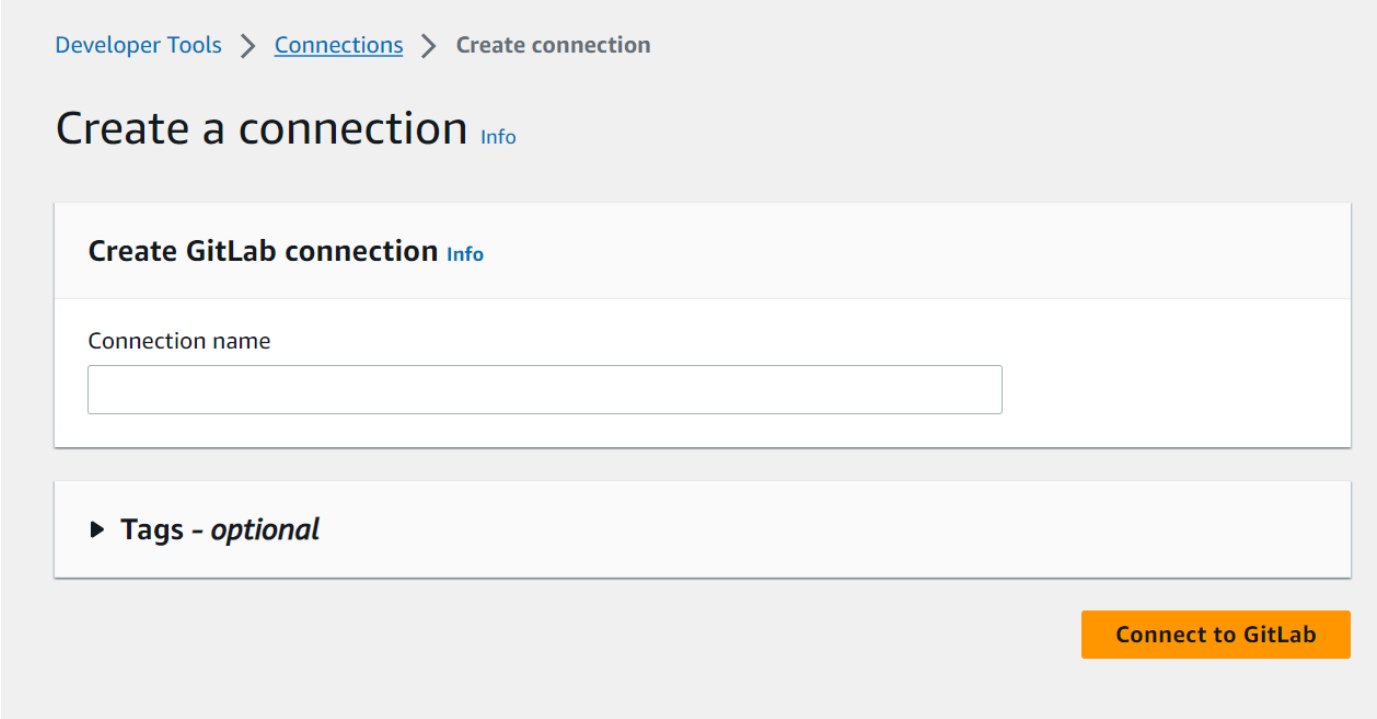
Gehen Sie wie folgt vor, um mithilfe der CodePipeline Konsole eine Verbindungsaktion für Ihr Projekt (Repository) in hinzuzufügen GitLab.

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.

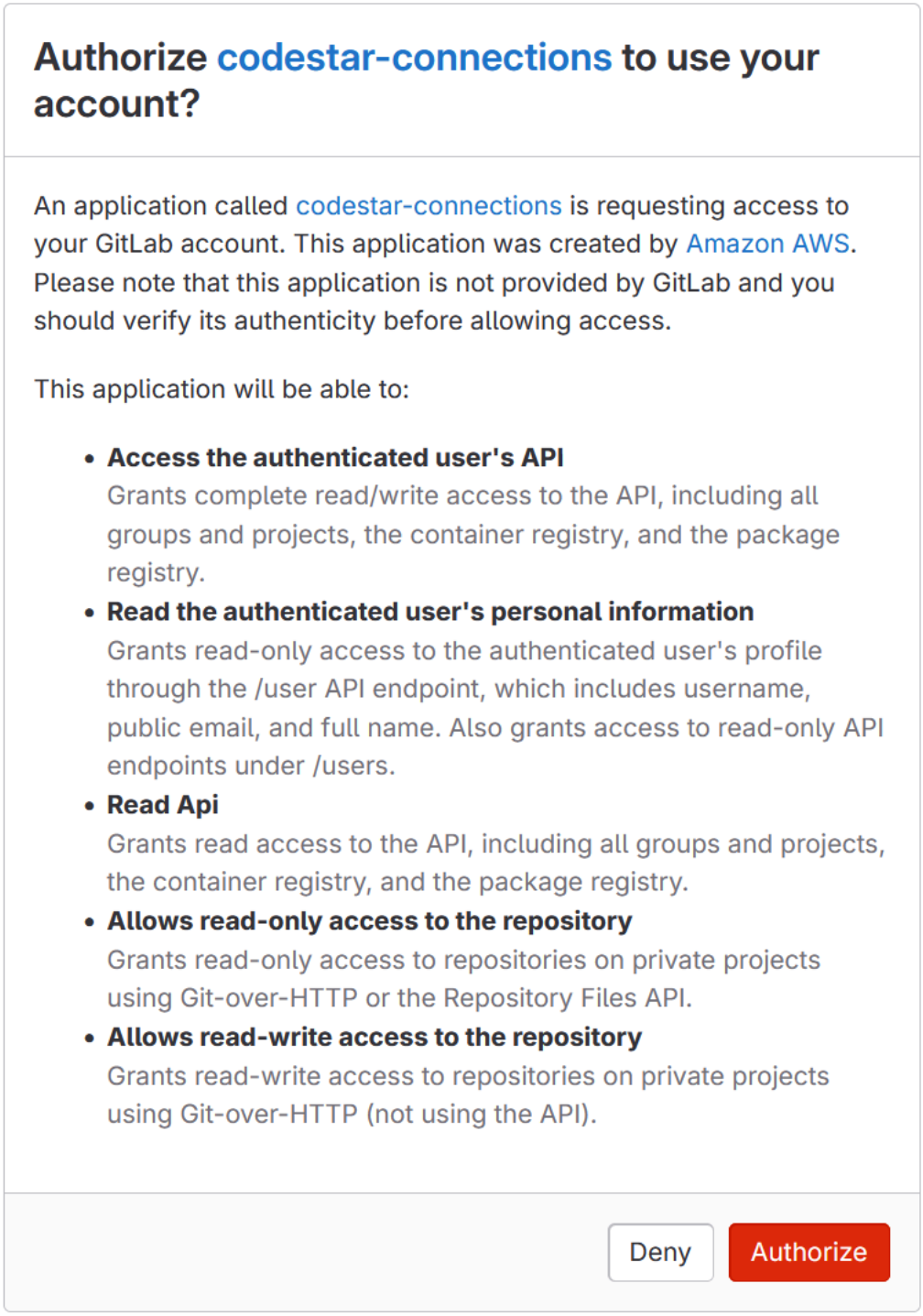
> **Note**
>
> Wenn Sie das Popup-Fenster schließen, bevor eine GitLab.com Verbindung hergestellt wurde, müssen Sie die Seite aktualisieren.

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 in https://docs.gitlab.com/ee/api/projects.html #get -single-project. Weitere Hinweise 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 AWS CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen.
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 die project/repository.

12. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

- Um die Ausgabeartefakte der GitLab.com Aktion 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 nachgeschalteten Aktionen verwendet werden.

  Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschrieben. Fügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub Enterprise Server oder hinzu GitHub GitLab.com 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 herstellen zu GitLab.com (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 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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen. 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 Selbstverwaltung

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**
>
> Anstatt eine bestehende Verbindung in Ihrem Konto zu erstellen oder zu verwenden, können Sie eine gemeinsame Verbindung zwischen anderen verwenden AWS-Konto. Siehe Verwenden Sie eine Verbindung, die mit einem anderen Konto geteilt wird.

> **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 und GitLab.com 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 Selbstverwaltung (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 und GitLab.com 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:

- 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 finden 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 kann die Verbindung mit Ressourcen wie CodePipeline verwendet werden. Bei Repositorys in Gruppen müssen Sie nicht der Gruppenbesitzer sein.

- Sie müssen bereits ein GitLab Personal Access Token (PAT) mit dem `api` Gültigkeitsbereich erstellt haben. Wenn der Admin-Modus auf Ihrer GitLab Instanz aktiviert ist, muss die PAT auch den `admin_mode` Bereich enthalten. Weitere Informationen finden Sie auf der GitLab Website unter Persönliche Zugriffstoken und Gültigkeitsbereiche für Zugriffstoken. 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 Selbstverwaltung (Konsole) her
- Erstellen Sie einen Host und eine Verbindung zu GitLab Self-Managed (CLI)

Stellen Sie eine Verbindung zur GitLab Selbstverwaltung (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 Self-managed herstellen.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie die Verbindung aus, und fahren Sie dann mit Schritt 3: Speichern Ihrer GitLab selbstverwalteten Quellaktion fort.

Schritt 2: Stellen Sie eine Verbindung zu GitLab Selfmanaged her

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die Seite Connect GitLab selbstverwalteter Verbindung herstellen angezeigt.

Um eine Verbindung zur selbstverwalteten Website herzustellen GitLab

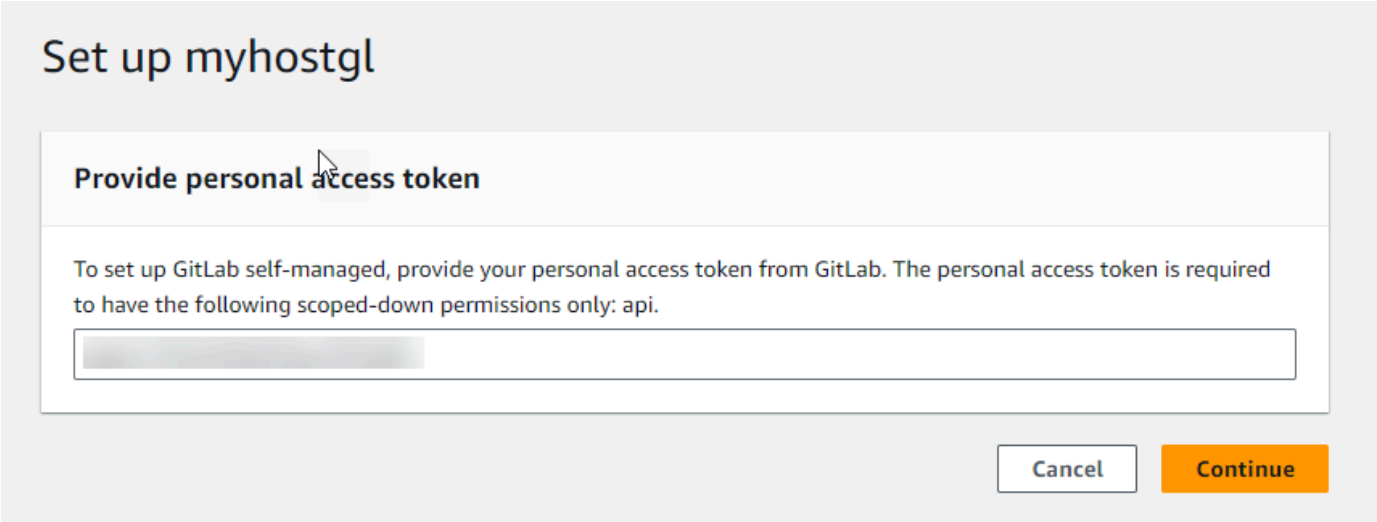
1. Geben Sie unter Connection Name (Verbindungsname) den Namen für die Verbindung ein.
2. Geben Sie unter URL den Endpunkt für Ihren Server ein.

   > **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 ***`host_name`***Einrichtungsseite wird angezeigt. Geben Sie im Feld Persönliches Zugriffstoken bereitstellen den `api` Geltungsbereich in Ihrem GitLab PAT an. Wenn der Admin-Modus auf Ihrer GitLab Instanz aktiviert ist, muss die PAT auch den `admin_mode` Bereich enthalten. Weitere Informationen finden Sie auf der GitLab Website unter Bereiche für Zugriffstoken.

   > **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 AWS CodeConnections Aktion handelt. Weitere Informationen zur Konfiguration der Pipeline-Trigger und zum optionalen Filtern mit Triggern finden Sie unterTrigger mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen.
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.

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 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`.

So erstellen Sie einen Host

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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen. 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"
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
     ```

Verwenden Sie eine Verbindung, die mit einem anderen Konto geteilt wird

Sie können eine gemeinsame Verbindung erstellen und verwalten mit AWS RAM. Auf diese Weise können Verbindungen AWS-Konten gemeinsam genutzt werden, um auf Repositorys von Drittanbietern zuzugreifen. Auf diese Weise kann eine einzige Verbindung in CodePipeline Pipelines zwischen Konten verwendet werden, während Benutzer weniger separate Verbindungen in jedem Konto verwalten und verwalten müssen.

Gehen Sie wie folgt vor CodePipeline, um gemeinsam genutzte Verbindungen in zu verwenden.

- Stellen Sie mithilfe der Developer Tools-Konsole unter Einstellungen eine Verbindung her. Weitere Informationen finden Sie unter Verbindung herstellen.
- Richten Sie die Ressourcenfreigabe ein mit AWS RAM. Siehe Verbindungen teilen mit AWS-Konten.
- Wenn du den CodePipeline Konsolenassistenten zum Erstellen von Pipelines oder die Aktionsseite „Aktion bearbeiten“ verwendest, um den Verbindungsanbieter auszuwählen, z. B. die Bitbucket-Provider-Option, kannst du die Verbindung auswählen, die mit dem Zielkonto geteilt wurde.

Automatisieren Sie das Starten von Pipelines mithilfe von Triggern und Filtern

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. Weitere Informationen zu Quellaktionen, die Verbindungen verwenden, findest du unterFügen Sie externe Quellanbieter zu Pipelines hinzu mit AWS CodeConnections.

Quellaktionen wie CodeCommit und S3 verwenden die automatische Änderungserkennung, um Pipelines zu starten, wenn eine Änderung vorgenommen wird. Weitere Informationen finden Sie unter CodeCommit Quellaktionen und EventBridge.

Sie geben Trigger über die Konsole oder CLI an.

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.
  - Push
    - Gültige Filterkombinationen sind:
      - Git-Tags

        Einschließen oder Ausschließen
      - Filialen

        Einschließen oder Ausschließen
      - Zweige + Dateipfade

Einschließen oder Ausschließen

- Pull-Anfrage
  - Gültige Filterkombinationen sind:
    - Zweige

      Einschließen oder Ausschließen
    - Zweige + Dateipfade

      Einschließen oder Ausschließen
- 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
Einen Auslöser hinzufügen — 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).

Felddefinitionen und weitere Hinweise zu Triggern findest du unter

Eine Liste der Felddefinitionen in der JSON-Struktur finden Sie unter`triggers`.

Für die Filterung werden Muster regulärer Ausdrücke im Glob-Format unterstützt, wie unter beschriebenArbeiten 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.

Überlegungen zu Triggerfiltern

Die folgenden Überlegungen gelten für die Verwendung von Triggern.

- Sie können nicht mehr als einen Trigger pro Quellaktion hinzufügen.
- Sie können einem Trigger mehrere Filtertypen hinzufügen. Ein Beispiel finden Sie unter 4: Ein Trigger mit zwei Push-Filtertypen mit widersprüchlichen Ein- und Ausschlüssen.
- 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 einer Pull-Anfrage kann in Fällen, in denen sich die Triggerkonfigurationen Push (Branches Filter) und Pull Request (Branches Filter) überschneiden, zwei Pipeline-Ausführungen auslösen.

- Bei einem Filter, der Ihre Pipeline bei Pull-Request-Ereignissen auslöst, hat der Drittanbieter-Repository-Anbieter für Ihre Verbindung beim Pull-Request-Ereignistyp „Geschlossen“ möglicherweise einen separaten Status für ein Merge-Ereignis. In Bitbucket ist das Git-Ereignis für eine Zusammenführung beispielsweise kein Pull-Request-Schließungsereignis. In GitHub ist das Zusammenführen einer Pull-Anfrage jedoch ein Abschlussereignis. Weitere Informationen finden Sie unter Pull-Request-Ereignisse für Trigger nach Anbieter.
- Wenn mehrere Quellaktionen in einer Pipeline über eine Verbindung auf verschiedene Zweige desselben Repositorys verweisen, löst nur ein Zweig zuverlässig die Pipeline aus. Das Webhook-Abonnement der Verbindung ist für die Kombination aus Pipeline und Repository registriert, nicht pro Zweig. Um das Problem zu umgehen, verwenden Sie für jeden Zweig eine separate Pipeline.

Pull-Request-Ereignisse für Trigger nach Anbieter

Die folgende Tabelle enthält eine Zusammenfassung der Git-Ereignisse, z. B. für das Schließen von Pull-Requests, die zu Pull-Request-Ereignistypen nach Anbietern führen.

	Repository-Anbieter für Ihre Verbindung			
PR-Ereignis für Trigger	Bitbucket	GitHub	JA	GitLab
Öffnen — Diese Option löst die Pipeline aus, wenn eine Pull-Anfrage für den Branch-/Dateipfad erstellt wird.	Das Erstellen einer Pull-Anfrage führt zu einem Opened Git-Ereignis.	Das Erstellen einer Pull-Anfrage führt zu einem Opened Git-Ereignis.	Das Erstellen einer Pull-Anfrage führt zu einem Opened Git-Ereignis.	Das Erstellen einer Pull-Anfrage führt zu einem Opened Git-Ereignis.
Update — Diese Option löst die Pipeline aus, wenn eine Pull-Request-Revision für den Branch-/	Die Veröffentlichung eines Updates führt zu einem aktualisierten Git-Ereignis.	Die Veröffentlichung eines Updates führt zu einem aktualisierten Git-Ereignis.	Die Veröffentlichung eines Updates führt zu einem aktualisierten Git-Ereignis.	Die Veröffentlichung eines Updates führt zu einem aktualisierten Git-Ereignis.

	Repository-Anbieter für Ihre Verbindung			
PR-Ereignis für Trigger	Bitbucket	GitHub	JA	GitLab
Dateipfad veröffentlicht wird.				
Geschlossen — Diese Option löst die Pipeline aus, wenn eine Pull-Anfrage für den branch/file Pfad geschloss en wird.	Das Zusammenf ühren einer Pull-Anfrage in Bitbucket führt zu einem Closed-Git-Event. Wichtig: Das manuelle Schließen einer Pull-Anfrage in Bitbucket ohne Zusammenf ührung führt nicht zu einem Closed Git-Ereig nis.	Das Zusammenf ühren oder manuelles Schließen einer Pull-Anfrage führt zu einem Closed-Git-Ereignis.	Das Zusammenf ühren oder manuelles Schließen einer Pull-Anfrage führt zu einem Closed-Git-Ereignis.	Das Zusammenf ühren oder manuelles Schließen einer Pull-Anfrage führt zu einem Closed-Git-Ereignis.

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. Ein Trigger kann mehrere Filtertypen enthalten, z. B. zwei Push-Filtertypen in der Trigger-Konfiguration, und die Push- und Pull-Request-Filtertypen verwenden eine OR-Operation zwischen ihnen, was bedeutet, dass bei jeder Übereinstimmung die Pipeline gestartet wird. Ebenso kann jeder Filtertyp mehrere Filter wie FilePaths und Branches enthalten. Diese Filter verwenden eine AND-Operation, was bedeutet, dass nur eine vollständige Übereinstimmung die Pipeline startet. Jeder Filtertyp kann Einschlüsse und Ausschlüsse enthalten, wobei Ausschlüsse Vorrang vor Einschlüssen haben. Wenn ein Zweig- oder Dateipfad

einem Ausschlussmuster entspricht, wird die Pipeline nicht ausgelöst, auch wenn er auch einem Include-Muster entspricht. Wenn ein Commit mehrere Dateien ändert, wird jede Datei unabhängig anhand des Filters bewertet. Wenn eine geänderte Datei erfolgreich ist (entspricht einem Include-Wert und entspricht nicht einem Exclude-Wert), wird die Pipeline gestartet. Namen innerhalb des Einschließen/Ausschließens, wie z. B. Zweignamen, verwenden eine OR-Operation. Die folgende Liste fasst die Operationen für jeden Teil des Git-Konfigurationsobjekts zusammen.

Eine Liste der Felddefinitionen in der JSON-Struktur und eine ausführliche Referenz zu Ein- und Ausschlüssen finden Sie unter. `triggers`

Example 1: Ein Filtertyp mit Filtern für Verzweigungen und Dateipfade (UND-Operation)

Für einen einzelnen Filtertyp, z. B. eine Pull-Anfrage, können Sie Filter kombinieren. Diese Filter verwenden eine AND-Operation, was bedeutet, dass nur eine vollständige Übereinstimmung die Pipeline startet. Das folgende Beispiel zeigt eine Git-Konfiguration für einen Push-Ereignistyp mit zwei verschiedenen Filtern (`filePathsundbranches`). Im folgenden Beispiel `filePaths` wird AND-verknüpft mit: `branches`

```
{
  "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. Mit anderen Worten, der Dateipfad `common/app.js` ist für den Filter enthalten, der die Pipeline als AND startet, auch wenn der `refs/heads/feature/triggers` angegebene Zweig keine Auswirkung hatte.

```
{
  "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "common/app.js"
      ]
```

```
      ...
    }
  ]
}
```

Das folgende Beispiel zeigt ein Ereignis für einen Trigger mit der obigen Konfiguration, der die Pipeline-Ausführung nicht startet, da der Zweig filtern kann, der Dateipfad jedoch nicht.

```
{
   "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "src/Main.java"
      ]
      ...
    }
  ]
}
```

Example 2: Ausschlüsse haben Vorrang vor Einschlüssen

Innerhalb eines einzelnen Filters haben Ausschlüsse Vorrang vor Einschlüssen. Das folgende Beispiel zeigt eine Git-Konfiguration mit einem einzigen Filter (`branches`) innerhalb des Konfigurationsobjekts. Das heißt, wenn ein Branch einem Ausschlussmuster entspricht (`feature-branch`im Beispiel), wird die Pipeline nicht ausgelöst, auch wenn sie auch einem Include-Muster entspricht. Wenn das Include-Muster übereinstimmt und kein Ausschlussmuster übereinstimmt, z. B. für den `main` Branch, dann wird die Pipeline ausgelöst.

Für das folgende JSON-Beispiel:

- Wenn ein Commit an den `main` Branch gesendet wird, wird die Pipeline ausgelöst
- Wenn ein Commit an den `feature-branch` Branch weitergeleitet wird, wird die Pipeline nicht ausgelöst.

```
{
  "branches": {
      "Includes": [
```

```
            "main"
        ],
        "Excludes": [
            "feature-branch"
        ]
    }
```

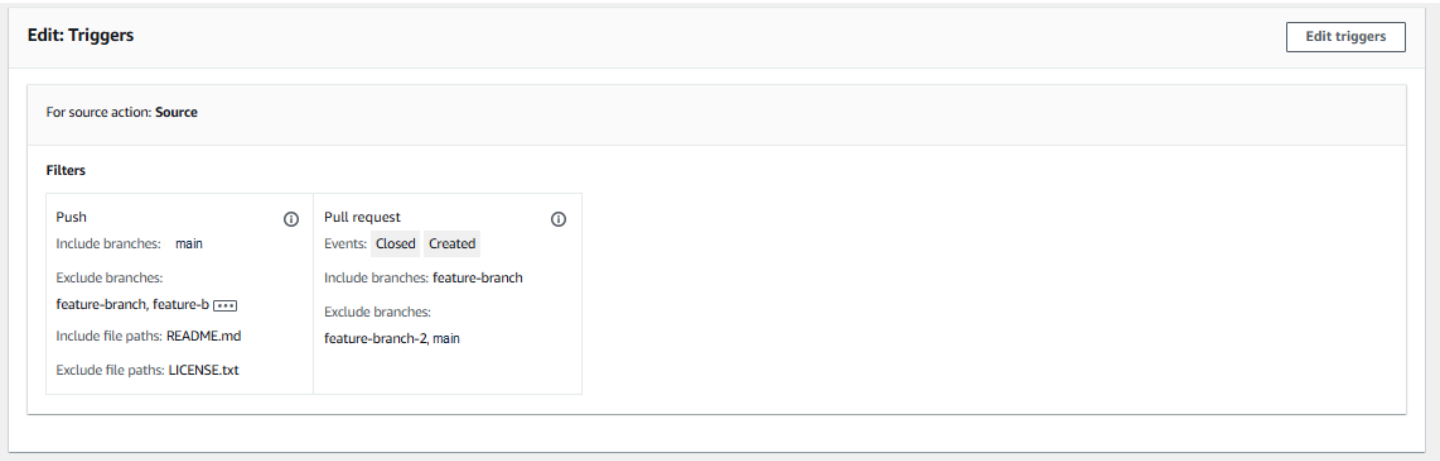
Example 3: Ein Trigger mit Push- und Pull-Request-Filtertypen (OR-Operation), Filtern für Dateipfade und Branches (AND-Operation) und includes/excludes (Ausschlüsse haben Vorrang)

Trigger-Konfigurationsobjekte, wie z. B. ein Trigger, der einen Push-Ereignistyp und einen Pull-Request-Ereignistyp enthält, verwenden eine OR-Operation zwischen den beiden Ereignistypen. Das folgende Beispiel zeigt eine Trigger-Konfiguration mit einem Push-Ereignistyp, bei dem der `main` Branch eingeschlossen ist, und einem Pull-Request-Ereignistyp, bei dem derselbe Branch `main` ausgeschlossen ist. Darüber hinaus ist beim Push-Ereignistyp ein Dateipfad `LICENSE.txt` ausgeschlossen und ein Dateipfad `README.MD` enthalten. Beim zweiten Ereignistyp startet eine Pull-Anfrage, die sich entweder `Created` auf `Closed` oder auf dem `feature-branch` Branch (eingeschlossen) befindet, die Pipeline, und die Pipeline startet nicht, wenn eine Pull-Anfrage in den `main` Zweigen `feature-branch-2` oder geschlossen wird (ausgeschlossen). Innerhalb jedes Ereignistyps haben Ausschlüsse Vorrang vor Einschlüssen. Beispiel: Bei einem Pull-Request-Ereignis im `feature-branch` Branch (das in der Pull-Anfrage enthalten ist) schließt der Push-Ereignistyp den `feature-branch` Branch aus, sodass der Push die Pipeline nicht auslöst.

Für das folgende Beispiel

- Wenn Sie einen Commit für den `README.MD` Dateipfad (im Lieferumfang enthalten) an den `main` Branch (im Lieferumfang enthalten) weiterleiten, wird die Pipeline ausgelöst.
- Beim `feature-branch` Branch (ausgeschlossen) wird durch das Pushen eines Commits die Pipeline nicht ausgelöst.
- Im eingeschlossenen Branch löst die Bearbeitung des `README.MD` Dateipfads (im Lieferumfang enthalten) die Pipeline aus.
- Im eingeschlossenen Zweig wird die Pipeline nicht ausgelöst, wenn nur der `LICENSE.TXT` Dateipfad (ausgeschlossen) bearbeitet wird. Wenn sich derselbe Commit jedoch ebenfalls ändert `README.MD` (eingeschlossen), wird die Pipeline ausgelöst, da jede Datei unabhängig ausgewertet wird.
- Auf dem `feature-branch` Branch löst das Schließen einer Pull-Anfrage die Pipeline aus, da sie für den Pull-Request-Ereignistyp enthalten `feature-branch` ist und der Ereignistyp CLOSED übereinstimmt.

Die folgende Abbildung zeigt die Konfiguration.



Im Folgenden finden Sie ein JSON-Beispiel für die Konfiguration.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "branches": {
                                "includes": [
                                    "main"
                                ],
                                "excludes": [
                                    "feature-branch",
                                    "feature-branch-2"
                                ]
                            },
                            "filePaths": {
                                "includes": [
                                    "README.md"
                                ],
                                "excludes": [
                                    "LICENSE.txt"
                                ]
                            }
                        }
                    ],
                    "pullRequest": [
                        {
```

```
                        "events": [
                            "CLOSED",
                            "OPEN"
                        ],
                        "branches": {
                            "includes": [
                                "feature-branch"
                            ],
                            "excludes": [
                                "feature-branch-2",
                                "main"
                            ]
                        }
                    }
                ]
            }
        }
    ]
},
```

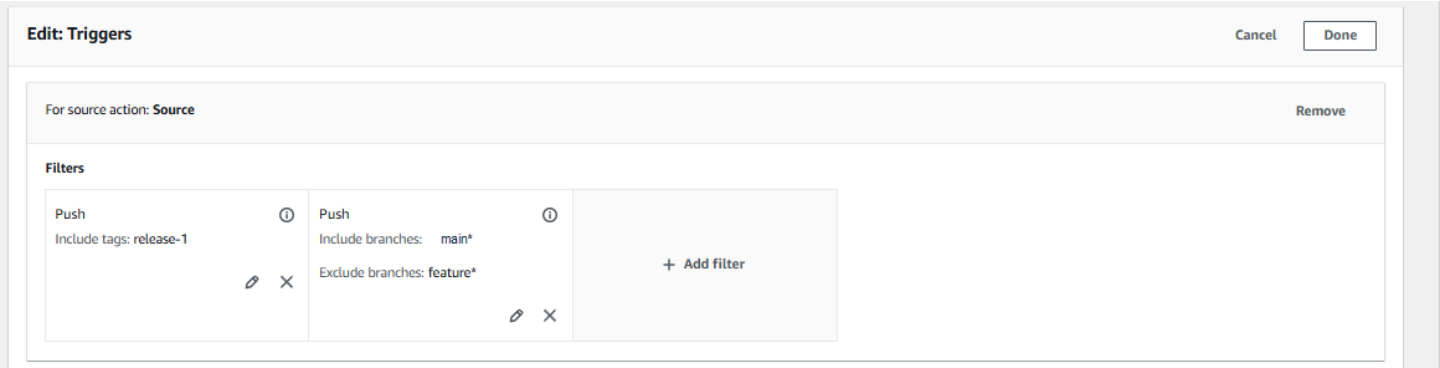
Example 4: Ein Trigger mit zwei Push-Filtertypen mit widersprüchlichen Ein- und Ausschlüssen

Die folgende Abbildung zeigt einen Push-Filtertyp, der angibt, dass nach dem Tag gefiltert werden soll `release-1` (im Lieferumfang enthalten). Ein zweiter Push-Filtertyp wird hinzugefügt, der angibt, dass nach dem Zweig gefiltert werden soll `main` (im Lieferumfang enthalten) und bei einem Push zu den `feature*` Branches (ausgeschlossen) nicht gestartet werden soll.

Bei folgendem Beispiel:

- Wenn ein Release vom Tag `release-1` (für den ersten Push-Filter enthalten) auf den `feature-branch` Branch übertragen wird (ausgenommen wie `feature*` beim zweiten Push-Filter), wird die Pipeline ausgelöst, da die beiden Push-Filtertypen eine ODER-Operation zwischen ihnen verwenden und der erste Push-Filter (Tag`release-1`) übereinstimmt.
- Durch das Pushen eines Releases aus dem `main` Branch (im zweiten Push-Filter enthalten) wird die Pipeline gestartet.

Das folgende Beispiel für die Bearbeitungsseite zeigt die beiden Push-Filtertypen und ihre Konfiguration für Ein- und Ausschlüsse.



Im Folgenden finden Sie ein JSON-Beispiel für die Konfiguration.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-1"
                                ]
                            }
                        },
                        {
                            "branches": {
                                "includes": [
                                    "main*"
                                ],
                                "excludes": [
                                    "feature*"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    },
```

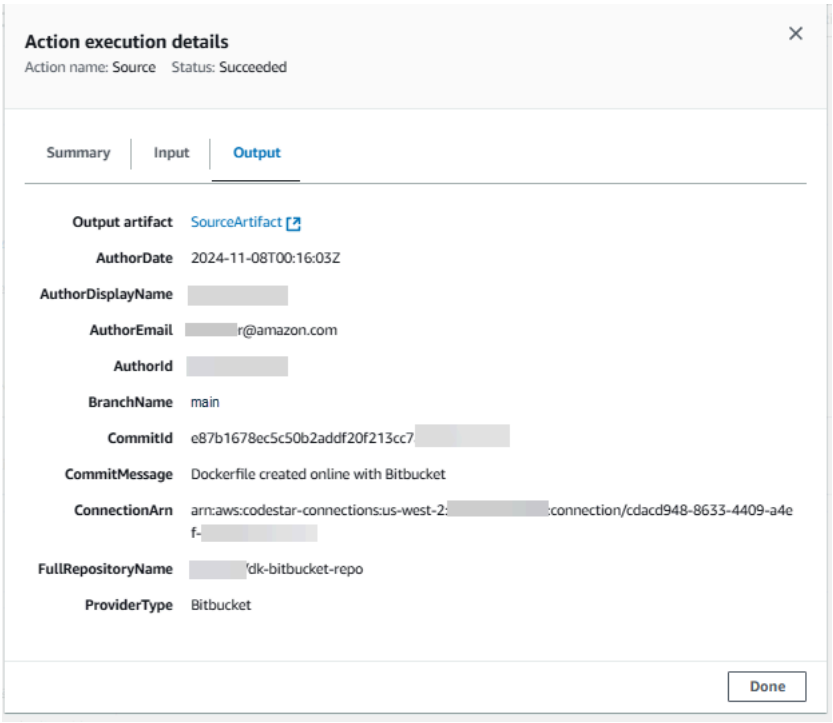
Example 5: Der Trigger wurde konfiguriert, während die Standardaktionskonfiguration für einen manuellen Start verwendet BranchName wird

Das `BranchName` Standardfeld für die Aktionskonfiguration definiert einen einzelnen Zweig, der verwendet wird, wenn die Pipeline manuell gestartet wird, wohingegen Trigger mit Filtern für alle von Ihnen angegebenen Verzweigungen verwendet werden können.

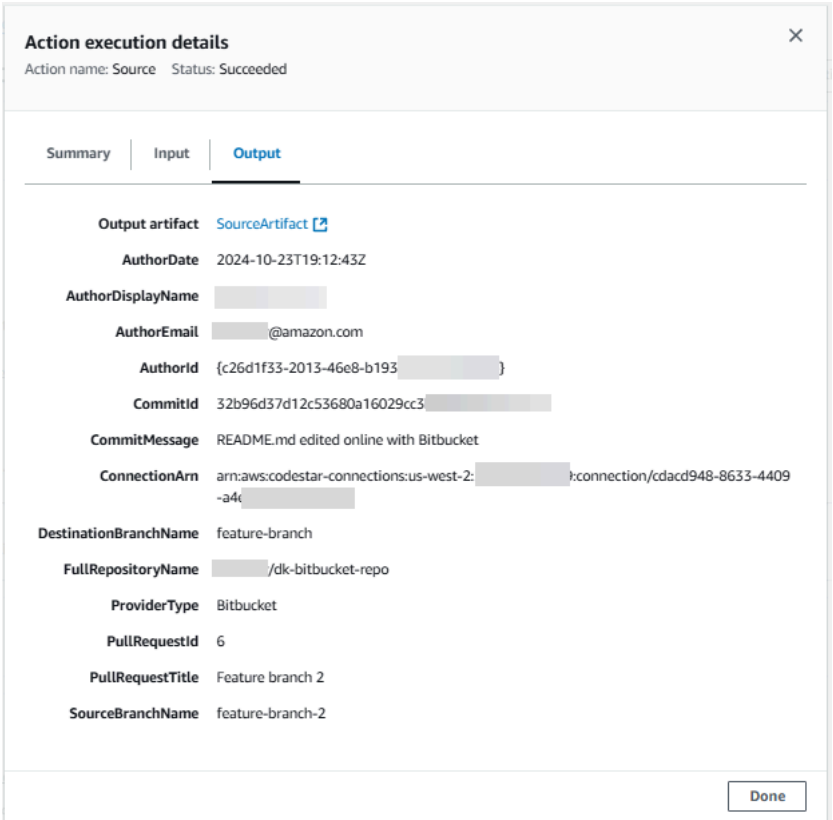
Im Folgenden finden Sie das JSON-Beispiel für die Aktionskonfiguration, in der das `BranchName` Feld angezeigt wird.

```
{
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "provider": "CodeStarSourceConnection",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "BranchName": "main",
                            "ConnectionArn": "ARN",
                            "DetectChanges": "false",
                            "FullRepositoryId": "owner-name/my-bitbucket-repo",
                            "OutputArtifactFormat": "CODE_ZIP"
                        },
                        "outputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "inputArtifacts": [],
                        "region": "us-west-2",
                        "namespace": "SourceVariables"
                    }
                ],
```

Das folgende Beispiel für eine Aktionsausgabe zeigt, dass der Standardzweig main verwendet wurde, als die Pipeline manuell gestartet wurde.



Die folgende Beispiel-Aktionsausgabe zeigt die Pull-Anfrage und den Branch, die für den Trigger verwendet wurden, wenn sie nach einer Pull-Anfrage gefiltert wurden.



Trigger beim Code-Push ohne Filter hinzufügen

Mit Triggern können Sie Ihre Pipeline so konfigurieren, dass sie bei einem bestimmten Ereignistyp startet, z. B. bei einem Code-Push oder einer Pull-Anfrage. Trigger können für Quellaktionen mit Verbindungen konfiguriert werden, die die CodeStarSourceConnection Aktion in verwenden CodePipeline GitHub, z. B. Bitbucket und GitLab.

Einen Trigger hinzufügen (Konsole)

1. Melden Sie sich bei der an AWS-Managementkonsole 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, ob Sie einen Trigger hinzufügen möchten.
5. Wählen Sie unter Triggertyp die Option Kein Filter aus.

Trigger mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen

Du kannst Filter für Pipeline-Trigger so konfigurieren, dass Pipeline-Ausführungen für verschiedene Git-Ereignisse wie Tag- oder Branch-Push, Änderungen in bestimmten Dateipfaden, ein in einem bestimmten Branch geöffneter Pull-Request usw. gestartet werden. Du kannst die AWS CodePipeline Konsole oder die update-pipeline Befehle create-pipeline und in der verwenden, um Triggerfilter AWS CLI zu konfigurieren.

> **Note**
>
> Das `BranchName` Feld für die Aktionskonfiguration definiert einen einzelnen Zweig, während Trigger mit Filtern für jeden Zweig oder jede von Ihnen angegebene Verzweigung verwendet werden können. Bei einer Pipeline, bei der Trigger verwendet werden, um Branches nach Push- oder Pull-Requests zu filtern, verwendet die Pipeline nicht das `BranchName` Standardfeld Branch in der Aktionskonfiguration. Die Verzweigung im `BranchName` Feld

in der Aktionskonfiguration ist jedoch die Standardeinstellung, wenn die Pipeline manuell gestartet wird. Ein Beispiel finden Sie unter 5: Der Trigger wurde konfiguriert, während die Standardaktionskonfiguration für einen manuellen Start verwendet BranchName wird.

Sie können Filter für die folgenden Triggertypen angeben:

- Push

  Ein Push-Trigger startet eine Pipeline, wenn eine Änderung an dein Quell-Repository übertragen wird. Bei der Ausführung wird das Commit aus dem Branch verwendet, zu dem Sie den Push senden (das ist der Ziel-Branch). Du kannst Push-Trigger nach Branches, Dateipfaden oder Git-Tags filtern.
- Pull-Anfrage

  Ein Pull-Request-Trigger startet eine Pipeline, wenn ein Pull Request in deinem Quell-Repository geöffnet, aktualisiert oder geschlossen wird. Bei der Ausführung wird der Commit aus dem Quell-Branch verwendet, aus dem du die Daten abrufst (also aus dem Quell-Branch). Du kannst Pull-Request-Trigger für Branches und Dateipfade filtern.

  Die folgenden Ereignistypen für Pull Requests werden unterstützt. Alle anderen Pull-Request-Ereignisse werden ignoriert.
  - Opened (Geöffnet)
  - Aktualisiert
  - Geschlossen (zusammengeführt)

> **Note**
>
> Das Verhalten bestimmter Pull-Request-Ereignisse kann je nach Anbieter unterschiedlich sein. Details hierzu finden Sie unter Pull-Request-Ereignisse für Trigger nach Anbieter.

Die Pipeline-Definition ermöglicht es Ihnen, verschiedene Filter innerhalb derselben Push-Trigger-Konfiguration zu kombinieren. Einzelheiten zur Pipeline-Definition finden Sie unterFügen Sie Filter für Push- und Pull-Request-Ereignistypen (CLI) hinzu. Eine Liste der Felddefinitionen finden Sie unter Trigger in der Pipeline-Strukturreferenz in diesem Handbuch.

Themen

- Fügen Sie Filter für Push- und Pull-Request-Ereignistypen hinzu (Konsole)
- Fügen Sie Filter für Push- und Pull-Request-Ereignistypen (CLI) hinzu
- Fügen Sie Filter für Push- und Pull-Request-Ereignistypen hinzu (CloudFormation Vorlagen)

Fügen Sie Filter für Push- und Pull-Request-Ereignistypen hinzu (Konsole)

Sie können die Konsole verwenden, um Filter für Push-Ereignisse hinzuzufügen und Branches oder Dateipfade ein- oder auszuschließen.

Filter hinzufügen (Konsole)

1. Melden Sie sich bei der an AWS-Managementkonsole 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 zur Pipeline-Erstellung.
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.

   - Wähle Push, um die Pipeline zu starten, wenn eine Änderung an dein Quell-Repository übertragen wird. Wenn Sie diese Option wählen, können die Felder Filter für Zweige und Dateipfade oder Git-Tags angeben.
   - Wähle Pull Request, um die Pipeline zu starten, wenn ein Pull Request in deinem 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 Push unter Filtertyp eine der folgenden Optionen aus.

   - Wählen Sie Branch, um die Branches in Ihrem Quell-Repository anzugeben, die der Trigger überwacht, um zu wissen, wann eine Workflow-Ausführung gestartet werden muss. Geben Sie im Feld 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 Triggerkonfiguration sie ignoriert

und Ihre Pipeline nicht bei Änderungen in den angegebenen Verzweigungen startet. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

> **Note**
>
> Wenn sowohl Include als auch Exclude dasselbe Muster haben, wird das Muster standardmäßig ausgeschlossen.

Sie können Glob-Muster verwenden, um Ihre Filialnamen zu definieren. Verwenden Sie zum Beispiel, `main*` um alle Zweige abzugleichen, die mit `main` beginnen. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

Geben Sie für einen Push-Trigger die Zweige an, zu denen Sie den Push ausführen, d. h. die Ziel-Branches. Geben Sie für einen Pull-Request-Trigger die Ziel-Branches an, für die Sie einen Pull Request öffnen.

- (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 Glob-Muster verwenden, um Ihre Dateipfadnamen zu definieren. Verwenden Sie zum Beispiel, `prod*` um alle Dateipfade abzugleichen, die mit `prod` beginnen. 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 im Feld 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 Tag-Namen im Glob-Format ein, die Sie angeben möchten, damit die Triggerkonfiguration sie ignoriert und Ihre Pipeline nicht bei der Veröffentlichung des oder der angegebenen Tags startet. Wenn sowohl Include als auch Exclude dasselbe Tag-Muster haben, wird das Tag-Muster standardmäßig ausgeschlossen.

7. Wählen Sie unter Push unter Filtertyp eine der folgenden Optionen aus.

   - Wählen Sie Branch, um die Branches in Ihrem Quell-Repository anzugeben, die der Trigger überwacht, um zu wissen, wann eine Workflow-Ausführung gestartet werden muss. Geben Sie im Feld 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 Triggerkonfiguration sie ignoriert und Ihre Pipeline nicht bei Änderungen in den angegebenen Verzweigungen startet. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

> **Note**
>
> Wenn sowohl Include als auch Exclude dasselbe Muster haben, wird das Muster standardmäßig ausgeschlossen.

Sie können Glob-Muster verwenden, um Ihre Filialnamen zu definieren. Verwenden Sie zum Beispiel, `main*` um alle Zweige abzugleichen, die mit `main` beginnen. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

Geben Sie für einen Push-Trigger die Zweige an, zu denen Sie den Push ausführen, d. h. die Ziel-Branches. Geben Sie für einen Pull-Request-Trigger die Ziel-Branches an, für die Sie einen Pull Request öffnen.

- (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 Glob-Muster verwenden, um Ihre Dateipfadnamen zu definieren. Verwenden Sie zum Beispiel, `prod*` um alle Dateipfade abzugleichen, die mit `prod` beginnen. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.
- Wählen Sie Pull Request, um die Pipeline-Trigger-Konfiguration so zu konfigurieren, dass sie mit von Ihnen angegebenen Pull-Request-Ereignissen beginnt.

Fügen Sie Filter für Push- und Pull-Request-Ereignistypen (CLI) hinzu

Sie können das Pipeline-JSON aktualisieren, um Filter für Trigger hinzuzufügen.

Verwenden Sie den `update-pipeline` Befehl or, AWS CLI um die Pipeline zu erstellen `create-pipeline` oder zu aktualisieren.

Das folgende Beispiel für eine JSON-Struktur bietet eine Referenz für die Felddefinitionen unter`create-pipeline`.

Eine Liste der Felddefinitionen finden Sie unter Trigger in der Pipeline-Strukturreferenz in diesem Handbuch.

```
{
    "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"
                    }
                }
            ]
        }
    ]
```

```
    }
}
```

Fügen Sie Filter für Push- und Pull-Request-Ereignistypen hinzu (CloudFormation Vorlagen)

Sie können die Pipeline-Ressource aktualisieren CloudFormation , um eine Triggerfilterung hinzuzufügen.

Das folgende Beispiel-Vorlagen-Snippet enthält eine YAML-Referenz für Trigger-Felddefinitionen. Eine Liste der Felddefinitionen finden Sie unter Trigger in der Pipeline-Strukturreferenz in diesem Handbuch.

Ein vollständiges Vorlagenbeispiel für eine Verbindungsquell- und Triggerfilterkonfiguration finden Sie im CloudFormation Benutzerhandbuch unter Pipeline mit zwei Stufen und Triggerkonfiguration.

```
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
```

Trigger hinzufügen, um die Änderungserkennung auszuschalten

Mit Triggern können Sie Ihre Pipeline so konfigurieren, dass sie bei einem bestimmten Ereignistyp beginnt, z. B. bei einem Code-Push- oder Pull-Request. Trigger sind für Quellaktionen mit Verbindungen konfigurierbar, in denen die `CodeStarSourceConnection` Aktion verwendet wird CodePipeline GitHub, z. B. Bitbucket und GitLab.

Hinzufügen eines Triggers zum Ausschalten der Änderungserkennung (Konsole)

1. Melden Sie sich bei der an AWS-Managementkonsole 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 zur Pipeline-Erstellung.
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, ob Sie einen Trigger hinzufügen möchten.
5. Wählen Sie unter Triggertyp die Option Keine Änderungen erkennen aus.

Manuelles Starten und Stoppen von Pipelines

Sie können die AWS CodePipeline Konsole oder die verwenden, um Pipelines manuell AWS CLI zu starten und zu beenden. Es gibt automatisierte Methoden zum Starten von Pipelines mit ereignisbasierter Änderungserkennung, wie unter beschrieben, oder Triggerkonfiguration, wie unter beschrieben.

Themen

- Starten Sie eine Pipeline in CodePipeline
- Stoppen Sie eine Pipeline-Ausführung in CodePipeline

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 in der Ausführungshistorie für eine Pipeline angezeigt. Die Triggertypen können wie folgt vom Quellaktions-Provider abhängen:

> **Note**
>
> Sie können nicht mehr als einen Trigger pro Quellaktion angeben.

- Pipeline-Erstellung: Wenn eine Pipeline erstellt wird, startet die Pipeline-Ausführung automatisch. 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 bei 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, startet Ihre Pipeline. Dieser Triggertyp verwendet eine automatische Änderungserkennung. Die Quell-Aktionsanbieter, 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 von Quellenänderungen: Diese Kategorie stellt den `PollForSourceChanges` Triggertyp in der Ausführungshistorie dar. Wenn beim Polling eine Änderung an einem Quell-Commit und einem Branch im Quell-Repository erkannt wird, startet Ihre Pipeline. Dieser Triggertyp wird nicht empfohlen und sollte migriert werden, um die automatische Änderungserkennung zu verwenden. Die Quell-Aktionsanbieter, die diesen Triggertyp verwenden, sind S3 und CodeCommit.
- Webhook-Ereignisse für Quellen von Drittanbietern: Diese Kategorie steht für den `Webhook` Triggertyp im Ausführungsverlauf. Wenn eine Änderung durch ein Webhook-Ereignis erkannt wird, startet Ihre Pipeline. Dieser Triggertyp verwendet eine automatische Änderungserkennung. Die Anbieter von Quellaktionen, die diesen Triggertyp verwenden, sind Verbindungen, die für Code-Push konfiguriert sind (Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab selbstverwaltete Verbindungen). GitLab.com
- 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 Veröffentlichung mit einem bestimmten Git-Tag erkannt wird, startet Ihre Pipeline. Du kannst Git-Tags verwenden, um einen Commit mit einem Namen oder einem anderen Bezeichner zu kennzeichnen, der anderen Repository-Benutzern hilft, seine Bedeutung zu verstehen. Du kannst Git-Tags auch verwenden, um einen bestimmten Commit in der Historie eines Repositorys zu identifizieren. Dieser Triggertyp deaktiviert die automatische Änderungserkennung. 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 im Ausführungsverlauf. 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 steht für den `RollbackStage` Triggertyp in der Ausführungshistorie. Sie können die Konsole oder die verwenden AWS CLI , um eine Phase manuell oder automatisch rückgängig zu machen. Weitere Informationen finden Sie unter Konfiguration des Stage-Rollbacks.

Wenn Sie Ihrer Pipeline eine Quellaktion hinzufügen, die automatische Triggertypen für die Änderungserkennung verwendet, funktionieren die Aktionen mit zusätzlichen Ressourcen. Aufgrund dieser zusätzlichen Ressourcen für die Änderungserkennung wird das Erstellen der einzelnen Quellaktionen in separaten Abschnitten detailliert beschrieben. Einzelheiten zu den einzelnen Quellanbietern und den Methoden zur Änderungserkennung, die für die automatische

Änderungserkennung erforderlich sind, finden Sie unterQuellaktionen und Methoden zur Erkennung von Änderungen.

Themen

- Manuelles Starten einer Pipeline
- Starten Sie eine Pipeline nach einem Zeitplan
- Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung

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 Revision manuell erneut in Ihrer Pipeline auszuführen.

Themen

- Manueller Start einer Pipeline (Konsole)
- Manuelles Starten einer Pipeline (CLI)

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-Managementkonsole 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 so konfiguriert ist, dass sie Parameter (Pipeline-Variablen) weiterleitet, wird bei Auswahl von Release change das Fenster Release-Änderung geöffnet. Geben Sie unter Pipeline-Variablen in das Feld oder die Felder für die Variablen auf Pipeline-Ebene den oder die Werte ein, die Sie für diese Pipeline-Ausführung übergeben möchten. Weitere Informationen finden Sie unter Variablenreferenz.

   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. Wenn Sie beispielsweise die letzte Änderung über eine Pipeline mit dem Namen *MyFirstPipeline* starten möchten:

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

   Um eine Pipeline zu starten, in der Variablen auf Pipeline-Ebene 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 erhalten 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 CodePipeline.
6. Geben Sie den Pipeline-ARN für die Pipeline-Ausführung für diesen Zeitplan ein.

   > **Note**
   >
   > Den Pipeline-ARN finden Sie in der Konsole unter Einstellungen. Siehe Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen.

7. Wählen Sie eine der folgenden Optionen, um eine IAM-Servicerolle zu erstellen oder anzugeben, die EventBridge Berechtigungen zum Aufrufen des mit Ihrer EventBridge Regel verknüpften Ziels erteilt (in diesem Fall lautet 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 in allen Pipelines, die Sie für Ihr AWS Konto erstellen CodePipeline , 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 erstelltMyRule2, die EventBridge nach einem Zeitplan filtert.

   ```
   aws events put-rule --schedule-expression 'cron(15 10 ? * 6L 2002-2005)' --name
    MyRule2
   ```

2. Um es CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die folgenden Parameter ein:

   - 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
   ```

3. Erteilen Sie Berechtigungen EventBridge zur Verwendung CodePipeline zum Aufrufen der Regel. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel zum Erstellen der Vertrauensrichtlinie, mit der EventBridge die Servicerolle übernehmen kann. Geben Sie ihr den Namen `trustpolicyforEB.json`.

      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`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
      ```

```
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333: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 Quellrevisions-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.

> **Note**
>
> Sie können auch eine Quellüberschreibung mithilfe eines Eingabe-Transformationseintrags erstellen, um das `revisionValue` In EventBridge für Ihr Pipeline-Ereignis zu verwenden, wobei `revisionValue` das von der Quellereignisvariablen für Ihren Objektschlüssel, Ihren Commit oder Ihre Bild-ID abgeleitet wird. Weitere Informationen finden Sie in dem optionalen Schritt für die Eingabe der Eingabetransformation, der in den Verfahren unter Amazon ECR-Quellaktionen und Ressourcen EventBridgeHerstellen einer Verbindung zu Amazon S3-Quellaktionen mit einer für Ereignisse aktivierten Quelle, oder enthalten istCodeCommit Quellaktionen und EventBridge.

Es gibt vier Arten der Quellrevision für`revisionType`:

- `COMMIT_ID`

- `IMAGE_DIGEST`
- `S3_OBJECT_VERSION_ID`
- `S3_OBJECT_KEY`

> **Note**
>
> Für die `IMAGE_DIGEST` Typen der Quellversionen `COMMIT_ID` und gilt die Quellrevisions-ID für den gesamten Inhalt des Repositorys in allen Zweigen.

> **Note**
>
> Bei den Quellrevisionen `S3_OBJECT_VERSION_ID` und `S3_OBJECT_KEY` kann jeder der Typen unabhängig voneinander verwendet werden, oder sie können zusammen verwendet werden, um die Quelle mit einer bestimmten ObjectKey VersionID zu überschreiben. Für `S3_OBJECT_KEY` `AllowOverrideForS3ObjectKey` muss der Konfigurationsparameter auf gesetzt werden. `true` Weitere Informationen zu S3-Quellkonfigurationsparametern finden Sie unterKonfigurationsparameter.

Themen

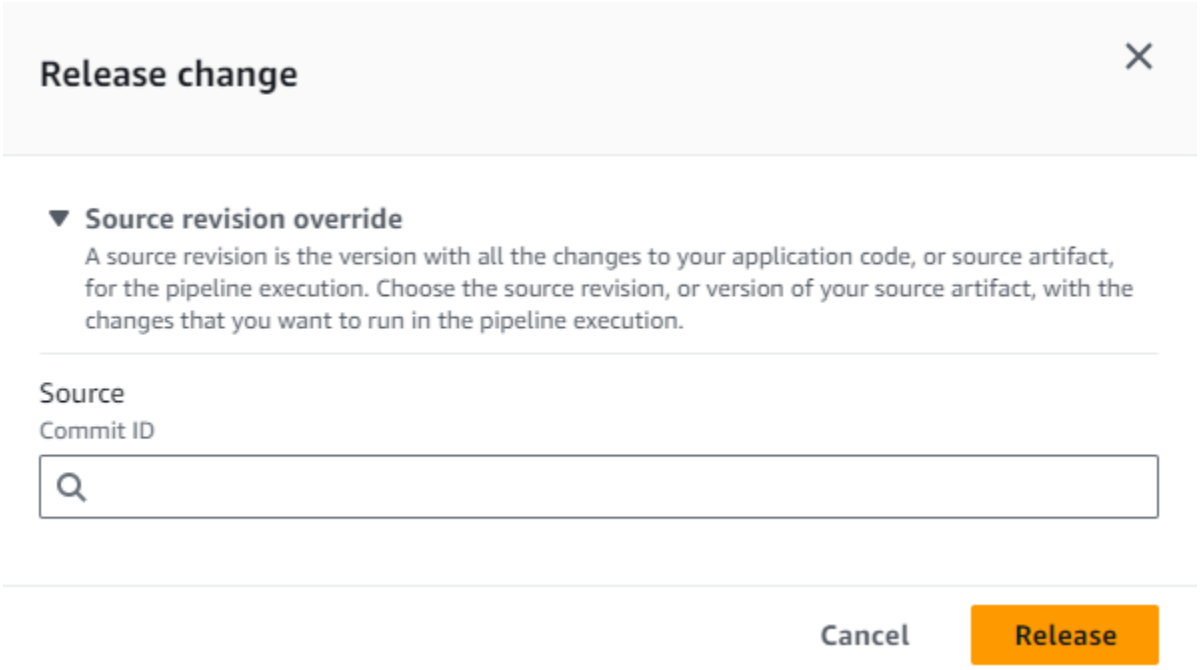
- Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung (Konsole)
- Starten Sie eine Pipeline mit einem Source Revision Override (CLI)

Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung (Konsole)

So starten Sie eine Pipeline und führen die letzte Revision manuell über eine Pipeline aus

1. Melden Sie sich bei der an AWS-Managementkonsole 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 Releasewechsel wählen, wird das Fenster Releasewechsel geöffnet. Wählen Sie für Quellversion überschreiben den Pfeil, um das Feld zu erweitern. Geben Sie im Feld Quelle die

Quell-Revision-ID ein. Wenn Ihre Pipeline beispielsweise eine CodeCommit Quelle hat, wählen Sie die Commit-ID aus dem Feld aus, das Sie verwenden möchten.



Starten Sie eine Pipeline mit einem Source Revision Override (CLI)

Um eine Pipeline manuell zu starten und die angegebene Quell-Revisions-ID für ein Artefakt durch eine Pipeline laufen zu lassen

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-Revisions-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 mit der Ausführung der angegebenen Änderung über eine Pipeline mit dem Namen zu beginnencodecommit-pipeline, gibt der folgende Befehl den Quellaktionsnamen 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 erhalten 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 Stoppen 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.

- Stoppen und abbrechen: AWS CodePipeline Stoppt die Ausführung, ohne auf den Abschluss laufender 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)
- Stoppen Sie eine eingehende Ausführung (Konsole)
- Pipeline-Ausführung anhalten (CLI)
- Stoppen Sie eine eingehende Ausführung (CLI)

Pipeline-Ausführung beenden (Konsole)

Sie können die Konsole verwenden, um eine Pipeline-Ausführung zu stoppen. Wählen Sie eine Ausführung und dann die Methode zum Stoppen der Pipeline-Ausführung.

> **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)

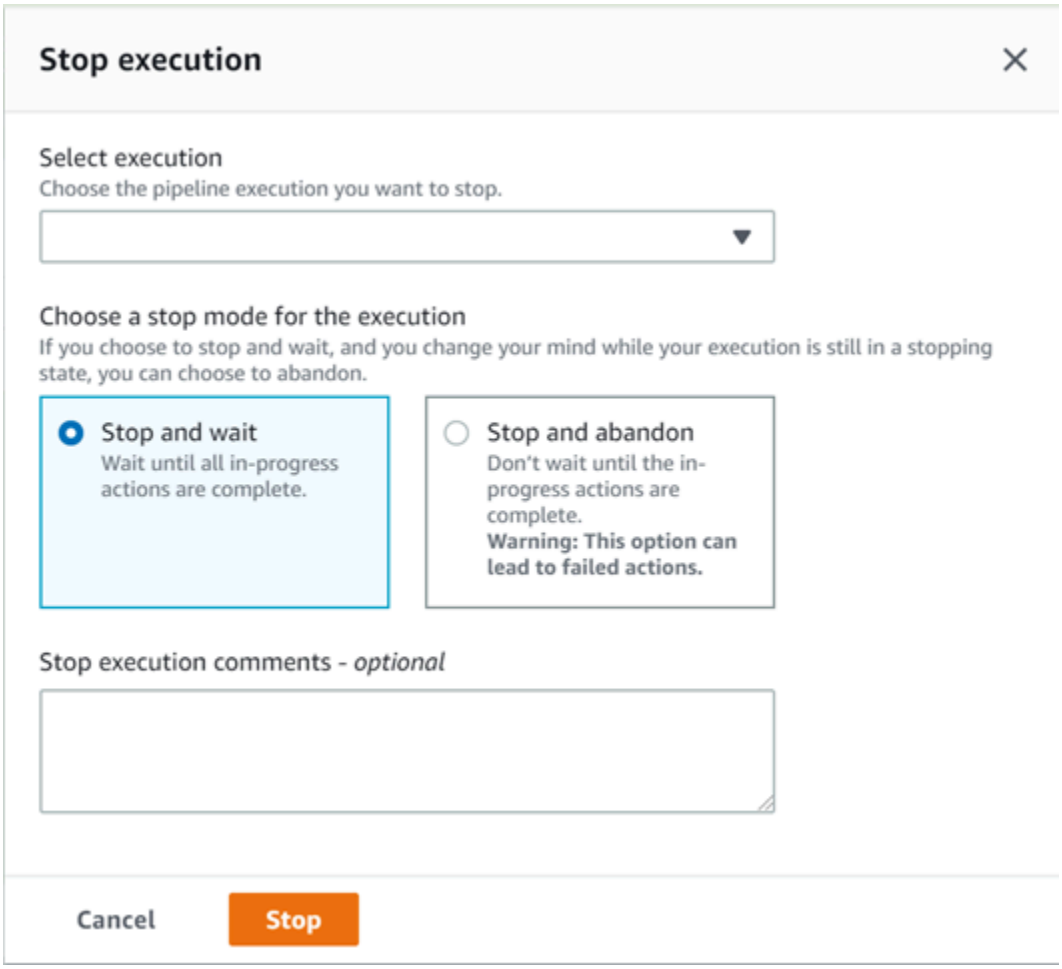
1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Führen Sie eine der folgenden Aktionen aus:

> **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 stoppen. Eine eingehende Ausführung ist eine Pipeline-Ausführung, die darauf wartet, in eine Phase einzutreten, in der der Übergang deaktiviert wurde. Wenn der Übergang aktiviert ist, tritt eine eingehende Ausführung `InProgress` weiterhin in die Phase ein. 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 bei einem deaktivierten Phasenübergang keine ausstehenden Ausführungen.

1. Melden Sie sich bei der an AWS-Managementkonsole 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, für die Sie die eingehende Ausführung stoppen möchten. Führen Sie eine der folgenden Aktionen aus:

   - Wählen Sie in der Pipeline-Ansicht die ID der eingehenden Ausführung und wählen 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 ID der eingehenden Ausführung aus und entscheiden Sie sich dann dafür, die Ausführung zu beenden.

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)

Verwenden Sie AWS CLI den stop-pipeline-execution Befehl mit den folgenden Parametern, um eine Pipeline manuell zu stoppen:

- 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. Beispiel: Eine Pipeline mit dem Namen *MyFirstPipeline* mit der Option "Anhalten und warten":

     ```
     aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
     pipeline-execution-id d-EXAMPLE --no-abandon
     ```

     Beispiel: Eine Pipeline mit dem Namen *MyFirstPipeline* anhalten, wobei standardmäßig die Option "Anhalten und warten" verwendet und die Option "Kommentar" gewählt wird:

```
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 *MyFirstPipeline* zu beenden, geben Sie die Beenden-Option an und fügen Kommentare ein:

  ```
  aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
  pipeline-execution-id d-EXAMPLE --abandon --reason "Stopping execution for a bug
   fix"
  ```

Stoppen Sie eine eingehende 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 einzutreten, in der der Übergang deaktiviert wurde. Wenn der Übergang aktiviert ist, tritt eine eingehende Ausführung `InProgress` weiterhin in die Phase ein. 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 bei einem deaktivierten Phasenübergang keine ausstehenden Ausführungen.

Wenn Sie den verwenden möchten AWS CLI , um eine eingehende Ausführung manuell zu stoppen, verwenden Sie den stop-pipeline-execution Befehl mit den folgenden Parametern:

- ID der eingehenden Ausführung (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]
```

Folgen Sie den Schritten des obigen Verfahrens, um den Befehl einzugeben und die Stoppmethode anzugeben.

Verwenden Sie den get-pipeline-state Befehl, um den Status der eingehenden Ausführung anzuzeigen.

Verlauf anzeigen und Modus für Pipeline-Ausführungen festlegen

Um den Fortschritt der Pipeline zu analysieren, können Sie den Verlauf der Pipeline- und Aktionsausführung einsehen. Um die Release-Methode für die Verarbeitung von Pipeline-Ausführungen zu konfigurieren, legen Sie den Ausführungsmodus fest.

Themen

- Hinrichtungen anzeigen in CodePipeline
- Den Pipeline-Ausführungsmodus festlegen oder ändern

Hinrichtungen anzeigen in CodePipeline

Sie können die AWS CodePipeline Konsole oder die verwenden, AWS CLI um den Ausführungsstatus und den Ausführungsverlauf einzusehen und fehlgeschlagene Phasen oder Aktionen zu wiederholen.

Themen

- Ausführungsverlauf einer Pipeline anzeigen (Konsole)
- Anzeige des Ausführungsstatus (Konsole)
- Eine eingehende Ausführung anzeigen (Konsole)
- Anzeigen von Revisionen der Pipeline-Ausführungsquelle (Konsole)
- Anzeigen von Aktionsausführungen (Konsole)
- Anzeigen von Informationen zu Aktionsartefakten und zum Artefaktspeicher (Konsole)
- Anzeigen von der Pipeline-Details und des Verlaufs (CLI)

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 Den Pipeline-Ausführungsmodus festlegen oder ändern.

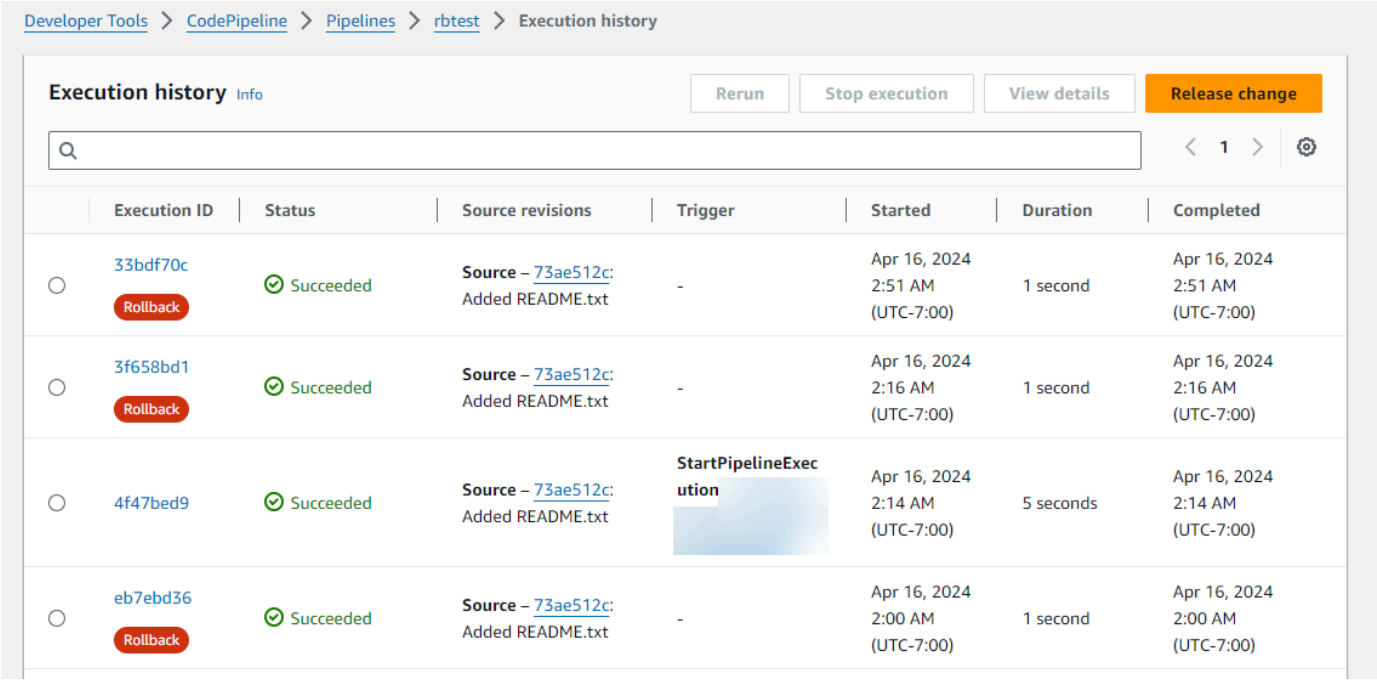
Der Ausführungsverlauf wird bis zu 12 Monate lang aufbewahrt.

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-Managementkonsole 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 PARALLELEN 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 die ID für die Ausführung, die Sie anzeigen möchten, auf der Seite mit dem Ausführungsverlauf auswählen. Wählen Sie in der linken Navigationsleiste Verlauf aus, wählen Sie die Ausführungs-ID für die parallele Ausführung aus, und sehen Sie sich 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 durchgeführt 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. Die Detailansicht zeigt Ausführungsdetails, die Registerkarte „Zeitleiste“, die Registerkarte „Visualisierung“ und die Registerkarte „Variablen“. 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 eingesehen 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 zum Anzeigen einer eingehenden Ausführung und ihres Status finden Sie unter. Eine eingehende Ausführung anzeigen (Konsole)

Pipeline-level Staaten

Pipelinestatus	Description
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.

Stage-level Staaten

Stufenstatus	Description
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	Description
Angehalten	Der Anhaltevorgang ist abgeschlossen, und die Phasenausführung wird angehalten.
Erfolgreich	Die Stufe wurde erfolgreich abgeschlossen.
Fehlgeschlagen	Die Stufe wurde nicht erfolgreich abgeschlossen.

Action-level Staaten

Aktionsstatus	Description
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 nicht alle laufenden Ausführungen fortgesetzt, und der Ausführungsstatus ändert sich in. `Failed`

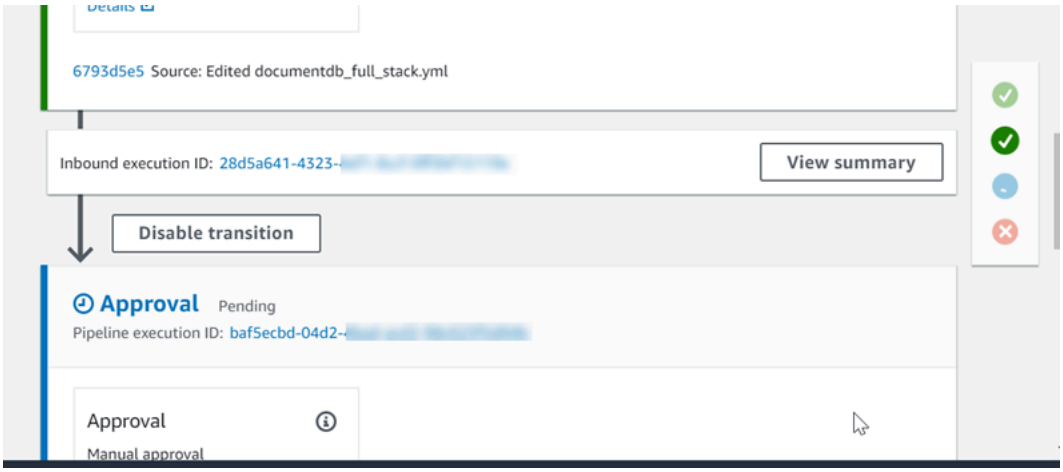
Wenn Sie keine eingehende Ausführung sehen, gibt es bei einem deaktivierten Phasenübergang keine ausstehenden Ausführungen.

1. Melden Sie sich bei der an AWS-Managementkonsole 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, für die Sie die eingehende Ausführung anzeigen möchten. Führen Sie einen der folgenden Schritte aus:

   - Wählen Sie View (Anzeige). Im Pipeline-Diagramm können Sie im Feld Inbound Execution ID vor Ihrem deaktivierten Übergang die ID der eingehenden Ausführung sehen.



     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 Repositorys GitHub und CodeCommit Repositorys die Commit-Nachricht. Für Amazon S3-Buckets oder Aktionen, die vom Benutzer bereitgestellten Inhalte eines in den Objektmetadaten angegebenen Codepipeline-Artefakt-Revisionszusammenfassungsschlüssels.
- revisionUrl: Die Revisions-URL für die Artefakt-Revision (z. B. die externe Repository-URL).
- revisionId: Die Versions-ID für die Artefakt-Revision. Bei einer Quelländerung in einem CodeCommit GitHub OR-Repository ist dies beispielsweise die Commit-ID. Für Artefakte, die in

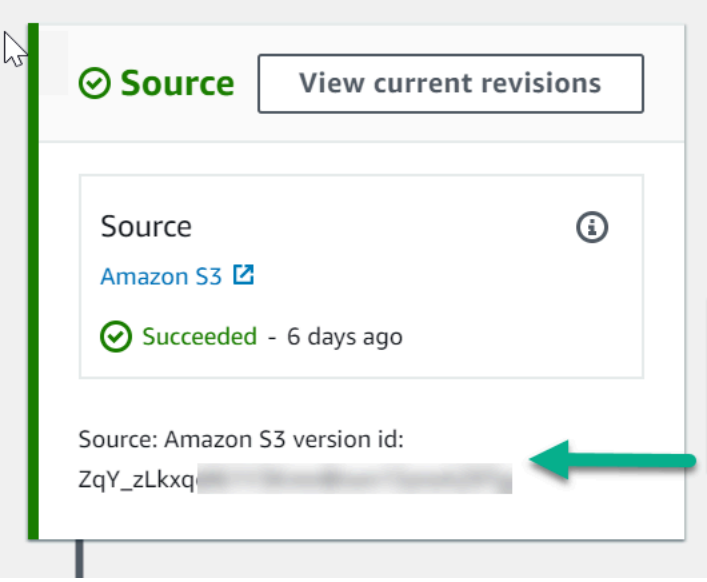
unseren GitHub CodeCommit Repositorys gespeichert sind, ist die Commit-ID mit einer Commit-Detailseite verknüpft.

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

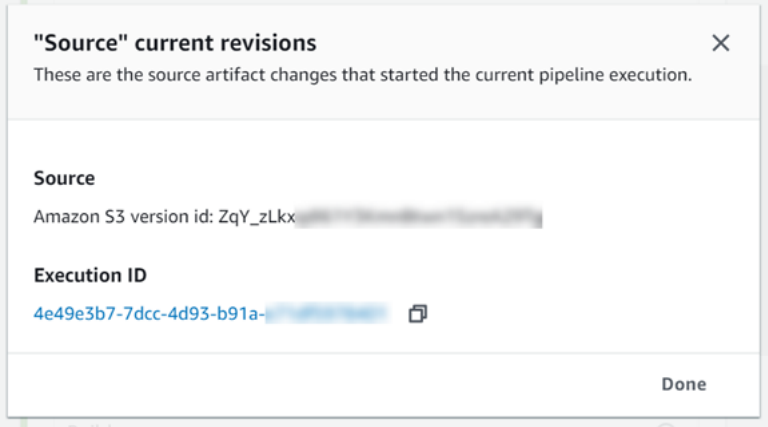
   Die Namen aller Pipelines, die mit Ihrem verknüpft sind AWS-Konto , werden angezeigt.

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-Buckets gespeichert sind, sind Identifikatoren wie Commit-IDs in dieser Informationsdetailansicht mit den 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 bei der an AWS-Managementkonsole 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 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 vorangegangenen Beispiel-Pipeline entweder die Phase Staging oder Production wählen CodeDeploy , wird die CodeDeploy Konsolenseite für die CodeDeploy Anwendung angezeigt, die für diese Phase konfiguriert ist.

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-Managementkonsole 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 Aktionsausführung 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 aus Aktionen an, die für die Aktionsausführung in der Pipeline sind.

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 Ihrer 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

- Zeigen Sie den Ausführungsverlauf mit List-Pipeline-Executions (CLI) an
- Zeigen Sie den Pipeline-Status mit get-pipeline-state (CLI) an
- Zeigen Sie den Status der eingehenden Ausführung mit get-pipeline-state (CLI) an
- Zeigen Sie Status- und Quellrevisionen mit get-pipeline-execution (CLI) an
- Zeigen Sie Aktionsausführungen mit List-Action-Executions (CLI) an

Zeigen Sie den Ausführungsverlauf mit List-Pipeline-Executions (CLI) an

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. Wenn Sie beispielsweise Details zum aktuellen Status einer Pipeline mit dem Namen *MyFirstPipeline* anzeigen möchten, 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 durchgeführt wurde, wird der Ausführungstyp angezeigt. `Rollback` 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* mit drei Ausführungen:


```
{
    "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"
            }
        ]
    }
}
```

Zeigen Sie den Pipeline-Status mit get-pipeline-state (CLI) an

Sie können die CLI verwenden, um den Pipeline-, Phase- 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. Wenn Sie

beispielsweise Details zum aktuellen Status einer Pipeline mit dem Namen *MyFirstPipeline* anzeigen möchten, 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 Pipeline mit drei Stufen und dem Namen *MyFirstPipeline*. Die ersten beiden Stufen und Aktionen wurden erfolgreich ausgeführt. Der Übergang zwischen der zweiten und dritten Stufe ist deaktiviert:

```
{
    "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"
        }
    ]
}
```

Zeigen Sie den Status der eingehenden Ausführung mit get-pipeline-state (CLI) an

Sie können die CLI verwenden, um den Status der 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 nicht alle laufenden Ausführungen fortgesetzt, und der Ausführungsstatus ändert sich in. `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. Wenn Sie beispielsweise Details zum aktuellen Status einer Pipeline mit dem Namen *MyFirstPipeline* anzeigen möchten, 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 eine eingehende Ausführungs-ID für eine Phase mit deaktiviertem Übergang 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 zeigt und die zweite Stufe, benannt`Beta`, einen deaktivierten Übergang und eine eingehende Ausführungs-ID zeigt. 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
}
```

Zeigen Sie Status- und Quellrevisionen mit get-pipeline-execution (CLI) an

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 Repositorys GitHub und AWS CodeCommit Repositorys die Commit-Nachricht. Für Amazon S3-Buckets oder Aktionen, die vom Benutzer bereitgestellten Inhalte eines in den Objektmetadaten angegebenen Codepipeline-Artefakt-Revisionszusammenfassungsschlüssels.
- revisionUrl: Die Commit-ID für die Artefaktrevision. Bei Artefakten, die in unseren GitHub 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 Quellrevisionen in einer Pipeline anzuzeigen

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) und verwenden Sie die AWS CLI , um den get-pipeline-state Befehl auszuführen. Für eine Pipeline mit dem Namen *MyFirstPipeline* geben Sie ein:

   ```
   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 über die Ausführung einer Pipeline mit dem Namen *MyFirstPipeline* und der Ausführungs-ID 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE anzuzeigen, geben Sie Folgendes ein:

   ```
   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 named*MyFirstPipeline*, in dem ein Artefakt mit dem Namen "MyApp" in einem GitHub Repository gespeichert ist:

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"
         }
    }
}
```

Zeigen Sie Aktionsausführungen mit List-Action-Executions (CLI) an

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. Wenn Sie beispielsweise die Aktionsausführungen in einer Pipeline mit dem Namen *MyFirstPipeline* anzeigen möchten, 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. Um beispielsweise die Aktionsausführungen für eine anzuzeigen, geben Sie Folgendes ein: *Execution_ID*

```
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"
                    }
                ]
            },


. . .
```

Den Pipeline-Ausführungsmodus festlegen oder ändern

Sie können den Ausführungsmodus für Ihre Pipeline festlegen, um anzugeben, wie mehrere Ausführungen behandelt werden.

Weitere Hinweise zu den Pipeline-Ausführungsmodi finden Sie unterSo funktionieren Pipeline-Ausführungen.

> **Important**
>
> Wenn Sie bei Pipelines im PARALLELMODUS den Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED ändern, zeigt der Pipeline-Status den aktualisierten Status nicht als PARALLEL an. Weitere Informationen finden Sie unter Bei Pipelines, die vom PARALLEL-Modus aus geändert wurden, wird ein früherer Ausführungsmodus angezeigt.

> **Important**
>
> Bei Pipelines im PARALLELMODUS wird die Pipeline-Definition für die Pipeline in den Modi QUEUED oder SUPERSEDED 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.

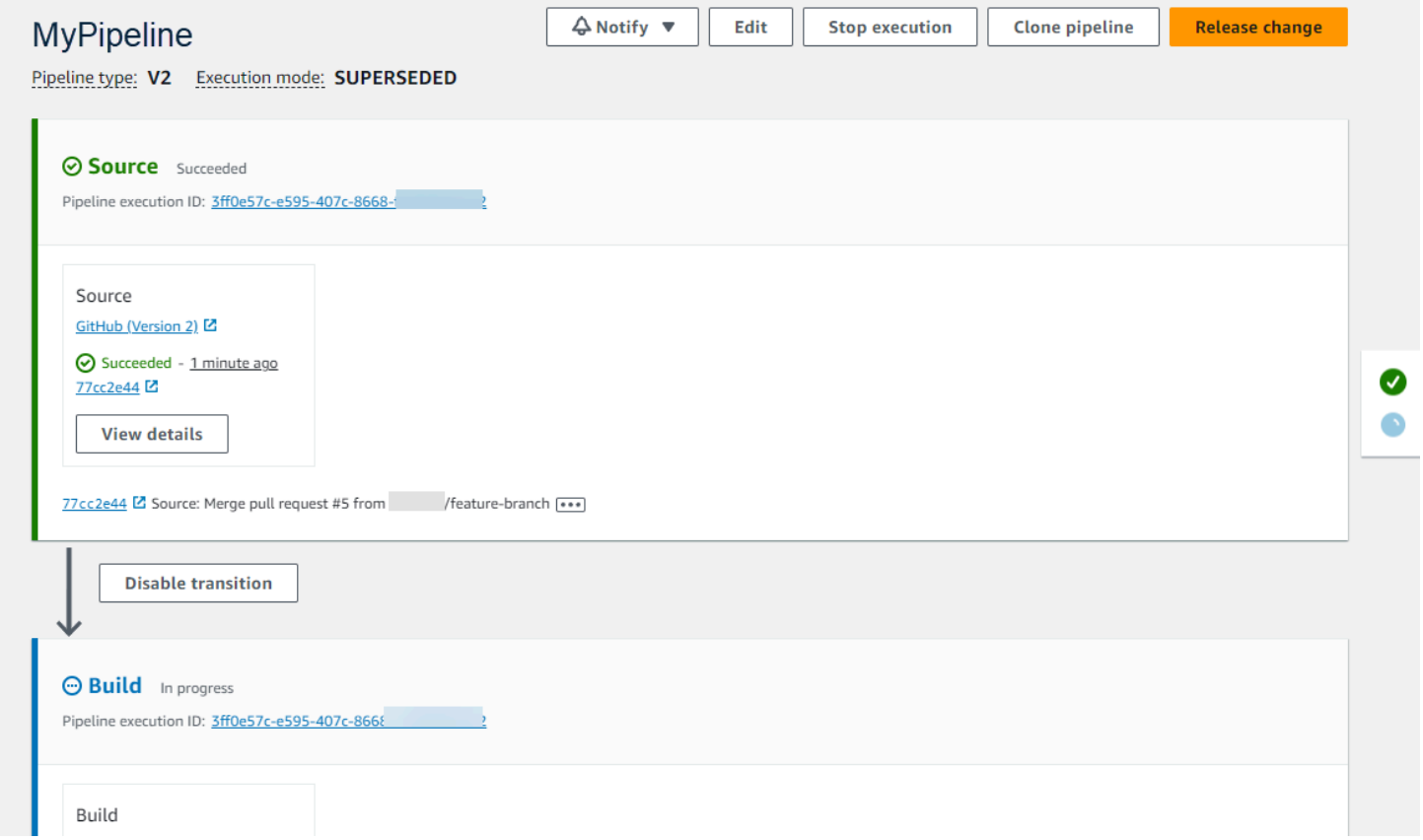
> **Important**
>
> Für Pipelines im PARALLELMODUS ist ein Stufen-Rollback nicht verfügbar. Ebenso können einer Pipeline im PARALLEL-Modus keine Fehlerbedingungen mit einem Rollback-Ergebnistyp hinzugefügt werden.

Überlegungen zur Anzeige der Ausführungsmodi

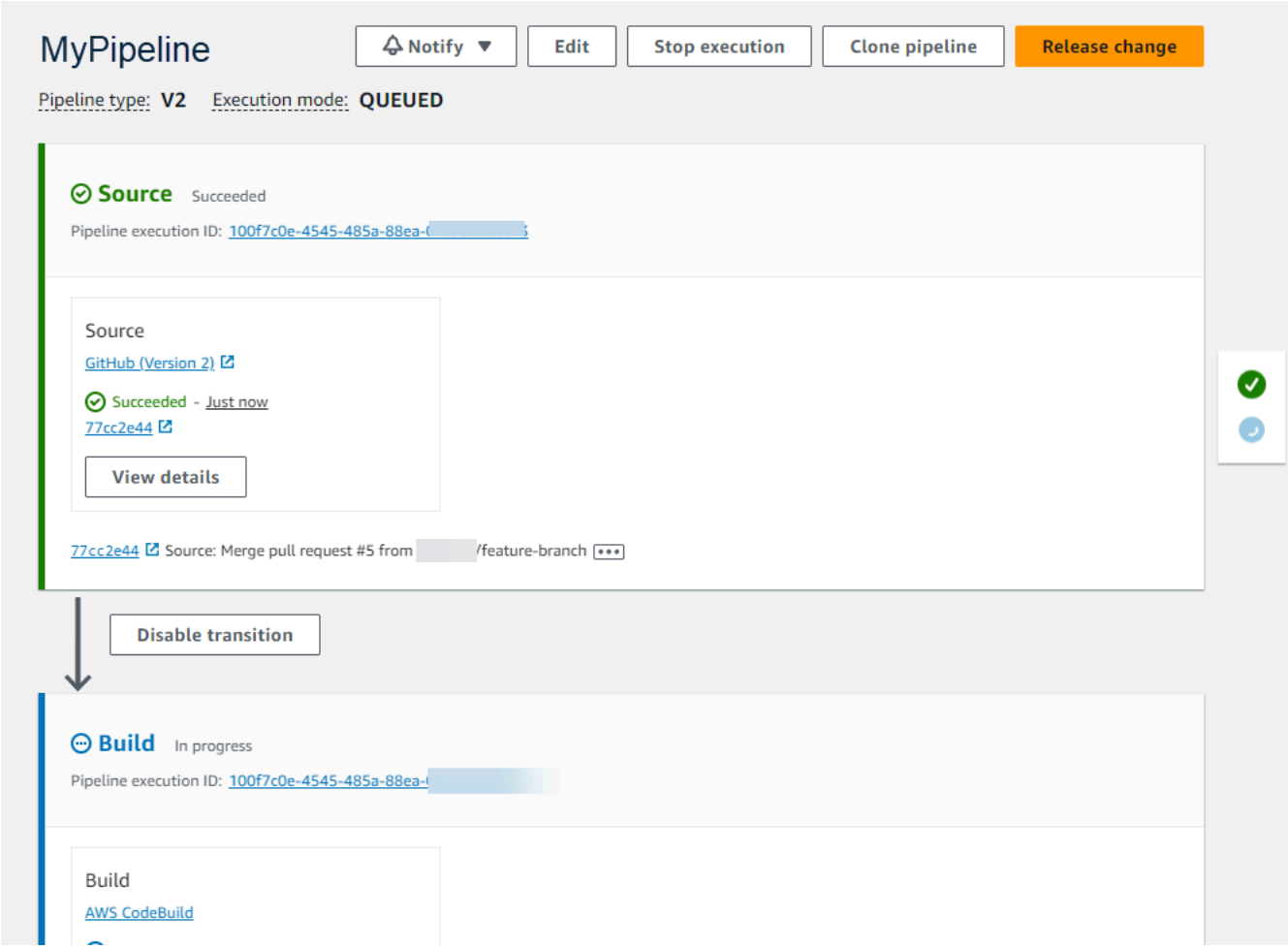
Bei der Anzeige von Pipelines in bestimmten Ausführungsmodi sind einige Überlegungen zu beachten.

Verwenden Sie für die Modi SUPERSEDED und QUEUED die Pipeline-Ansicht, um laufende Ausführungen zu sehen, und klicken Sie auf die Ausführungs-ID, um Details und den Verlauf anzuzeigen. Klicken Sie im PARALLEL-Modus auf die Ausführungs-ID, um die laufende Ausführung auf der Registerkarte Visualisierung anzuzeigen.

Im Folgenden wird die Ansicht für den SUPERSEDED-Modus in gezeigt. CodePipeline

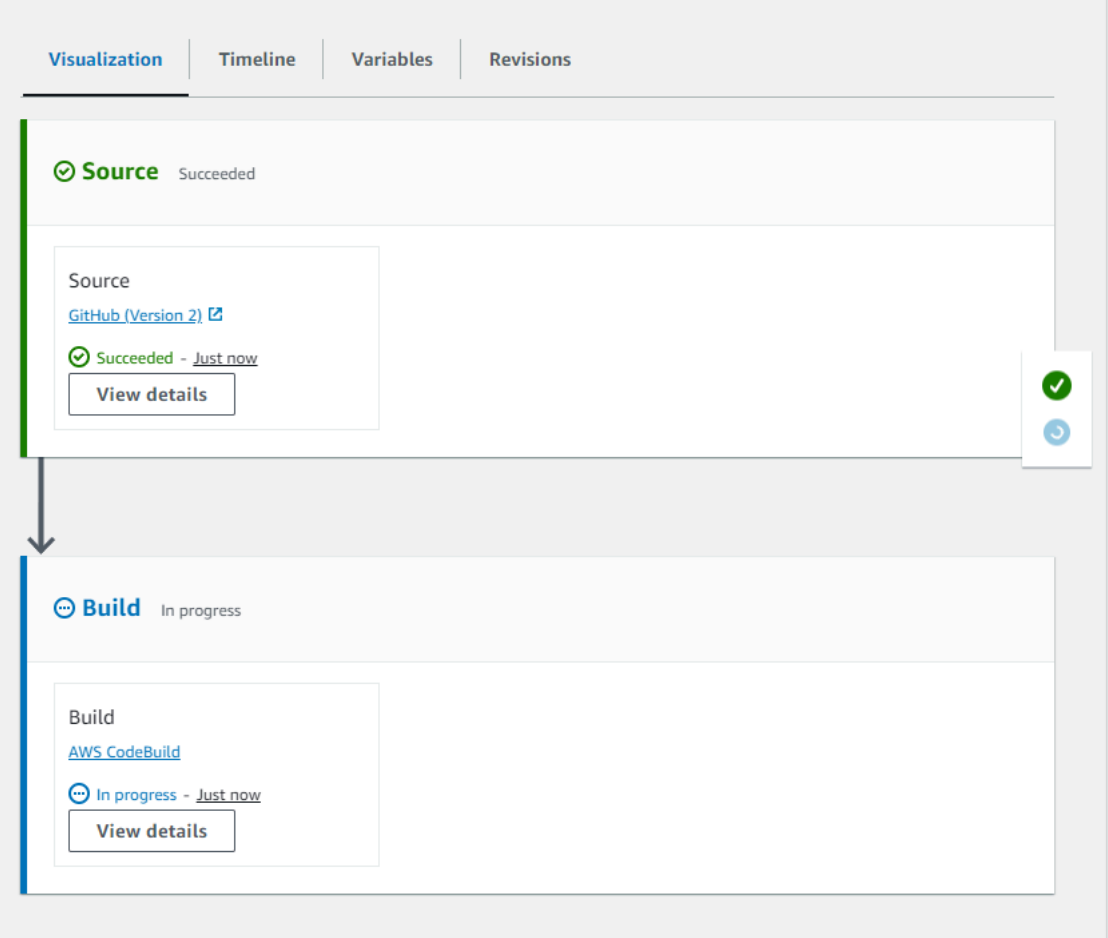


Im Folgenden wird die Ansicht für den QUEUED-Modus in CodePipeline gezeigt.



Im Folgenden wird die Ansicht für den PARALLEL-Modus in gezeigt CodePipeline.

> **Important**
>
> Für Pipelines im PARALLEL-MODUS ist ein Stufen-Rollback nicht verfügbar. Ebenso können einer Pipeline im PARALLEL-Modus keine Fehlerbedingungen mit einem Rollback-Ergebnistyp hinzugefügt werden.



Überlegungen zum Umschalten zwischen den 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 dann speichern, passen sich möglicherweise bestimmte Ansichten oder Zustände an.

Wenn Sie beispielsweise vom PARALLELMODUS in den QUEUED- oder SUPERSEDED-Modus wechseln, läuft die im PARALLEL-MODUS gestartete Ausführung weiter. 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 ausgeführt wurde, oder andernfalls in einem leeren Zustand.

Ein weiteres Beispiel: Wenn Sie vom QUEUED- oder SUPERSEDED-Modus in den PARALLEL-Modus wechseln, wird die Pipeline-Seite nicht mehr angezeigt. view/state Um eine Ausführung im PARALLELMODUS anzusehen, 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	Details zur ausstehenden und aktiven Ausführung	Einzelheiten zum Status der Pipeline
ERSETZT DURCH ERSETZT/ ERSETZT DURCH WARTESCHLANGE	• Aktive Ausführungen werden abgebrochen, nachdem laufende Aktionen abgeschlossen sind. • Ausstehende Ausführungen werden storniert.	Der Pipelinestatus, z. B. abgebrochen, wird zwischen der Version des ersten Modus und dem zweiten Modus beibehalten.
IN DIE WARTESCHL ANGE GESTELLT IN DIE WARTESCHL ANGE GESTELLT/IN DIE WARTESCHLANGE GESTELLT BIS ERSETZT	• Aktive Ausführungen werden abgebrochen, nachdem laufende Aktionen abgeschlossen sind. • Ausstehende Ausführungen werden storniert.	Der Pipelinestatus, z. B. abgebrochen, wird zwischen der Version des ersten Modus und dem zweiten Modus beibehalten.
PARALLEL ZU PARALLEL	Alle Ausführungen dürfen unabhängig von Aktualisi erungen der Pipeline-Definitio nen ausgeführt werden.	Leer. Der Parallelmodus hat keinen Pipeline-Status.
DURCH PARALLEL ERSETZT/IN DIE WARTESCHLANGE GESTELLT	• Aktive Ausführungen werden abgebrochen, nachdem laufende 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-Managementkonsole 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“.
5. Wählen Sie den Modus für Ihre Pipeline.

   - Ersetzt
   - In Warteschlange gestellt (Pipeline-Typ V2 erforderlich)
   - Parallel (Pipeline-Typ V2 erforderlich)
6. Wählen Sie auf der Seite „Bearbeiten“ die Option „Fertig“.

Stellen Sie den Pipeline-Ausführungsmodus (CLI) ein

Verwenden Sie den `update-pipeline` Befehl `create-pipeline` or, 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 Klartext-Editor und ändern Sie die Struktur der Datei so, dass sie dem Pipeline-Ausführungsmodus entspricht, 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 setzen 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.

Stufen rückgängig machen oder erneut versuchen

Sie können die AWS CodePipeline Konsole oder die verwenden AWS CLI , um eine Phase oder Aktionen in einer Phase manuell rückgängig zu machen oder erneut zu versuchen. Informationen zum Konfigurieren von Bedingungen für Phasen, in denen Sie Rollback oder Retry als konfiguriertes Ergebnis verwenden möchten, finden Sie unter. Bedingungen für eine Phase konfigurieren

Themen

- Konfiguration des Wiederholungsversuchs für eine fehlgeschlagene Phase oder für fehlgeschlagene Aktionen
- Konfiguration des Stage-Rollbacks

Konfiguration des Wiederholungsversuchs für eine fehlgeschlagene Phase oder für fehlgeschlagene Aktionen

Sie können einen Schritt, bei dem ein Fehler aufgetreten ist, wiederholen, ohne eine Pipeline erneut von Anfang an ausführen zu müssen. Sie tun dies, indem Sie entweder die fehlgeschlagenen Aktionen in einer Phase wiederholen 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, weiter und fehlgeschlagene Aktionen werden erneut ausgelöst. Wenn Sie eine fehlgeschlagene Phase von der ersten Aktion in der Phase aus wiederholen, können in der Phase keine laufenden Aktionen ausgeführt werden. Bevor eine Phase erneut versucht werden kann, müssen entweder alle Aktionen fehlgeschlagen sein oder einige Aktionen sind fehlgeschlagen und andere waren erfolgreich.

> **Important**
>
> Beim erneuten Versuch einer fehlgeschlagenen Phase werden alle Aktionen in der Phase ab der ersten Aktion in der Phase wiederholt, und beim erneuten Versuch einer fehlgeschlagenen Aktion werden alle fehlgeschlagenen Aktionen in der Phase wiederholt. Dadurch werden Ausgabeartefakte von zuvor erfolgreichen 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, wird auf der Stufe entweder eine Schaltfläche „Phase wiederholen“ oder eine Schaltfläche „Fehlgeschlagene Aktionen wiederholen“ angezeigt, 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 sich die Phase nicht im Status Fehlgeschlagen befindet.
> - Die gesamte Pipeline-Struktur änderte sich, nachdem die Phase fehlgeschlagen war.
> - Es wird bereits ein anderer Wiederholungsversuch in der Phase ausgeführt.

Themen

- Überlegungen zur Wiederholung der Phase
- Wiederholen Sie eine fehlgeschlagene Phase manuell
- Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler

Überlegungen zur Wiederholung der Phase

Für eine Wiederholung der Phase sind folgende Überlegungen zu beachten:

- Sie können die automatische Wiederholung bei einem Stufenfehler nur für einen Wiederholungsversuch konfigurieren.
- Sie können die automatische Wiederholung bei einem Ausfall einer Phase für alle Aktionen, einschließlich Aktionen, konfigurieren. `Source`

Wiederholen Sie eine fehlgeschlagene Phase manuell

Sie können eine fehlgeschlagene Phase mithilfe der Konsole oder der CLI manuell wiederholen.

Sie können auch eine Phase für die automatische Wiederholung bei einem Fehler konfigurieren, wie unter beschrieben. Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler

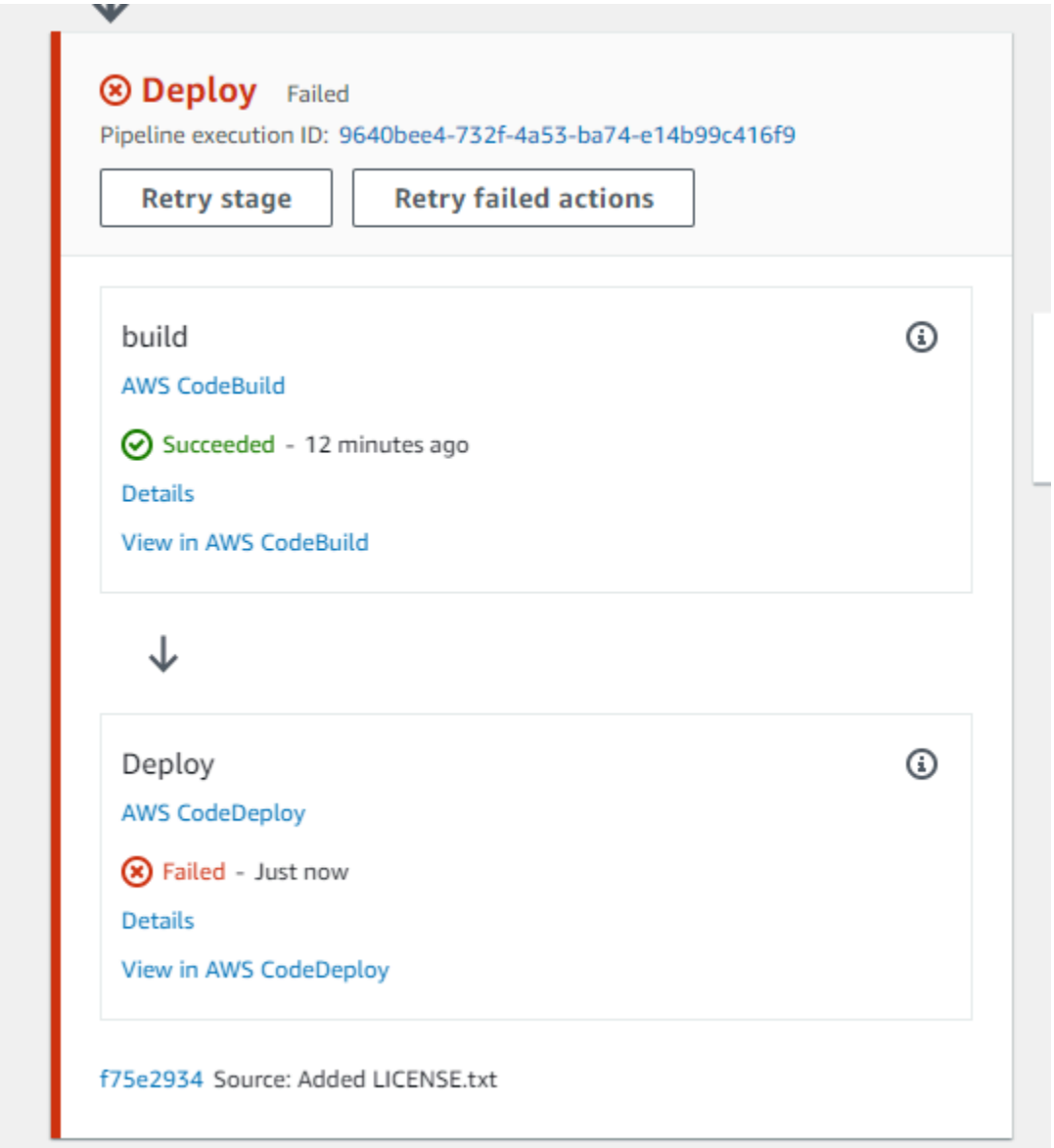
Einen fehlgeschlagenen Schritt manuell wiederholen (Konsole)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase zu wiederholen — Konsole

1. Melden Sie sich bei der an AWS-Managementkonsole 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 zu wiederholen, wählen Sie Phase wiederholen.
   - Um nur fehlgeschlagene Aktionen in der Phase zu wiederholen, wählen Sie Fehlgeschlagene Aktionen wiederholen.



Wenn alle wiederholten Aktionen in der Stufe erfolgreich abgeschlossen sind, wird die Pipeline weiter ausgeführt.

Wiederholen Sie eine fehlgeschlagene Phase manuell (CLI)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase zu wiederholen — CLI

Um die zu verwenden AWS CLI , um alle Aktionen oder alle fehlgeschlagenen Aktionen erneut 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-state Befehlszeile (Windows) in einer Pipeline aus. Für eine Pipeline mit dem Namen würden Sie MyFirstPipeline beispielsweise etwas eingeben, das dem Folgenden ähnelt:

      ```
      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 ungefähr 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. Beispiel:

> **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.

Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler

Sie können eine Phase für die automatische Wiederholung bei einem Fehler konfigurieren. Die Phase unternimmt einen Wiederholungsversuch und zeigt auf der Seite „Pipeline anzeigen“ einen Wiederholungsstatus für die fehlgeschlagene Phase an.

Sie können den Wiederholungsmodus konfigurieren, indem Sie angeben, dass die Phase automatisch entweder alle Aktionen in der fehlgeschlagenen Phase oder nur fehlgeschlagene Aktionen in der Phase wiederholen soll.

Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler (Konsole)

Sie können die Konsole verwenden, um eine Phase für die automatische Wiederholung zu konfigurieren.

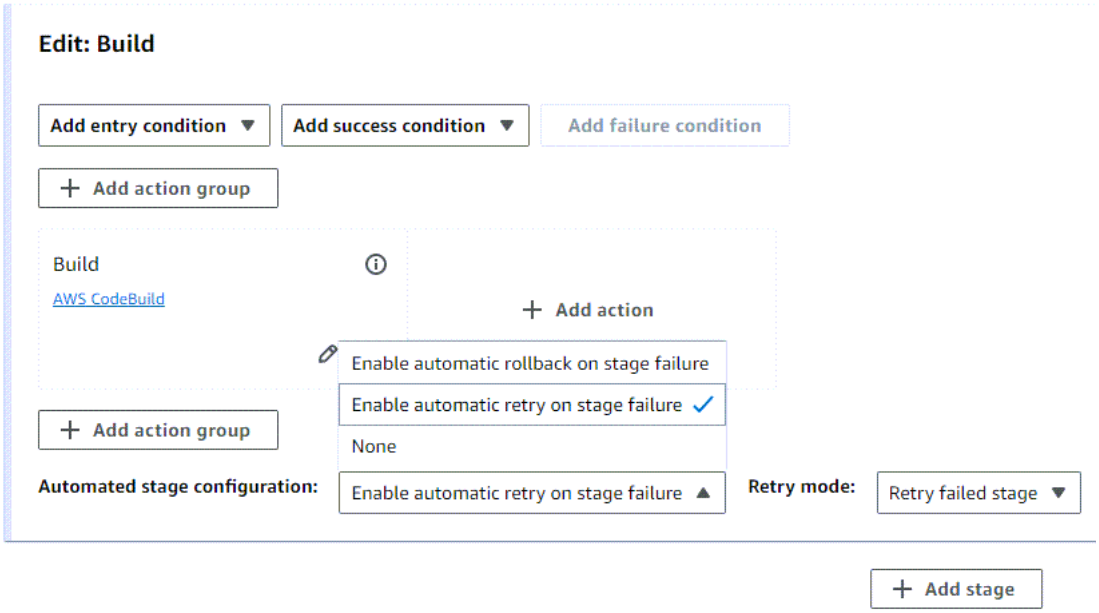
Konfigurieren Sie eine Phase für die automatische Wiederholung (Konsole)

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

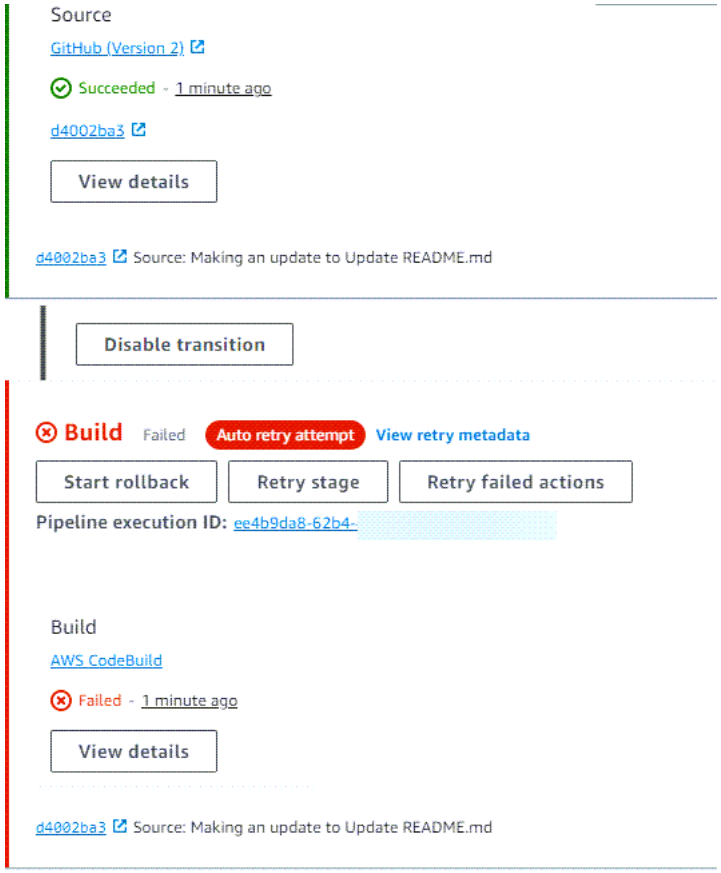
   Die Namen und der Status aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie auf der Seite „Bearbeiten“ für die Aktion, die Sie bearbeiten möchten, die Option Phase bearbeiten aus.
5. Wählen Sie Automatisierte Stufenkonfiguration: und wählen Sie dann Automatische Wiederholung bei Ausfall einer Phase aktivieren aus. Speichern Sie die Änderungen an Ihrer Pipeline.
6. Wählen Sie unter Automatisierte Stufenkonfiguration: einen der folgenden Wiederholungsmodi aus:

   - Um anzugeben, dass der Modus alle Aktionen in der Phase wiederholt, wählen Sie „Fehlgeschlagene Phase wiederholen“.
   - Um festzulegen, dass der Modus nur fehlgeschlagene Aktionen in der Phase wiederholt, wählen Sie Fehlgeschlagene Aktionen wiederholen.

   Speichern Sie die Änderungen an Ihrer Pipeline.



7. Wenn die Phase nach der Ausführung Ihrer Pipeline ausfällt, wird der automatische Wiederholungsversuch unternommen. Die folgenden Beispiele zeigen eine Build-Phase, die automatisch wiederholt wurde.



8. Um Details zum Wiederholungsversuch anzuzeigen, wählen Sie. Das Fenster wird angezeigt.



Konfigurieren Sie eine Phase für die automatische Wiederholung (CLI)

Um eine Phase so AWS CLI zu konfigurieren, dass sie bei einem Fehler automatisch erneut versucht, verwenden Sie die Befehle, um eine Pipeline zu erstellen oder zu aktualisieren, wie in Eine Pipeline, Phasen und Aktionen erstellen und beschrieben. Eine Pipeline bearbeiten in CodePipeline

- Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) und verwenden Sie den, AWS CLI um den `update-pipeline` Befehl auszuführen. Geben

Sie dabei die Fehlerbedingung in der Pipeline-Struktur an. Im folgenden Beispiel wird die automatische Wiederholung für eine Stufe 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": "RETRY",
                    "retryConfiguration": {
                        "retryMode": "ALL_ACTIONS",
                    },
            }
```

Konfigurieren Sie eine Phase für die automatische Wiederholung (CloudFormation)

Um eine Phase für die automatische Wiederholung bei einem Fehler CloudFormation zu konfigurieren, verwenden Sie den `OnFailure` Stage-Lifecycle-Parameter. Verwenden Sie den `RetryConfiguration` Parameter, um den Wiederholungsmodus zu konfigurieren.

```
OnFailure:
     Result: RETRY
     RetryConfiguration:
         RetryMode: ALL_ACTIONS
```

- Aktualisieren Sie die Vorlage wie im folgenden Codeausschnitt gezeigt. Im folgenden Beispiel wird die automatische Wiederholung für eine Stufe 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: RETRY
         RetryConfiguration:
            RetryMode: ALL_ACTIONS
  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 von Wiederholungsversuchen im Fehlerfall finden Sie OnFailure unter `StageDeclaration` im CloudFormation Benutzerhandbuch.

Konfiguration des Stage-Rollbacks

Sie können einen Schritt auf eine Ausführung zurücksetzen, die in dieser Phase erfolgreich war. Sie können eine Phase so vorkonfigurieren, dass sie bei einem Fehler rückgängig gemacht wird, 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 verwendet, um Quellrevisionen und Variablen abzurufen.

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
- Führt eine Stufe manuell zurück
- Konfigurieren Sie eine Phase für das automatische Rollback
- Sehen Sie sich den Rollback-Status in der Ausführungsliste an
- Details zum Rollback-Status anzeigen

Überlegungen zu Rollbacks

Beim Stufen-Rollback sind folgende Überlegungen zu beachten:

- Ein Rollback einer Quellstufe ist nicht möglich.
- Die Pipeline kann nur dann zu einer früheren Ausführung zurückkehren, wenn die vorherige Ausführung in der aktuellen Pipeline-Strukturversion gestartet wurde.
- Sie können kein Rollback zu einer Zielausführungs-ID durchführen, bei der es sich um einen Rollback-Ausführungstyp handelt.
- CodePipeline verwendet die Variablen und Artefakte der Ausführung, zu der das Rollback durchgeführt wird.

Führt eine Stufe manuell zurück

Sie können eine Phase mithilfe der Konsole oder der CLI manuell rückgängig machen. Die Pipeline kann nur dann zu einer früheren Ausführung zurückkehren, wenn die vorherige Ausführung in der aktuellen Pipeline-Strukturversion gestartet wurde.

Sie können eine Phase auch so konfigurieren, dass sie bei einem Fehler automatisch rückgängig gemacht wird, wie unter beschriebenKonfigurieren Sie eine Phase für das automatische Rollback.

Führen Sie ein manuelles Rollback einer Stufe durch (Konsole)

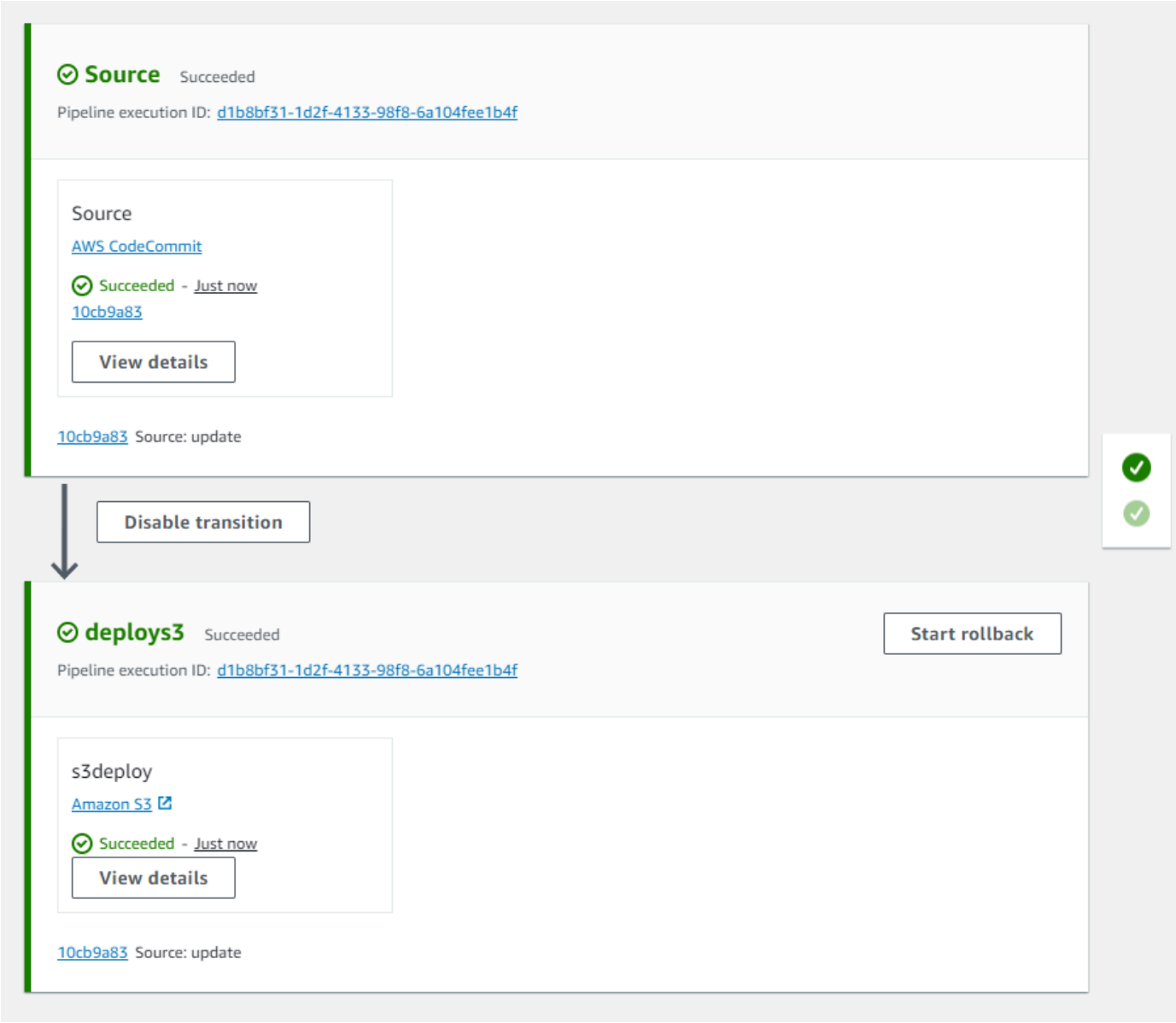
Sie können die Konsole verwenden, um ein manuelles Rollback einer Phase auf eine Ziel-Pipeline-Ausführung durchzuführen. Wenn für eine Phase ein Rollback durchgeführt wird, wird in der Pipeline-Visualisierung in der Konsole ein Rollback-Label angezeigt.

Führen Sie ein manuelles Rollback einer Stufe durch (Konsole)

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

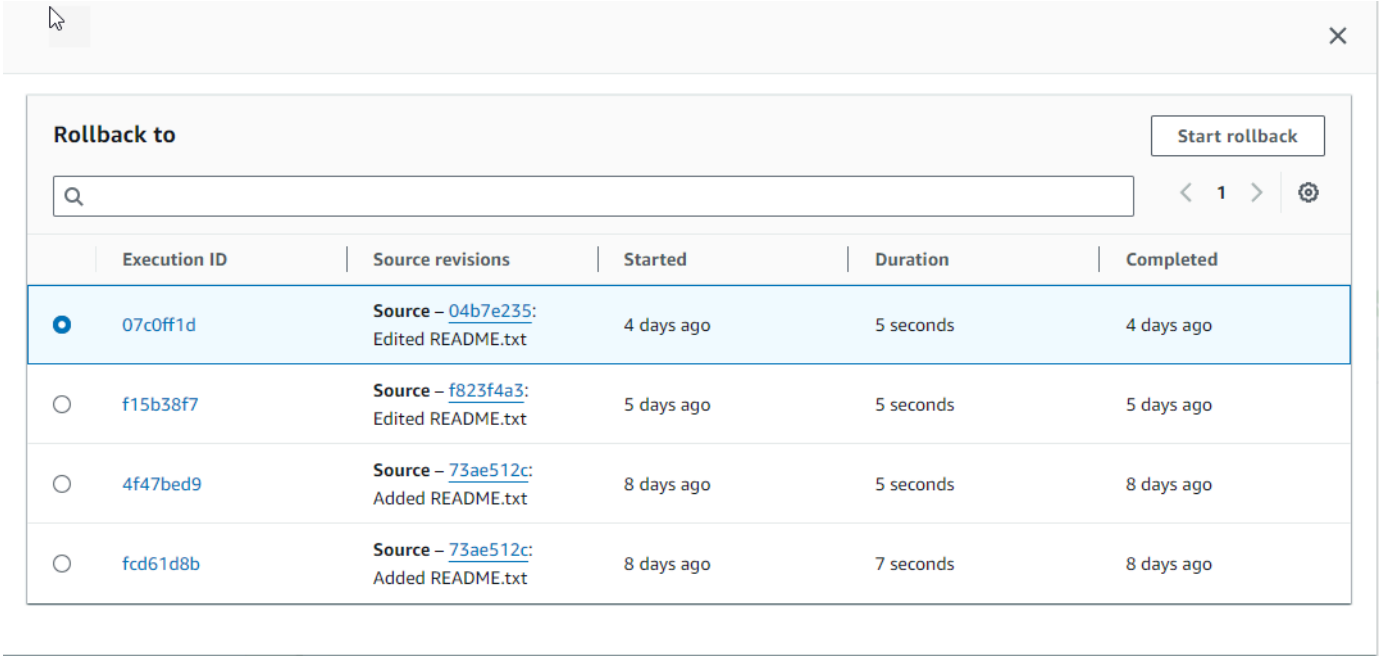
   Die Namen und der Status aller mit Ihrem AWS Konto verknüpften Pipelines werden angezeigt.

2. Wählen Sie unter Name den Namen der Pipeline mit der Phase aus, für die ein Rollback ausgeführt werden soll.

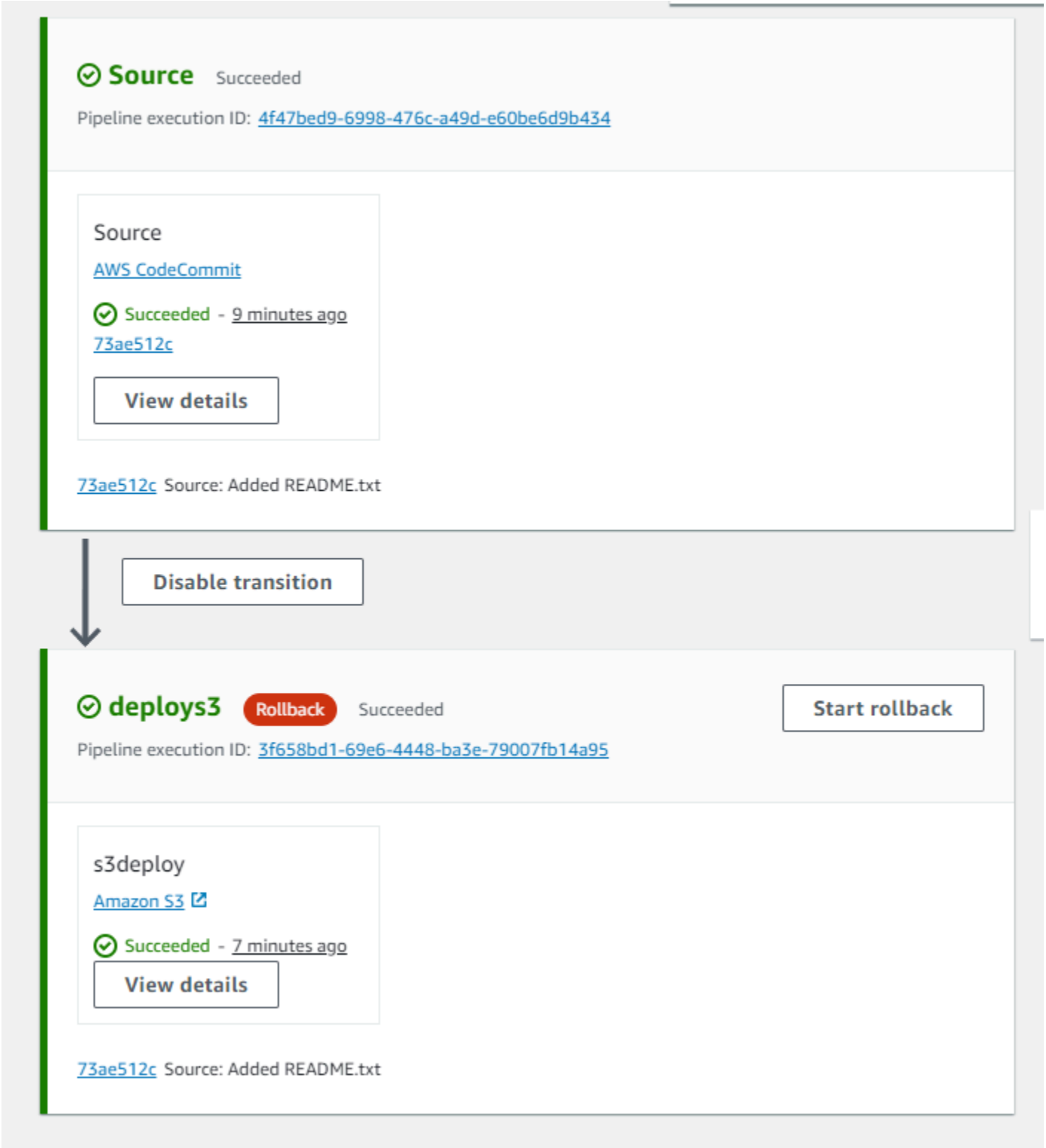


3. Wählen Sie auf der Bühne die Option Rollback starten aus. Das Fenster Zurück zur Seite wird angezeigt.
4. Wählen Sie die Zielausführung aus, zu der Sie die Phase zurücksetzen möchten.

> **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.



Führen Sie ein manuelles Rollback einer Phase durch (CLI)

Verwenden Sie den `rollback-stage` Befehl AWS CLI , um ein manuelles Rollback einer Stufe durchzuführen.

Sie können eine Stufe auch manuell rückgängig machen, wie unter beschriebenFührt eine Stufe manuell zurück.

> **Note**
>
> Die Liste der verfügbaren Ziel-Pipeline-Ausführungen umfasst alle Ausführungen in der aktuellen Pipeline-Version, die am 1. Februar 2024 beginnt.

Um eine Phase manuell rückgängig zu machen (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 Ausführungs-ID für die Zielpipeline abzurufen, verwenden Sie den Befehl list-pipeline-executions 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 listet die Ausgabe Pipelineausführungen für die benannte Pipeline MyFirstPipeline und für erfolgreiche Ausführungen in der genannten Phase auf. `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 Ziel-Ausführungs-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 Stufe und die Zielausführung an, zu der Sie ein Rollback durchführen möchten. Um beispielsweise einen Schritt namens Deploy für eine Pipeline mit dem Namen 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 Rollback-Ausführung zurück. Dies ist eine separate ID, die die Quellrevisionen und Parameter der angegebenen Zielausführung verwendet.

Konfigurieren Sie eine Phase für das automatische Rollback

Sie können Stufen in einer Pipeline so konfigurieren, dass sie bei einem Ausfall automatisch rückgängig gemacht werden. Wenn die Phase fehlschlägt, wird die Phase auf die letzte erfolgreiche Ausführung zurückgesetzt. Die Pipeline kann nur dann zu einer früheren Ausführung zurückkehren, wenn die vorherige Ausführung in der aktuellen Pipeline-Strukturversion gestartet wurde. Da die automatische Rollback-Konfiguration Teil der Pipeline-Definition ist, erfolgt für Ihre Pipeline-Phase ein automatisches Rollback erst, wenn in der Pipeline-Phase eine erfolgreiche Pipeline-Ausführung erfolgt ist.

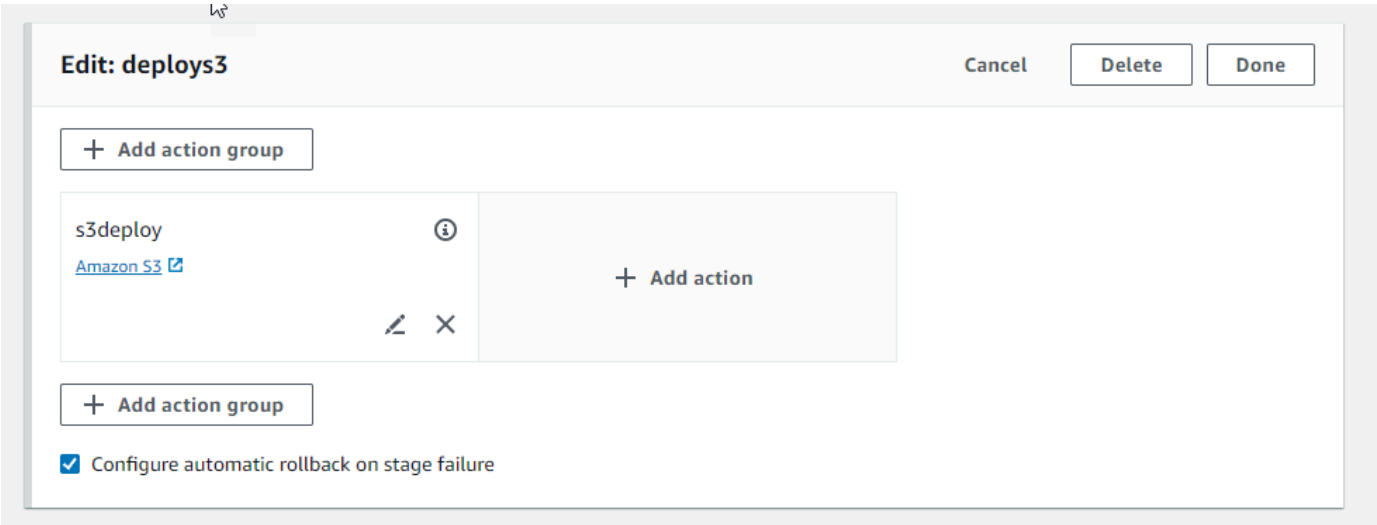
Konfigurieren Sie eine Phase für das automatische Rollback (Konsole)

Sie können eine Phase auf eine angegebene vorherige erfolgreiche Ausführung zurücksetzen. Weitere Informationen finden Sie RollbackStage im CodePipeline API-Handbuch.

Konfigurieren Sie eine Stufe für das automatische Rollback (Konsole)

1. Melden Sie sich bei der an AWS-Managementkonsole 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 Automatisierte Stufenkonfiguration: und wählen Sie dann Automatisches Rollback bei Stufenausfall konfigurieren aus. Speichern Sie die Änderungen an Ihrer Pipeline.



Konfigurieren Sie eine Phase für das automatische Rollback (CLI)

Um eine fehlgeschlagene Phase so zu konfigurieren, dass sie automatisch zur letzten erfolgreichen Ausführung zurückkehrt, verwenden Sie die Befehle, um eine Pipeline zu erstellen oder zu aktualisieren, wie in Eine Pipeline, Phasen und Aktionen erstellen und Eine Pipeline bearbeiten in CodePipeline beschrieben. AWS CLI

- Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie die, AWS CLI um den `update-pipeline` Befehl auszuführen. Geben Sie dabei die Fehlerbedingung in der Pipeline-Struktur an. Im folgenden Beispiel wird das automatische Rollback für eine Stufe 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 Stagen-Rollback finden Sie FailureConditions in der CodePipeline API-Referenz.

Konfigurieren Sie eine Phase für das automatische Rollback (CloudFormation)

Verwenden Sie CloudFormation den `OnFailure` Parameter, um eine Stufe so zu konfigurieren, dass sie bei einem Ausfall automatisch rückgängig gemacht wird. Bei einem Ausfall wird die Stufe automatisch auf die letzte erfolgreiche Ausführung zurückgesetzt.

```
OnFailure:
     Result: ROLLBACK
```

- Aktualisieren Sie die Vorlage wie im folgenden Codeausschnitt gezeigt. Im folgenden Beispiel wird das automatische Rollback für eine Stufe 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 Stagen-Rollback finden Sie OnFailure unter `StageDeclaration` im CloudFormation Benutzerhandbuch.

Sehen Sie sich den Rollback-Status in der Ausführungsliste an

Sie können den Status und die Zielausführungs-ID für eine Rollback-Ausführung anzeigen.

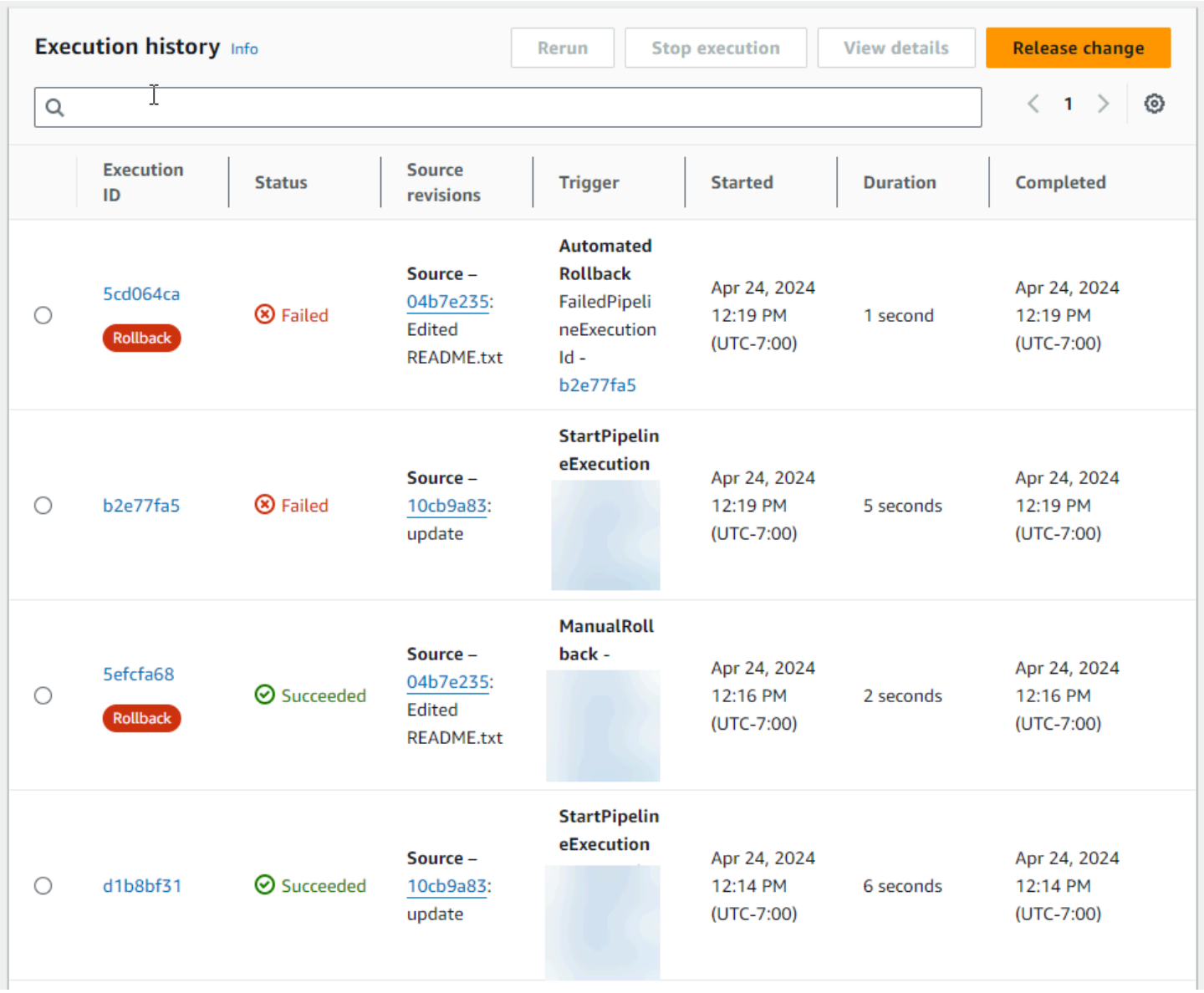
Den Rollback-Status in der Liste der Ausführungen (Konsole) anzeigen

Sie können die Konsole verwenden, um den Status und die Zielausführungs-ID für eine Rollback-Ausführung in der Ausführungsliste anzuzeigen.

Den Ausführungsstatus des Rollbacks in der Liste der Ausführungen (Konsole) anzeigen

1. Melden Sie sich bei der an AWS-Managementkonsole 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.

Zeigen Sie den Rollback-Status mit List-Pipeline-Executions (CLI) an

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 Eingabeaufforderung (Windows) und verwenden Sie die 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 der Pipeline zugeordnet sind.

Das folgende Beispiel zeigt die zurückgegebenen Daten für eine Pipeline mit dem Namen, *MyFirstPipeline* an dem 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",
```

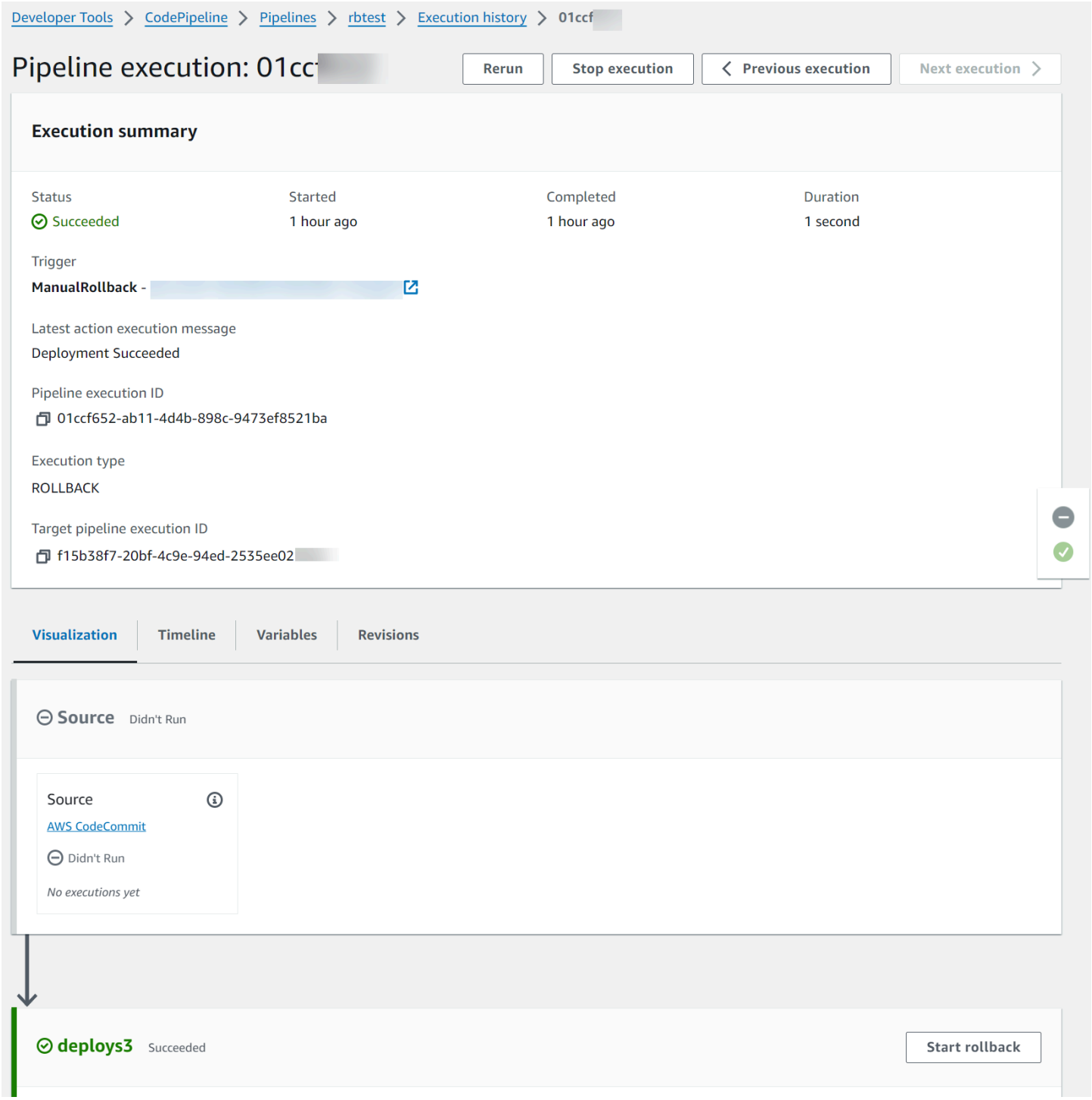
```
                    "revisionUrl": "console_URL"
                }
            ],
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
            },
            "executionMode": "SUPERSEDED"
        },
        {
            "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 Ausführungs-ID der Zielpipeline für eine Rollback-Ausführung anzuzeigen.



Zeigen Sie Rollback-Details mit **`get-Pipeline-Execution`** (CLI) an

Pipeline-Ausführungen, für die ein Rollback durchgeführt wurde, werden in der Ausgabe für das 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. Wenn Sie beispielsweise Details zu einer Pipeline mit dem Namen *MyFirstPipeline* anzeigen möchten, 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"
          }
  ```

```
    }
}
```

Zeigen Sie den Rollback-Status mit get-pipeline-state (CLI) an

Pipeline-Ausführungen, für die ein Rollback durchgeführt wurde, werden in der Ausgabe für das 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"
}
```

Bedingungen für eine Phase konfigurieren

Sie können eine Bedingung für eine Phase angeben, indem Sie beispielsweise im Pipeline-Lauf nach einer bestimmten Variablen suchen, und dann ein Ergebnis für die Bedingung eingeben, z. B. das Überspringen der Phase oder das Nichtbestehen der Phase. Eine Pipeline kann eingerichtet werden, um während des Durchlaufs nach Phasenbedingungen zu suchen. Dabei geben Sie Prüfungen für eine Phase an und geben dann an, wie die Phase fortgesetzt werden soll, wenn bestimmte Bedingungen erfüllt sind. Bedingungen enthalten eine oder mehrere Regeln, die in einer Liste von Regeln in verfügbar sind CodePipeline. Wenn alle Regeln in einer Bedingung erfolgreich sind, ist die Bedingung erfüllt. Sie können Bedingungen so konfigurieren, dass das angegebene Ergebnis aktiviert wird, wenn die Kriterien nicht erfüllt sind.

Jede Bedingung hat einen Regelsatz, bei dem es sich um einen geordneten Satz von Regeln handelt, die zusammen ausgewertet werden. Wenn also eine Regel in der Bedingung fehlschlägt, schlägt die Bedingung fehl. Sie können die Regelbedingungen zur Laufzeit der Pipeline überschreiben.

Bedingungen werden für bestimmte Arten von Ausdrücken verwendet, und für jeden gibt es spezifische Optionen für Ergebnisse, die wie folgt verfügbar sind:

- Eingabe — Die Bedingungen für die Durchführung von Prüfungen, die, wenn sie erfüllt sind, den Eintritt in eine Phase ermöglichen. Die Regeln sind mit den folgenden Ergebnisoptionen verknüpft: Fehlgeschlagen oder Überspringen
- Bei Misserfolg — Die Bedingungen, unter denen Prüfungen für die Phase durchgeführt werden, wenn sie fehlschlägt. Regeln verwenden die folgende Ergebnisoption: Rollback
- Bei Erfolg — Die Bedingungen, unter denen überprüft wird, ob die Phase erfolgreich abgeschlossen ist. Regeln werden mit den folgenden Ergebnisoptionen aktiviert: Rollback oder Fail

Bedingungen werden durch eine Reihe von Regeln für jeden Bedingungstyp unterstützt.

Für jeden Bedingungstyp gibt es spezifische Aktionen, die durch die Bedingung eingerichtet werden. Die Aktion ist das Ergebnis der erfolgreichen oder fehlgeschlagenen Zustandsprüfung. Beispiel: Die Bedingung für den Eintritt (Eingangsbedingung) trifft auf einen Alarm (Regel), dann ist die Prüfung erfolgreich und das Ergebnis (Aktion) ist, dass der Stufeneintrag blockiert ist.

Sie können auch die AWS CodePipeline Konsole oder die verwenden AWS CLI , um eine Stufe oder Aktionen in einer Phase manuell rückgängig zu machen oder erneut zu versuchen. Siehe Bedingungen für eine Phase konfigurieren.

Themen

- Anwendungsfälle für Bühnenbedingungen
- Überlegungen zu Ergebnissen, die für die Bedingungen der Phase konfiguriert wurden
- Überlegungen zu Regeln, die für die Bedingungen der Phase konfiguriert sind
- Zugangsbedingungen erstellen
- Bedingungen bei Ausfall erstellen
- Bedingungen bei Erfolg erstellen
- Stage-Bedingungen löschen
- Übergeordnete Stufenbedingungen

Anwendungsfälle für Bühnenbedingungen

Stufenbedingungen haben mehrere Anwendungsfälle für die Einrichtung der Freigabe- und Wechselsicherheit in Rohrleitungen. Im Folgenden finden Sie Beispiele für Anwendungsfälle für Stufenbedingungen.

- Verwenden Sie eine Eingabebedingung, um eine Bedingung zu definieren, die den CloudWatch Alarmstatus überprüft. Eine Änderung wird dann blockiert, wenn sich die Produktionsumgebung nicht in einem fehlerfreien Zustand befindet.
- Verwenden Sie eine Entry-Bedingung mit einer Wartezeit von 60, um eine Bedingung zu definieren, die bewertet wird, wenn alle Aktionen in einer Phase erfolgreich abgeschlossen wurden. Machen Sie die Änderungen dann rückgängig, wenn ein CloudWatch Alarm innerhalb von 60 Minuten in den ALARM-Zustand übergeht.
- Verwenden Sie die Bedingung „Bei Erfolg“, um eine Bedingung zu definieren, sodass die Regel nach erfolgreichem Abschluss der Phase prüft, ob die aktuelle Uhrzeit im Bereitstellungsfenster liegt, und dann auslöst, wenn die Regel erfolgreich ist.

Überlegungen zu Ergebnissen, die für die Bedingungen der Phase konfiguriert wurden

Die folgenden Überlegungen zu den Stufenbedingungen lauten wie folgt:

- Sie können die automatische Wiederholung von Phasen nicht mit OnFailure-Bedingungen verwenden.
- Wenn Sie eine Bedingung mit einem Rollback-Ergebnis konfigurieren, kann die Phase nur dann zu einer früheren Ausführung zurückkehren, wenn sie in der aktuellen Pipeline-Strukturversion verfügbar ist.
- Wenn Sie eine Bedingung mit einem Rollback-Ergebnis konfigurieren, können Sie nicht zu einer Zielausführungs-ID zurückkehren, bei der es sich um einen Rollback-Ausführungstyp handelt.
- Bei Entry-Bedingungen, die das Skip-Ergebnis verwenden, um die Phase zu überspringen, falls die Bedingung fehlschlägt, werden nur die `LambdaInvoke` `VariableCheck` UND-Regeln unterstützt.
- Sie können keinen manuellen Wiederholungsversuch für eine Phase mit dem Status Übersprungen durchführen.
- Sie können keinen manuellen Rollback zu einer Phase mit dem Status Übersprungen durchführen.
- Sie können eine Bedingung nicht überschreiben, wenn die Bedingung mit dem Ergebnis „Überspringen“ konfiguriert ist.
- Mit Ausnahme von Skip-Ergebnissen können Sie eine Stufenbedingung überschreiben, wenn Sie eine Pipeline-Ausführung starten. Bei einer Stufenbedingung, bei der eine Überschreibung ausgelöst wird, erfolgt die Ausführung wie in der folgenden Tabelle beschrieben.
-

Typ	Konfiguriertes Ergebnis bei Ausfall einer Bedingung	Status der Phase	Verhalten überschreiben
Eintrag	Fehler	In-progress	Die Phase geht weiter.
Eintrag	Überspringen	Übersprungen	Nicht zutreffend.
OnFailure	Rollback	Fehlgeschlagen	Die Phase ist gescheitert.

Typ	Konfiguriertes Ergebnis bei Ausfall einer Bedingung	Status der Phase	Verhalten überschreiben
OnSuccess	Rollback	Erfolgreich	Die Phase geht weiter.
OnSuccess	Fehler	Fehlgeschlagen	Die Phase geht weiter.

Überlegungen zu Regeln, die für die Bedingungen der Phase konfiguriert sind

Die folgenden Überlegungen zu den verfügbaren Regeln für Stufenbedingungen lauten wie folgt:

- Für die `LambdaInvoke` Regel müssen Sie zuerst die Lambda-Funktion so konfigurieren, dass sie in der Regel verwendet wird. Halten Sie den ARN der Lambda-Funktion bereit, ihn bereitzustellen, wenn Sie die Regel konfigurieren.
- Für die `CloudWatchAlarm` Regel müssen Sie zuerst das Ereignis CloudWatch Ereignisse so konfigurieren, dass es in der Regel verwendet werden soll. Halten Sie den Event-ARN bereit, damit Sie ihn angeben können, wenn Sie die Regel konfigurieren.

Zugangsbedingungen erstellen

Sie können die Eingabebedingungen für eine Phase mithilfe der Konsole oder der CLI konfigurieren. Sie konfigurieren die entsprechenden Regeln und Ergebnisse für jede Bedingung. Bei einem Rollback-Ergebnis kann die Pipeline nur dann zu einer früheren Ausführung zurückkehren, wenn die vorherige Ausführung in der aktuellen Pipeline-Strukturversion gestartet wurde.

Die Schritte enthalten ein Beispiel für eine Eingabebedingung, die eine Überwachungsregel verwendet.

Weitere Informationen finden Sie unter Condition und RuleExecution im CodePipeline API-Handbuch. RuleTypeId

Eingabebedingungen erstellen — Beispiel für eine CloudWatchAlarm Regel (Konsole)

Sie können die Eingabebedingungen für eine Phase zusammen mit den Regeln und Ergebnissen konfigurieren, die die Phase ausführen soll, wenn die Bedingungen erfüllt sind.

Konfigurieren Sie eine Eingabebedingung (Konsole)

1. Erfüllen Sie alle Voraussetzungen, z. B. das Erstellen der Ressource und des ARN für eine Regel, in der AWS CloudWatchAlarm eine Ressource bereitgestellt wird, wie z. B.
2. Melden Sie sich bei der an AWS-Managementkonsole 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.
3. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten.
4. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
5. Wählen Sie auf der Seite „Bearbeiten“ für die Aktion, die Sie bearbeiten möchten, die Option Phase bearbeiten aus.
6. Wählen Sie Eingabebedingung hinzufügen aus. Die Karte mit den Bedingungen vor dem Eintritt in die Phase wird angezeigt. Für diese Bedingung ist die Option Fehlgeschlagen verfügbar.
7. Wählen Sie Regel hinzufügen aus, und gehen Sie dann wie folgt vor.

   a. Geben Sie im Feld Regelname einen Namen für Ihre Regel ein. Geben Sie für dieses Beispiel `MyAlarmRule` ein.
   b. Wählen Sie unter Regelanbieter den vorkonfigurierten Regelanbieter aus, der zu Ihrer Bedingung hinzugefügt werden soll. Wählen Sie für dieses Beispiel aus AWS CloudWatchAlarm, und führen Sie dann die folgenden Schritte aus.
   c. Wählen Sie unter Region die Region für Ihre Erkrankung aus, oder belassen Sie die Standardeinstellung.
   d. Wählen Sie unter Alarmname die CloudWatch Ressource aus, die für die Regel verwendet werden soll. Sie müssen die Ressource bereits in Ihrem Konto erstellt haben.
   e. (Optional) Geben Sie im Feld Wartezeit die Wartezeit ein, wenn CodePipeline sich der Alarm bei der ersten Auswertung im ALARM-Zustand befindet. Wenn der Alarm bei der ersten Überprüfung der Regel den Status OK hat, hat die Regel sofort Erfolg.

f. (Optional) Geben Sie alle spezifischen Alarmzustände ein, die überwacht werden sollen, und geben Sie gegebenenfalls den Rollen-ARN ein.

g. Wenn Sie mit der Bearbeitung der Phase fertig sind, wählen Sie Fertig aus. Wählen Sie auf der Pipeline-Bearbeitungsseite Speichern aus.

8. Sehen Sie sich nach dem Lauf das Ergebnis an.

Eingabebedingungen mit Ergebnis und **`VariableCheck`** Regel überspringen (Konsole) erstellen

Sie können die Eintrittsbedingungen für eine Phase so konfigurieren, dass die Phase übersprungen wird, wenn die Einstiegsbedingung nicht erfüllt ist. Wenn die Bedingung nicht erfüllt ist, wird das Ergebnis aktiviert und die Phase wird übersprungen. Wenn eine Phase übersprungen wird, lautet der Status Übersprungen und der Aktionsstatus lautet Nicht ausgeführt. Überlegungen zu Phasenbedingungen mit Ergebnissen vom Typ „Überspringen“ finden Sie unter. Überlegungen zu Ergebnissen, die für die Bedingungen der Phase konfiguriert wurden

Im folgenden Beispiel stellt die Regel zur Variablenüberprüfung fest, dass der Wert nicht übereinstimmt, und die Erstellungsphase wird übersprungen.

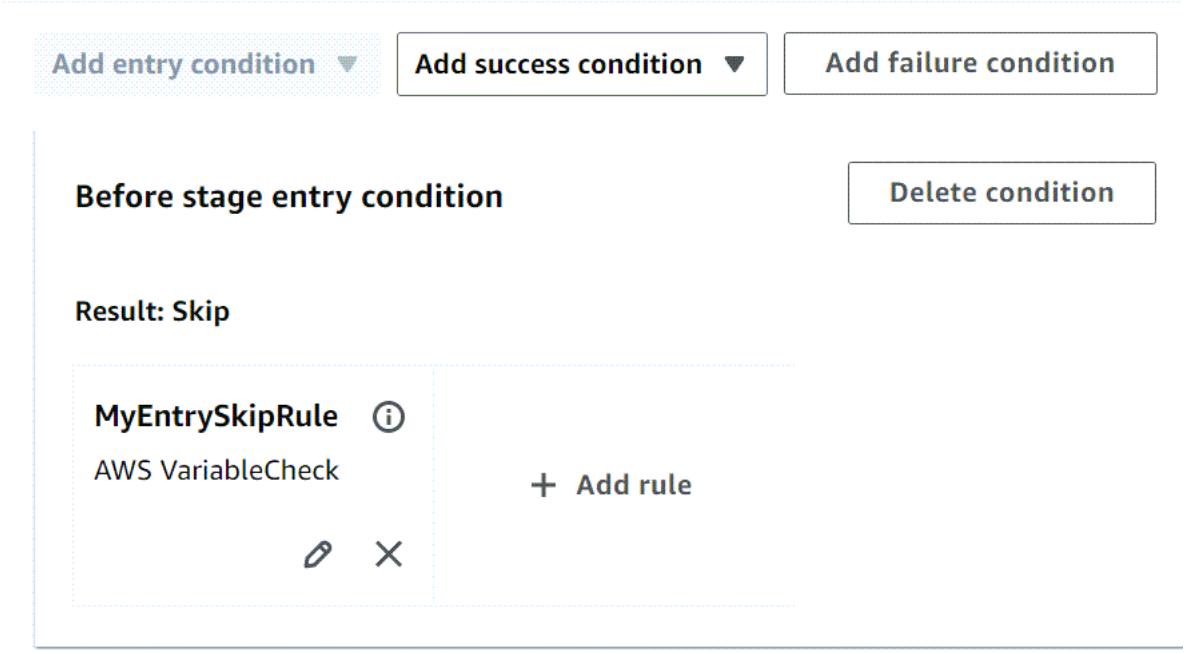
Konfigurieren Sie eine Eingabebedingung mit einer Überspringen Ergebnis (Konsole)

1. Erfüllen Sie alle Voraussetzungen, z. B. das Erstellen der Ressource und des ARN für eine Regel, in der AWS CloudWatchAlarm eine Ressource bereitgestellt wird, wie z. B.
2. Melden Sie sich bei der an AWS-Managementkonsole 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.
3. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten.
4. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
5. Wählen Sie auf der Seite „Bearbeiten“ für die Aktion, die Sie bearbeiten möchten, die Option Phase bearbeiten aus.
6. Wählen Sie Eingabebedingung hinzufügen und wählen Sie dann als Ergebnis Überspringen aus.
7. Wählen Sie Regel hinzufügen aus, und gehen Sie dann wie folgt vor.

a. Geben Sie im Feld Regelname einen Namen für Ihre Regel ein. Geben Sie für dieses Beispiel `MyAlarmRule` ein.

b. Wählen Sie unter Regelanbieter den vorkonfigurierten Regelanbieter aus, der zu Ihrer Bedingung hinzugefügt werden soll. Wählen Sie für dieses Beispiel aus VariableCheck, und führen Sie dann die folgenden Schritte aus.

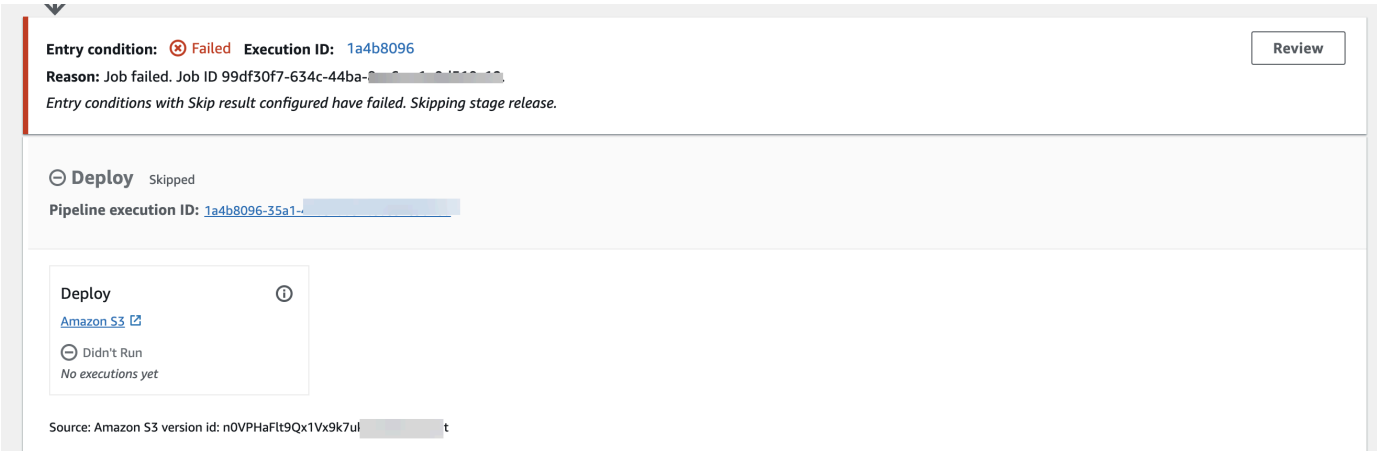


c. Wählen Sie unter Region die Region für Ihre Erkrankung aus oder belassen Sie die Standardeinstellung.

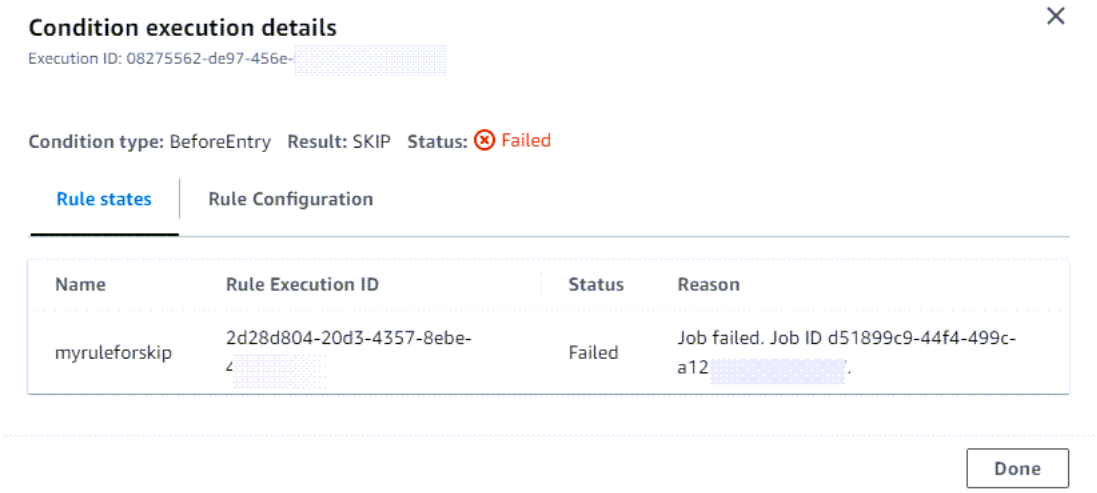
d. Wählen Sie unter Variable die Variable aus, mit der Sie vergleichen möchten, z. B. `#{SourceVariables.FullRepositoryName}` für eine Pipeline, die über eine Quellaktion GitHub (per GitHub App) verfügt. Geben Sie den Repository-Namen ein und wählen Sie den Operator aus, z. B. Equals.

e. Wenn Sie mit der Bearbeitung der Stufe fertig sind, wählen Sie Fertig aus. Wählen Sie auf der Pipeline-Bearbeitungsseite Speichern aus.

8. Sehen Sie sich nach dem Lauf das Ergebnis an.



9. Um die Details zu überprüfen, wählen Sie Überprüfen. Das Detail im folgenden Beispiel zeigt, dass das konfigurierte Ergebnis für die Bedingung „Überspringen“ lautet. Dieses Ergebnis kann nicht außer Kraft gesetzt werden. Der Regelstatus lautet Fehlgeschlagen, da die Bedingung nicht erfüllt ist.



Eingabebedingungen erstellen (CLI)

Um die zur Konfiguration einer Eingabebedingung AWS CLI zu verwenden, verwenden Sie die Befehle, um eine Pipeline zu erstellen oder zu aktualisieren, wie in Eine Pipeline, Phasen und Aktionen erstellen und beschriebenEine Pipeline bearbeiten in CodePipeline.

Konfigurieren Sie die Bedingung und die Regel oder Regeln (CLI)

- Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) und verwenden Sie die, AWS CLI um den `update-pipeline` Befehl auszuführen. Geben Sie

dabei die Fehlerbedingung in der Pipeline-Struktur an. Im folgenden Beispiel wird eine Entry-Bedingung für eine Stufe mit dem Namen `Deploy` konfiguriert:

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "S3",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "BucketName": "MyBucket",
                "Extract": "false",
                "ObjectKey": "object.xml"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1",
            "namespace": "DeployVariables"
        }
    ],
    "beforeEntry": {
        "conditions": [
            {
                "result": "FAIL",
                "rules": [
                    {
                        "name": "MyAlarmRule",
                        "ruleTypeId": {
                            "category": "Rule",
                            "owner": "AWS",
                            "provider": "CloudWatchAlarm",
                            "version": "1"
                        },
```

```
                        "configuration": {
                            "AlarmName": "CWAlarm",
                            "WaitTime": "1"
                        },
                        "inputArtifacts": [],
                        "region": "us-east-1"
                    }
                ]
            }
        ]
    }
}
```

Weitere Informationen zur Konfiguration der Erfolgsbedingungen für das Phase-Rollback finden Sie SuccessConditions in der CodePipeline API-Referenz.

Zugangsbedingungen (CFN) erstellen

Verwenden Sie den `beforeEntry` Parameter, CloudFormation um eine Eingabebedingung zu konfigurieren. Bei der Eingabe führt die Stufe die Regel aus und führt das Ergebnis aus.

```
beforeEntry:
     Result: FAIL
```

- Aktualisieren Sie die Vorlage wie im folgenden Codeausschnitt gezeigt. Im folgenden Beispiel wird eine Entry-Bedingung mit einer Regel namens konfiguriert: `MyMonitorRule`

```
Name: Deploy
Actions:
- Name: Deploy
  ActionTypeId:
    Category: Deploy
    Owner: AWS
    Provider: S3
    Version: '1'
  RunOrder: 1
  Configuration:
    BucketName: MyBucket
    Extract: 'false'
    ObjectKey: object.xml
  OutputArtifacts: []
```

```
  InputArtifacts:
  - Name: SourceArtifact
  Region: us-east-1
  Namespace: DeployVariables
BeforeEntry:
  Conditions:
  - Result: FAIL
    Rules:
    - Name: MyMonitorRule
      RuleTypeId:
        Category: Rule
        Owner: AWS
        Provider: CloudWatchAlarm
        Version: '1'
      Configuration:
        AlarmName: CWAlarm
        WaitTime: '1'
      InputArtifacts: []
      Region: us-east-1
```

Weitere Informationen zur Konfiguration von BeforeEntry-Bedingungen finden Sie BeforeEntryConditions unter AWS::CodePipeline: :Pipeline `StageDeclaration` im CloudFormation Benutzerhandbuch.

Bedingungen bei Ausfall erstellen

Sie können die On-Failure-Bedingungen für eine Phase mithilfe der Konsole oder der CLI konfigurieren. Sie konfigurieren die entsprechenden Regeln und Ergebnisse für jede Bedingung. Bei einem Rollback-Ergebnis kann die Pipeline nur dann zu einer früheren Ausführung zurückkehren, wenn die vorherige Ausführung in der aktuellen Pipeline-Strukturversion gestartet wurde.

Bedingungen bei Ausfall erstellen (Konsole)

Sie können die Bedingungen für den Fall eines Fehlers für eine Phase zusammen mit den Regeln und Ergebnissen konfigurieren, die die Phase ausführen soll, wenn die Bedingungen erfüllt sind.

Konfigurieren Sie die Bedingung „Bei Ausfall“ (Konsole)

1. Erfüllen Sie alle Voraussetzungen, z. B. das Erstellen der Ressource und des ARN für eine Regel, in der eine Ressource bereitgestellt wird, wie z. B. die LambdaInvoke Regel.

2. Melden Sie sich bei der an AWS-Managementkonsole 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.

3. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten.
4. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
5. Wählen Sie auf der Seite „Bearbeiten“ für die Aktion, die Sie bearbeiten möchten, die Option Phase bearbeiten aus.
6. Wählen Sie Fehlerbedingung hinzufügen aus. Die Karte mit den Fehlerbedingungen wird mit der Rollback-Option angezeigt, die für diesen Zustand verfügbar ist.
7. Wählen Sie Regel hinzufügen aus, und gehen Sie dann wie folgt vor.

   a. Geben Sie im Feld Regelname einen Namen für Ihre Regel ein. Geben Sie für dieses Beispiel `MyLambdaRule` ein.

   b. Wählen Sie unter Regelanbieter den vorkonfigurierten Regelanbieter aus, der zu Ihrer Bedingung hinzugefügt werden soll. Wählen Sie für dieses Beispiel aus AWS LambdaInvoke, und führen Sie dann die folgenden Schritte aus.

   c. Wählen Sie unter Region die Region für Ihre Erkrankung aus, oder belassen Sie die Standardeinstellung.

   d. Wählen Sie unter Eingabe-Artefakte das Quell-Artefakt aus.

   e. Wählen Sie unter Funktionsname die Lambda-Ressource aus, die für die Regel verwendet werden soll. Sie müssen die Ressource bereits in Ihrem Konto erstellt haben.

   f. (Optional) Geben Sie im Feld Benutzerparameter alle Paare ein, die Parameter für zusätzliche Konfigurationen darstellen.

   g. (Optional) Geben Sie im Feld Role Arn den Rollen-ARN ein, sofern konfiguriert.

   h. (Optional) Geben Sie im Feld Timeout in Minuten die Zeit in Minuten ein, die die Regel vor dem Timeout warten soll.

   i. Wenn Sie mit der Bearbeitung der Phase fertig sind, wählen Sie Fertig aus. Wählen Sie auf der Pipeline-Bearbeitungsseite Speichern aus.

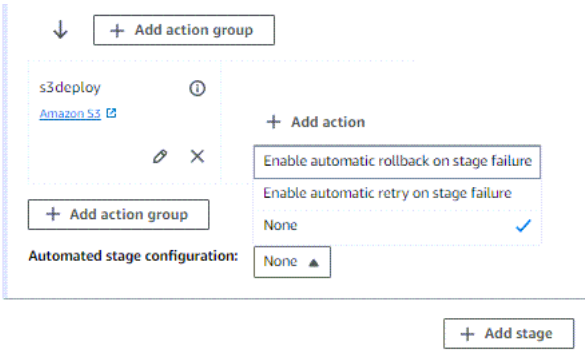
Erstellen von OnFailure-Bedingungen mit einem Wiederholungsergebnis (Beispiel) (Konsole)

Sie können die OnFailure-Bedingungen für eine Stufe so konfigurieren, dass die Phase erneut versucht wird, wenn die Eingabebedingung nicht erfüllt ist. Als Teil dieses Ergebnisses konfigurieren Sie den Wiederholungsmodus und geben an, ob die fehlgeschlagenen Aktionen oder die fehlgeschlagene Phase erneut versucht werden sollen.

Konfigurieren Sie eine OnFailure-Bedingung mit einem Wiederholungsergebnis (Konsole)

1. Erfüllen Sie alle Voraussetzungen, z. B. das Erstellen der Ressource und des ARN für eine Regel, in der eine Ressource bereitgestellt wird, z. B. AWS CloudWatchAlarm
2. Melden Sie sich bei der an AWS-Managementkonsole 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.
3. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten.
4. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
5. Wählen Sie auf der Seite „Bearbeiten“ für die Aktion, die Sie bearbeiten möchten, die Option Phase bearbeiten aus.
6. Wählen Sie unten in der Phase unter Automatisierte Stufenkonfiguration: die Option Automatische Wiederholung bei Stufenausfall aktivieren aus. Wählen Sie im Modus „Wiederholen“ entweder „Fehlgeschlagene Phase wiederholen“ oder „Fehlgeschlagene Aktionen wiederholen“.



7. Wählen Sie, ob Sie eine OnFailure-Bedingung hinzufügen möchten, und wählen Sie dann Regel hinzufügen und geben Sie eine Regel für die Bedingung ein.

a. Geben Sie im Feld Regelname einen Namen für Ihre Regel ein. Geben Sie für dieses Beispiel `MyAlarmRule` ein.

b. Wählen Sie unter Regelanbieter den vorkonfigurierten Regelanbieter aus, der zu Ihrer Bedingung hinzugefügt werden soll. Wählen Sie für dieses Beispiel aus CloudWatchAlarm, und führen Sie dann die folgenden Schritte aus.

c. Wählen Sie unter Region die Region für Ihre Erkrankung aus, oder belassen Sie die Standardeinstellung.

d. Wählen Sie unter Alarmname die konfigurierte Ressource für die Warnung aus.

e. Wenn Sie mit der Bearbeitung der Phase fertig sind, wählen Sie Fertig aus. Wählen Sie auf der Pipeline-Bearbeitungsseite Speichern aus.

8. Sehen Sie sich nach dem Lauf das Ergebnis an.

Bedingungen bei Ausfall erstellen (CLI)

Um die Bedingung „Bei Ausfall“ AWS CLI zu konfigurieren, verwenden Sie die Befehle, um eine Pipeline zu erstellen oder zu aktualisieren, wie unter Eine Pipeline, Phasen und Aktionen erstellen und beschriebenEine Pipeline bearbeiten in CodePipeline.

Konfigurieren Sie die Bedingung und die Regel oder Regeln (CLI)

- Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) und verwenden Sie die, AWS CLI um den `update-pipeline` Befehl auszuführen. Geben Sie dabei die Fehlerbedingung in der Pipeline-Struktur an. Im folgenden Beispiel wird die Bedingung On Failure für eine Stufe mit dem Namen `Deploy` konfiguriert:

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "S3",
                "version": "1"
            },
            "runOrder": 1,
```

```
            "configuration": {
                "BucketName": "MyBucket",
                "Extract": "false",
                "ObjectKey": "object.xml"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1",
            "namespace": "DeployVariables"
        }
    ],
    "onFailure": {
        "conditions": [
            {
                "result": "ROLLBACK",
                "rules": [
                    {
                        "name": "MyLambdaRule",
                        "ruleTypeId": {
                            "category": "Rule",
                            "owner": "AWS",
                            "provider": "LambdaInvoke",
                            "version": "1"
                        },
                        "configuration": {
                            "FunctionName": "my-function"
                        },
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "region": "us-east-1"
                    }
                ]
            }
        ]
    }
}
```

Weitere Informationen zur Konfiguration von Fehlerbedingungen finden Sie FailureConditions in der CodePipeline API-Referenz.

Bedingungen bei Ausfall erstellen (CFN)

Verwenden Sie CloudFormation den `OnFailure` Parameter, um eine On-Failure-Bedingung zu konfigurieren. Bei einem Fehler führt die Stufe die Regel aus und führt das Ergebnis aus.

```
OnFailure:
     Result: ROLLBACK
```

- Aktualisieren Sie die Vorlage wie im folgenden Codeausschnitt gezeigt. Im folgenden Beispiel wird eine OnFailure Bedingung mit einer Regel namens konfiguriert: `MyMonitorRule`

```
name: Deploy
actions:
- name: Deploy
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: S3
    version: '1'
  runOrder: 1
  configuration:
    BucketName: MyBucket
    Extract: 'false'
    ObjectKey: object.xml
  outputArtifacts: []
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
  namespace: DeployVariables
OnFailure:
  conditions:
  - result: ROLLBACK
    rules:
    - name: MyMonitorRule
      ruleTypeId:
        category: Rule
        owner: AWS
```

```
        provider: CloudWatchAlarm
        version: '1'
      configuration:
        AlarmName: AlarmOnHelloWorldInvocation
        AlarmStates: ALARM
        WaitTime: '1'
      inputArtifacts: []
      region: us-east-1
```

Weitere Informationen zur Konfiguration von Fehlerbedingungen finden Sie OnFailure unter `StageDeclaration` im CloudFormation Benutzerhandbuch.

Bedingungen bei Erfolg erstellen

Sie können die On-Success-Bedingungen für eine Phase mithilfe der Konsole oder der CLI konfigurieren. Sie konfigurieren die entsprechenden Regeln und Ergebnisse für jede Bedingung. Bei einem Rollback-Ergebnis kann die Pipeline nur dann zu einer früheren Ausführung zurückkehren, wenn die vorherige Ausführung in der aktuellen Pipeline-Strukturversion gestartet wurde.

Die Schritte enthalten ein Beispiel für die Bedingung „Bei Erfolg“, die eine Regel für das Bereitstellungsfenster verwendet.

Weitere Informationen finden Sie unter Condition und RuleExecution im CodePipeline API-Handbuch. RuleTypeId

Bedingungen bei Erfolg erstellen (Konsole)

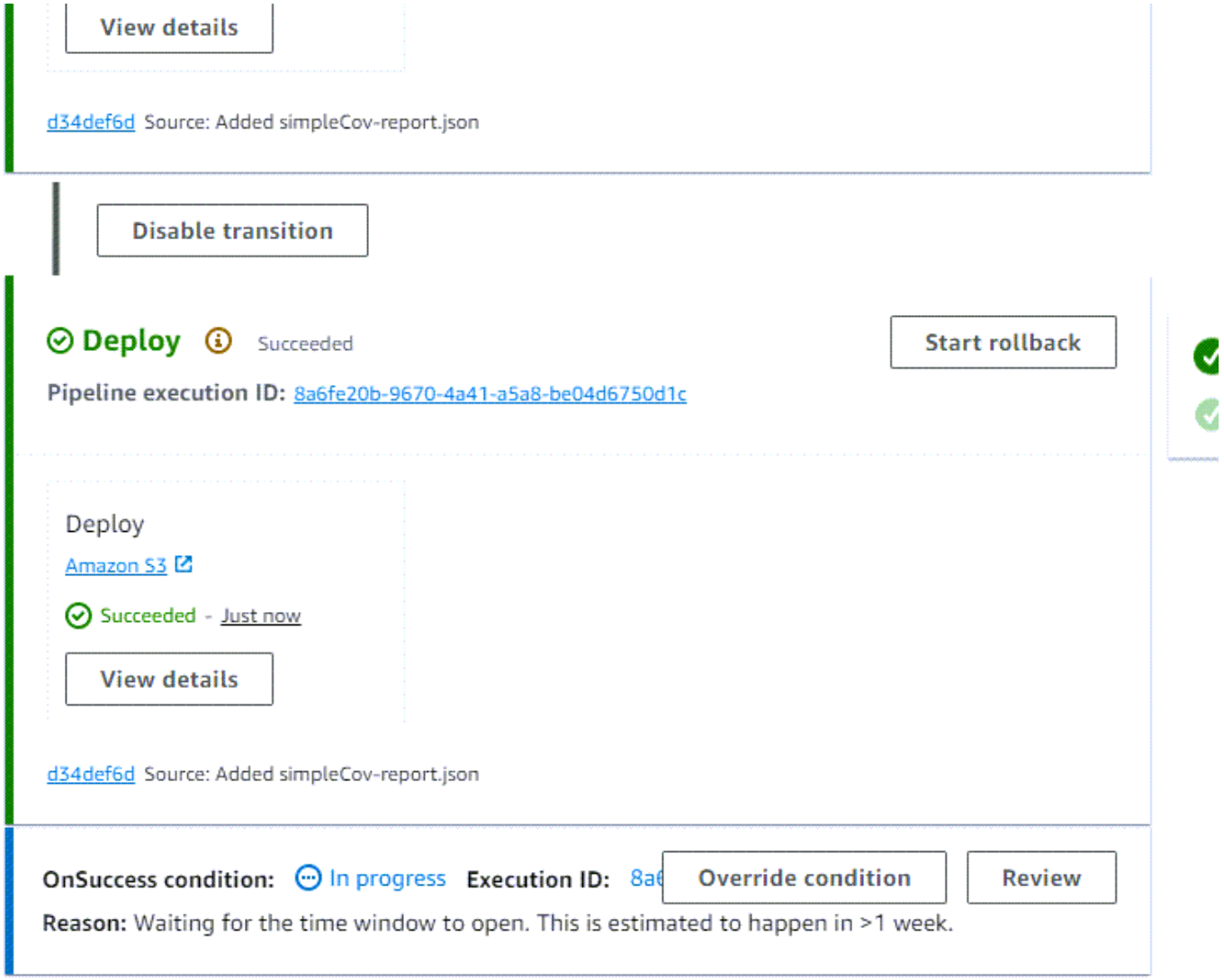
Sie können „Bei Erfolg“ -Bedingungen für eine Phase zusammen mit den Regeln und Ergebnissen konfigurieren, die die Phase ausführen soll, wenn die Bedingungen erfüllt sind.

Konfigurieren Sie eine On-Success-Bedingung (Konsole)

1. Erfüllen Sie alle Voraussetzungen, z. B. das Erstellen der Ressource und des ARN für eine Regel, in der AWS LambdaRule eine Ressource bereitgestellt wird, wie z. B.
2. Melden Sie sich bei der an AWS-Managementkonsole 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.

3. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten.
4. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
5. Wählen Sie auf der Seite „Bearbeiten“ für die Aktion, die Sie bearbeiten möchten, die Option Phase bearbeiten aus.
6. Wählen Sie Erfolgsbedingung hinzufügen aus. Die Karte mit den Erfolgsbedingungen auf der Bühne wird angezeigt. Wählen Sie die Option Rollback oder Fail, die als verfügbare Ergebnisse für diesen Bedingungstyp angezeigt werden.
7. Wählen Sie Regel hinzufügen aus, und führen Sie dann die folgenden Schritte aus.
   a. Geben Sie im Feld Regelname einen Namen für Ihre Bedingung ein. Geben Sie für dieses Beispiel `MyDeploymentRule` ein.
   b. Wählen Sie unter Regelanbieter die vorkonfigurierte Regel aus, die Sie Ihrer Bedingung hinzufügen möchten. Wählen Sie für dieses Beispiel aus AWS DeploymentWindow, und führen Sie dann die folgenden Schritte aus.
   c. Wählen Sie unter Region die Region für Ihre Erkrankung aus, oder belassen Sie die Standardeinstellung.
   d. Geben Sie im Feld Cron den Cron-Ausdruck für das Bereitstellungsfenster ein. Der Cron-Ausdruck definiert die Tage und Uhrzeiten, an denen die Bereitstellung zulässig sein sollte. Referenzinformationen zu Cron-Ausdrücken finden Sie unter Verwenden von Cron- und Rate-Ausdrücken zum Planen von Regeln.
   e. (Optional) Geben Sie TimeZone unter die Zeitzone für das Bereitstellungsfenster ein.
8. Sehen Sie sich nach dem Lauf das Ergebnis an.



Bedingungen bei Erfolg erstellen (CLI)

Um die Bedingung On Success AWS CLI zu konfigurieren, verwenden Sie die Befehle, um eine Pipeline zu erstellen oder zu aktualisieren, wie unter Eine Pipeline, Phasen und Aktionen erstellen und beschriebenEine Pipeline bearbeiten in CodePipeline.

Konfigurieren Sie die Bedingung und die Regel oder Regeln (CLI)

- Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) und verwenden Sie die, AWS CLI um den `update-pipeline` Befehl auszuführen. Geben Sie dabei die Fehlerbedingung in der Pipeline-Struktur an. Im folgenden Beispiel wird eine On-

Success-Bedingung für eine Stufe mit dem Namen konfiguriert`Deploy`, in der die Regel benannt `MyDeploymentRule` ist:

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "S3",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "BucketName": "MyBucket",
                "Extract": "false",
                "ObjectKey": "object.xml"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1",
            "namespace": "DeployVariables"
        }
    ],
    "onSuccess": {
       "conditions": [
            {
                "result": "FAIL",
                "rules": [
                    {
                        "name": "MyAlarmRule",
                        "ruleTypeId": {
                            "category": "Rule",
                            "owner": "AWS",
                            "provider": "CloudWatchAlarm",
                            "version": "1"
                        },
```

```
                        "configuration": {
                            "AlarmName": "CWAlarm",
                            "WaitTime": "1"
                        },
                        "inputArtifacts": [],
                        "region": "us-east-1"
                    }
                ]
            }
        ]
    }
}
```

Weitere Informationen zur Konfiguration von Erfolgsbedingungen finden Sie SuccessConditions in der CodePipeline API-Referenz.

Erstellen Sie eine On Success-Bedingung (CFN)

Verwenden Sie CloudFormation den `OnSuccess` Parameter, um eine On-Success-Bedingung zu konfigurieren. Bei Erfolg führt die Stufe die Regel aus und führt das Ergebnis aus.

```
OnSuccess:
     Result: ROLLBACK
```

- Aktualisieren Sie die Vorlage wie im folgenden Codeausschnitt gezeigt. Im folgenden Beispiel wird eine OnSuccess Bedingung mit einer Regel namens konfiguriert: `MyDeploymentWindowRule`

```
name: Deploy
actions:
- name: Deploy
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: S3
    version: '1'
  runOrder: 1
  configuration:
    BucketName: MyBucket
    Extract: 'false'
    ObjectKey: object.xml
```

```
  outputArtifacts: []
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
  namespace: DeployVariables
onSuccess:
   conditions:
  - result: FAIL
    rules:
    - name: MyMonitorRule
      ruleTypeId:
        category: Rule
        owner: AWS
        provider: CloudWatchAlarm
        version: '1'
      configuration:
        AlarmName: CWAlarm
        WaitTime: '1'
      inputArtifacts: []
      region: us-east-1
```

Weitere Informationen zur Konfiguration von Fehlerbedingungen für das Stagen-Rollback finden Sie OnFailure unter `StageDeclaration` im CloudFormation Benutzerhandbuch.

Stage-Bedingungen löschen

Sie können Phasenbedingungen löschen, die für Ihre Pipeline konfiguriert wurden.

Um eine Phasenbedingung zu löschen

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

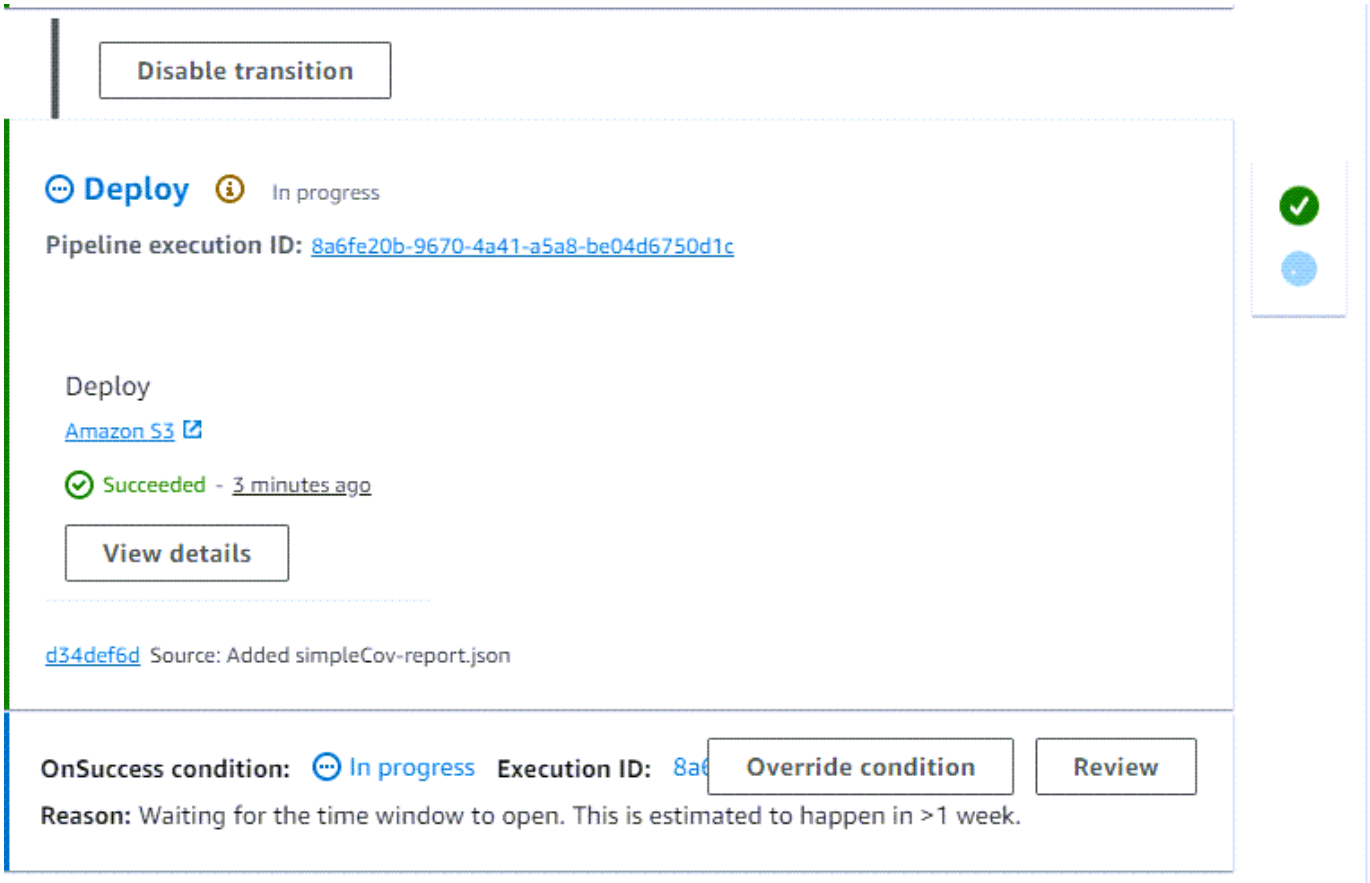
   Die Namen und der Status aller Pipelines, die mit Ihrer verknüpft AWS-Konto sind, werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten.
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie auf der Seite „Bearbeiten“ für die Bedingung, die Sie bearbeiten möchten, die Option Stufe bearbeiten aus.
5. Wählen Sie neben der Bedingung, die Sie löschen möchten, die Option Bedingung löschen aus.

Übergeordnete Stufenbedingungen

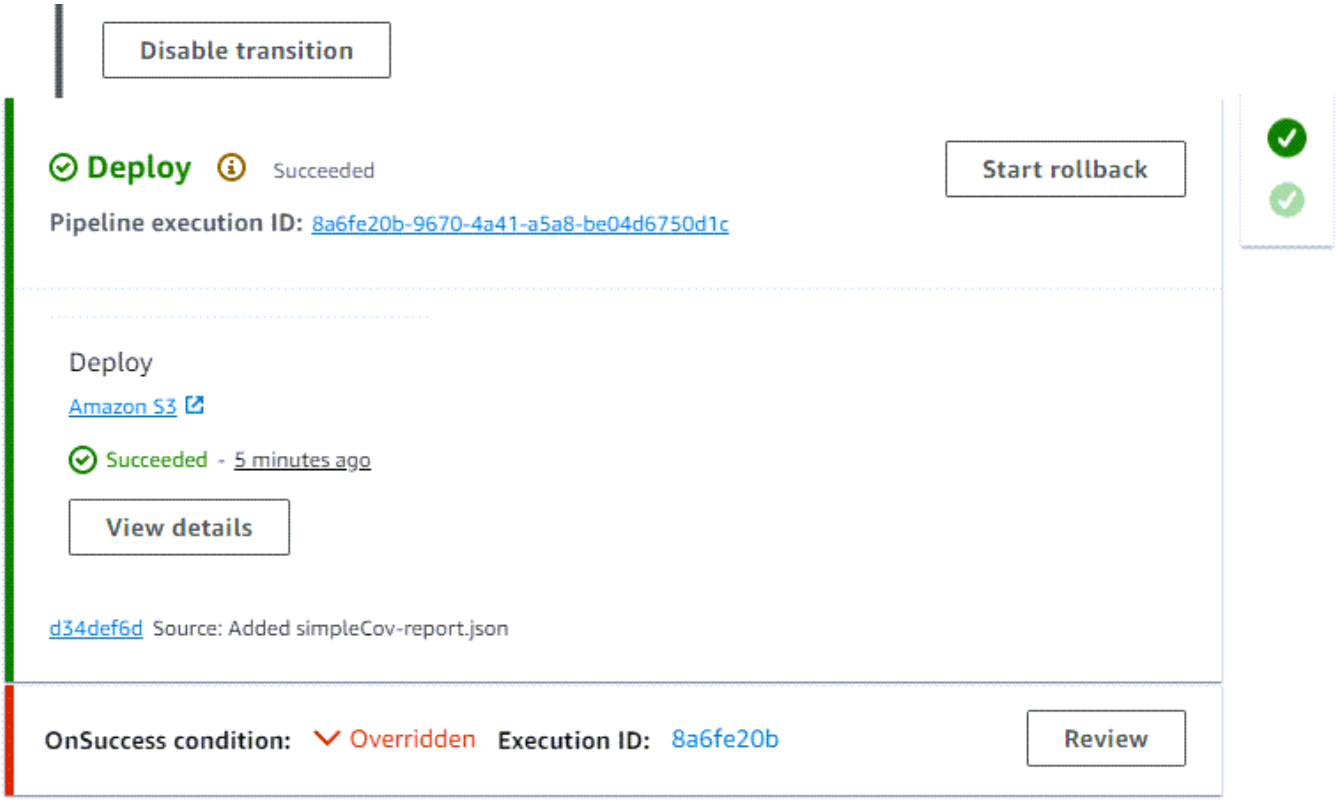
Sie können die Stufenbedingungen überschreiben, die für Ihre Pipeline konfiguriert wurden. Wenn die Phase und die Regel in der Konsole ausgeführt werden, können Sie wählen, ob Sie die Staging-Bedingung außer Kraft setzen möchten. Dies führt dazu, dass die Phase ausgeführt wird.

Um eine Stufenbedingung zu überschreiben

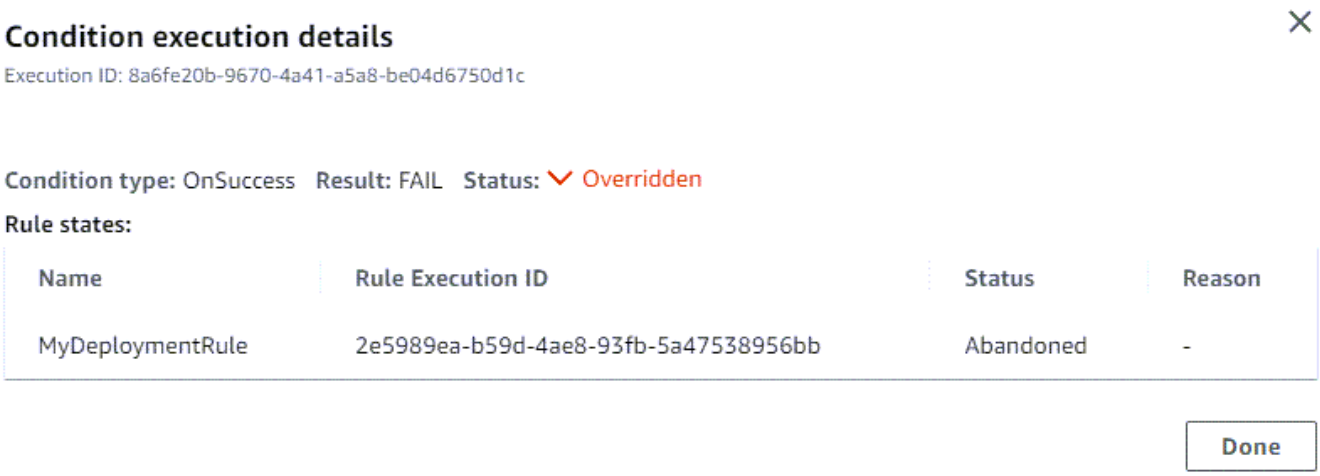
1. In diesem Beispiel wird die Pipeline-Phase mit einer Bedingung ausgeführt. Die Schaltfläche „Überschreiben“ ist aktiviert.



2. Wählen Sie neben der Bedingung, die Sie überschreiben möchten, die Option Überschreiben aus.



3. Um die Details zu überprüfen, wählen Sie Überprüfen. Das Detail im folgenden Beispiel zeigt, dass das konfigurierte Ergebnis für die Bedingung „Fehlgeschlagen“ lautet. Dieses Ergebnis wurde außer Kraft gesetzt. Der Regelstatus lautet aufgrund der Überschreibung aufgegeben.



Verwenden Sie Aktionstypen, benutzerdefinierte Aktionen und Genehmigungsaktionen

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
- Invoke

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
- Erstellen Sie eine benutzerdefinierte Aktion und fügen Sie sie hinzu in CodePipeline
- Kennzeichnen Sie eine benutzerdefinierte Aktion in CodePipeline
- Ruft einen auf AWS Lambda Funktion in einer Pipeline in CodePipeline
- Hinzufügen einer manuellen Genehmigungsaktion zu einer Phase
- Fügen Sie eine regionsübergreifende Aktion hinzu in CodePipeline
- Arbeiten mit Variablen

Mit Aktionstypen arbeiten

Aktionstypen sind vorkonfigurierte Aktionen, die Sie als Anbieter mithilfe eines der unterstützten Integrationsmodelle für Kunden erstellen. AWS CodePipeline

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 ihren Pipelines hinzufügen, sobald sie verfügbar ist. CodePipeline

> **Note**
>
> Sie erstellen Aktionen `custom` im `owner` Feld, die mit einem Jobworker ausgeführt werden. Sie erstellen sie nicht mit einem Integrationsmodell. Hinweise zu benutzerdefinierten Aktionen finden Sie unterErstellen Sie eine benutzerdefinierte Aktion und fügen Sie sie hinzu in CodePipeline.

Komponenten vom Aktionstyp

Die folgenden Komponenten bilden einen Aktionstyp.

- Aktionstyp-ID — Die ID besteht aus Kategorie, Besitzer, Anbieter und Version. Das folgende Beispiel zeigt eine Aktionstyp-ID mit einem Besitzer von`ThirdParty`, einer Kategorie von`Test`, einem Namen `TestProvider` eines Anbieters 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 Eigentümer 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 den Kunden-Pipelines nach verfügbaren Jobs fragt. Der Job-Worker führt dann den Job aus und sendet das Auftragsergebnis mithilfe CodePipeline CodePipeline von APIs an ihn zurück.

> **Note**
>
> Das Jobworker-Integrationsmodell ist nicht das bevorzugte Integrationsmodell.

- Eingabe- und Ausgabeartefakte: Grenzwerte für die Artefakte, die der Eigentümer des Aktionstyps den Kunden der Aktion zuweist.
- Berechtigungen: Die Berechtigungsstrategie, die Kunden festlegt, die auf den Aktionstyp des 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 Eigentümers des Aktionstyps.

Themen

- Fordern Sie einen Aktionstyp an
- Fügen Sie einer Pipeline (Konsole) einen verfügbaren Aktionstyp hinzu
- Zeigen Sie einen Aktionstyp an
- Aktualisieren Sie einen Aktionstyp

Fordern Sie einen Aktionstyp an

Wenn ein neuer CodePipeline Aktionstyp von einem Drittanbieter angefordert wird, wird der Aktionstyp für den Eigentümer des Aktionstyps 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. Um zu beantragen, dass ein Aktionstyp in eine öffentliche Aktion geändert wird, wenden Sie sich an das CodePipeline Serviceteam.

Bevor Sie Ihre Aktionsdefinitionsdatei, die Executor-Ressourcen und die Aktionstyp-Anfrage für das CodePipeline Team erstellen, müssen Sie ein Integrationsmodell auswählen.

Schritt 1: Wählen Sie Ihr Integrationsmodell

Wählen Sie Ihr Integrationsmodell aus 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 die Berechtigungen zu geben, ihn mithilfe des Dienstprinzips aufzurufen:. CodePipeline `codepipeline.amazonaws.com` Die Berechtigungen können mithilfe der Befehlszeile CloudFormation oder über die Befehlszeile hinzugefügt werden.
  - Beispiel für das Hinzufügen von Berechtigungen mithilfe von 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 Jobworker-Integrationsmodell erstellen Sie eine Integration mit einer Liste zulässiger Konten, in denen der Jobworker mithilfe der CodePipeline APIs nach Jobs fragt.

Schritt 2: Erstellen Sie eine Definitionsdatei für den Aktionstyp

Sie definieren einen Aktionstyp in einer Aktionstyp-Definitionsdatei mithilfe von JSON. In der Datei enthalten Sie die Aktionskategorie, das Integrationsmodell, das zur Verwaltung des Aktionstyps verwendet wird, und Konfigurationseigenschaften.

> **Note**
>
> Nachdem Sie eine öffentliche Aktion erstellt haben, können Sie die Aktionstypeigenschaft unter `properties` von `optional` bis nicht ändern`required`. Sie können das auch nicht ändern`owner`.

Weitere Informationen zu den Parametern der Aktionstyp-Definitionsdatei finden Sie unter ActionTypeDeclaration und UpdateActionType in 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 erstellt wurde, entweder oder`Lambda`. `job worker` Sie können je nach `lambdaExecutorConfiguration` Executor-Typ nur entweder `jobWorkerExecutorConfiguration` oder angeben.
  - `configuration`: Ressourcen für die Konfiguration des Aktionstyps, basierend auf dem ausgewählten Integrationsmodell. Verwenden Sie für das Lambda-Integrationsmodell den ARN der Lambda-Funktion. Verwenden Sie für das Jobworker-Integrationsmodell das Konto oder die Liste der Konten, von denen aus der Job Worker ausgeführt wird.
  - `jobTimeout`: Der Timeout in Sekunden für den Job. Eine Aktionsausführung kann aus mehreren Jobs bestehen. Dies ist der Timeout für einen einzelnen Job und nicht für die gesamte Aktionsausführung.

    > **Note**
    >
    > Für das Lambda-Integrationsmodell beträgt der maximale Timeout 15 Minuten.

  - `policyStatementsTemplate`: Die Richtlinienerklärung, die die Berechtigungen im CodePipeline Kundenkonto angibt, die für die erfolgreiche Ausführung einer Aktion erforderlich sind.
  - `type`: Das Integrationsmodell, das zum Erstellen und Aktualisieren des Aktionstyps verwendet wird, 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 ausgeführt werden: Quelle, Build, Deploy, Test, Invoke oder Approval.
  - `provider`: Der Anbieter des aufgerufenen Aktionstyps, z. B. der Name des Anbieters, das Unternehmen oder das Produkt. Der Anbietername wird angegeben, wenn der Aktionstyp erstellt wird.
  - `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 aufgrund des Ergebnisses der Aktionstyp-Phase 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 aus dem Protokoll weggelassen wird. 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`: Eine Liste der URLs CodePipeline wird Ihren Benutzern 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 Dienst registrieren und die Erstkonfiguration der von diesem Dienst 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 zu registrieren CodePipeline, wenden Sie sich mit Ihrer Anfrage an das CodePipeline Serviceteam.

Das CodePipeline Serviceteam registriert die Integration des neuen Aktionstyps, 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 mithilfe des folgenden Formulars 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 Integration von Arbeitnehmern 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 nutzen.

Fügen Sie einer Pipeline (Konsole) einen verfügbaren Aktionstyp hinzu

Sie fügen Ihren Aktionstyp einer Pipeline hinzu, damit Sie ihn testen können. Sie können dies tun, indem Sie eine neue Pipeline erstellen oder eine vorhandene bearbeiten.

> **Note**
>
> Wenn es sich bei Ihrem Aktionstyp um eine Aktion der Kategorie Quell, Build oder Deploy handelt, können Sie sie 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 von der CodePipeline Konsole aus zu einer vorhandenen Pipeline hinzuzufügen

1. Melden Sie sich bei der an AWS-Managementkonsole 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, zu 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, 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 Definitionsdatei für den Aktionstyp `inputArtifactDetails` angegebenen Werte definiert werden.
8. Wählen Sie Verbinden mit < 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 unternimmt, um Ihren Aktionstyp zu verwenden. Deine 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 CodePipeline „Aktion 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 Definitionsdatei für den Aktionstyp `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 in der Quellphase der Pipeline angegebenen Quelle oder folgen Sie den Schritten unter Manuelles Starten einer Pipeline.

Um eine Pipeline mit Ihrem Aktionstyp zu erstellen, folgen Sie den Schritten unter Eine Pipeline, Phasen und Aktionen erstellen 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 benennen Sie die Datei`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 Ausgabe der Aktionsdefinition 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 erforderliche Eigenschaften 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 benennen Sie sie`action.json`. Fügen Sie die Aktionstyp-Struktur 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 Definitionsdatei für den Aktionstyp.

   Das folgende Beispiel zeigt, wie ein Beispiel-Aktionstyp aktualisiert wird, der mit dem Lambda-Integrationsmodell erstellt wurde. In diesem Beispiel werden die folgenden Änderungen vorgenommen:

   - Ändert den `provider` Namen in`TestProvider1`.
   - Fügen Sie 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 Sie eine benutzerdefinierte Aktion und fügen Sie sie hinzu in CodePipeline

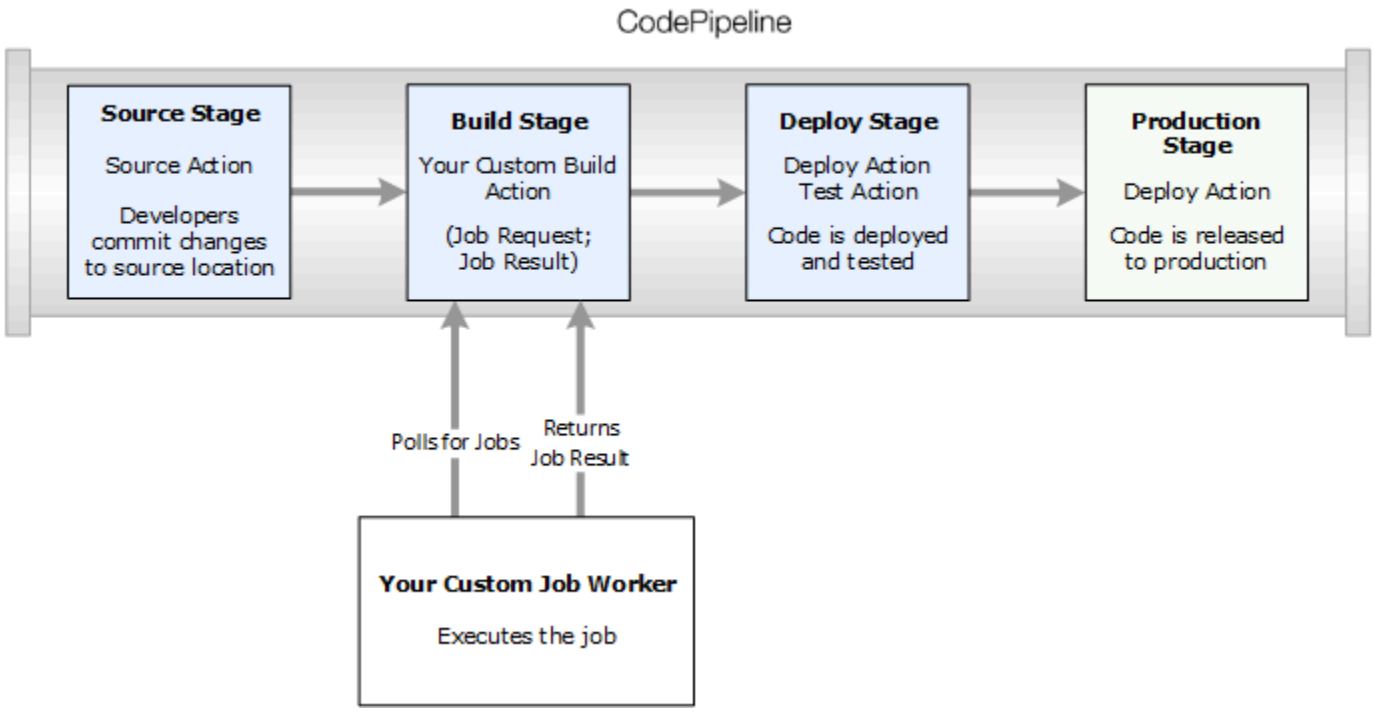
AWS CodePipeline enthält eine Reihe von Aktionen, mit denen Sie Ressourcen für Ihren automatisierten Release-Prozess konfigurieren, testen und bereitstellen können. Falls Ihr Freigabeverfahren Aktivitäten beinhaltet, die nicht durch die Standardaktionen abgedeckt sind (wie z. B. ein intern entwickelter Erstellungsprozess oder eine Testsuite), können Sie dafür eine benutzerdefinierte Aktion erstellen und diese Ihrer Pipeline hinzufügen. Sie können das verwenden AWS CLI , um benutzerdefinierte Aktionen in Pipelines zu erstellen, die Ihrem AWS Konto zugeordnet 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 Jobanfragen CodePipeline für diese benutzerdefinierte Aktion abfragt, den Job ausführt und das Statusergebnis an zurückgibt CodePipeline. Dieser Auftragsworker kann sich auf einem beliebigem Computer oder einer beliebigen Ressource befinden, solange er Zugriff auf den öffentlichen Endpunkt für CodePipeline besitzt. Um Zugriff und Sicherheit einfach zu verwalten, sollten Sie erwägen, Ihren Job Worker auf einer Amazon 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 zu erstellen, verwenden Sie 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:

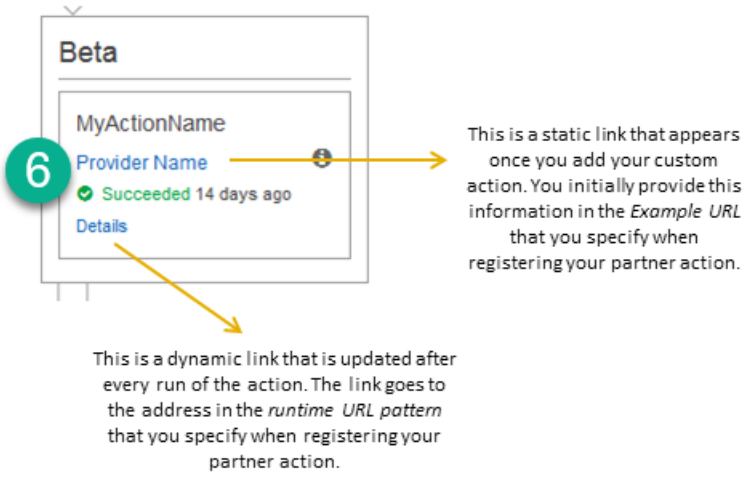
```
{
    "category": "Build",
    "provider": "My-Build-Provider-Name",
    "version": "1",
    "settings": {
        "entityUrlTemplate": "https://my-build-instance/job/{Config:ProjectName}/",
        "executionUrlTemplate": "https://my-build-instance/job/
{Config:ProjectName}/lastSuccessfulBuild/{ExternalExecutionId}/"
    },
    "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 Hinweise zum Taggen von Ressourcen in finden Sie CodePipeline unterTaggen 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 ActionTypeSettings und CreateCustomActionType in 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 create-custom-action-type Befehlsausgabe umfasst der `id` Abschnitt Folgendes`"owner": "Custom"`: CodePipeline weist benutzerdefinierte Aktionstypen automatisch `Custom` als Eigentümer 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 an CodePipeline zurückgibt. CodePipeline Der Auftragsworker kann sich auf einem beliebigen Computer oder einer beliebigen Ressource befinden, solange er Zugriff auf den öffentlichen Endpunkt für CodePipeline besitzt.

Sie können Ihren Auftragsworker auf vielerlei Arten gestalten. In den folgenden Abschnitten erhalten Sie praktische Anleitungen für die Entwicklung Ihres benutzerdefinierten Auftragsworkers 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-Instances mit einer IAM-Instance-Rolle erstellen, sodass Sie die Ressourcen, die Sie für Ihre Integration benötigen, problemlos skalieren können. Sie können die integrierte Integration mit verwenden AWS , um die Interaktion zwischen Ihrem benutzerdefinierten Job Worker und zu vereinfachen. CodePipeline

Um Amazon EC2-Instances einzurichten

1. Erfahren Sie mehr über Amazon EC2 und finden Sie heraus, ob es die richtige Wahl für Ihre Integration ist. Informationen finden Sie unter Amazon EC2 — Virtual Server Hosting.
2. Beginnen Sie mit der Erstellung Ihrer Amazon EC2-Instances. Weitere Informationen finden Sie unter Erste Schritte mit Amazon EC2 Linux-Instances.

Eine weitere Strategie, die Sie in Betracht ziehen sollten, ist die Verwendung von Identity Federation mit IAM, um Ihr vorhandenes Identitätsanbietersystem und Ihre Ressourcen zu integrieren. Diese Strategie ist besonders nützlich, wenn Sie bereits einen Unternehmensidentitätsanbieter haben oder eine Konfiguration vorgenommen wurde, um Benutzer mit Web-Identitätsanbietern zu unterstützen. Mit einem 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 im IAM-Benutzerhandbuch unter Identitätsanbieter und Verbund.

Erstellen Sie eine der folgenden Optionen, um sie unter Ihrem zu verwenden AWS-Konto , wenn Sie Ihre benutzerdefinierte Aktion und Ihren Job Worker ausführen.

Benutzer benötigen programmgesteuerten Zugriff, wenn sie AWS außerhalb von interagieren möchten. AWS-Managementkonsole Die Art und Weise, wie programmgesteuerter 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
IAM	(Empfohlen) Verwenden Sie Konsolenanmeldedaten als temporäre Anmeldeinformationen, um programmatische Anforderungen an die AWS SDKs AWS CLI oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu finden Sie im AWS Command Line Interface Benutzerhandbuch unter Anmeldung für AWS lokale Entwicklung. AWS CLI • Informationen zu AWS SDKs finden Sie unter Anmeldung für AWS lokale Entwicklung im Referenzh

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		andbuch für AWS SDKs und Tools.
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS SDKs oder AWS CLI APIs zu signieren. AWS	Befolgen Sie die Anweisung en für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu finden Sie AWS IAM Identity Center im AWS Command Line Interface Benutzerh andbuch unter Configuring AWS CLI to use. AWS CLI • Informationen zu AWS SDKs, Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifiz ierung im Referenzh andbuch für AWS SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS SDKs oder APIs AWS CLI zu signieren. AWS	Folgen Sie den Anweisungen unter Verwenden temporäre r Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
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 im AWS CLI Benutzerhandbuch unter Authentifizierung mit IAM-Benutzeranmeldeinformationen. AWS Command Line Interface • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch für AWS SDKs und Tools. • Informationen zu AWS APIs finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

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.

JSON

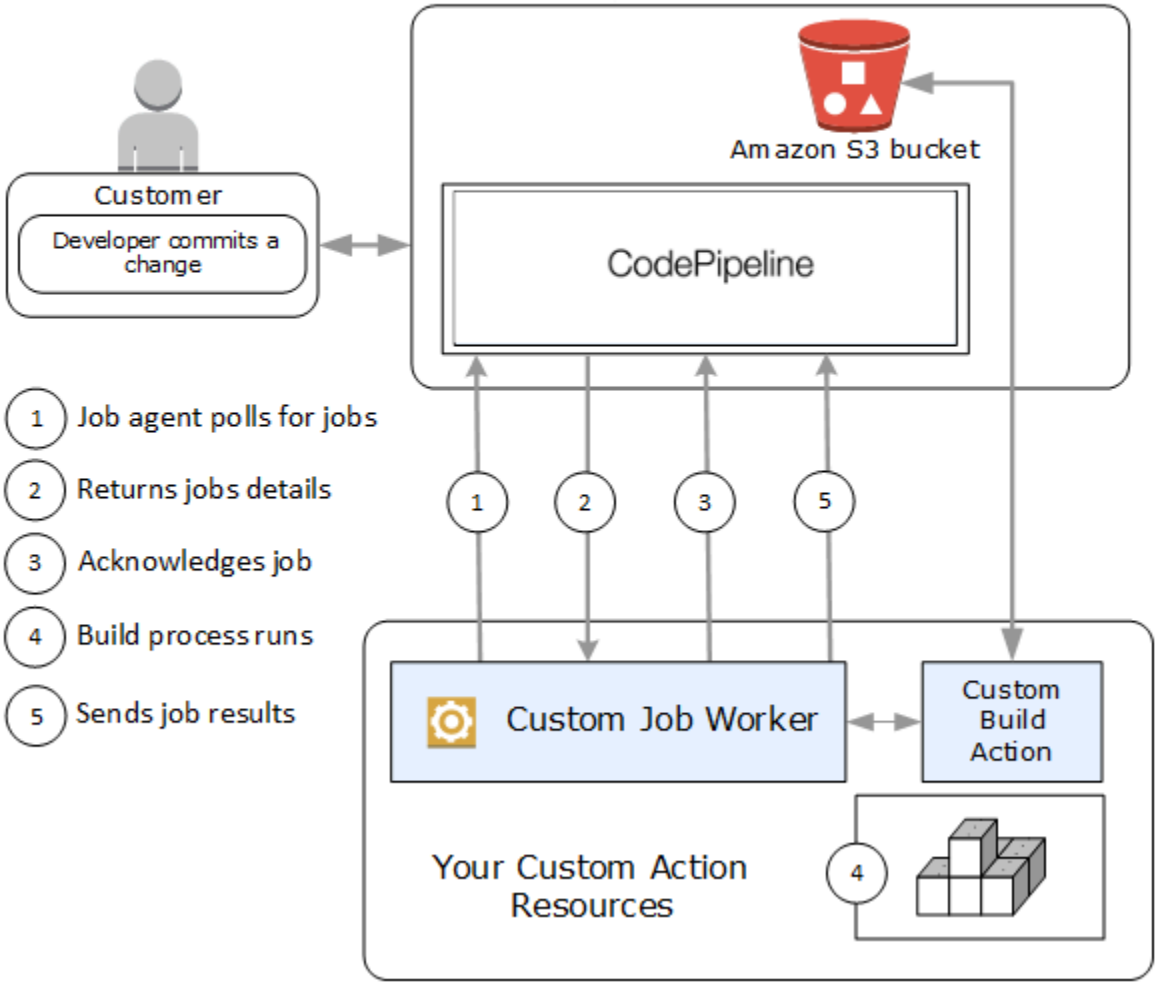
```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "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 der Bearbeitung umgehen wird CodePipeline. Das folgende übersichtliche Diagramm zeigt den Arbeitsablauf einer benutzerdefinierten Aktion und eines Jobworkers für einen Build-Prozess.



1. Ihr Jobbearbeiter fragt CodePipeline nach Jobs mithilfe von`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 bis zu der Phase fortgesetzt, in der Ihre benutzerdefinierte Aktion konfiguriert wurde. Wenn Ihre Aktion in dieser Phase erreicht wird, wird ein Auftrag 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 Jobworker ruft an`AcknowledgeJob`, um CodePipeline eine Auftragsbestätigung zu senden. CodePipeline gibt eine Bestätigung zurück, aus der hervorgeht, dass der für die Stelle zuständige Mitarbeiter den Job fortsetzen soll (`InProgress`). Wenn mehrere Mitarbeiter nach Stellen fragen und ein anderer Mitarbeiter die Stelle 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-Bucket ab, erstellt sie und schiebt erfolgreich erstellte Artefakte zurück in den Amazon S3-Bucket.
5. Während die Aktion ausgeführt wird, kann der Job-Worker einen Continuation-Token (die Serialisierung des vom Job-Worker generierten Status des Jobs, z. B. eine Build-ID im JSON-Format oder einen Amazon S3-Objektschlüssel) aufrufen`PutJobSuccessResult`, sowie die `ExternalExecutionId` Informationen, die zum Ausfüllen des Links verwendet werden`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 an zurück CodePipeline, ohne dass der letzte Schritt ein Fortsetzungstoken enthält.

Weitere Informationen zur Verwendung des Fortsetzungstokens finden Sie in den Spezifikationen für `PutJobSuccessResult` unter CodePipeline -API-Referenz.

6. Sobald die benutzerdefinierte Aktion abgeschlossen ist, gibt der Jobworker das Ergebnis der benutzerdefinierten Aktion an 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:

- Es wird nach Aufträgen gesucht, die CodePipeline verwendet werden`PollForJobs`.
- Bestätigen von Aufträgen und Zurückgeben der Ergebnisse zur CodePipeline Verwendung von `AcknowledgeJobPutJobSuccessResult`, und. `PutJobFailureResult`
- Artefakte werden abgerufen, and/or indem Artefakte für die Pipeline in den Amazon S3-Bucket eingegeben werden. Um Artefakte aus dem Amazon S3-Bucket herunterzuladen, müssen Sie einen Amazon S3-Client erstellen, der Signature Version 4-Signing (Sig V4) verwendet. Sig V4 ist erforderlich für AWS KMS.

  Um Artefakte in den Amazon S3-Bucket hochzuladen, müssen Sie die Amazon `PutObject` S3-Anfrage zusätzlich so konfigurieren, dass sie Verschlüsselung verwendet. Derzeit wird nur 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 den, 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 Specifying the AWS Key Management Service in Amazon S3 Using the AWS SDKs. Weitere Informationen zum Amazon S3-Bucket für finden Sie CodePipeline unterCodePipeline Konzepte.

Ein komplexeres Beispiel für einen benutzerdefinierten Job-Worker ist verfügbar unter GitHub. Dieses Beispiel ist ein Open Source-Beispiel und wird unverändert bereitgestellt.

- Beispiel für einen 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 haben, können Sie Ihre benutzerdefinierte Aktion zu einer Pipeline hinzufügen, indem Sie eine neue erstellen und sie auswählen, wenn Sie den Assistenten zum Erstellen einer Pipeline verwenden, indem Sie eine vorhandene 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 auf Ihre benutzerdefinierten Aktionen in CodePipeline Tags anzuwenden. Hinweise zu CodePipeline Ressourcen-Tagging, Anwendungsfällen, Tag-Schlüssel- und Werteinschränkungen und unterstützten Ressourcentypen finden Sie unter. Taggen 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 mithilfe der ein Tag hinzuzufügen. Informationen zum Hinzufügen eines Tags zu einer benutzerdefinierten Aktion während der Erstellung finden Sie unter Erstellen Sie eine benutzerdefinierte Aktion und fügen Sie sie 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, der *TestActionType* mit dem Tag-Wert von benannt ist*UnitTest*, und einem Tag-Schlüssel, der *ApplicationName* mit dem Tag-Wert von benannt ist*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

Folgen Sie diesen Schritten, um das AWS CLI zum Bearbeiten eines Tags für eine benutzerdefinierte Aktion zu verwenden. 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 mit dem 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 für eine benutzerdefinierte 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.

Ruft einen auf AWS Lambda Funktion in einer Pipeline 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 Logs protokolliert werden. CloudWatch Die CodePipeline Rolle verwendet ein JSON-Ereignis, um temporäre Anmeldeinformationen im Feld an Lambda zu übergeben. `artifactCredentials` Ein Beispiel für ein Ereignis finden Sie unter JSON-Beispielereignis.

Hier sind einige Möglichkeiten, wie Lambda-Funktionen in Pipelines verwendet werden können:

- Um Ressourcen bei Bedarf in einer Phase einer Pipeline zu erstellen, sie in einer anderen Phase zu verwenden CloudFormation und zu löschen.
- Zur Bereitstellung von Anwendungsversionen ohne Ausfallzeiten AWS Elastic Beanstalk mit einer Lambda-Funktion, die CNAME-Werte austauscht.
- 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 Belastungen Ihres AWS Kontos führen. Weitere Informationen finden Sie unter – Preise.

In diesem Thema wird davon ausgegangen, dass Sie mit Pipelines, Funktionen AWS CodePipeline AWS Lambda und den IAM-Richtlinien und -Rollen, von denen sie abhängen, vertraut sind und wissen, wie man sie erstellt. In diesem Thema wird Folgendes veranschaulicht:

- Erstellen Sie eine Lambda-Funktion, die testet, ob eine Webseite erfolgreich bereitgestellt wurde.
- Konfigurieren Sie die 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, PutJobFailureResult sollte der Status der Lambda-Ausführung mit PutJobSuccessResult und 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 folgenden Bereichen zu demonstrieren: CodePipeline

- Basic Lambda function
  - Erstellen einer grundlegenden Lambda-Funktion zur Verwendung mit. CodePipeline
  - Wenn Sie Erfolg oder Misserfolg zurückgeben, wird CodePipeline der Link Details für die Aktion angezeigt.
- Python-Beispielfunktion, die eine verwendet AWS CloudFormation Vorlage
  - Verwenden von JSON-encoded Benutzerparametern, um mehrere Konfigurationswerte an die Funktion (`get_user_params`) zu übergeben.
  - 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 werden und die Funktion ist auch dann erfolgreich, 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 finden Sie unter https://docs.aws.amazon.com/lambda/latest/dg/limits.html 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
- Schritt 2: Erstellen Sie die Lambda-Funktion
- Schritt 3: Fügen Sie die Lambda-Funktion zu einer Pipeline in der Konsole hinzu CodePipeline
- Schritt 4: Testen Sie die Pipeline mit der Lambda-Funktion
- Schritt 5: Nächste Schritte
- JSON-Beispielereignis
- Weitere Beispielfunktionen

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 Tutorials erstellt haben. Wenn diese Pipeline immer 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 SchrittenTutorial: Erstellen einer einfachen Pipeline (S3-Bucket), um einen Amazon 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. Die Schritte in diesem Thema verwenden den Namen MyLambdaTestPipeline.

2. Wählen Sie auf der Statusseite Ihrer Pipeline in der CodeDeploy Aktion 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-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 Standard-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, die Lambda-Aktionen enthalten, fehl.

Schritt 2: Erstellen Sie die Lambda-Funktion

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-Managementkonsole 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.

   JSON

```
{
  "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 so ändern, dass sie die für diese Lambda-Funktionen erforderlichen Aktionen zulässt.

5. Wählen Sie auf der Seite mit dem Richtlinien-Dashboard Roles (Rollen) aus und wählen Sie dann Create role (Rolle erstellen) aus.
6. Wählen AWS-Service Sie auf der Seite „Rolle erstellen“. Wählen Sie Lambda und klicken Sie dann auf Next: Permissions (Weiter: Berechtigungen).
7. Aktivieren Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien anfügen) das Kontrollkästchen neben CodePipelineLambdaExecPolicy und wählen Sie dann Next: Tags (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 zur Verwendung mit zu erstellen CodePipeline

1. Melden Sie sich bei der an AWS-Managementkonsole 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 unter Runtime die Option 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 Event-Objekt unter dem CodePipeline.job Schlüssel enthält die Jobdetails. Ein vollständiges Beispiel für die CodePipeline Rückkehr des JSON-Ereignisses an 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 Konsole hinzu CodePipeline

In diesem Schritt fügen Sie Ihrer Pipeline eine neue Stufe 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-Managementkonsole 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“ + Stufe hinzufügen aus, um nach der Bereitstellungsphase eine Phase mit der CodeDeploy Aktion 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 den Fortschritt der Phase leicht verfolgen können, wenn Artefakte eine Pipeline durchlaufen.

5. Wählen Sie + Add action group (Aktionsgruppe hinzufügen). Geben Sie unter Aktion bearbeiten im Feld Aktionsname einen Namen für Ihre Lambda-Aktion ein (z.

B.**`MyLambdaAction`**). Wählen Sie in Provider (Anbieter) AWS Lambda aus. Wählen Sie im Feld 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-Instance an, die Sie zuvor kopiert haben (z. B.**`http://`*`192.0.2.4`***), und wählen Sie dann Fertig aus.

> **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 Eventdaten 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

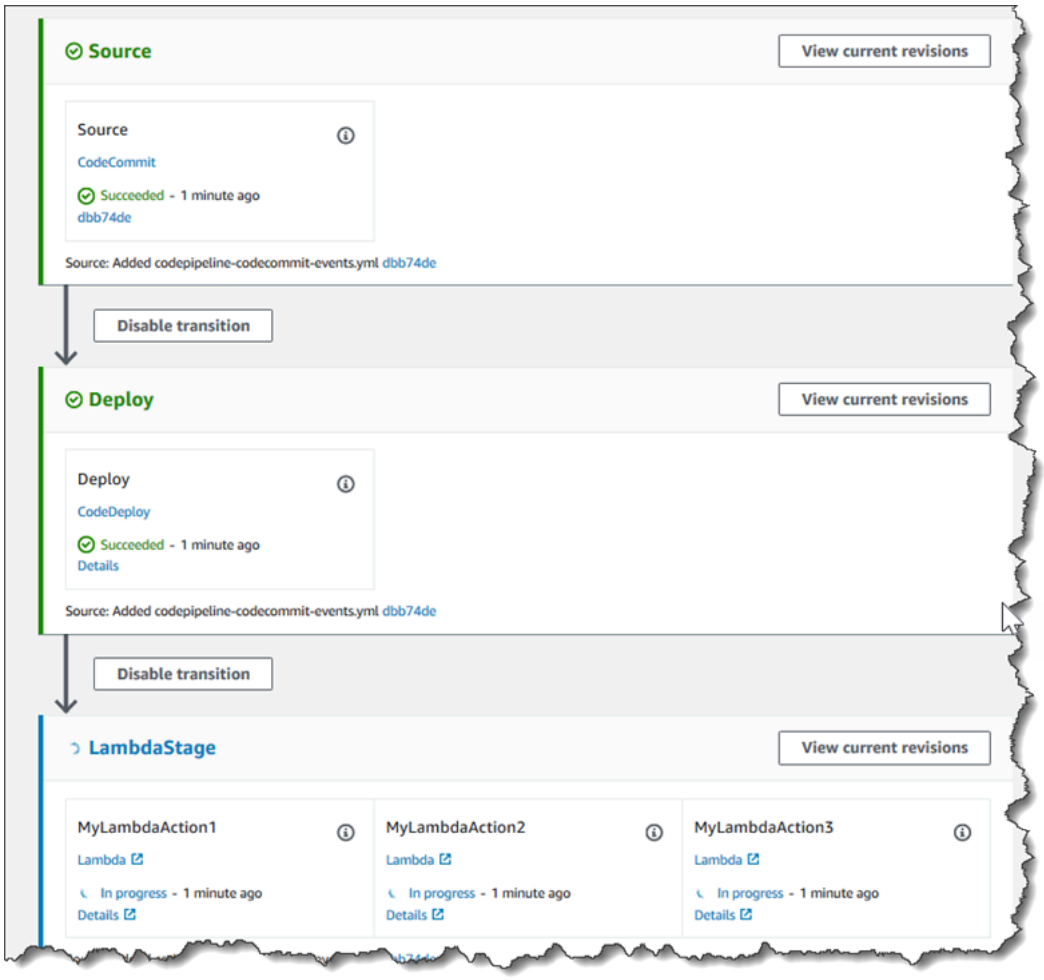
1. Wählen Sie auf der Seite mit den Pipeline-Details die Option Release-Änderung aus. Dadurch wird die letzte Revision gestartet, die in jedem in einer Quellaktion der Pipeline angegebenen Quellspeicherort vorhanden ist.
2. Wenn die Lambda-Aktion abgeschlossen ist, wählen Sie den Link Details, um den Log-Stream für die Funktion in Amazon anzuzeigen CloudWatch, einschließlich der fakturierten Dauer des Ereignisses. Wenn die Funktion fehlgeschlagen ist, enthält das CloudWatch Protokoll Informationen über die Ursache.

Schritt 5: Nächste Schritte

Nachdem Sie nun erfolgreich eine Lambda-Funktion erstellt und als Aktion zu einer Pipeline hinzugefügt haben, können Sie Folgendes versuchen:

- Fügen Sie Ihrer Stage 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 das Experimentieren mit der Lambda-Funktion abgeschlossen haben, 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 Beispiel für ein JSON-Ereignis, das von an Lambda gesendet wurde. CodePipeline 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 *red italic text* sind Beispiele oder Erklärungen, keine echten 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 veranschaulichen 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 beschrieben.

Themen

- Python-Beispielfunktion, die eine verwendet AWS CloudFormation Vorlage

Python-Beispielfunktion, die eine verwendet AWS CloudFormation Vorlage

Das folgende Beispiel zeigt eine Funktion, die einen Stack auf der Grundlage einer bereitgestellten CloudFormation Vorlage erstellt oder aktualisiert. Die Vorlage erstellt einen Amazon 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-Bucket als Quellaktion verwendet, oder dass Sie Zugriff auf einen versionierten Amazon S3-Bucket haben, den Sie mit der Pipeline verwenden können. Sie erstellen die 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-encoded Benutzerparametern, um mehrere Konfigurationswerte an die Funktion () `get_user_params` zu übergeben.
- 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 werden und die Funktion ist erfolgreich, 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 für Amazon S3 und CloudFormation, wie in dieser Beispielrichtlinie gezeigt CodePipeline, verfügen:

JSON

```
{
    "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 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 (.zip) -Datei mit diesem Verzeichnis und der benannten **`template-package.zip`** Datei und laden Sie die komprimierte Datei in einen versionierten Amazon 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 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 zu ermöglichen, die für Ihre CloudFormation Vorlage erforderlich sind.

Um den folgenden Code als Funktion in Lambda hinzuzufügen

1. Öffnen Sie die Lambda-Konsole und wählen Sie Funktion erstellen.
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 Eine 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 für Handler den Standardwert und für Rolle den 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 Input Artifacts die Option`MyTemplate`. UserParametersIn müssen Sie eine JSON-Zeichenfolge mit drei Parametern angeben:

- Stack name
- CloudFormation Name der Vorlage und Pfad zur Datei
- Artefakt eingeben

Verwenden Sie geschweifte Klammern ({}) und trennen Sie die Parameter durch Kommas. Um beispielsweise einen Stapel mit dem Namen *`MyTestStack`* für eine Pipeline mit dem Eingabeartefakt in zu erstellen *`MyTemplate`* UserParameters, geben Sie Folgendes ein: {"stack“:“ *`MyTestStack`* „, "file“ :"template- package/template .json“, "artifact“:“ „}. *`MyTemplate`*

> **Note**
>
> Obwohl Sie das Eingabeartefakt in angegeben haben UserParameters, müssen Sie dieses Eingabeartefakt auch für die Aktion in Eingabeartefakte 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.

Hinzufügen einer manuellen Genehmigungsaktion zu einer Phase

In können Sie einer Phase in einer Pipeline an dem Punkt AWS CodePipeline, an dem die Pipeline-Ausführung gestoppt werden soll, eine Genehmigungsaktion hinzufügen, sodass jemand 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 beendet hat, niemand die Aktion genehmigt oder ablehnt, ist das Ergebnis dasselbe, als ob eine Aktion fehlschlägt und die Pipeline-Ausführung nicht fortgesetzt wird.

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 zu einem Thema von Amazon Simple Notification Service 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 für Amazon SNS-Themen veröffentlichen, können Sie zwischen 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-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 Endpunkte HTTP/HTTPS
- 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 zum Amazon 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 für die Genehmigung oder Ablehnung von Genehmigungsaktionen erforderlich sind.
2. (Optional) Wenn Sie Amazon SNS-Benachrichtigungen verwenden, stellen Sie sicher, dass die Servicerolle, die Sie in Ihrem CodePipeline Betrieb verwenden, die Berechtigung hat, auf Amazon SNS-Ressourcen zuzugreifen.
3. (Optional) Wenn Sie Amazon SNS-Benachrichtigungen verwenden, erstellen Sie ein Amazon SNS-Thema und fügen 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-Themen sind wie folgt strukturiert. *arn:aws:sns:us-east-2:80398EXAMPLE:MyApprovalTopic* Weitere Informationen finden

Sie unter Amazon-Ressourcennamen (ARNs) und AWS-Service Namespaces im.) Allgemeine Amazon Web Services-Referenz

5. Die Pipeline stoppt, wenn sie die Genehmigungsaktion erreicht. Wenn ein ARN für ein Amazon SNS-Thema in der Konfiguration der Aktion enthalten war, wird eine Benachrichtigung zum Amazon SNS-Thema veröffentlicht. 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 für die Aktualisierung des Status von Genehmigungsaktionen erteilt werden. Sie können die Berechtigung zum Zugriff auf alle Pipelines und Genehmigungsaktionen in Ihrem Konto gewähren, indem Sie die `AWSCodePipelineApproverAccess` verwaltete Richtlinie an einen IAM-Benutzer, eine IAM-Rolle oder eine Gruppe anhängen. Sie können auch eingeschränkte Berechtigungen gewähren, indem Sie die einzelnen Ressourcen angeben, auf die ein IAM-Benutzer, eine 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 verwalteten Richtlinien, die für verfügbar sind, 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 Anleitung unter Eine Rolle für einen externen Identitätsanbieter (Verbund) erstellen im IAM-Benutzerhandbuch.

- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter Eine Rolle für einen IAM-Benutzer erstellen 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. Beispielsweise gewährt die folgende Richtlinie Benutzern die Befugnis, `MyApprovalAction` in der Region USA Ost (Ohio) (us-east-2) nur die Aktion zu genehmigen oder abzulehnen, die in der `MyFirstPipeline` Pipeline genannt wird:

> **Note**
>
> Die `codepipeline:ListPipelines` Berechtigung ist nur erforderlich, wenn IAM-Benutzer auf das CodePipeline Dashboard zugreifen müssen, um diese Liste von Pipelines 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-Managementkonsole 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-Berechtigungen für eine Servicerolle CodePipeline

Wenn Sie Amazon SNS verwenden möchten, um Benachrichtigungen zu Themen zu veröffentlichen, wenn Genehmigungsmaßnahmen überprüft werden müssen, muss der Servicerolle, die Sie in Ihrem CodePipeline Betrieb verwenden, die Berechtigung zum Zugriff auf die Amazon 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-Managementkonsole mit denselben Kontoinformationen angemeldet sind, die Sie verwendet haben. Erste Schritte mit CodePipeline

JSON

```
{
    "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-Managementkonsole 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-Managementkonsole, können Sie eine JSON- oder YAML-Richtlinienvorlage generieren, die Sie in CloudFormation Vorlagen verwenden können.

   Wählen Sie dazu im Richtlinieneditor Aktionen und dann CloudFormation Vorlage generieren aus. Weitere Informationen 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 Tags für AWS Identity and Access Management 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 eine Genehmigungsaktion an dem Punkt hinzufügen, an dem die Pipeline beendet werden soll, damit 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-Ressourcen zuzugreifen. Weitere Informationen finden Sie unter Gewähren Sie Amazon SNS-Berechtigungen für eine Servicerolle CodePipeline.
- 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.

Fügen Sie einer CodePipeline Pipeline eine manuelle Genehmigungsaktion hinzu (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 Eine Pipeline, Phasen und Aktionen erstellen 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".

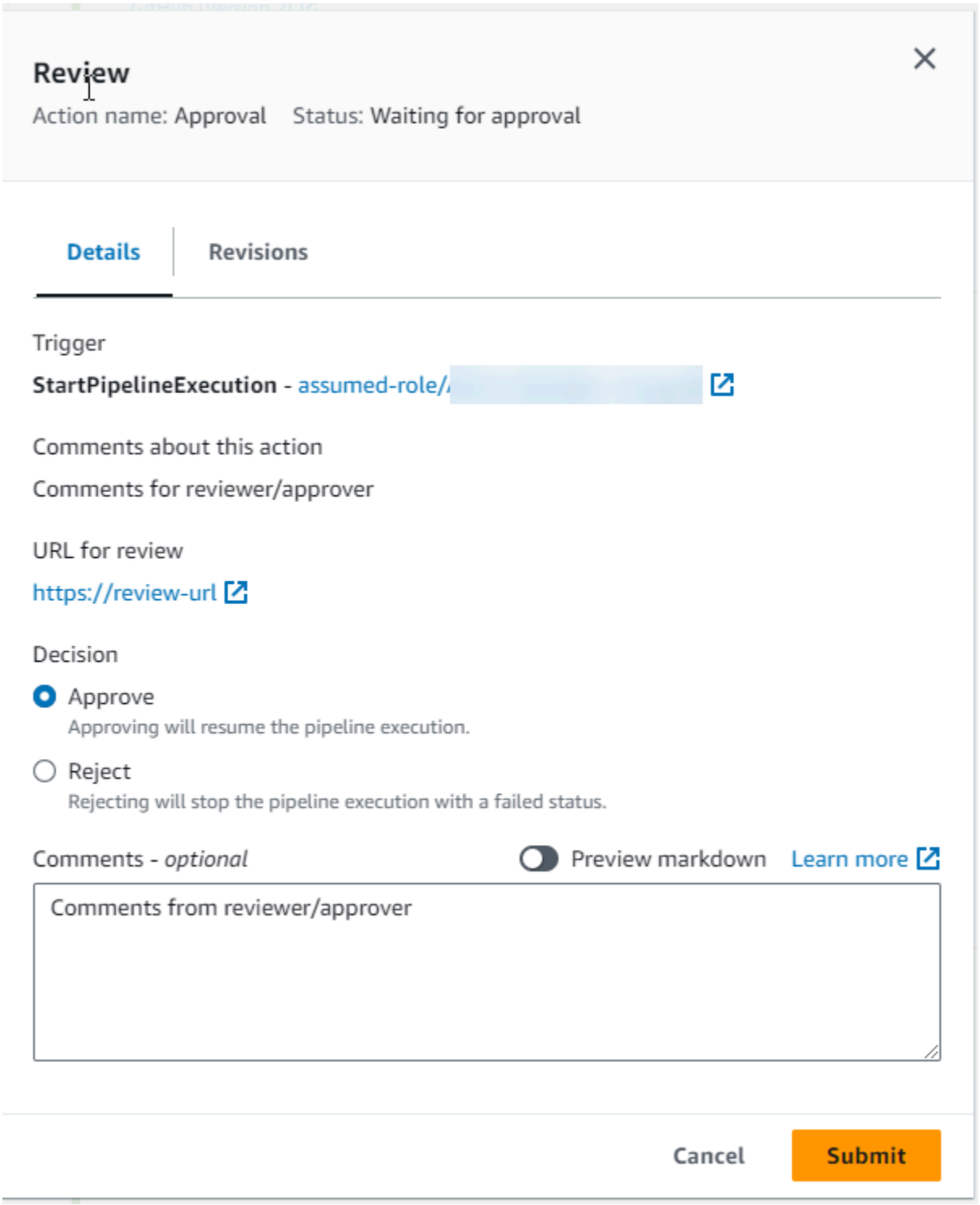
Wenn die Person, die die Genehmigungsaktion zur Pipeline hinzugefügt hat, Benachrichtigungen konfiguriert hat, erhalten Sie möglicherweise eine E-Mail mit den Pipeline-Informationen und dem Status zur Genehmigung.

Erlauben oder Ablehnen einer Genehmigungsaktion (Konsole)

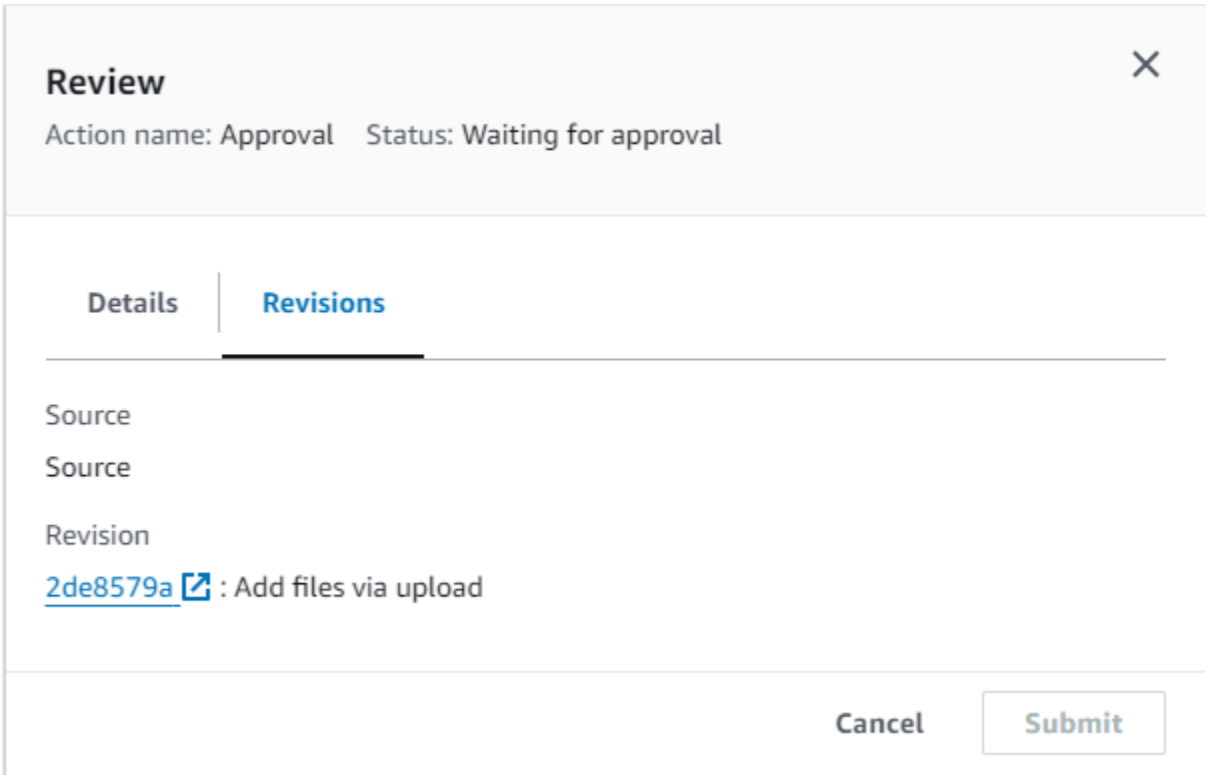
Wenn Sie eine Benachrichtigung erhalten, die einen direkten Link zu einer Genehmigungsaktion enthält, klicken Sie auf den Link Approve or reject (Genehmigen oder ablehnen), melden Sie sich bei der Konsole an und fahren Sie dann mit Schritt 7 hier fort. Führen Sie anderenfalls alle der folgenden Schritte 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 Quellrevisionen 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 die URL für den Link zur Überprüfung aus und überprüfen Sie dann den Inhalt.
6. Geben Sie im Fenster „Überprüfen“ Kommentare zur Bewertung ein, z. B. warum Sie die Aktion genehmigen oder ablehnen, und wählen Sie dann „Genehmigen“ oder „Ablehnen“.
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 Befehl get-pipeline-state für die Pipeline aus, die die Genehmigungsaktion enthält. Geben Sie für eine Pipeline mit dem Namen *MyFirstPipeline* beispielsweise 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.

   Bei vorhergehendem Beispiel *MyFirstPipeline* sollte Ihre Datei folgendermaßen 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 Befehl put-approval-result mit Angabe des Namens der Genehmigungs-JSON-Datei aus, beispielsweise folgendermaßen:

> **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 beendet wird. Sie können die JSON-Ausgabe verwenden, um Nachrichten an Amazon 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-Nachrichten an Amazon SQS-Warteschlangen und Aufrufen von Lambda-Funktionen mithilfe von Amazon SNS-Benachrichtigungen im Amazon SNS-Entwicklerhandbuch.

Im folgenden Beispiel wird die Struktur der JSON-Ausgabe bei CodePipeline-Genehmigungen dargestellt.

```
{
    "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 einer Aktion AWS-Service ist und sich dieser type/provider Aktionstyp in einer anderen AWS Region als Ihre Pipeline befindet, handelt es sich um eine regionsübergreifende Aktion.

> **Note**
>
> Cross-region Aktionen werden unterstützt und können nur in den AWS Regionen erstellt werden, in denen sie CodePipeline unterstützt werden. Eine Liste der unterstützten AWS Regionen für CodePipeline finden Sie unterKontingente in AWS CodePipeline.

Sie können die Konsole verwenden oder CloudFormation um regionsübergreifende Aktionen in Pipelines hinzuzufügen. AWS CLI

> **Note**
>
> Bestimmte Aktionstypen 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 übernimmt 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 die Standard-Artefakt-Buckets CodePipeline in den Regionen konfiguriert, in denen Sie Aktionen haben. Wenn Sie das AWS CLI, oder ein SDK verwenden 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
- Third-party Aktionen
- Benutzerdefinierte Aktionen

> **Note**
>
> Wenn Sie die regionsübergreifende Lambda-Aufrufaktion in verwenden CodePipeline, PutJobFailureResult sollte der Status der Lambda-Ausführung mit PutJobSuccessResult und 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 Event-Änderungen verwaltet werden, bleiben unverändert. Die Region, in der Ihre Pipeline gehostet wird, ändert sich nicht.

Verwalte regionsübergreifende 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 Sie eine benutzerdefinierte Pipeline (Konsole).

In der Konsole erstellen Sie eine regionsübergreifende Aktion in einer Pipeline-Phase, indem Sie den Aktionsanbieter und das Feld Region auswählen, in dem die Ressourcen aufgeführt sind, die Sie in dieser Region für diesen Anbieter erstellt haben. Wenn Sie eine regionsübergreifende Aktion hinzufügen, CodePipeline wird ein separater Artefakt-Bucket in der Region der Aktion verwendet. Weitere Informationen zu regionsübergreifenden Artefakt-Buckets finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

Fügt einer Pipeline-Phase (Konsole) eine regionsübergreifende Aktion hinzu

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 an http://console.aws.amazon.com/codesuite/codepipeline/home.
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 der Aktionsanbieter ein ist. AWS-Service Das Feld Region hat standardmäßig den gleichen Wert AWS-Region wie Ihre Pipeline.
   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 Sie eine regionsübergreifende 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 bei der Konsole unter an http://console.aws.amazon.com/codesuite/codepipeline/home.
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 Sie eine regionsübergreifende 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 an http://console.aws.amazon.com/codesuite/codepipeline/home.
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.

Fügen Sie einer Pipeline eine regionsübergreifende Aktion hinzu (CLI)

Sie können die 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. Sie müssen außerdem 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 exemplarischen Vorgehensweise und ihren Beispielen *`RegionA`* handelt es sich um die Region, in der die Pipeline erstellt wird. Sie hat Zugriff auf den *`RegionA`* Amazon S3-Bucket, der zum Speichern von Pipeline-Artefakten verwendet wird, sowie auf die von CodePipeline verwendete Servicerolle. *`RegionB`*ist die Region, in der die von verwendete CodeDeploy Anwendung, Bereitstellungsgruppe und Servicerolle erstellt CodeDeploy werden.

Voraussetzungen

Sie müssen Folgendes erstellt haben:

- Eine Pipeline ist da*RegionA*.
- Ein Amazon S3-Artefakt-Bucket ist da. *RegionB*
- Die Ressourcen für Ihre Aktion, z. B. Ihre CodeDeploy Anwendung und Bereitstellungsgruppe für eine regionsübergreifende Bereitstellungsaktion, in. *RegionB*

Fügen Sie einer Pipeline eine regionsübergreifende Aktion hinzu (CLI)

Verwenden Sie die AWS CLI , um einer Pipeline eine regionsübergreifende Aktion hinzuzufügen.

So fügen Sie eine regionsübergreifende Aktion hinzu

1. Führen Sie für einen Pipeline-Eingang den get-pipeline Befehl aus*RegionA*, 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. 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 `us-east-1` befindet.

   ```
   {
                   "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 im Mapping müssen sich die Ressourcen in der jeweiligen AWS Region befinden. Im folgenden Beispiel `ID-A` ist das die Verschlüsselungsschlüssel-ID für *`RegionA`* und `ID-B` ist 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 regionsübergreifende Aktion in der Konsole angezeigt.



Fügen Sie einer Pipeline eine regionsübergreifende Aktion hinzu (CloudFormation)

Sie können CloudFormation es verwenden, um einer vorhandenen Pipeline eine regionsübergreifende Aktion hinzuzufügen.

Um eine regionsübergreifende Aktion hinzuzufügen mit 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 als `us-west-2` und fügt den neuen *RegionB* Bucket hinzu: `eu-central-1`

```
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 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 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 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 CloudFormation Aktion, die auf Stack-Ausgabevariablen aus einer CloudFormation Upstream-Aktion verweist, finden Sie unterTutorial: Erstellen Sie eine Pipeline, die Variablen von verwendet AWS CloudFormation Aktionen zur Bereitstellung.
- 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
- Anzeigen von Ausgabevariablen
- Beispiel: Variablen in manuellen Genehmigungen verwenden
- Beispiel: Verwenden Sie eine BranchName Variable mit CodeBuild Umgebungsvariablen

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.

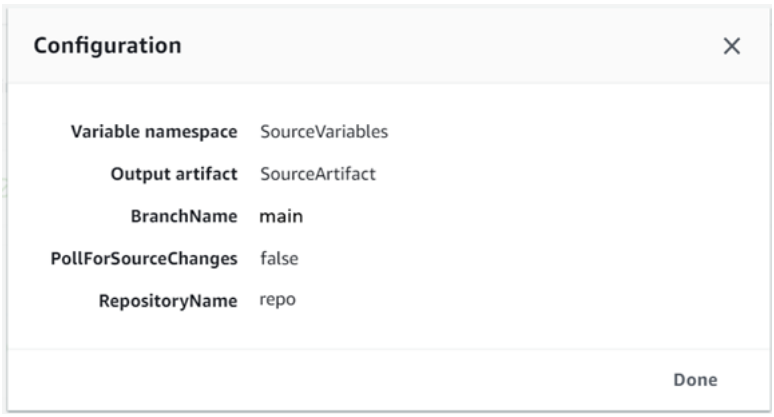
So erstellen Sie eine Pipeline mit Variablen

1. Melden Sie sich bei der an AWS-Managementkonsole 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 aus.
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-Managementkonsole 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-Managementkonsole 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 create-pipeline Befehl zum Erstellen einer Pipeline oder den update-pipeline Befehl zum Bearbeiten einer Pipeline verwenden, können Sie reference/use Variablen in der Konfiguration einer Aktion 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 Eine Pipeline, Phasen und Aktionen erstellen 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. Beispiel:

```
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. Beispiel:

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

Anzeigen von Ausgabevariablen

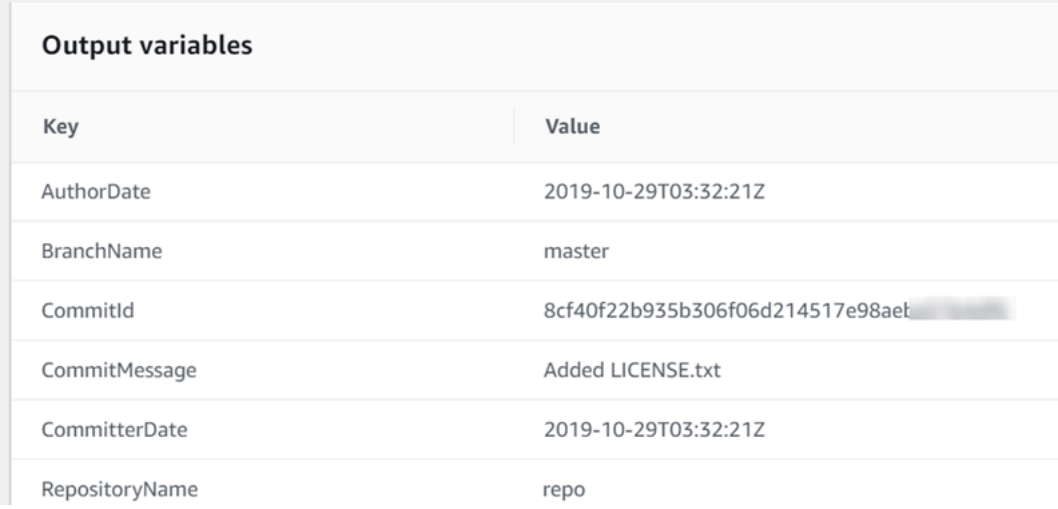
Sie können die Details der Aktionsausführung anzeigen, um die Variablen für diese Aktion anzuzeigen, spezifisch für jede Ausführung.

Anzeigen von Variablen (Konsole)

Sie können die Konsole verwenden, um Variablen für eine Aktion anzuzeigen.

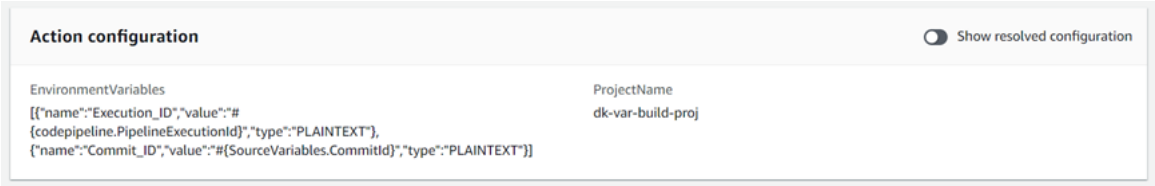
1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon. com/codesuite/codepipeline/home.

   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.

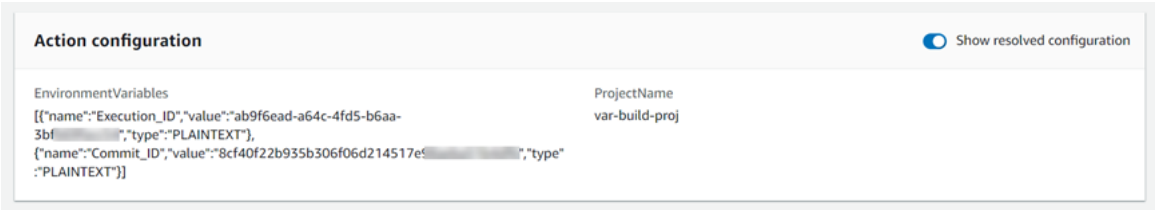


5. Nach erfolgreicher Ausführung der Pipeline können Sie die von der Build-Aktion erzeugten Variablen anzeigen. Wählen Sie View history (Verlauf anzeigen). Wählen Sie in der Aktionsliste

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-Managementkonsole 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. Beispielsweise lautet die Variablensyntax für eine CodeCommit-Aktion mit dem Standardnamespace `SourceVariables` `#{SourceVariables.CommitId}`.

   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-Managementkonsole 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 enthält, 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.

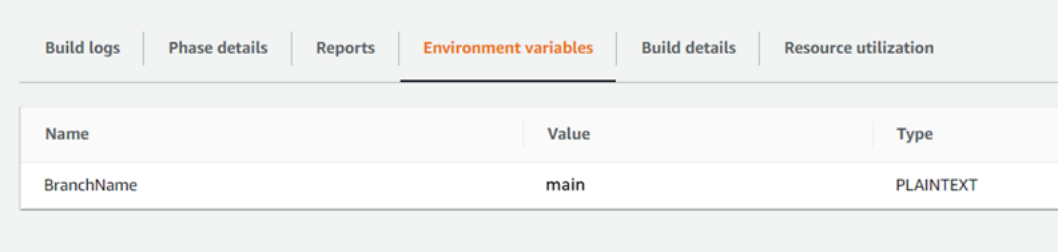
5. Nachdem die Pipeline erfolgreich ausgeführt wurde, können Sie sehen, wie die aufgelöste Ausgabevariable dem Wert in der Umgebungsvariablen entspricht. Wählen Sie eine der folgenden Optionen:

   - 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 Stufen 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 Hinzufügen einer manuellen Genehmigungsaktion zu einer Phase.

Themen

- Deaktivieren oder Aktivieren von Übergängen (Konsole)
- Deaktivieren oder Aktivieren von Übergängen (CLI)

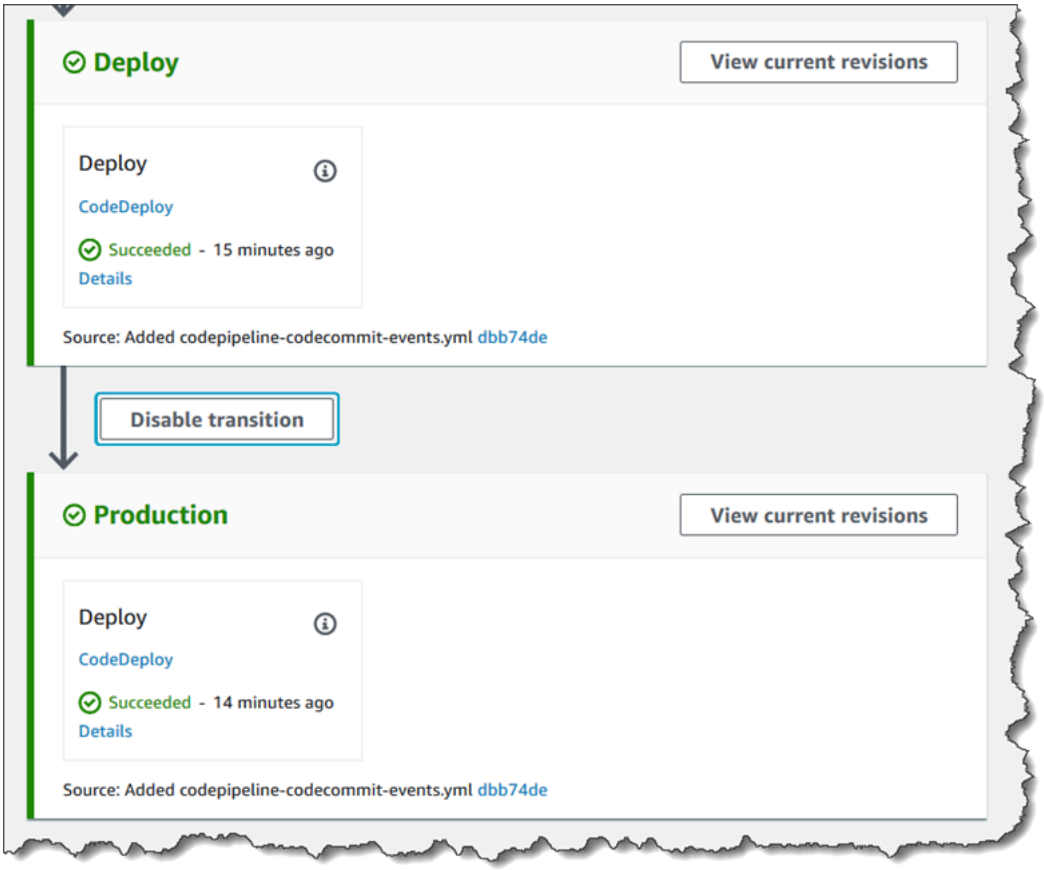
Deaktivieren oder Aktivieren von Übergängen (Konsole)

So deaktivieren oder aktivieren Sie Übergänge in einer Pipeline

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

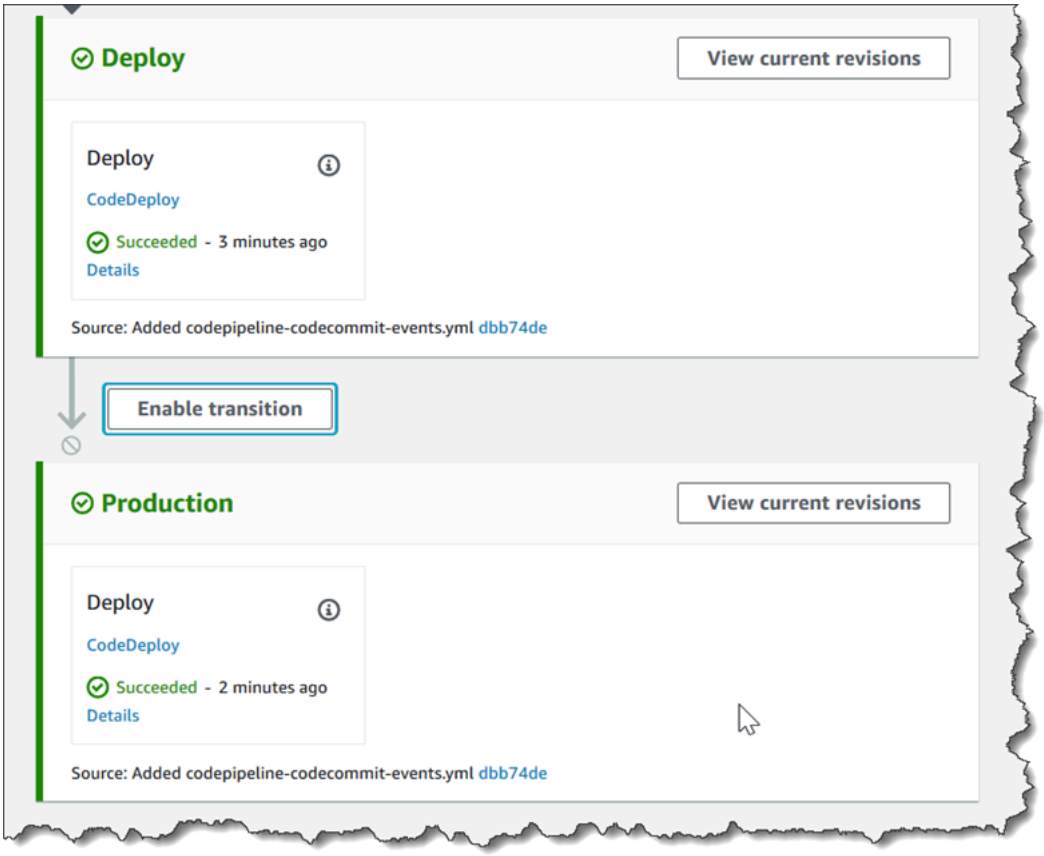
   Es werden die Namen aller Ihrem AWS -Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, für die Sie Übergänge aktivieren oder deaktivieren möchten. Eine detaillierte Ansicht der Pipeline wird geöffnet (einschließlich der Übergänge zwischen den Stufen der Pipeline).
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 Eingabeaufforderung (Windows) und verwenden Sie die, AWS CLI um den Befehl disable-stage-transition auszuführen. Geben Sie dabei den Namen der Pipeline, den Namen der Stufe, für die Sie Übergänge deaktivieren möchten, den Übergangstyp und den Grund für die Deaktivierung der Übergänge zu dieser Stufe 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 Stufe zu deaktivieren, die *Staging* in einer Pipeline mit dem Namen benannt ist, geben Sie einen Befehl ein*MyFirstPipeline*, 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 sicherzustellen, dass der Übergang deaktiviert wurde, zeigen Sie entweder die Pipeline in der CodePipeline -Konsole an oder führen den Befehl get-pipeline-state aus. Weitere Informationen erhalten 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 Eingabeaufforderung (Windows) und verwenden Sie die, AWS CLI um den Befehl enable-stage-transition auszuführen. Geben Sie dabei den Namen der Pipeline, den Namen der Stufe, für die Sie Übergänge aktivieren möchten, und den Übergangstyp an.

   Um beispielsweise den Übergang zu einer Stufe zu ermöglichen, die *Staging* in einer Pipeline mit dem Namen benannt ist*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 aktiviert wurde, sehen Sie sich entweder die Pipeline in der CodePipeline Konsole an oder führen Sie den get-pipeline-state Befehl aus. Weitere Informationen erhalten Sie unter Pipelines anzeigen (Konsole) und Anzeigen von der Pipeline-Details und des Verlaufs (CLI).

Überwachen 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

- Überwachung von CodePipeline Ereignissen
- Platzhalter-Bucket-Referenz für Ereignisse
- Protokollierung von CodePipeline API-Aufrufen mit AWS CloudTrail
- CodePipeline CloudWatch Metriken

Überwachung von CodePipeline Ereignissen

Sie können CodePipeline Ereignisse in überwachen EventBridge, was einen Strom von Echtzeitdaten aus Ihren eigenen Anwendungen, Software-as-a-Service (SaaS) -Anwendungen usw. liefert. AWS-Services EventBridge leitet diese Daten an Ziele wie Amazon Simple AWS Lambda Notification Service weiter. Bei diesen Ereignissen handelt es sich um dieselben, die in Amazon CloudWatch Events angezeigt werden. 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 zugrundeliegende Service und dieselbe API, EventBridge bieten jedoch mehr Funktionen. Änderungen, die Sie entweder in CloudWatch Events vornehmen oder EventBridge die auf jeder Konsole angezeigt werden.

Ereignisse bestehen aus Regeln. Eine Regel wird konfiguriert, indem Sie Folgendes auswählen:

- Event-Muster. Jede Regel wird als 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 Maßnahmen

zu ergreifen. Wenn Sie Ihr Ziel hinzufügen, müssen Sie ihm auch Berechtigungen gewähren, 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. Lesen Sie den GetPipelineExecution API-Aufruf 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 jedes Mal ein Ereignis, EventBridge wenn sich der Status einer Ressource in Ihrem Ordner AWS-Konto ändert. Ereignisse werden garantiert mindestens einmal für die folgenden Ressourcen ausgelöst:

- Pipeline-Ausführungen
- Hinrichtungen in Etappen
- Aktionsausführungen

Ereignisse werden EventBridge mit dem oben beschriebenen Ereignismuster und Schema ausgegeben. Bei verarbeiteten Ereignissen, z. B. Ereignissen, die Sie durch Benachrichtigungen erhalten, die Sie in der Entwicklertools-Konsole konfiguriert haben, enthält die Ereignisnachricht Felder für das Ereignismuster mit einigen Variationen. Das `detail-type` Feld wird beispielsweise in konvertiert`detailType`. Weitere Informationen finden Sie im PutEvents API-Aufruf in der EventBridge Amazon-API-Referenz.

Die folgenden Beispiele zeigen Ereignisse für CodePipeline. Wo möglich, zeigt jedes Beispiel das Schema für ein ausgegebenes Ereignis zusammen mit dem Schema für ein verarbeitetes Ereignis.

Themen

- Detailtypen
- Pipeline-level Ereignisse

- Stage-level Ereignisse
- Action-level Ereignisse
- Erstellen Sie eine Regel, die bei einem Pipeline-Ereignis eine Benachrichtigung sendet

Detailtypen

Wenn Sie Ereignisse zur 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	Description
CodePipeline Änderung des Ausführungsstatus der Pipeline	CANCELED	Die Pipelineausführung wurde abgebrochen, da die Pipelinestruktur aktualisiert wurde.
	FEHLGESCHLAGEN	Die Pipelineausführung wurde nicht erfolgreich abgeschlossen.

Typ des Details	Status	Description
	RESUMED	Eine fehlgeschlagene Pipelineausführung wurde als Reaktion auf den API-Aufruf `RetryStageExecution` erneut ausgeführt.
	STARTED	Die Pipelineausführung wird derzeit ausgeführt.
	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.

Typ des Details	Status	Description
	SUCCEEDED	Die Stufe wurde erfolgreich abgeschlossen.
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.

Pipeline-level Ereignisse

Pipeline-level Ereignisse werden ausgegeben, wenn sich der Status einer Pipeline-Ausführung ändert.

Themen

- Ereignis „Pipeline GESTARTET“
- Ereignis „Pipeline STOPING“
- Ereignis „Pipeline WAR ERFOLGREICH“
- Pipeline ERFOLGREICH (Beispiel mit Git-Tags)
- Ereignis Pipeline IST FEHLGESCHLAGEN
- Die Pipeline IST FEHLGESCHLAGEN (Beispiel mit Git-Tags)

Ereignis „Pipeline GESTARTET“

Wenn eine Pipeline-Ausführung beginnt, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die `"myPipeline"` in der `us-east-1`

Region benannte Pipeline. Das `id` Feld steht für die Ereignis-ID, und das `account` Feld steht für die Konto-ID, unter der die Pipeline erstellt wird.

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 „Pipeline STOPING“

Wenn eine Pipeline-Ausführung gestoppt 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 benannte Pipeline.

```
{
    "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 WAR ERFOLGREICH“

Wenn eine Pipeline-Ausführung erfolgreich ist, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die `myPipeline` in der Region benannte Pipeline. `us-east-1`

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 eine Phase hat, die wiederholt und erfolgreich versucht wurde, gibt sie 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 ist, in der sie für Git-Tags konfiguriert `execution-trigger` 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 IST FEHLGESCHLAGEN

Wenn eine Pipeline-Ausführung 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 benannte Pipeline.

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"
    }
```

Die Pipeline IST FEHLGESCHLAGEN (Beispiel mit Git-Tags)

Sofern sie nicht auf der Quellphase fehlschlägt, gibt sie bei einer Pipeline-Konfiguration mit Triggern 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 sie 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"
    }
```

Stage-level Ereignisse

Stage-level Ereignisse werden ausgegeben, wenn sich der Status einer Bühnenausführung ändert.

Themen

- Ereignis „Stage STARTED“
- Ereignis „Stage STOPING“
- Ereignis „Stage STOPED“
- Die Phase wurde nach dem Ereignis „Wiederholungsversuch“ wieder aufgenommen

Ereignis „Stage STARTED“

Wenn die Ausführung einer Phase beginnt, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die `"myPipeline"` in der `us-east-1` Region benannte Pipeline 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 STOPING“

Wenn die Ausführung einer Phase gestoppt 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 benannte Pipeline 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 „Stage STOPED“

Wenn die Ausführung einer Phase gestoppt wird, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die `myPipeline` in der `us-west-2` Region benannte Pipeline 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 Ereignis „Wiederholungsversuch“ wieder aufgenommen

Wenn die Ausführung einer Phase wieder aufgenommen wird und eine Phase erneut versucht wurde, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden 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, wenn ein Wiederholungsversuch durchgeführt wurde.

> **Note**
>
> Das `stage-last-retry-attempt-time` Feld wird in allen nachfolgenden Etappenereignissen angezeigt, nachdem eine Stufe 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
    }
}
```

Action-level Ereignisse

Action-level Ereignisse werden ausgelöst, wenn sich der Status einer Aktionsausführung ändert.

Themen

- Ereignis „Aktion GESTARTET“
- Ereignis „Aktion ERFOLGREICH“
- Ereignis „Aktion FEHLGESCHLAGEN“
- Aktion ABANDONED, Ereignis

Ereignis „Aktion GESTARTET“

Wenn eine Aktionsausführung beginnt, wird ein Ereignis ausgelöst, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die `myPipeline` in der `us-east-1` Region benannte 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": {}
}
```

Ereignis „Aktion ERFOLGREICH“

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 benannte Pipeline für 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 eine Aktionsausführung 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 benannte 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 ABANDONED, Ereignis

Wenn eine Aktionsausführung 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 benannte 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)
- Eine Benachrichtigung senden, wenn sich der Pipeline-Status ändert (CLI)

Eine Benachrichtigung senden, wenn sich der Pipeline-Status ändert (Konsole)

Diese Schritte zeigen, wie Sie mithilfe 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-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 CodePipeline Pipeline Execution State Change aus.
    - Für eine Regel, die für Ereignisse auf Etappenebene gilt, wählen Sie CodePipeline Stage Execution State Change aus.
    - Für eine Regel, die für Ereignisse auf Aktionsebene gilt, wählen Sie CodePipeline Action Execution State Change 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. Für Ereignismuster, 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**
   >
   > Wenn nicht anders angegeben, wird das Ereignismuster für alle pipelines/stages /actions und Status erstellt.

   Für detailliertere Ereignismuster können Sie die folgenden Beispielereignismuster kopieren und in das Fenster „Event-Muster“ 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 CodePipeline. Geben Sie im Feld Pipeline-ARN den Pipeline-ARN für die Pipeline ein, die nach 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 account pipeline-name*
    > Pipeline-Beispiel-ARN:
    > arn: aws: codepipeline: us-east- 2:80398 EXAMPLE:MyFirstPipeline

12. So erstellen oder spezifizieren Sie eine IAM-Servicerolle, 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 zum Starten Ihrer Pipeline-Ausführungen gewährt.
    - Wählen Sie Bestehende Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihnen EventBridge Berechtigungen zum Starten 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 mithilfe der CLI eine CloudWatch Ereignisregel erstellen, um Benachrichtigungen über Änderungen in 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 in allen Pipelines, die Sie für Ihr AWS Konto erstellen CodePipeline , 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 --event-pattern mit dem Namen erstellt"`MyPipelineStateChanges`", die das CloudWatch Ereignis ausgibt, wenn eine Pipelineausführung für die Pipeline mit dem Namen „MyPipeline“ fehlschlägt.

   ```
   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 ein:

   - Der Parameter `--rule` wird für den `rule_name` verwendet, den Sie mit put-rule erstellt haben.
   - Der `--targets` Parameter wird zusammen mit der Liste `Id` des Ziels in der Liste der Ziele und der Liste `ARN` des Amazon 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-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 Erkennung von Änderungen.

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 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

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

Protokollierung von CodePipeline API-Aufrufen mit AWS CloudTrail

AWS CodePipeline ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem AWS-Service Benutzer ausgeführten Aktionen bereitstellt CodePipeline. CloudTrail erfasst alle API-Aufrufe CodePipeline als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der CodePipeline Konsole und Codeaufrufe an die CodePipeline API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übertragung von CloudTrail Ereignissen an einen Amazon S3-Bucket aktivieren, einschließlich Ereignissen für CodePipeline. Wenn Sie keinen Trail konfigurieren, können Sie trotzdem die neuesten Ereignisse in der CloudTrail Konsole im Event-Verlauf einsehen. Anhand der gesammelten Informationen können Sie ermitteln CloudTrail, an welche Anfrage gestellt wurde CodePipeline, von welcher IP-Adresse 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 stattfindet CodePipeline, wird diese Aktivität zusammen mit anderen AWS-Service Ereignissen im CloudTrail Eventverlauf in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS Konto anzeigen, suchen und herunterladen. Weitere Informationen finden Sie unter Ereignisse mit dem CloudTrail Eventverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem Verzeichnis AWS-Konto, einschließlich der Ereignisse für CodePipeline, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon 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 in der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3-Bucket. Darüber hinaus können Sie andere so konfigurieren, AWS-Services dass die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter analysiert und entsprechend reagiert wird. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon 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 darin 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 CloudTrail userIdentity-Element.

Grundlegendes zu CodePipeline Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon 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 Logdateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, daher erscheinen sie in keiner bestimmten Reihenfolge.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für ein Aktualisierungspipeline-Ereignis, bei dem eine Pipeline mit dem Namen von dem Benutzer 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 sowohl das Element `requestParameters` als auch das Element `responseElements` 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":"amzn-s3-demo-source-bucket",
    "S3ObjectKey":"sampleapp_linux.zip"
```

```
      }
      }
     ],
       "name":"Source"
     },
     (...)
                },
 "responseElements":{
   "pipeline":{
     "version":2,
     (...)
           },
     "requestID":"2c4af5c9-7ce8-EXAMPLE",
     "eventID":""c53dbd42-This-Is-An-Example"",
     "eventType":"AwsApiCall",
     "recipientAccountId":"80398EXAMPLE"
     }
      ]
}
```

CodePipeline CloudWatch Metriken

Sie können Metriken verwenden CloudWatch , um CodePipeline Pipelines zu messen. Sie können die folgenden Metriken verwenden, um die Dauer von Pipelines und fehlgeschlagene Pipeline-Ausführungen zu messen.

Sie können Ihre Pipelines auf zwei Ebenen überwachen:

Pipeline-Ebene

Diese Kennzahlen beziehen sich auf Pipeline-Ebene. Wählen Sie In die Option Nach Pipeline aus CloudWatch. Verfügbare Pipelines werden unter aufgelistet. CloudWatch

AWS Kontoebene

Diese Metriken gelten für alle Pipelines in einem Konto. Um Metriken auf AWS Kontoebene anzuzeigen, wählen Sie Account-Metriken in CloudWatch aus.

Weitere Informationen zum Anzeigen von Metriken finden Sie unter Verfügbare Metriken anzeigen im CloudWatch Amazon-Benutzerhandbuch.

Themen

- PipelineDuration
- FailedPipelineExecutions

PipelineDuration

Die `PipelineDuration` Metrik misst die Dauer der Pipeline-Ausführung. Diese Metrik ist nur für die Pipeline-Ebene verfügbar.

Einheiten: Sekunden

FailedPipelineExecutions

Die `FailedPipelineExecutions` Metrik misst die Anzahl der fehlgeschlagenen Pipeline-Ausführungen. Diese Metrik ist auf Pipeline- und Kontoebene verfügbar.

Diese Metrik beinhaltet eine separate Anzahl für Pipeline-Ausführungen mit einem erneuten Versuch einer fehlgeschlagenen Aktion. Mit anderen Worten, wenn eine fehlgeschlagene Aktion in einer Pipeline wiederholt wird, wird dies als separate Metrik zur Pipeline-Ausführung gezählt. Bei einer Pipeline mit einer S3-Quellaktion, bei der die S3-Quellaktion beim ersten Mal fehlschlägt, wird eine Ausführung ausgeführt, und dann wird die Quellaktion erneut versucht und schlägt erneut fehl. In diesem Beispiel lautet die Metrik wie folgt:

```
FailedPipelineExecutionAttempts: 2
```

Einheiten: Anzahl

Problembehebung CodePipeline

Die folgenden Informationen helfen Ihnen möglicherweise bei der Lösung häufiger Probleme in AWS CodePipeline.

Themen

- Pipeline-Fehler: Eine Pipeline, die konfiguriert ist mit AWS Elastic Beanstalk gibt eine Fehlermeldung zurück: „Die Bereitstellung ist fehlgeschlagen. Die angegebene Rolle verfügt nicht über ausreichende Berechtigungen: Service:AmazonElasticLoadBalancing“
- Bereitstellungsfehler: Eine Pipeline, die mit einem konfiguriert ist AWS Elastic Beanstalk Die Bereitstellungsaktion bleibt hängen und schlägt nicht fehl, wenn die DescribeEvents "" - Berechtigung fehlt
- Pipeline-Fehler: Eine Quellaktion gibt die Meldung mit unzureichenden Berechtigungen zurück: „Auf den Repository-Namen konnte nicht zugegriffen werden. CodeCommit Überprüfen Sie, ob die Pipeline-IAM-Rolle über ausreichende Berechtigungen für den Zugriff auf das Repository verfügt.“
- Pipeline-Fehler: Ein Jenkins-Build oder eine Testaktion werden über eine lange Zeit ausgeführt und schlagen dann aufgrund fehlender Anmeldeinformationen oder Berechtigungen fehl.
- Pipeline-Fehler: Eine Pipeline wurde in einer Pipeline erstellt AWS Region, die einen in einem anderen erstellten Bucket verwendet AWS Region gibt ein "InternalError" mit dem Code "JobFailed“ zurück
- Bereitstellungsfehler: Eine ZIP-Datei, die eine WAR-Datei enthält, wurde erfolgreich bereitgestellt für AWS Elastic Beanstalk, aber die Anwendungs-URL meldet einen 404-Fehler „Nicht gefunden“
- Pipeline-Artefakt-Ordnernamen werden gekürzt
- Fügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub Enterprise Server oder hinzu GitHub GitLab.com
- Fügen Sie CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen hinzu
- Pipeline-Fehler: Bei einer Bereitstellung mit der CodeDeployToECS Aktion wird die folgende Fehlermeldung zurückgegeben: „Ausnahme beim Versuch, die Artefaktdatei für die Aufgabendefinition zu lesen aus: Name des < Quellartefakts“ >
- GitHub Quellaktion (über die OAuth-App): Die Repository-Liste zeigt verschiedene Repositorys
- GitHub Quellaktion (über GitHub App): Die Verbindung für ein Repository konnte nicht hergestellt werden
- Amazon S3 S3-Fehler: Der < ARN für die CodePipeline Servicerolle > erhält S3-Zugriff für den S3-Bucket verweigert < BucketName >

- 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
- Die EC2-Bereitstellungsaktion schlägt fehl und es wird die Fehlermeldung Keine solche Datei angezeigt
- Die EKS Deploy-Aktion schlägt fehl und es wird die Fehlermeldung „Cluster nicht erreichbar“ angezeigt
- Die Pipeline wird nicht für alle Branches ausgelöst, wenn mehrere Quellaktionen auf dasselbe Repository verweisen
- Benötigen Sie Hilfe für ein anderes Problem?

Pipeline-Fehler: Eine Pipeline, die konfiguriert ist mit AWS Elastic Beanstalk gibt eine Fehlermeldung zurück: „Die Bereitstellung ist 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 Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu 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 Pipeline, die mit einem konfiguriert ist AWS Elastic Beanstalk Die Bereitstellungsaktion 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 Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu, 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 den Repository-Namen konnte nicht zugegriffen werden. `CodeCommit` 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 Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu.

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 wurde in einer Pipeline erstellt AWS Region, die einen in einem anderen erstellten Bucket verwendet AWS Region gibt ein "InternalError" mit dem Code "JobFailed“ zurück

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 für AWS Elastic Beanstalk, aber die Anwendungs-URL meldet einen 404-Fehler „Nicht gefunden“

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. Beispiel:

```
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, GitHub Enterprise Server oder hinzu GitHub GitLab.com

Wenn du eine Aktion AWS CodeConnections in einer Quelle und eine CodeBuild Aktion verwendest, gibt es zwei Möglichkeiten, wie das Eingabeartefakt an den Build übergeben werden kann:

- Standard: Die Quellaktion erstellt eine ZIP-Datei, die den CodeBuild heruntergeladenen Code enthält.
- Vollständiger Klon: Der Quellcode kann direkt in die Build-Umgebung heruntergeladen werden.

  Der vollständige Klonmodus 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 gewähren.

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 angegeben wird, der Verbindungs-ARN im `Resource` Feld angegeben wird und die Quell-Repository-ID über begrenzt wird`Condition`.

Um die Konsole zum Hinzufügen der UseConnection Berechtigungen zu verwenden

1. Um den Verbindungs-ARN und die Quell-Repository-ID für Ihre Pipeline zu finden, öffnen Sie Ihre Pipeline, klicken Sie auf das (i) -Symbol in Ihrer Quellaktion und wechseln Sie zur Registerkarte Eingabe.

   Ein Beispiel für eine Verbindungs-ARN ist:

   ```
   arn:aws:codeconnections:eu-central-1:123456789123:connection/
   sample-1908-4932-9ecc-2ddacee15095
   ```

   Ein Beispiel für die Quell-Repository-ID:

   ```
   owner/test-app
   ```

Sie fügen den Verbindungs-ARN und die Repository-ID zu Ihrer CodeBuild Servicerollenrichtlinie hinzu.

2. Um Ihre CodeBuild Servicerolle zu finden, wählen Sie das in Ihrer Pipeline verwendete Build-Projekt aus 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 Berechtigungen hinzufügen und anschließend Inline-Richtlinie erstellen aus.

   Verwenden Sie die folgende Beispielrichtlinienvorlage. Fügen Sie Ihren Verbindungs-ARN in das `Resource` Feld und Ihre Repository-ID in `codeconnections:FullRepositoryId` das `Condition` Feld ein, wie in diesem Beispiel gezeigt:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codeconnections:UseConnection",
            "Resource": "arn:aws:codeconnections:eu-
central-1:123456789123:connection/my-connection-id",
            "Condition": {
                "StringEquals": {
                    "codeconnections:FullRepositoryId": "my-repository-id"
                }
            }
        }
    ]
}
```

   Verwenden Sie das `Condition` Feld, um Ihre Policy-Berechtigungen auf der Grundlage Ihrer Build-Spezifikationen weiter einzuschränken (siehe Dokumentation zu den `CodeConnection` Bedingungen).

   Fügen Sie auf der Registerkarte JSON Ihre Richtlinie ein.

5. Wählen Sie Weiter aus. Geben Sie einen Namen für die Richtlinie ein (beispielsweise **`connection-permissions`**) und wählen Sie dann Create policy (Richtlinie erstellen) aus.

   Im Anhang zu Ihren **`connection-permissions`** Rollenberechtigungsrichtlinien wird eine Richtlinie angezeigt.

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.

Pipeline-Fehler: Bei einer Bereitstellung mit der CodeDeployToECS Aktion wird die folgende Fehlermeldung zurückgegeben: „Ausnahme beim Versuch, die Artefaktdatei für die Aufgabendefinition zu lesen aus: Name des < Quellartefakts“ >

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 (über die OAuth-App): Die Repository-Liste zeigt verschiedene Repositorys

Problem:

Nach erfolgreicher Autorisierung für eine Aktion GitHub (über die OAuth-App) in der CodePipeline Konsole können Sie aus einer Liste Ihrer Repositorys wählen. GitHub Wenn die Liste nicht die Repositorys enthält, die Sie erwartet haben, können Sie Probleme mit dem für die Autorisierung verwendeten Konto beheben.

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 (über GitHub App): Die Verbindung für ein Repository konnte nicht hergestellt werden

Problem:

Da bei einer Verbindung zu einem GitHub Repository der AWS Connector für verwendet wird GitHub, benötigen Sie die Rechte des Organisationsinhabers oder Administratorberechtigungen für das Repository, um die Verbindung herzustellen.

Mögliche Korrekturen: Informationen zu den Berechtigungsstufen für ein GitHub Repository finden Sie unter https://docs.github.com/en/free-pro-team@ latest/github teams/permission /setting-up-and-managing-organizations-and- -levels-for-an-organization.

Amazon S3 S3-Fehler: Der < ARN für die CodePipeline Servicerolle > erhält S3-Zugriff für den S3-Bucket verweigert < BucketName >

Problem:

Während der Ausführung 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 Für die Servicerolle „arn:aws:iam::*`AccountID`*: role/service -role/*`RoleID`*" wurde der S3-Zugriff für den S3-Bucket verweigert "“ *`BucketName`*

In den CloudTrail Protokollen für die Aktion wird auch der Fehler protokolliert. `AccessDenied`

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 beschriebenFügen Sie der CodePipeline Servicerolle Berechtigungen hinzu.
- 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, bei dem sich die Beispielrollen-ID für die Servicerolle befindet*AROAEXAMPLEID*.

     ```
     {
         "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.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringLike": {
                    "aws:userid": "AROAEXAMPLEID:*"
                }
            }
        },
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "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:

Aktion an der Quelle	Parameter/Trigger-Identifikatoren (Konsole)
Amazon ECR	Repository-Name Bild-Tag
Amazon S3	Bucket S3-Objektschlüssel
CodeCommit	Repository-Name

Aktion an der Quelle	Parameter/Trigger-Identifikatoren (Konsole)
	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 `release-branch` in`2nd-release-branch`. Vermeiden Sie eine Änderung am Ende des Namens, `release-branch-2` z.
- Ändern Sie die CodeCommit/S3/ECR Konfigurationseinstellungen für jede Pipeline.

  Beispiel: Ändern Sie Ihren Filialnamen `myRepo/myBranch` in`myDeployRepo/myDeployBranch`. Vermeiden Sie eine Änderung am Ende des Namens, `myRepo/myBranch`*`2`* z.
- Verwenden Sie statt der Konsole die CLI oder CloudFormation um Ihre Regeln für Änderungserkennungsereignisse zu erstellen und zu aktualisieren. Anweisungen zum Erstellen von Ereignisregeln für eine S3-Quellaktion finden Sie unter. Verbindung zu Amazon S3-Quellaktionen herstellen, die EventBridge und verwenden AWS CloudTrail Anweisungen zum Erstellen von Ereignisregeln für eine Amazon ECR-Aktion finden Sie unterAmazon ECR-Quellaktionen und Ressourcen EventBridge. Anweisungen zum Erstellen von Ereignisregeln für eine CodeCommit Aktion finden Sie unterCodeCommit 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 AWS CodeConnections Verbindung dadurch nicht in der Lage ist, die geänderten Dateien aufzulösen. Wenn die Git-Konfiguration für den Trigger so eingerichtet ist, dass sie nach Dateipfaden filtert,

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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen.

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 sie nach Dateipfaden filtert, startet die Pipeline daher möglicherweise nicht, wenn mehr als 100 Dateien vorhanden sind. Weitere Informationen zum Filtern nach Dateipfaden finden Sie unterTrigger mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen.

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.

Die EC2-Bereitstellungsaktion schlägt fehl und es wird die Fehlermeldung **`Keine solche Datei`** angezeigt

Beschreibung: Nachdem die EC2-Bereitstellungsaktion die Artefakte im Zielverzeichnis der Instances entpackt hat, führt die Aktion das Skript aus. Wenn sich das Skript im Zielverzeichnis befindet, die Aktion das Skript jedoch nicht ausführen kann, schlägt die Aktion auf dieser Instance fehl und die Bereitstellung der verbleibenden Instances schlägt fehl.

In den Protokollen einer Bereitstellung, bei der sich das Zielverzeichnis `/home/ec2-user/deploy/` und der Quell-Repository-Pfad befinden, wird ein Fehler ähnlich den folgenden Fehlermeldungen angezeigt`myRepo/postScript.sh`.

- ```
  Instance i-0145a2d3f3EXAMPLE is FAILED on event AFTER_DEPLOY, message:
  ----------ERROR-------
  chmod: cannot access '/home/ec2-user/deploy/myRepo/postScript.sh': No such file or
   directory
  /var/lib/<path>/_script.sh: line 2: /home/ec2-user/deploy/myRepo/postScript.sh: No
   such file or directory
  failed to run commands: exit status 127
  ```
- ```
  Executing commands on instances i-0145a2d3f3EXAMPLE, SSM command id <ID>, commands:
   chmod u+x /home/ec2-user/deploy/script.sh
  ----------ERROR-------: No such file or directory
  ```

Ergebnis: Die Bereitstellungsaktion schlägt in der Pipeline fehl.

Mögliche Lösungen: Gehen Sie wie folgt vor, um Fehler zu beheben.

1. Sehen Sie sich die Protokolle an, um zu überprüfen, welche Instanz den Skriptfehler verursacht hat.
2. Ändern Sie das Verzeichnis (`cd`) in das Zielverzeichnis auf Ihrer Instance. Führen Sie das Skript testweise auf der Instanz aus.

3. Bearbeiten Sie in Ihrem Quell-Repository die Skriptdatei, um alle Kommentare oder Befehle zu entfernen, die das Problem verursachen könnten.

Die EKS Deploy-Aktion schlägt fehl und es wird die Fehlermeldung „`Cluster nicht erreichbar`“ angezeigt

Beschreibung: Nachdem die EKS-Bereitstellungsaktion ausgeführt wurde, schlägt die Aktion mit einer `cluster unreachable` Fehlermeldung fehl. Die Meldung zeigt ein Zugriffsproblem auf dem Cluster aufgrund fehlender Berechtigungen. Je nach Dateityp (Helm-Diagramm oder Kubernetes-Manifestdateien) wird die Fehlermeldung wie folgt angezeigt.

- Bei einer EKS-Bereitstellungsaktion, die ein Helm-Diagramm verwendet, wird ein Fehler ähnlich der folgenden angezeigt.

```
error message:

helm upgrade --install my-release test-chart --wait
Error: Kubernetes cluster unreachable: the server has asked for the client to provide
 credentials
```

- Bei einer EKS-Bereitstellungsaktion, die Kubernetes-Manifestdateien verwendet, wird ein Fehler ähnlich der folgenden angezeigt.

```
kubectl apply -f deployment.yaml
Error: error validating "deployment.yaml": error validating data: failed to download
 openapi: the server has asked for the client to provide credentials
```

Ergebnis: Die Bereitstellungsaktion schlägt in der Pipeline fehl.

Mögliche Lösungen: Wenn Sie eine vorhandene Rolle verwenden, muss die CodePipeline Servicerolle mit den erforderlichen Berechtigungen aktualisiert werden, um die EKS-Bereitstellungsaktion verwenden zu können. Um der CodePipeline Servicerolle Zugriff auf Ihren Cluster zu gewähren, müssen Sie Ihrem Cluster außerdem einen Zugriffseintrag hinzufügen und die Servicerolle für den Zugriffseintrag angeben.

1. Stellen Sie sicher, dass die CodePipeline Servicerolle über die erforderlichen Berechtigungen für die EKS-Bereitstellungsaktion verfügt. Die Referenz zu den Berechtigungen finden Sie unter Berechtigungen für Richtlinien zur Servicerolle.

2. Fügen Sie Ihrem Cluster einen Zugriffseintrag hinzu und geben Sie die CodePipeline Servicerolle für den Zugriff an. Ein Beispiel finden Sie unter Schritt 4: Erstellen Sie einen Zugriffseintrag für die CodePipeline Servicerolle.

Die Pipeline wird nicht für alle Branches ausgelöst, wenn mehrere Quellaktionen auf dasselbe Repository verweisen

Problem: Eine Pipeline hat mehrere Quellaktionen, die Verbindungen verwenden (z. B. einen Bitbucket oder eine GitLab Verbindung), wobei jede Quellaktion auf einen anderen Zweig desselben Repositorys verweist. GitHub Nur einer der Branches löst die Pipeline aus, wenn du Änderungen pushst. Das Webhook-Abonnement der Verbindung ist für die Kombination aus Pipeline und Repository registriert, nicht pro Zweig. Daher werden mehrere Quellaktionen, die auf verschiedene Zweige desselben Repositorys innerhalb einer einzigen Pipeline abzielen, nicht unterstützt.

Mögliche Lösungen: Verwenden Sie für jeden Branch, den Sie unabhängig auslösen möchten, eine separate Pipeline.

Benötigen Sie Hilfe für ein anderes Problem?

Sie haben folgende weitere Möglichkeiten:

- AWS Support kontaktieren.
- Stellen Sie eine Frage im CodePipeline-Forum.
- 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 bei AWS CodePipeline

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

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, auf der AWS Dienste in der ausgeführt AWS Cloud werden. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Third-party Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für gelten AWS CodePipeline, finden Sie unter AWS Services im Umfang nach Compliance-Programmen AWS .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, einschließlich der Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Nutzung 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 AWS Dienste nutzen können, die Sie bei der Überwachung und Sicherung Ihrer CodePipeline Ressourcen unterstützen.

Topics

- Datenschutz in AWS CodePipeline
- Identitäts- und Zugriffsmanagement für AWS CodePipeline
- Anmeldung und Überwachung CodePipeline
- Konformitätsvalidierung für AWS CodePipeline
- Resilienz in AWS CodePipeline
- Sicherheit der Infrastruktur in AWS CodePipeline
- Bewährte Methoden für die Gewährleistung der Sicherheit

Datenschutz in AWS CodePipeline

Das Modell der AWS gemeinsamen Verantwortung gilt für den Datenschutz in. 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 . Weitere Informationen zum Datenschutz in Europa finden Sie im Zentrum für die Datenschutz-Grundverordnung (DSGVO).

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).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- 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 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

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 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 umfassen ebenfalls den Adressdatenschutz in CodePipeline:

- Konfigurieren Sie die serverseitige Verschlüsselung für in Amazon S3 gespeicherte Artefakte für CodePipeline
- Verwenden Sie AWS Secrets Manager um Datenbankpasswörter oder API-Schlüssel von Drittanbietern zu verfolgen

Richtlinie für den Datenverkehr zwischen Netzwerken

Amazon VPC ist eine AWS-Service , die Sie verwenden können, um AWS Ressourcen in einem von Ihnen definierten virtuellen Netzwerk (virtuelle private Cloud) zu starten. CodePipelineunterstützt Amazon VPC-Endpoints powered by AWS PrivateLink, eine AWS Technologie, die die private Kommunikation zwischen AWS-Services einer 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 war für Anwendungen, die innerhalb einer VPC ausgeführt wurden, ein Internetzugang erforderlich, um eine Verbindung mit CodePipeline herzustellen. 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

Die gesamte Kommunikation zwischen Diensten wird bei der Übertragung verschlüsselt mithilfe von. SSL/TLS

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. Die folgende Bucket-Richtlinie verweigert beispielsweise allen Benutzern die Berechtigung zum Hochladen von Objekt (`s3:PutObject`), wenn die Anfrage nicht den `x-amz-server-side-encryption` Header enthält, mit dem die serverseitige Verschlüsselung angefordert wird. SSE-KMS

JSON

```
{
    "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 durch Verschlüsselung und AWS KMS Schützen von Daten mithilfe serverseitiger Server-Side Verschlüsselung mit KMS-Schlüsseln, die in () gespeichert sind. AWS Key Management Service SSE-KMS

Weitere Informationen zu AWS KMS finden Sie im AWS Key Management Service Entwicklerhandbuch.

Themen

- Sehen Sie sich Ihre an Von AWS verwalteter Schlüssel
- Konfigurieren Sie die serverseitige Verschlüsselung für S3-Buckets mit CloudFormation oder das AWS 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 dasselbe SSE-KMS Verfahren, um das Artefakt zu entschlüsseln.

Um Informationen zu Ihrem einzusehen Von AWS verwalteter Schlüssel

1. Melden Sie sich bei der an AWS-Managementkonsole 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 CloudFormation oder das AWS CLI

Wenn Sie 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 einer anderen verwendet werden AWS Konto.

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.

Verwenden Sie AWS Secrets Manager um Datenbankpasswörter 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 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 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/.

Identitäts- und Zugriffsmanagement für AWS CodePipeline

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf AWS Ressourcen sicher zu 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
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- Wie AWS CodePipeline funktioniert mit IAM
- AWS CodePipeline Beispiele für identitätsbasierte Richtlinien
- AWS CodePipeline Beispiele für ressourcenbasierte Richtlinien
- Fehlerbehebung AWS CodePipeline Identität und Zugriff
- Berechtigungsreferenz
- Verwalten Sie die CodePipeline Servicerolle

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe Fehlerbehebung AWS CodePipeline Identität und Zugriff).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe Wie AWS CodePipeline funktioniert mit IAM).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe 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 sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

Root-Benutzer des AWS-Kontos

Wenn Sie eine erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem so genannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle Ressourcen hat. AWS-Services Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer für den Zugriff AWS mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2

laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM 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 definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identity-based Richtlinien

Identity-based Richtlinien sind Richtliniendokumente für JSON-Berechtigungen, die Sie an eine Identität (Benutzer, Gruppe oder Rolle) anhängen. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identity-based Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mehreren Identitäten zugeordnet sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Resource-based Richtlinien

Resource-based Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen 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. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Resource-based Richtlinien sind Inline-Richtlinien, die sich in diesem Dienst befinden. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs) – SCPs legen die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in AWS Organizations fest. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Ressourcen-Kontrollrichtlinien (RCPs) – RCPs definieren die maximale Anzahl an Berechtigungen, die Ressourcen in Ihren Konten zur Verfügung stehen. Weitere Informationen finden Sie unter Ressourcen-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. 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

- CodePipeline identitätsbasierte Richtlinien
- CodePipeline ressourcenbasierte Richtlinien
- Autorisierung auf der Grundlage von Tags CodePipeline
- CodePipeline IAM-Rollen

CodePipeline identitätsbasierte Richtlinien

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen erteilt 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 Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs 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 heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Ressourcen und Abläufe

In ist die primäre Ressource eine Pipeline. In einer Richtlinie identifizieren Sie die Ressource, für welche die Richtlinie gilt, mithilfe eines Amazon-Ressourcennamens (ARN). unterstützt weitere Ressourcen, die mit der primären Ressource verwendet werden können, wie 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 account pipeline-name*
Stage	arn:aws:codepipeline:::/*regionaccountpipeline-name stage-name*
Action	arn:aws:codepipeline:::/*regionaccountpipeline-name stage-name action-name*

Ressourcentyp	ARN-Format
Benutzerdefinierte Aktion	arn:aws:codepipeline: ::Aktions typ:///*regionaccountownercategoryproviderversion*
Alle Ressourcen	arn:aws:codepipeline: *
Alle -Ressourcen, die sich im Besitz des angegebenen Kontos in der angegebenen Region befinden	arn:aws:codepipeline::: * *region account*

> **Note**
>
> Die meisten Dienste in AWS behandeln einen Doppelpunkt (:) oder einen Schrägstrich (/) als dasselbe Zeichen in ARNs. verwendet jedoch eine genaue Übereinstimmung in 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, die Berechtigungen auf Ressourcenebene unterstützen. Resource-level Berechtigungen geben an, ob ein API-Aufruf einen Ressourcen-ARN angeben kann oder ob der API-Aufruf nur alle Ressourcen mithilfe des Platzhalters angeben kann. Eine Berechtigungsreferenz ausführliche Beschreibung der Berechtigungen auf Ressourcenebene und eine Liste der CodePipeline API-Aufrufe, die Berechtigungen auf Ressourcenebene unterstützen, finden Sie unter.

Sie können beispielsweise eine bestimmte Pipeline (*myPipeline*) in Ihrer Anweisung angeben, indem Sie ihren ARN wie folgt verwenden:

```
"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**
>
> Wenn Sie IAM-Richtlinien erstellen, befolgen Sie die standardmäßigen Sicherheitsempfehlungen, 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 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"]
```

stellt eine Reihe von Operationen bereit, um mit den Ressourcen zu arbeiten. Eine Liste der verfügbaren Operationen finden Sie unter Berechtigungsreferenz.

Bedingungsschlüssel

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. 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. 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 AWS CodePipeline Beispiele für ressourcenbasierte Richtlinien

Autorisierung auf der Grundlage von Tags CodePipeline

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 über das Markieren von CodePipeline - Ressourcen mit Tags finden Sie unter Taggen 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

IAM-Benutzer besitzen keine Berechtigungen zum Erstellen oder Ändern von CodePipeline -Ressourcen. Sie können auch keine Aufgaben mit der AWS-Managementkonsole 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 einer anderen verwendet werden AWS Konto.

Themen

- Best Practices für Richtlinien
- Anzeigen von Ressourcen in der Konsole

- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer
- Identity-based Beispiele für Richtlinien (IAM)
- Verwenden von Tags zur Steuerung des Zugriffs auf Ressourcen CodePipeline
- Erforderliche Berechtigungen für die Verwendung der Konsole
- Zum Anzeigen von Rechenprotokollen in der Konsole sind Berechtigungen erforderlich
- AWS verwaltete Richtlinien für AWS CodePipeline
- Beispiele für vom Kunden verwaltete Richtlinien

Best Practices für Richtlinien

Identity-based 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. Beachten Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads 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 Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen 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. 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 mit 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 Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Anzeigen von Ressourcen in der Konsole

Die Konsole benötigt die `ListRepositories` Erlaubnis, eine Liste der 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:

- AWS 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": "*"
        }
    ]
}
```

Identity-based Beispiele für 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 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:

JSON

```
{
  "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. 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 eine der verwalteten Richtlinien in Betracht ziehen, die von erstellt und verwaltet wurden. 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`

JSON

```
{
  "Version":"2012-10-17",
  "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": {
        "ArnLike": {
          "codestar-notifications:NotificationsForResource":
 "arn:aws:iam::*:role/Service*"
        }
      }
    }
  ]
}
```

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 Kontos ermöglicht, Rollen zu übernehmen, die im *111111111111* AWS Konto 111222333444 definiert sind:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Das folgende Beispiel zeigt eine im Konto erstellte Richtlinie, die es Benutzern ermöglicht, die *CrossAccountPipelineViewers* im *111111111111* AWS Konto 111222333444 angegebene Rolle anzunehmen:

JSON

```
{
    "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 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 CodePipeline Aktionen erforderlich sind. 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. Taggen 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
- war: RequestTag/tag-key
- Zustandsschlüssel für CodePipeline

Die folgenden Beispiele zeigen, wie Sie Tag-Bedingungen in Richtlinien für CodePipeline -Benutzer festlegen.

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 auch für nicht autorisierte Benutzer mit Administratorrechten einrichten.

JSON

```
{
  "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.

JSON

```
{
  "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

Mit der folgenden Richtlinie wird Benutzern die Berechtigung zum Erstellen von Entwicklungs-Pipelines in CodePipeline erteilt.

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 Wert muss sein`ProjectA`.

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.

JSON

```
{
  "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.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Erforderliche Berechtigungen für die Verwendung der Konsole

Um in der -Konsole zu verwenden, müssen Sie über einen Mindestsatz von Berechtigungen der folgenden Services verfügen:

- 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
- AWS CodeBuild
- 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 Konsole weiterhin verwenden können, fügen Sie dem Benutzer auch die verwaltete Richtlinie `AWSCodePipeline_ReadOnlyAccess` an. Einzelheiten dazu finden Sie unter AWS verwaltete Richtlinien für AWS CodePipeline.

Sie müssen Benutzern, die die API AWS CLI oder die API aufrufen, keine Mindestberechtigungen für die Konsole gewähren.

Zum Anzeigen von Rechenprotokollen in der Konsole sind Berechtigungen erforderlich

Um die Protokolle in der Aktion Befehle auf der CodePipeline Konsole anzeigen zu können, muss die Konsolenrolle über Berechtigungen verfügen. Um die Protokolle in der Konsole anzuzeigen, fügen Sie der Konsolenrolle die `logs:GetLogEvents` entsprechenden Berechtigungen hinzu.

Beschränken Sie die Berechtigungen in der Richtlinienerklärung zur Konsolenrolle auf die Pipeline-Ebene, wie im folgenden Beispiel gezeigt.

```
{
    "Effect": "Allow",
    "Action": [
        "Action": "logs:GetLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

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 vom Kunden verwaltete 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. Das 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 Amazon Q Developer in Chat-Anwendungen zu verwalten.

- `cloudformation`— Erteilt Berechtigungen, die es Prinzipalen ermöglichen, Ressourcenstapel in zu verwalten. 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 für Bereitstellungen CodeCatalyst zur Verwaltung von Elastic Load Balancing in Amazon EC2.
- `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.

Informationen zur Richtlinie finden Sie unter AWSCodePipeline_ FullAccess.

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.

Die Richtlinie finden Sie unter AWSCodePipeline _. ReadOnlyAccess

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.

Informationen zur Richtlinie finden Sie unter AWSCodePipelineApproverAccess.

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.

Informationen zur Richtlinie finden Sie unter AWSCodePipelineCustomActionAccess.

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. 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 Amazon Q Developer in Chat-Anwendungsclients auflisten, die für Slack konfiguriert sind.

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 für AWS Verwaltete Richtlinien

Hier finden Sie Informationen zu Aktualisierungen der AWS verwalteten Richtlinien, die CodePipeline seit Beginn der Nachverfolgung dieser Änderungen durch diesen Dienst vorgenommen wurden. Um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der CodePipeline Seite -Dokumentverlauf.

Änderungen	Beschreibung	Date
AWSCodePipeline_ FullAccess — Aktualisierungen der bestehenden Richtlinie	CodePipeline dieser Richtlinie wurde die Erlaubnis hinzugefügt, `ListStacks` in zu unterstützen CloudFormation.	15. März 2024
AWSCodePipeline_ FullAccess — Aktualisierungen der bestehenden Richtlinie	Diese Richtlinie wurde aktualisiert, um Berechtigungen für Amazon Q Developer in Chat-Anwendungen hinzuzufügen. Weitere Informationen finden Sie unter CodePipeline verwaltete Richtlinien und Benachrichtigungen.	21. Juni 2023
AWSCodePipeline_ FullAccess und AWSCodePipeline_ ReadOnlyAccess verwaltete Richtlinien — Aktualisierungen der bestehenden Richtlinie	CodePipeline hat diesen Richtlinien die Erlaubnis hinzugefügt, einen zusätzlichen Benachrichtigungstyp mithilfe von Amazon Q	16. Mai 2023

Änderungen	Beschreibung	Date
	Developer in Chat-Anwendungen zu unterstützen,`chatbot:ListMicrosoftTeamsChannelConfigurations` .	
AWSCodePipelineFullAccess – als veraltet gekennzeichnet	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 verwaltete Richtlinien für AWS CodePipeline.	17. November 2022
AWSCodePipelineReadOnlyAccess – als veraltet gekennzeichnet	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

Änderungen	Beschreibung	Date
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 Beispiele für Benutzerrichtlinien, die Berechtigungen für verschiedene -Aktionen gewähren. Diese Richtlinien funktionieren, wenn Sie die 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 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 1: Gewähren von Berechtigungen zum Abrufen des Status einer Pipeline

Im folgenden Beispiel werden Berechtigungen vergeben, um den Status der Pipeline namens `MyFirstPipeline` abzurufen:

JSON

```
{
    "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:

JSON

```
{
    "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:

JSON

```
{
    "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:

JSON

```
{
    "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.

JSON

```
{
```

```
    "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. Die Richtlinie ist die gleiche wie die verwaltete Richtlinie `AWSCodePipelineCustomActionAccess`. Das folgende Beispiel zeigt eine Richtlinie für die Jenkins-Integration:

JSON

```
{
    "Version":"2012-10-17",

    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:AcknowledgeJob",
                "codepipeline:GetJobDetails",
                "codepipeline:PollForJobs",
                "codepipeline:PutJobFailureResult",
                "codepipeline:PutJobSuccessResult"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

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 Exemplarische Vorgehensweise: Cross-Account Zugriff mithilfe von Rollen.

Das folgende Beispiel zeigt eine Richtlinie im 80398EXAMPLE-Konto, die es Benutzern ermöglicht, die in der Konsole angegebene `MyFirstPipeline` Pipeline anzuzeigen, aber nicht zu ändern. 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 eine der verwalteten Richtlinien in Betracht ziehen, die von erstellt und verwaltet wurden. Weitere Informationen finden Sie unter Arbeiten mit verwalteten Richtlinien. Sie müssen diese Richtlinie an eine IAM-Rolle anhängen, die Sie für den Zugriff erstellen, z. B. an eine Rolle mit dem Namen: `CrossAccountPipelineViewers`

JSON

```
{
    "Version":"2012-10-17",

    "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-2:111122223333:MyFirstPipeline"
        }
    ]
}
```

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 im *111111111111* AWS Konto erstellte Richtlinie, die es Benutzern ermöglicht, die `CrossAccountPipelineViewers` im 80398EXAMPLE-Konto angegebene Rolle anzunehmen:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource":
 "arn:aws:iam::111122223333:role/CrossAccountPipelineViewers"
        }
    ]
}
```

AWS CodePipeline Beispiele für ressourcenbasierte Richtlinien

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. Es 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 für 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:

JSON

```
{
    "Version":"2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
```

```
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": false
                }
            }
        }
    ]
}
```

Fehlerbehebung 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 Konto, um auf meine CodePipeline Ressourcen zuzugreifen

Ich bin nicht berechtigt, eine Aktion durchzuführen in CodePipeline

Wenn Ihnen AWS-Managementkonsole 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 um die Aktualisierung Ihrer Richtlinien, um eine Rolle an CodePipeline übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte 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 CodePipeline auszuführen. 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 den Personen oder Anwendungen, die Zugriff benötigen, die entsprechenden Berechtigungen erteilen. Wenn Sie Personen und Anwendungen verwalten, weisen Sie Benutzern oder Gruppen Berechtigungssätze zu, um deren Zugriffsebene zu definieren. AWS IAM Identity Center Mit Berechtigungssätzen werden automatisch IAM-Richtlinien erstellt und den IAM-Rollen zugewiesen, die der Person oder Anwendung zugeordnet sind. Weitere Informationen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Berechtigungssätze.

Wenn Sie IAM Identity Center nicht verwenden, müssen Sie IAM-Entitäten (Benutzer oder Rollen) für die Personen oder Anwendungen erstellen, die Zugriff benötigen. Anschließend müssen Sie der

Entität eine Richtlinie anfügen, die dieser die korrekten Berechtigungen in CodePipeline gewährt. Nachdem die Berechtigungen erteilt wurden, stellen Sie dem Benutzer oder Anwendungsentwickler die Anmeldeinformationen zur Verfügung. Sie werden diese Anmeldeinformationen für den Zugriff verwenden AWS. Weitere Informationen zum Erstellen von IAM-Benutzern, -Gruppen, -Richtlinien und -Berechtigungen finden Sie im IAM-Benutzerhandbuch unter IAM-Identitäten sowie Richtlinien und Berechtigungen in IAM.

Ich möchte Personen außerhalb meines AWS Konto, um auf meine CodePipeline Ressourcen zuzugreifen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen 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 Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Berechtigungsreferenz

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 werden alle API-Operationen und die zugehörigen Aktionen aufgeführt,

für die Sie Berechtigungen zur Durchführung 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.

Resource-level Mit Berechtigungen 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.

-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*`/`*versio*

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 erstellen.

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 CodePipeline , auf die reagiert werden 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:

Action

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

PutApprovalResult

Aktion: `codepipeline:PutApprovalResult`

Erforderlich zum Melden der Reaktion auf eine manuelle Genehmigungsanfrage an CodePipeline. Gültige Antworten sind `Approved` und `Rejected`.

Ressourcen:

Action

`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/versio`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UntagResource

Aktion: `codepipeline:UntagResource`

Erforderlich, um das Tag der angegebenen Ressource zu entfernen.

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`

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`

Verwalten Sie die CodePipeline Servicerolle

Die 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. Vergewissern Sie sich über die Auswirkungen, bevor Sie die Servicerolle für auf irgendeine Weise verändern. 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`.

CodePipeline Richtlinie für die Servicerolle

Die Richtlinie zur CodePipeline Servicerolle enthält die Mindestberechtigungen für die Verwaltung von Pipelines. Sie können die Diensterollenanweisung bearbeiten, um den Zugriff auf Ressourcen, die Sie

nicht verwenden, zu entfernen oder hinzuzufügen. In der entsprechenden Aktionsreferenz finden Sie die für jede Aktion erforderlichen Mindestberechtigungen CodePipeline .

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowS3BucketAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketVersioning",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::[[pipeArtifactBucketNames]]"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "{{accountId}}"
        }
      }
    },
    {
      "Sid": "AllowS3ObjectAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:PutObjectTagging",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::[[pipeArtifactBucketNames]]/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "{{accountId}}"
```

```
        }
      }
    }
  ]
}
```

> **Note**
>
> In der Richtlinie sind die folgenden Berechtigungen erforderlich, wenn die S3-Objekte in Ihrem Quell-Bucket Tags enthalten:
>
> ```
> s3:PutObjectTagging
> s3:GetObjectTagging
> s3:GetObjectVersionTagging
> ```

Entfernen Sie 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"
},
```

Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu

Bevor Sie sie in Ihren Pipelines verwenden können, müssen Sie Ihre Dienstrollen-Richtlinienerklärung mit Berechtigungen für eine Richtlinie aktualisieren, die noch AWS-Service nicht in der Standardrichtlinie für die Servicerolle enthalten ist.

Dies ist besonders wichtig, wenn die Servicerolle, die Sie für Ihre Pipelines verwenden, erstellt wurde, bevor der Support für eine hinzugefügt wurde. AWS-Service

Die folgende Tabelle zeigt, wann Unterstützung für andere AWS-Services hinzugefügt wurde.

AWS-Service	CodePipeline Datum der Unterstützung
CodePipeline Unterstützung für Aktionen aufrufen hinzugefügt. Siehe Berechtigungen für die Aktion zum CodePipeline Aufrufen der Aktion „Dienst“.	14. März 2025
EC2Unterstützung für Aktionen hinzugefügt. Siehe Richtlinienberechtigungen für die EC2-Bereitstellungsaktion.	21. Februar 2025
EKSUnterstützung für Aktionen hinzugefügt. Siehe Berechtigungen für Richtlinien zur Servicerolle.	20. Februar 2025
Unterstützung für Amazon Elastic Container Registry-Aktionen `ECRBuildAndPublish` hinzugefügt. Siehe Berechtigungen für Servicerollen: Aktion `ECRBuildAndPublish`.	22. November 2024
Unterstützung für Amazon Inspector Inspector-Aktionen `InspectorScan` hinzugefügt. Siehe Berechtigungen für Servicerollen: `InspectorScan` Aktion.	22. November 2024

AWS-Service	CodePipeline Datum der Unterstützung
Unterstützung für Befehlsaktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: Befehle, Aktion.	03. Oktober 2024
CloudFormation Unterstützung für Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: Aktion `CloudFormationStackSet` und Berechtigungen für Servicerollen: `CloudFormationStackInstances` Aktion.	30. Dezember 2020
CodeCommit Unterstützung für Aktionen im vollständigen Klonausgabeartefaktformat hinzugefügt. Siehe Berechtigungen für Servicerollen: CodeCommit Aktion.	11. November 2020
AWS CodeBuild Unterstützung für Batch-Build-Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: CodeCommit Aktion.	30. Juli 2020
AWS AppConfig Unterstützung für Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: Aktion `AppConfig` .	22. Juni 2020
AWS Step Functions Aktionsunterstützung hinzugefügt. Siehe Berechtigungen für Servicerollen: `StepFunctions` Aktion.	27. Mai 2020
AWS CodeStar Unterstützung für Verbindungsaktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: AWS CodeConnections action.	18. Dezember 2019
Unterstützung für S3-Bereitstellungsaktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: Aktion „S3-Bereitstellung“.	16. Januar 2019

AWS-Service	CodePipeline Datum der Unterstützung
Die Unterstützung für `CodeDeployToECS` Aktionsaktionen wurde hinzugefügt. Siehe Berechtigungen für Servicerollen: `CodeDeployToECS` Aktion.	27. November 2018
Unterstützung für Amazon ECR-Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: Amazon ECR-Aktion.	27. November 2018
Unterstützung Service Catalog Servicekatalog-Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: Aktion „Servicekatalog“.	16. Oktober 2018
AWS Device Farm Unterstützung für Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: AWS Device Farm action.	19. Juli 2018
Unterstützung für Amazon ECS-Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: Amazon ECS-Standardaktion.	12. Dezember 2017//Update für die Anmeldung zur Tagging-Autorisierung am 21. Juli 2017
CodeCommit Unterstützung für Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: CodeCommit Aktion.	18. April 2016
AWS OpsWorks Aktionsunterstützung hinzugefügt. Siehe Berechtigungen für Servicerollen: AWS OpsWorks action.	2. Juni 2016
CloudFormation Aktionsunterstützung hinzugefügt. Siehe Berechtigungen für Servicerollen: CloudFormation Aktion.	3. November 2016
AWS CodeBuild Aktionsunterstützung hinzugefügt. Siehe Berechtigungen für Servicerollen: CodeBuild Aktion.	1. Dezember 2016

AWS-Service	CodePipeline Datum der Unterstützung
Unterstützung für Elastic Beanstalk Beanstalk -Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: Aktion `ElasticBeanstalk` bereitstellen.	Erster Start des Dienstes
CodeDeploy Unterstützung für Aktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: AWS CodeDeploy Aktion.	Erster Start des Dienstes
Unterstützung für S3-Quellaktionen hinzugefügt. Siehe Berechtigungen für Servicerollen: S3-Quellaktion.	Erster Start des Dienstes

Gehen Sie folgendermaßen vor, um Berechtigungen für einen unterstützten Service hinzuzufügen:

1. Melden Sie sich bei der an AWS-Managementkonsole 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

5. 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 Protokollierung von CodePipeline API-Aufrufen 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 Überwachung von CodePipeline Ereignissen.

Konformitätsvalidierung für AWS CodePipeline

Unser neues AWS Anmeldeerlebnis ist nicht für regulierte Workloads konzipiert. Wenn Sie unser neues AWS Anmeldeformular nutzen, es aber AWS für regulierte Workloads nutzen möchten, können Sie sich für AWS (erweiterte) Funktionen registrieren oder erweiterte Funktionen für Ihre Umgebung aktivieren. AWS

Informationen darüber, ob AWS-Service ein in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter AWS-Services Umfang nach Compliance-Programm unter Umfang

nach Compliance-Programm AWS-Services das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Mithilfe von Artifact können Sie Prüfberichte von Drittanbietern herunterladen. 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. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der AWS Sicherheitsdokumentation.

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.
- Cipher-Suites mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

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 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 beschriebenVerwenden Sie AWS Secrets Manager um Datenbankpasswörter 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 Sie für die Jenkins-Integration verwenden möchten.

CodePipeline Referenz zur Pipeline-Struktur

Sie können CodePipeline es verwenden, um eine CI/CD Pipeline automatisierter Schritte zu strukturieren, mit denen Aufgaben zum Erstellen, Testen und Bereitstellen Ihres Anwendungsquellcodes ausgeführt werden. Dieser Referenzabschnitt enthält Details zur JSON-Struktur und zu den Parametern in Ihrer Pipeline. Eine allgemeine Liste von Konzepten, die beschreiben, wie Pipelines verwendet werden, finden Sie unterCodePipeline Konzepte.

- Wenn Sie eine Pipeline erstellen, wählen Sie eine verfügbare Quellaktion und einen verfügbaren Anbieter aus, z. B. einen S3-Bucket, ein CodeCommit Repository, ein Bitbucket-Repository oder ein GitHub Repository, das Ihren Quellcode enthält und Ihre Pipeline startet, wenn Sie eine Quellcodeänderung festschreiben. Dieser Referenzabschnitt enthält Referenzinformationen zu den verfügbaren Quellen für deine Pipeline. Weitere Informationen zum Arbeiten mit Quellaktionen finden Sie unterStarten Sie eine Pipeline in CodePipeline.
- Sie können die Test-, Build- und Deployment-Aktionen und Anbieter auswählen, die Sie bei der Ausführung Ihrer Pipeline automatisch einbeziehen möchten. Dieser Referenzabschnitt enthält Referenzinformationen zu den verfügbaren Aktionen und dazu, wie sie in Ihr Pipeline-JSON passen.
- Ihre fertige Pipeline besteht aus einer Quellphase sowie weiteren Stufen, in denen Sie Aktionen zum Bereitstellen und Testen Ihrer Anwendung konfigurieren. Ein konzeptionelles Beispiel für eine DevOps Pipeline, die Ihre Anwendung bereitstellt, finden Sie unterDevOps Beispiel für eine Pipeline.

Standardmäßig hat jede Pipeline, die Sie erfolgreich erstellen, AWS CodePipeline eine gültige Struktur. Wenn Sie jedoch manuell eine JSON-Datei erstellen oder bearbeiten, um eine Pipeline zu erstellen, oder eine Pipeline von der aus aktualisieren AWS CLI, erstellen Sie möglicherweise versehentlich eine Struktur, 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.

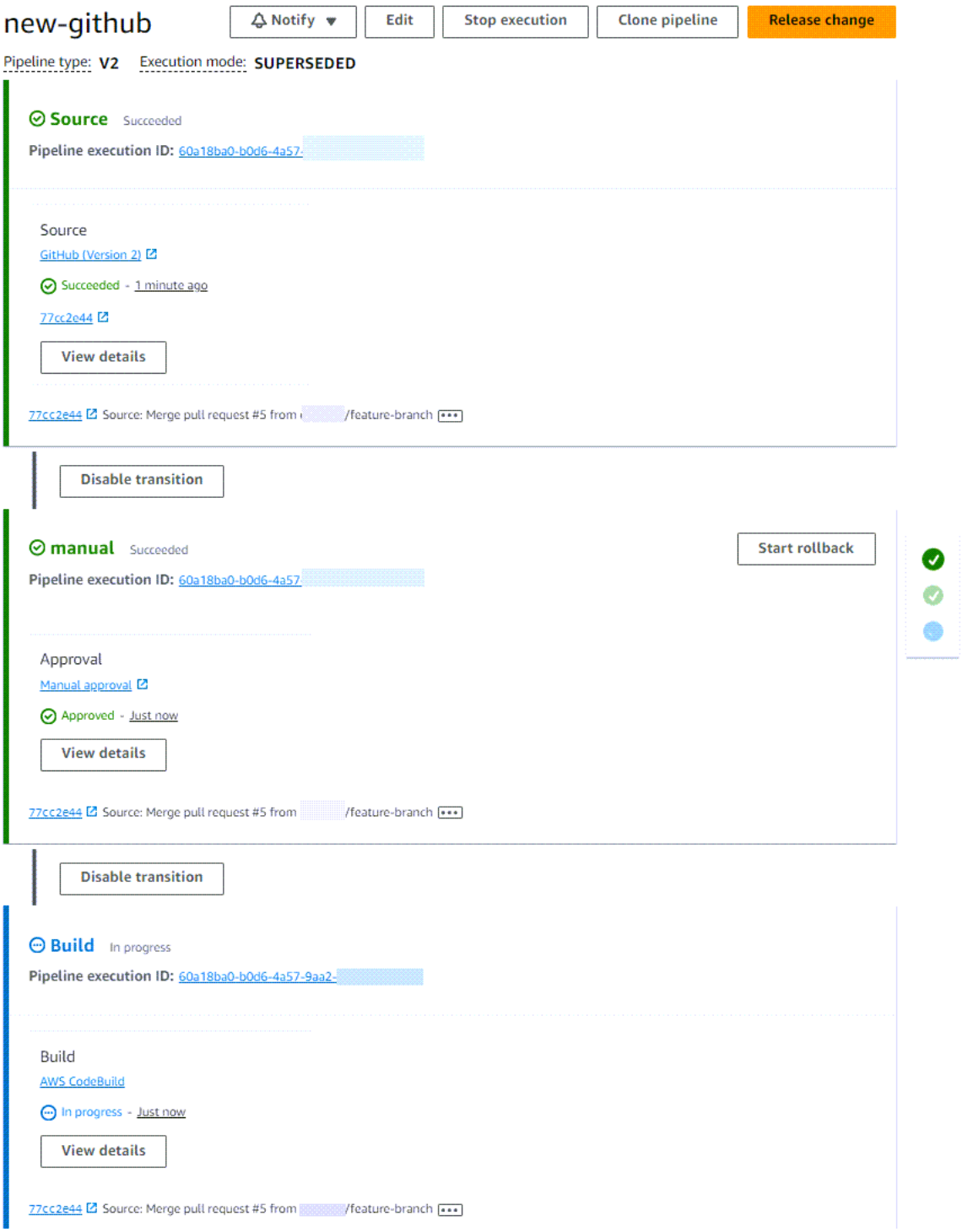
Die folgenden Abschnitte enthalten allgemeine Parameter und ihre Position in der Pipeline-Struktur. Die Anforderungen an die Pipeline-Struktur werden in den einzelnen Abschnitten für die folgenden Pipeline-Komponententypen detailliert beschrieben:

- Feldreferenz für den Pipeline-Erklärung

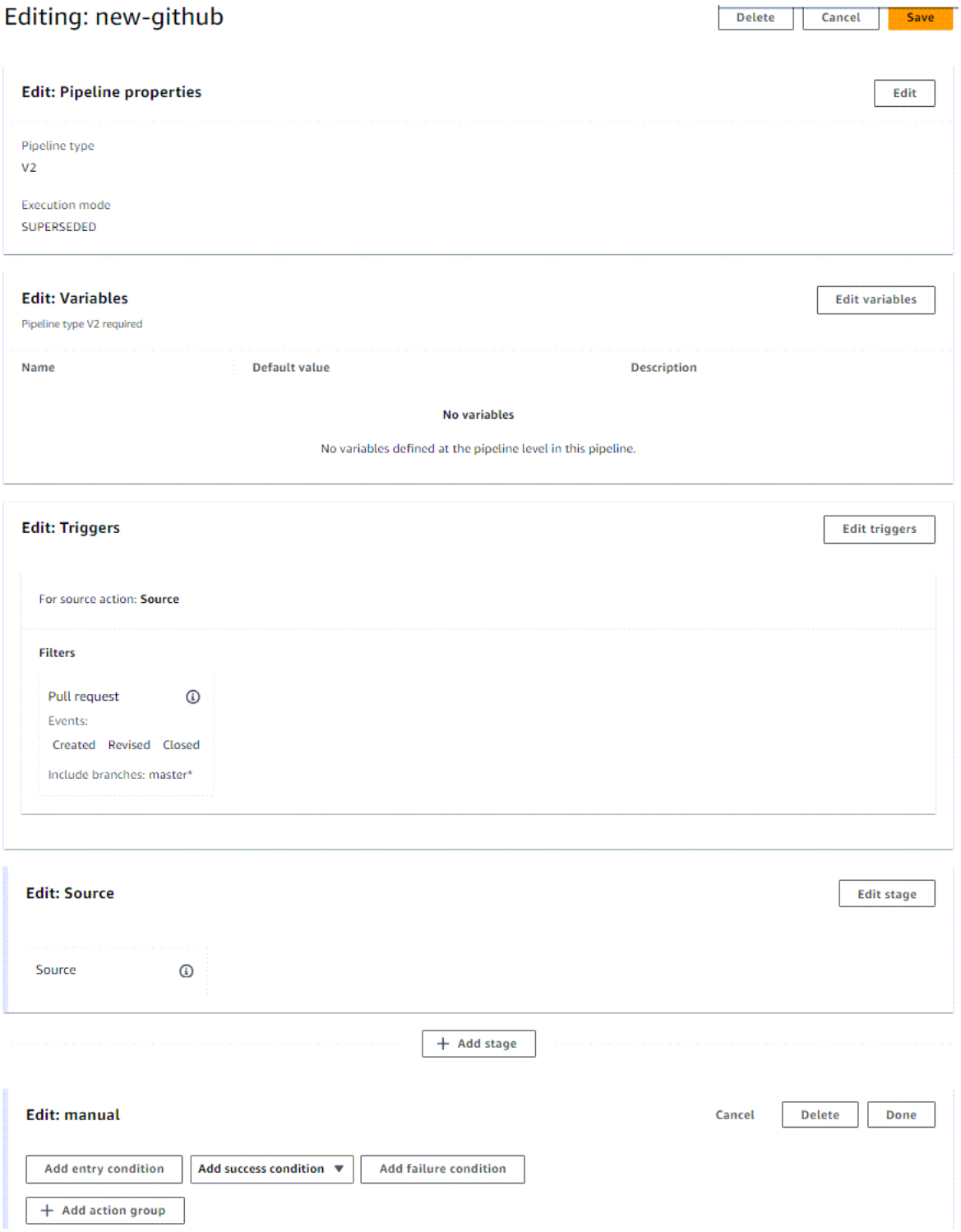
- Feldreferenz für den Deklaration der Stufe
- Feldreferenz für den Aktionsdeklaration
- Liste von Gültige Aktionsanbieter in CodePipeline nach Aktionstyp
- Referenz für Gültige Einstellungen für den `PollForSourceChanges` Parameter
- Referenz für Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
- Liste der Links für Gültige Konfigurationsparameter für jeden Anbietertyp

Weitere Informationen finden Sie im CodePipeline API-Handbuch zu dem PipelineDeclaration Objekt.

Das folgende Beispiel für eine Pipeline-Konsolenansicht zeigt die Pipeline mit dem Namen new-github `Sourcemanual`, die Stufen mit dem Namen und und die Aktionen von GitHub (via GitHub App), manuellen Genehmigungsanbietern und CodeBuild Aktionsanbietern. `Build`



Im Pipeline-Bearbeitungsmodus können Sie, wenn er im Konsolendiagramm angezeigt wird, Quellüberschreibungen, Trigger und Aktionen bearbeiten, wie im folgenden Beispiel gezeigt.



Themen

- Pipeline-Erklärung
- Deklaration der Stufe
- Aktionsdeklaration

- Gültige Aktionsanbieter in CodePipeline
- Gültige Einstellungen für den PollForSourceChanges Parameter
- Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
- Gültige Konfigurationsparameter für jeden Anbietertyp

Pipeline-Erklärung

Die Pipeline- und Metadatenebene einer Pipeline hat eine grundlegende Struktur, die die folgenden Parameter und Syntax umfasst. Der Pipeline-Parameter stellt die Struktur der Aktionen und Stufen dar, die in der Pipeline ausgeführt werden sollen.

Weitere Informationen finden Sie im CodePipeline API-Handbuch zu dem PipelineDeclaration Objekt.

Das folgende Beispiel zeigt die Pipeline- und Metadatenebene der Pipeline-Struktur sowohl in JSON als auch in YAML für eine Pipeline vom Typ V2.

YAML

```
pipeline:
  name: MyPipeline
  roleArn: >-
    arn:aws:iam::ACCOUNT_ID:role/service-role/AWSCodePipelineServiceRole-us-west-2-
MyPipeline
  artifactStore:
    type: S3
    location: amzn-s3-demo-bucket
  stages:
    ...
  version: 6
  executionMode: SUPERSEDED
  pipelineType: V2
  variables:
  - name: MyVariable
    defaultValue: '1'
  triggers:
  - providerType: CodeStarSourceConnection
    gitConfiguration:
      sourceActionName: Source
      push:
```

```
      - branches:
          includes:
          - main
          excludes:
          - feature-branch
        pullRequest:
        - events:
          - CLOSED
          branches:
            includes:
            - main*
metadata:
  pipelineArn: 'arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline'
  created: '2019-12-12T06:49:02.733000+00:00'
  updated: '2020-09-10T06:34:07.447000+00:00'
  pollingDisabledAt: '2020-09-10T06:34:07.447000+00:00'
```

JSON

```
{
    "pipeline": {
        "name": "MyPipeline",
        "roleArn": "arn:aws:iam::ACCOUNT_ID:role/service-role/
AWSCodePipelineServiceRole-us-west-2-MyPipeline",
        "artifactStore": {
            "type": "S3",
            "location": "amzn-s3-demo-bucket"
        },
        "stages": {
            ...
    },
        "version": 6,
        "executionMode": "SUPERSEDED",
                "pipelineType": "V2",
        "variables": [
            {
                "name": "MyVariable",
                "defaultValue": "1"
            }
        ],
        "triggers": [
            {
                "providerType": "CodeStarSourceConnection",
```

```
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "branches": {
                                "includes": [
                                    "main"
                                ],
                                "excludes": [
                                    "feature-branch"
                                ]
                            }
                        }
                    ],
                    "pullRequest": [
                        {
                            "events": [
                                "CLOSED"
                            ],
                            "branches": {
                                "includes": [
                                    "main*"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    },
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline",
        "created": "2019-12-12T06:49:02.733000+00:00",
        "updated": "2020-09-10T06:34:07.447000+00:00",
        "pollingDisabledAt": "2020-09-10T06:34:07.447000+00:00"
    }
}
```

Name

Der Name der Pipeline. 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.

`roleArn`

Der IAM-ARN für die CodePipeline Servicerolle, z. B. arn:aws:iam: :80398 _. EXAMPLE:role/ CodePipeline Service_Role

Um die Konsole zum Anzeigen des ARN der Pipeline-Servicerolle anstelle der JSON-Struktur zu verwenden, wählen Sie Ihre Pipeline in der Konsole aus und klicken Sie dann auf Einstellungen. Auf der Registerkarte Allgemein wird das ARN-Feld Servicerolle angezeigt.

`ArtifactStore` ODER `ArtifactStores`

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.

> **Note**
>
> In der Pipeline-Struktur müssen Sie entweder `artifactStore` oder `artifactStores` in Ihre Pipeline aufnehmen, aber Sie können nicht beide verwenden. Wenn Sie eine regionsübergreifende Aktion in Ihrer Pipeline erstellen, müssen Sie `artifactStores` verwenden.

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 amzn-s3-demo-bucket"
            },
            "us-west-2": {
                "type": "S3",
                "location": "S3 artifact bucket name, such as amzn-s3-demo-bucket"
            }
        },
        "stages": [
            {

...
```

`type`

Der Standorttyp für den Artefakt-Bucket, angegeben als Amazon S3.

`location`

Der Name des Amazon S3-Buckets, der automatisch für Sie generiert wird, wenn Sie zum ersten Mal mithilfe der Konsole eine Pipeline erstellen, z. B. codepipeline-us-east-2-1234567890, oder ein beliebiger Amazon S3-Bucket, den Sie für diesen Zweck bereitstellen

`Stufen`

Dieser Parameter enthält den Namen jeder Phase in der Pipeline. Weitere Informationen zu den Parametern und der Syntax auf Stufenebene der Pipeline-Struktur finden Sie im CodePipeline API-Handbuch zu dem StageDeclaration Objekt.

Für die Pipeline-Struktur für Stufen gelten die folgenden 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.

- Die Namen der Stufen können in der CodePipeline Konsole nicht bearbeitet werden. Wenn Sie einen Stufennamen mithilfe von bearbeiten und der AWS CLI Stagingbereich eine Aktion mit einem oder mehreren geheimen Parametern (z. B. einem OAuth-Token) enthält, wird der Wert dieser geheimen Parameter nicht beibehalten. Sie müssen die Werte der Parameter (die in der von der AWS CLI zurückgegebenen JSON mit vier Sternchen maskiert sind) manuell in die JSON-Struktur eingeben.

> **Important**
>
> Bei Pipelines, die länger als 30 Tage inaktiv sind, wird die Abfrage für die Pipeline deaktiviert. Weitere Informationen finden Sie unter Polling DisabledAt in der Pipeline-Strukturreferenz. Die Schritte zur Migration Ihrer Pipeline vom Polling zur ereignisbasierten Änderungserkennung finden Sie unter Methoden zur Änderungserkennung.

`version`

Die Versionsnummer einer Pipeline wird bei jedem Pipeline-Update automatisch generiert und aktualisiert.

`executionMode`

Sie können den Pipeline-Ausführungsmodus so einstellen, dass Sie das Pipeline-Verhalten für aufeinanderfolgende Läufe angeben können, z. B. in die Warteschlange gestellt, ersetzt oder im Parallelmodus ausgeführt werden. Weitere Informationen finden Sie unter Den Pipeline-Ausführungsmodus festlegen oder ändern.

> **Important**
>
> Für Pipelines im PARALLELMODUS ist das Stufen-Rollback nicht verfügbar. Ebenso können einer Pipeline im PARALLEL-Modus keine Fehlerbedingungen mit einem Rollback-Ergebnistyp hinzugefügt werden.

`PipelineType`

Der Pipeline-Typ spezifiziert die verfügbare Struktur und die verfügbaren Funktionen in der Pipeline, z. B. für eine Pipeline vom Typ V2. Weitere Informationen finden Sie unter Pipeline-Typen.

`variables`

Variablen auf Pipeline-Ebene werden definiert, wenn die Pipeline erstellt und zur Laufzeit der Pipeline aufgelöst wird. Weitere Informationen finden Sie unter Variablenreferenz. Ein Tutorial mit einer Variablen auf Pipeline-Ebene, die zum Zeitpunkt der Pipeline-Ausführung übergeben wird, finden Sie unter. Tutorial: Variablen auf Pipeline-Ebene verwenden

`triggers`

Mit Triggern können Sie Ihre Pipeline so konfigurieren, dass sie mit einem bestimmten Ereignistyp oder einem gefilterten Ereignistyp beginnt, z. B. wenn eine Änderung an einem bestimmten Branch oder Pull Request erkannt wird. Trigger sind für Quellaktionen mit Verbindungen konfigurierbar, in denen die `CodeStarSourceConnection` Aktion verwendet wird CodePipeline GitHub, z. B. Bitbucket und GitLab. Weitere Informationen zu Quellaktionen, die Verbindungen verwenden, findest du unterFügen Sie externe Quellanbieter zu Pipelines hinzu mit AWS CodeConnections.

Weitere Informationen und detailliertere Beispiele finden Sie unterAutomatisieren Sie das Starten von Pipelines mithilfe von Triggern und Filtern.

Zum Filtern werden Muster regulärer Ausdrücke im Glob-Format unterstützt, wie unter beschriebenArbeiten mit Glob-Mustern in der Syntax.

> **Note**
>
> Für die Aktionen „Quelle“ CodeCommit und „S3“ ist entweder eine konfigurierte Ressource zur Änderungserkennung (eine EventBridge Regel) erforderlich, oder Sie verwenden die Option, das Repository nach Änderungen an der Quelle abzufragen. Für Pipelines mit einer Bitbucket GitHub - oder GitHub Enterprise Server-Quellaktion müssen Sie keinen Webhook einrichten oder standardmäßig Polling verwenden. Die Aktion „Verbindungen“ verwaltet die Änderungserkennung für dich.

> **Important**
>
> Bei Pipelines, die länger als 30 Tage inaktiv sind, wird die Abfrage für die Pipeline deaktiviert. Weitere Informationen finden Sie unter Polling DisabledAt in der Pipeline-Strukturreferenz. Die Schritte zur Migration Ihrer Pipeline vom Polling zur ereignisbasierten Änderungserkennung finden Sie unter Methoden zur Änderungserkennung.

GitConfiguration-Felder

Die Git-Konfiguration für den Trigger, einschließlich der Ereignistypen und aller Parameter zum Filtern nach Branches, Dateipfaden, Tags oder Pull-Request-Ereignissen.

Die Felder in der JSON-Struktur sind wie folgt definiert:

- `sourceActionName`: Der Name der Pipeline-Quellaktion mit der Git-Konfiguration.
- `push`: Ereignisse mit Filterung übertragen. 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 Inklusionen und Ausschlüssen.
    - `includes`: Muster, nach denen für Zweige gefiltert werden soll, die einbezogen werden. Beinhaltet die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Zweigen gefiltert werden soll, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.
  - `filePaths`: Die Dateipfadnamen, nach denen gefiltert werden soll.
    - `includes`: Muster, nach denen für Dateipfade gefiltert werden soll, die aufgenommen werden. Beinhaltet die Verwendung einer OR-Operation.
    - `excludes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.
  - `tags`: Die Tagnamen, nach denen gefiltert werden soll.
    - `includes`: Muster, nach denen nach Tags gefiltert werden soll, die aufgenommen werden. Beinhaltet die Verwendung einer OR-Operation.
    - `excludes`: Muster, nach denen nach Stichwörtern gefiltert werden soll, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.
- `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 Zweige, nach denen gefiltert werden soll. Zweige verwenden eine UND-Operation zwischen Inklusionen und Ausschlüssen.
    - `includes`: Muster, nach denen für Zweige gefiltert werden soll, die einbezogen werden. Beinhaltet die Verwendung einer ODER-Operation.

- `excludes`: Muster, nach denen nach Zweigen gefiltert werden soll, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.

- `filePaths`: Die Dateipfadnamen, nach denen gefiltert werden soll.
  - `includes`: Muster, nach denen für Dateipfade gefiltert werden soll, die aufgenommen werden. Beinhaltet die Verwendung einer OR-Operation.
  - `excludes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.

Das Folgende ist ein Beispiel für die Triggerkonfiguration für Push- und Pull-Request-Ereignistypen.

```
"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"
                            ]
                        }
                    }
                ]
            }
        }
    ],
```

Push-Felder vom Ereignistyp für Einschließen und Ausschließen

Das Ein- und Ausschlussverhalten für Ebenen von Git-Konfigurationsfeldern für Push-Ereignistypen wird in der folgenden Liste aufgeführt:

```
push (OR operation is used between push and pullRequest or multiples)
```

```
    filePaths (AND operation is used between filePaths, branches, and tags)
        includes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
        excludes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
    branches (AND operation is used between filePaths, branches, and tags)
        includes (AND operation is used between includes and excludes)
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
        excludes (AND operation is used between includes and excludes)
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
    tags (AND operation is used between filePaths, branches, and tags)
         includes (AND operation is used between includes and excludes)
            TAG/**", "TAG2/** (OR operation is used between tag names)
         excludes (AND operation is used between includes and excludes)
            TAG/**", "TAG2/** (OR operation is used between tag names)
```

Pull-Request-Felder vom Ereignistyp für Einschließen und Ausschließen

Das Ein- und Ausschlussverhalten für Ebenen von Git-Konfigurationsfeldern für Pull-Request-Ereignistypen wird in der folgenden Liste aufgeführt:

```
pullRequest (OR operation is used between push and pullRequest or multiples)
    events (AND operation is used between events, filePaths, and branches). Includes/
excludes are N/A for pull request events.
    filePaths (AND operation is used between events, filePaths, and branches)
        includes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
        excludes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
    branches (AND operation is used between events, filePaths, and branches)
        includes (AND operation is used between includes and excludes)
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
        excludes (AND operation is used between includes and excludes)
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
```

Metadaten

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.

`pipelineArn`

Der Amazon-Ressourcenname (ARN) der Pipeline.

Um die Konsole zum Anzeigen des Pipeline-ARN anstelle der JSON-Struktur zu verwenden, wählen Sie Ihre Pipeline in der Konsole aus und klicken Sie dann auf Einstellungen. Auf der Registerkarte Allgemein wird das Feld Pipeline-ARN angezeigt.

`created`

Datum und Uhrzeit der Erstellung der Pipeline.

`Aktualisiert`

Datum und Uhrzeit der letzten Aktualisierung der Pipeline.

`Umfrage DisabledAt`

Datum und Uhrzeit der Deaktivierung der Abfrage für eine Pipeline, die für die Erkennung von Änderungen konfiguriert ist.

Bei Pipelines, die länger als 30 Tage inaktiv sind, wird die Abfrage für die Pipeline deaktiviert.

- Bei inaktiven Pipelines wird das Polling nach 30 Tagen ohne Ausführung deaktiviert.
- Pipelines EventBridge, die CodeStar Verbindungen oder Webhooks verwenden, sind davon nicht betroffen.
- Aktive Pipelines sind nicht betroffen.

Weitere Informationen finden Sie im CodePipeline API-Handbuch unter dem `pollingDisabledAt` Parameter unter PipelineMetadata Objekt. Die Schritte zur Migration Ihrer Pipeline vom Polling zur ereignisbasierten Änderungserkennung finden Sie unter Methoden zur Änderungserkennung.

Deklaration der Stufe

Die Stufenebene einer Pipeline hat eine grundlegende Struktur, die die folgenden Parameter und Syntax umfasst. Weitere Informationen finden Sie im CodePipeline API-Handbuch zu dem StageDeclaration Objekt.

Das folgende Beispiel zeigt die Stufenebene der Pipeline-Struktur sowohl in JSON als auch in YAML. Das Beispiel zeigt zwei Stufen mit dem Namen `Source` und`Build`. Das Beispiel enthält zwei Bedingungen, eine für on`Success` und eine für`beforeEntry`.

YAML

```
pipeline:
  name: MyPipeline
  roleArn: >-
    arn:aws:iam::ACCOUNT_ID:role/service-role/AWSCodePipelineServiceRole-us-west-2-
MyPipeline
  artifactStore:
    type: S3
    location: amzn-s3-demo-bucket
  stages:
    - name: Source
      actions:
        - name: Source
          ...
    - name: Build
      actions:
        - name: Build
          ...
      onSuccess:
        conditions:
        - result: ROLLBACK
          rules:
          - name: DeploymentWindowRule
         ...
      beforeEntry:
        conditions:
        - result: FAIL
          rules:
          - name: MyLambdaRule
         ...
  version: 6
metadata:
  pipelineArn: 'arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline'
  created: '2019-12-12T06:49:02.733000+00:00'
  updated: '2020-09-10T06:34:07.447000+00:00'
```

JSON

```
{
    "pipeline": {
        "name": "MyPipeline",
        "roleArn": "arn:aws:iam::ACCOUNT_ID:role/service-role/
AWSCodePipelineServiceRole-us-west-2-MyPipeline",
        "artifactStore": {
            "type": "S3",
            "location": "amzn-s3-demo-bucket"
        },
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        ...
                    }
                ]
            },
            {
                "name": "Build",
                "actions": [
                    {
                        "name": "Build",
                        ...
                    }
                ],
                "onSuccess": {
                    "conditions": [
                        {
                            "result": "ROLLBACK",
                            "rules": [
                                {
                                    "name": "DeploymentWindowRule",
                                    ...
                                }
                            ]
                        }
                    ]
                },
                "beforeEntry": {
                    "conditions": [
```

```
                        {
                            "result": "FAIL",
                            "rules": [
                                {
                                    "name": "MyLambdaRule",
                                     ...
                                }
                            ]
                        }
                    ]
                }
            }
        ],

            }
        ],
        "version": 6
    },
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline",
        "created": "2019-12-12T06:49:02.733000+00:00",
        "updated": "2020-09-10T06:34:07.447000+00:00"
    }
}
```

Name

Den Namen der -Stufe.

actions

Die Aktionsebene einer Pipeline hat eine grundlegende Struktur, die die folgenden Parameter und Syntax umfasst. Parameter und Beispiele finden Sie unterAktionsdeklaration.

conditions

Bedingungen enthalten eine oder mehrere Regeln, die in einer Liste von Regeln in verfügbar sind CodePipeline. Wenn alle Regeln in einer Bedingung erfolgreich sind, ist die Bedingung erfüllt. Sie können Bedingungen so konfigurieren, dass das angegebene Ergebnis aktiviert wird, wenn die Kriterien nicht erfüllt sind.

Sie können die folgenden Arten von Bedingungen konfigurieren:

- `beforeEntry`
- `onFailure`
- `onSuccess`

Weitere Informationen und Beispiele finden Sie unter Bedingungen für eine Phase konfigurieren.

Regeln

Jede Bedingung hat einen Regelsatz, bei dem es sich um einen geordneten Satz von Regeln handelt, die zusammen ausgewertet werden. Wenn also eine Regel in der Bedingung fehlschlägt, schlägt die Bedingung fehl. Sie können die Regelbedingungen zur Laufzeit der Pipeline überschreiben.

Die verfügbaren Regeln finden Sie in der Regelreferenz. Weitere Informationen finden Sie in der Referenz zur Regelstruktur unterReferenz zur Regelstruktur.

Aktionsdeklaration

Die Aktionsebene einer Pipeline hat eine grundlegende Struktur, die die folgenden Parameter und Syntax umfasst. Weitere Informationen finden Sie im CodePipeline API-Handbuch zu dem ActionDeclaration Objekt.

Das folgende Beispiel zeigt die Aktionsebene der Pipeline-Struktur sowohl in JSON als auch in YAML.

YAML

```
. . .

  stages:
    - name: Source
      actions:
        - name: Source
          actionTypeId:
            category: Source
            owner: AWS
```

```
        provider: S3
        version: '1'
      runOrder: 1
      configuration:
        PollForSourceChanges: 'false'
        S3Bucket: amzn-s3-demo-bucket
        S3ObjectKey: codedeploy_linux.zip
      outputArtifacts:
        - name: SourceArtifact
      inputArtifacts: []
      region: us-west-2
      namespace: SourceVariables
  - name: Build
    actions:
      - name: Build
        actionTypeId:
          category: Build
          owner: AWS
          provider: CodeBuild
          version: '1'
        runOrder: 1
        configuration:
          EnvironmentVariables: >-
            [{"name":"ETag","value":"#{SourceVariables.ETag}","type":"PLAINTEXT"}]
          ProjectName: my-project
        outputArtifacts:
          - name: BuildArtifact
        inputArtifacts:
          - name: SourceArtifact
        region: us-west-2
        namespace: BuildVariables
        runOrder: 1
        configuration:
          CustomData: >-
            Here are the exported variables from the build action: S3 ETAG:
            #{BuildVariables.ETag}
        outputArtifacts: []
        inputArtifacts: []
        region: us-west-2
```

JSON

```
. . .

        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "provider": "S3",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "PollForSourceChanges": "false",
                            "S3Bucket": "amzn-s3-demo-bucket",
                            "S3ObjectKey": "aws-codepipeline-s3-aws-
codedeploy_linux.zip"
                        },
                        "outputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "inputArtifacts": [],
                        "region": "us-west-2",
                        "namespace": "SourceVariables"
                    }
                ]
            },
            {
                "name": "Build",
                "actions": [
                    {
                        "name": "Build",
                        "actionTypeId": {
                            "category": "Build",
                            "owner": "AWS",
                            "provider": "CodeBuild",
                            "version": "1"
                        },
                        "runOrder": 1,
```

```
                        "configuration": {
                            "EnvironmentVariables": "[{\"name\":\"ETag\",\"value\":
\"#{SourceVariables.ETag}\",\"type\":\"PLAINTEXT\"}]",
                            "ProjectName": "my-build-project"
                        },
                        "outputArtifacts": [
                            {
                                "name": "BuildArtifact"
                            }
                        ],
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "region": "us-west-2",
                        "namespace": "BuildVariables"
                    }
                ]

. . .
```

Eine Liste der Beispiel-`configuration` Details für den jeweiligen Anbietertyp finden Sie unter Gültige Konfigurationsparameter für jeden Anbietertyp.

Die Aktionsstruktur hat folgende Anforderungen:

- Alle Aktionsnamen in einer Phase müssen eindeutig sein.
- Für jede Pipeline ist eine Quellaktion erforderlich.
- Quellaktionen, die keine Verbindung verwenden, können für die Änderungserkennung oder zum Ausschalten der Änderungserkennung konfiguriert werden. Weitere Informationen finden Sie unter Methoden zur Änderungserkennung.
- 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.
- Wenn Sie einen Amazon 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-4bf 3-9013-db3a9 EXAMPLE.zip

Name

Den Namen der Aktion.

Region

Für Aktionen, bei denen der Anbieter ein ist, der der Ressource. AWS-Service AWS-Region

Cross-Region Aktionen verwenden Sie das `Region` Feld, um anzugeben AWS-Region , wo die Aktionen erstellt werden sollen. Die für diese Aktion erstellten AWS Ressourcen müssen in derselben Region erstellt werden, die im `region` Feld angegeben ist. Für die folgenden Aktionstypen können Sie keine regionsübergreifenden Aktionen erstellen:

- Quellaktionen
- Aktionen von Drittanbietern
- Aktionen von benutzerdefinierten Anbietern

`roleArn`

Der ARN der IAM-Servicerolle, die die angegebene Aktion ausführt. Dies wird durch das RolEearn vorausgesetzt, das auf Pipeline-Ebene angegeben wird.

`Namespace`

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 Variablenreferenz.

> **Note**
>
> Für Amazon ECR, Amazon S3 oder CodeCommit Quellen können Sie auch eine Quellüberschreibung mithilfe eines Eingabe-Transformationseintrags erstellen, um das `revisionValue` In EventBridge für Ihr Pipeline-Ereignis zu verwenden, wobei das von der Quellereignisvariablen für Ihren Objektschlüssel, Ihren Commit oder Ihre Bild-ID abgeleitet `revisionValue` wird. Weitere Informationen finden Sie in dem optionalen Schritt für die Eingabe der Eingabetransformation, der in den Verfahren unter Amazon ECR-Quellaktionen und Ressourcen EventBridgeHerstellen einer Verbindung zu Amazon S3-Quellaktionen mit einer für Ereignisse aktivierten Quelle, oder CodeCommit Quellaktionen und EventBridge enthalten ist.

`Aktion TypeId`

Die Aktionstyp-ID wird als Kombination der folgenden vier Felder identifiziert.

`category`

Die Art der Aktion oder des Schritts in der Pipeline, z. B. eine Quellaktion. Jeder Aktionstyp hat einen bestimmten Satz gültiger Aktionsanbieter. Eine Liste der gültigen Anbieter nach Aktionstyp finden Sie unterReferenz der Aktionsstruktur.

Dies sind die gültigen `actionTypeId` Kategorien (Aktionstypen) für CodePipeline:

- `Source`
- `Build`
- `Approval`
- `Deploy`
- `Test`
- `Invoke`
- `Compute`

`owner`

Für alle derzeit unterstützten Aktionstypen ist die einzig gültige BesitzerzeichenfolgeAWS,`ThirdParty`, oder`Custom`. Die gültige Besitzerzeichenfolge für eine bestimmte Aktion finden Sie imReferenz der Aktionsstruktur.

Weitere Informationen finden Sie in der CodePipeline -API-Referenz.

`version`

Die Version der Aktion.

`provider`

Der Aktionsanbieter, z. CodeBuild B.

- Welche Anbietertypen für eine Aktionskategorie gültig sind, hängt von der Kategorie ab. Für eine Quellaktionskategorie ist beispielsweise ein gültiger Anbietertyp `S3CodeStarSourceConnection`,`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"},
```

InputArtifacts

Dieses Feld enthält die Eingabe-Artefaktstruktur, sofern dies für die Aktionskategorie unterstützt wird. Das Eingabeartefakt einer Aktion muss genau mit dem Ausgabeartefakt übereinstimmen, das in einer vorangegangenen Aktion deklariert wurde. 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"
    }
],
```

Beispielsweise kann eine Quellaktion keine Eingabeartefakte haben, da es sich um die erste Aktion in der Pipeline handelt. Eine Quellaktion hat jedoch immer Ausgabeartefakte, die von der folgenden Aktion verarbeitet werden. Die Ausgabeartefakte für eine Quellaktion sind die Anwendungsdateien aus dem Quell-Repository, die gezippt und über den Artefakt-Bucket bereitgestellt werden. Sie werden von der folgenden Aktion verarbeitet, z. B. einer CodeBuild Aktion, die mit Build-Befehlen auf die Anwendungsdateien einwirkt.

Ein Beispiel für Aktionen, die keine Ausgabeartefakte haben können, sind Bereitstellungsaktionen, die keine Ausgabeartefakte haben, da es sich bei diesen Aktionen in der Regel um die letzte Aktion in einer Pipeline handelt.

Name

Der Artefaktname für die Eingabeartefakte der Aktion.

Ausgabe-Artefakte

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.

Dieses Feld enthält die Ausgabe-Artefaktstruktur, sofern dies für die Aktionskategorie unterstützt wird. Das Ausgabeartefakt einer Aktion muss exakt mit dem Ausgabeartefakt übereinstimmen, das in einer vorangegangenen Aktion deklariert wurde. 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"
    }
],
```

Beispielsweise kann eine Quellaktion keine Eingabeartefakte haben, da es sich um die erste Aktion in der Pipeline handelt. Eine Quellaktion hat jedoch immer Ausgabeartefakte, die von der folgenden Aktion verarbeitet werden. Die Ausgabeartefakte für eine Quellaktion sind die Anwendungsdateien aus dem Quell-Repository, die gezippt und über den Artefakt-Bucket bereitgestellt werden. Sie werden von der folgenden Aktion verarbeitet, z. B. einer CodeBuild Aktion, die mit Build-Befehlen auf die Anwendungsdateien einwirkt.

Ein Beispiel für Aktionen, die keine Ausgabeartefakte haben können, sind Bereitstellungsaktionen, die keine Ausgabeartefakte haben, da es sich bei diesen Aktionen in der Regel um die letzte Aktion in einer Pipeline handelt.

`Name`

Der Artefaktname für die Ausgabeartefakte der Aktion.

`Konfiguration` (nach Aktionsanbieter)

Die Aktionskonfiguration enthält Details und Parameter, die dem Anbietertyp entsprechen. Im folgenden Abschnitt sind die Beispielparameter für die Aktionskonfiguration spezifisch für die S3-Quellaktion.

Die Aktionskonfiguration und die input/output Artefaktlimits können je nach Aktionsanbieter variieren. Eine Liste mit Beispielen für die Aktionskonfiguration nach Aktionsanbieter finden Sie unter Referenz der Aktionsstruktur und in Gültige Konfigurationsparameter für jeden Anbietertyp der Tabelle unter. Die Tabelle enthält einen Link zur Aktionsreferenz für jeden Anbietertyp, in der die Konfigurationsparameter für jede Aktion detailliert aufgeführt sind. Eine Tabelle mit den Grenzwerten für Eingabe- und Ausgabeartefakte für jeden Aktionsanbieter finden Sie unterGültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp.

Die folgenden Überlegungen gelten für die Arbeit mit Aktionen:

- Quellaktionen haben keine Eingabeartefakte, und Bereitstellungsaktionen haben keine Ausgabeartefakte.
- Bei Anbietern von Quellaktionen, die keine Verbindung verwenden, wie S3, müssen Sie mithilfe des `PollForSourceChanges` Parameters angeben, ob Ihre Pipeline automatisch gestartet werden soll, wenn eine Änderung erkannt wird. Siehe Gültige Einstellungen für den `PollForSourceChanges` Parameter.
- Informationen zum Konfigurieren der automatischen Änderungserkennung zum Starten Ihrer Pipeline oder zum Deaktivieren der Änderungserkennung finden Sie unterQuellaktionen und Methoden zur Erkennung von Änderungen.
- Um Trigger mit Filterung zu konfigurieren, verwenden Sie die Quellaktion für Verbindungen und finden Sie dann weitere Informationen unterAutomatisieren Sie das Starten von Pipelines mithilfe von Triggern und Filtern.
- Die Ausgabevariablen für jede Aktion finden Sie unterVariablenreferenz.

> **Note**
>
> Für Amazon ECR, Amazon S3 oder CodeCommit Quellen können Sie auch eine Quellüberschreibung mithilfe eines Eingabe-Transformationseintrags erstellen, um das `revisionValue` In EventBridge für Ihr Pipeline-Ereignis zu verwenden, wobei das aus der Quellereignisvariablen für Ihren Objektschlüssel, Ihren Commit oder Ihre Bild-ID abgeleitet `revisionValue` wird. Weitere Informationen finden Sie in dem optionalen Schritt für die Eingabe der Eingabetransformation, der in den Verfahren unter Amazon ECR-Quellaktionen und Ressourcen EventBridgeHerstellen einer Verbindung zu Amazon S3-Quellaktionen mit einer für Ereignisse aktivierten Quelle, oder CodeCommit Quellaktionen und EventBridge enthalten ist.

> **Important**
>
> Bei Pipelines, die länger als 30 Tage inaktiv sind, ist das Polling für die Pipeline deaktiviert. Weitere Informationen finden Sie unter Polling DisabledAt in der Pipeline-Strukturreferenz. Die Schritte zur Migration Ihrer Pipeline vom Polling zur ereignisbasierten Änderungserkennung finden Sie unter Methoden zur Änderungserkennung.

> **Note**
>
> Für die Aktionen „Quelle“ CodeCommit und „S3“ ist entweder eine konfigurierte Ressource zur Änderungserkennung (eine EventBridge Regel) erforderlich, oder Sie verwenden die Option, das Repository nach Quelländerungen abzufragen. Für Pipelines mit einer Bitbucket GitHub - oder GitHub Enterprise Server-Quellaktion müssen Sie keinen Webhook einrichten oder standardmäßig Polling verwenden. Die Aktion „Verbindungen“ verwaltet die Änderungserkennung für dich.

`Bestellung ausführen`

Eine positive Ganzzahl, die die Ausführungsreihenfolge der Aktion innerhalb der Phase angibt. Parallele Aktionen in der Phase werden so dargestellt, dass sie dieselbe Ganzzahl haben.

Beispielsweise werden zwei Aktionen mit einer Ausführungsreihenfolge von zwei parallel ausgeführt, nachdem die erste Aktion in der Phase ausgeführt wird.

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 Sequenz für eine Aktion angeben, indem Sie auf der Ebene in der Phase, auf der sie ausgeführt werden soll, die Option Aktionsgruppe hinzufügen auswählen. Sie können auch eine parallele Reihenfolge angeben, indem Sie Aktion hinzufügen wählen. Aktionsgruppe bezieht sich auf eine Ausführungsreihenfolge von einer oder mehreren 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 von Serienaktionen muss nicht in strikter Reihenfolge erfolgen. 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.

Gültige Aktionsanbieter 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.

Jede Aktionskategorie hat eine gültige Liste von Aktionsanbietern. Informationen zu den gültigen Aktionsanbietern für jede Aktionskategorie finden Sie unterReferenz der Aktionsstruktur.

Jede Aktionskategorie verfügt über einen designierten Satz von Anbietern. Jeder Aktionsanbieter, wie Amazon S3, hat einen Anbieternamen wie`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`.

Den Anbieternamen und die Inhaberinformationen für Ihren Aktionsanbieter finden Sie unter Referenz der Aktionsstruktur oderGültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp.

Diese Tabelle führt die gültigen Anbieter nach Aktionstyp auf.

> **Note**
>
> Informationen zu Bitbucket GitHub - oder GitHub Enterprise Server-Aktionen findest du im Thema CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen Aktionsreferenz.

Gültige Aktionsanbieter nach Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Der Pipeline-Typ wird unterstützt	Aktionsreferenz
Quelle	Amazon S3	V1, V2	Referenz zur Amazon S3-Quellaktion
	Amazon ECR	V1, V2	Referenz zur Amazon ECR-Quellaktion
	CodeCommit	V1, V2	CodeCommit Quellaktionsreferenz
	CodeStarSourceConnection (für Bitbucket- GitHub, GitHub Enterprise Server-Aktionen)	V1, V2	CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen

Aktionskategorie	Gültige Aktionsanbieter	Der Pipeline-Typ wird unterstützt	Aktionsreferenz
Entwicklung	Amazon ECR-Aktion `ECRBuildAndPublish`	Nur V2	`ECRBuildAndPublish` Aktionsreferenz erstellen
	CodeBuild	V1, V2	AWS CodeBuild Aktionsreferenz erstellen und testen
	Befehle und Aktion (siehe Compute)	Nur V2	
	Benutzerdefiniert CloudBees	V1, V2	Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Benutzerdefiniert Jenkins	V1, V2	Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Benutzerdefiniert TeamCity	V1, V2	Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
Test	CodeBuild	V1, V2	AWS CodeBuild Aktionsreferenz erstellen und testen
	AWS Device Farm	V1, V2	Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Benutzerdefiniert BlazeMeter	V1, V2	Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Der Pipeline-Typ wird unterstützt	Aktionsreferenz
	ThirdParty GhostInspector		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Benutzerdefiniert Jenkins		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	ThirdParty Micro Focus StormRunner Load		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	ThirdParty Nouvola		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	ThirdParty Runscope		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
Bereitstellen	Amazon S3		Referenz zur Amazon S3-Bereitstellungsaktion
	CloudFormation		CloudFormation Aktionsreferenz bereitstellen
	CodeDeploy		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Aktion EC2 Deploy	Nur V2	Amazon EC2-Aktionsreferenz

Aktionskategorie	Gültige Aktionsanbieter	Der Pipeline-Typ wird unterstützt	Aktionsreferenz
	Amazon ECS		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Amazon ECS (Blue/Green) (das ist die `CodeDeployToECS` Aktion)		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Amazon EKS-Aktion	Nur V2	Referenz zur Amazon Elastic Kubernetes Service EKS-Bereitstellungsaktion
	Elastische Bohnenstange		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	AWS AppConfig		AWS AppConfig Aktionsreferenz bereitstellen
	OpsWorks		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Servicekatalog		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Amazon Alexa		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
	Benutzerdefiniert XebiaLabs		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Der Pipeline-Typ wird unterstützt	Aktionsreferenz
Genehmigung	Manuell		Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
Invoke	CodePipelineAktion aufrufen		AWS CodePipeline Aktionsreferenz aufrufen
	AWS Lambda		AWS Lambda Aktionsreferenz aufrufen
	AWS Step Functions		AWS Step Functions Aktionsreferenz aufrufen
	InspectorScan		Referenz zum `InspectorScan` Aufrufen der Aktion durch Amazon Inspector
Datenverarbeitung	Befehle, Aktion		Aktionsreferenz für Befehle

Einige Aktionstypen 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.

Gültige Einstellungen für den **`PollForSourceChanges`** Parameter

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 CloudFormation -Vorlage hinzu.
- Erstellen Sie Ressourcen zur Änderungserkennung (CloudWatch Event-Regel, falls zutreffend).
- Stellen Sie den `PollForSourceChanges`-Parameter auf "false" ein.

> **Note**
>
> Wenn Sie eine CloudWatch Events-Regel oder einen Webhook erstellen, müssen Sie den Parameter auf false setzen, um zu vermeiden, dass die Pipeline mehrmals ausgelöst wird.

Der `PollForSourceChanges` Parameter wird nicht für Amazon ECR-Quellaktionen verwendet.

- **`PollForSourceChanges`**Standardwerte für Parameter

Quelle	Methode der Erstellung	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 CloudFormation, und der `PollForSo urceChanges` Parameter wird in der JSON-Ausgabe nicht angezeigt, er wird jedoch auf ² festgelegt`true`.	`BranchName": "main", "RepositoryName": "my- repo"`
Amazon S3	Pipeline wird mit der Konsole erstellt (und die Konsole erstellt Änderungserkennung	`"S3Bucket": "my-bucke t",`

Quelle	Methode der Erstellung	Beispiel für „Konfiguration“ JSON-Struktur-Ausgabe
	sressourcen). Der Parameter wird in der Pipeline-Strukturausgabe angezeigt und ist standardmäßig `false`.	`"S3ObjectKey": "object.zip", "PollForSourceChanges" : "false"`
	Die Pipeline wird mit der CLI oder erstellt CloudFormation, und der `PollForSourceChanges` Parameter wird in der JSON-Ausgabe nicht angezeigt, er wird jedoch auf `true` [2] festgelegt.	`"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 CloudFormation, und der `PollForSourceChanges` Parameter wird in der JSON-Ausgabe nicht angezeigt, er wird jedoch auf `true` [2] festgelegt.	`"Owner": "MyGitHubAccountName ", "Repo": "MyGitHubRepositoryName ", "Branch": " main", "OAuthToken": " ****"`

Quelle	Methode der Erstellung	Beispiel für „Konfiguration“ JSON-Struktur-Ausgabe
	² Wenn `PollForSourceChanges` er zu einem beliebigen Zeitpunkt zur JSON-Struktur oder zur CloudFormation Vorlage hinzugefügt wurde, wird er wie folgt angezeigt: `"PollForSourceChanges": "true",` ³ Informationen zu den Ressourcen zur Änderungserkennung, die für die einzelnen Quellanbieter gelten, finden Sie unter Methoden zur Änderungserkennung.	

Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp

Je nach Aktionstyp und Anbieter können Sie die folgende Anzahl von Eingabe- und Ausgabeartefakten haben.

Aktionstypbedingungen für Artefakte

Eigentümer	Aktionstyp	Anbieter	Gültige Zahl der Eingabeartefakte	Gültige Zahl der Ausgabeartefakte
`AWS`	Quelle	`S3`	0	1
`AWS`	Quelle	`CodeCommit`	0	1
`AWS`	Quelle	`ECR`	0	1
`ThirdParty`	Quelle	`CodeStarSourceConnection`	0	1
`AWS`	Entwicklung	`CodeBuild`	1 bis 5	0 bis 5
`AWS`	Test	`CodeBuild`	1 bis 5	0 bis 5

Eigentümer	Aktionstyp	Anbieter	Gültige Zahl der Eingabeartefakte	Gültige Zahl der Ausgabeartefakte
AWS	Test	DeviceFarm	1	0
AWS	Genehmigung	ThirdParty	0	0
AWS	Bereitstellen	S3	1	0
AWS	Bereitstellen	CloudFormation	0 bis 10	0 bis 1
AWS	Bereitstellen	CodeDeploy	1	0
AWS	Bereitstellen	ElasticBeanstalk	1	0
AWS	Bereitstellen	OpsWorks	1	0
AWS	Bereitstellen	ECS	1	0
AWS	Bereitstellen	ServiceCatalog	1	0
AWS	Invoke	Lambda	0 bis 5	0 bis 5
ThirdParty	Bereitstellen	AlexaSkillsKit	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

Gültige Konfigurationsparameter für jeden Anbietertyp

In diesem Abschnitt werden gültige `configuration`-Parameter für jeden Aktionsanbieter aufgeführt.

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	Required/Optional
Amazon S3 (Aktionsanbieter bereitstellen)	Weitere Informationen, einschließlich Beispielen zu den Aktionsparametern von Amazon S3 Deploy, finden Sie unterReferenz zur Amazon S3-Bereitstellungsaktion.		
Amazon S3 (Anbieter von Quellaktionen)	Weitere Informationen, einschließlich Beispielen zu Amazon S3-Quellaktionsparametern, finden Sie unterReferenz zur Amazon S3-Quellaktion.		
Amazon ECR	Weitere Informationen, einschließlich Beispielen zu Amazon ECR-Parametern, finden Sie unterReferenz zur Amazon ECR-Quellaktion.		
CodeCommit	Weitere Informationen, einschließlich Beispielen zu CodeCommit Parametern, finden Sie unterCodeCommit Quellaktionsreferenz.		
CodeStarSourceConnection Aktion für Bitbucket GitHub (per GitHub	Weitere Informationen, einschließlich Beispielen für die Aktionskonfiguration, findest du unterKonfigurationsparameter.		

Name des Anbieters	Anbietername in Aktionstyp	Konfigurationseigenschaften	Required/Optional
App), GHES und GitLab			
GitHub (über die OAuth-App)	Weitere Informationen, einschließlich Beispielen zu GitHub Parametern, finden Sie unter. GitHub Referenz zur Quellaktion (über die OAuth-App) Dies ist die GitHub Aktion der Version 1.		
AWS CloudFormation	Weitere Informationen, einschließlich Beispielen zu AWS CloudFormation Parametern, finden Sie unterCloudFormation Aktionsreferenz bereitstellen.		
CodeBuild	Weitere Beschreibungen und Beispiele zu CodeBuild Parametern finden Sie unterAWS CodeBuild Aktionsreferenz erstellen und testen.		
CodeDeploy	Weitere Beschreibungen und Beispiele zu CodeDeploy Parametern finden Sie unterAWS CodeDeploy Aktionsreferenz bereitstellen.		
AWS Device Farm	Weitere Beschreibungen und Beispiele zu AWS Device Farm Parametern finden Sie unterAWS Device Farm Referenz zur Testaktion.		
AWS Elastic Beanstalk	`ElasticBeanstalk`	`ApplicationName`	Erforderlich
		`EnvironmentName`	Erforderlich
AWS Lambda	Weitere Informationen, einschließlich Beispielen zu AWS Lambda Parametern, finden Sie unterAWS Lambda Aktionsreferenz aufrufen.		
AWS OpsWorks Stacks	`OpsWorks`	`Stack`	Erforderlich
		`Layer`	Optional
		`App`	Erforderlich
Amazon ECS	Weitere Beschreibungen und Beispiele zu Amazon ECS-Parametern finden Sie unterReferenz zur Amazon Elastic Container Service-Bereitstellungsaktion.		

Name des Anbieters	Anbietername in Aktionstyp	Konfigurationseigenschaften	Required/Optional
Amazon ECS und CodeDeploy (Blue/Green)	Weitere Beschreibungen und Beispiele zu Amazon ECS und CodeDeploy blue/green Parametern finden Sie unterAmazon Elastic Container Service und Referenz für Bereitstellungsaktionen in CodeDeploy Blaugrün.		
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

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

- Amazon EC2-Aktionsreferenz
- Referenz zur Amazon ECR-Quellaktion
- ECRBuildAndPublishAktionsreferenz erstellen
- Amazon Elastic Container Service und Referenz für Bereitstellungsaktionen in CodeDeploy Blaugrün
- Referenz zur Amazon Elastic Container Service-Bereitstellungsaktion
- Referenz zur Amazon Elastic Kubernetes Service EKS-Bereitstellungsaktion
- AWS Lambda Aktionsreferenz bereitstellen
- Referenz zur Amazon S3-Bereitstellungsaktion
- Referenz zur Amazon S3-Quellaktion
- AWS AppConfig Aktionsreferenz bereitstellen
- CloudFormation Aktionsreferenz bereitstellen
- CloudFormation StackSets Aktionsreferenz bereitstellen
- AWS CodeBuild Aktionsreferenz erstellen und testen

- AWS CodePipeline Aktionsreferenz aufrufen
- CodeCommit Quellaktionsreferenz
- AWS CodeDeploy Aktionsreferenz bereitstellen
- CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen
- Aktionsreferenz für Befehle
- AWS Device Farm Referenz zur Testaktion
- Referenz zur Elastic Beanstalk-Bereitstellungsaktion
- Referenz zum InspectorScan Aufrufen der Aktion durch Amazon Inspector
- AWS Lambda Aktionsreferenz aufrufen
- AWS OpsWorks Aktionsreferenz bereitstellen
- AWS Service Catalog Aktionsreferenz bereitstellen
- AWS Step Functions Aktionsreferenz aufrufen

Amazon EC2-Aktionsreferenz

Sie verwenden eine Amazon EC2 EC2-Aktion, um Anwendungscode für Ihre Bereitstellungsflotte bereitzustellen. Ihre Bereitstellungsflotte kann aus Amazon EC2-Linux-Instances oder SSM-managed Linux-Knoten bestehen. Auf Ihren Instances muss der SSM-Agent installiert sein.

> **Note**
>
> Diese Aktion unterstützt nur Linux-Instanztypen. Die maximal unterstützte Flottengröße beträgt 500 Instances.

Die Aktion wählt eine Anzahl von Instances auf der Grundlage eines angegebenen Maximums aus. Die fehlgeschlagenen Instanzen aus früheren Instanzen werden zuerst ausgewählt. Die Aktion überspringt die Bereitstellung auf bestimmten Instances, wenn die Instanz bereits die Bereitstellung desselben Eingabeartefakts erhalten hat, z. B. in einem Fall, in dem die Aktion zuvor fehlgeschlagen ist.

> **Note**
>
> Diese Aktion wird nur für Pipelines vom Typ V2 unterstützt.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Richtlinienberechtigungen für die EC2-Bereitstellungsaktion
- Referenz zur Spezifikationsdatei bereitstellen
- Aktionsdeklaration
- Aktionsdeklaration mit Beispiel für eine Deploy-Spezifikation
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `EC2`
- Version: `1`

Konfigurationsparameter

InstanceTagKey

Erforderlich: Ja

Der Tag-Schlüssel der Instances, die Sie in Amazon EC2 erstellt haben, z. B. `Name`

InstanceTagValue

Erforderlich: Nein

Der Tag-Wert der Instances, die Sie in Amazon EC2 erstellt haben, z. B. `my-instances`

Wenn dieser Wert nicht angegeben ist, InstanceTagKey werden alle Instances mit abgeglichen.

InstanceType

Erforderlich: Ja

Der Typ der in Amazon EC2 erstellten Instanzen oder SSM-Knoten. Die gültigen Werte sind `EC2` und `SSM_MANAGED_NODE`.

Sie müssen den SSM-Agenten bereits auf allen Instances erstellt, markiert und installiert haben.

> **Note**
>
> Wenn Sie die Instance erstellen, erstellen oder verwenden Sie eine vorhandene EC2-Instance-Rolle. Um `Access Denied` Fehler zu vermeiden, müssen Sie der Instance-Rolle S3-Bucket-Berechtigungen hinzufügen, um die Instance-Berechtigungen für den CodePipeline Artefakt-Bucket zu erteilen. Erstellen Sie eine Standardrolle oder aktualisieren Sie Ihre bestehende Rolle mit der `s3:GetObject` Berechtigung, die auf den Artefakt-Bucket für die Region Ihrer Pipeline beschränkt ist.

TargetDirectory

Erforderlich: Ja (wenn ein Skript angegeben ist)

Das Verzeichnis, das auf Ihrer Amazon EC2-Instance zum Ausführen von Skripten verwendet werden soll.

DeploySpec

Erforderlich: Ja (wenn die Bereitstellungsspezifikation angegeben ist)

Die Datei, die zur Konfiguration von Bereitstellungs-, Installations- und Lebenszyklusereignissen verwendet werden soll. Beschreibungen und Informationen zu den Feldern für die Bereitstellungsspezifikation finden Sie unterReferenz zur Spezifikationsdatei bereitstellen. Informationen zum Anzeigen einer Aktionskonfiguration mit der angegebenen Bereitstellungsspezifikationsdatei finden Sie im Beispiel unter. Aktionsdeklaration mit Beispiel für eine Deploy-Spezifikation

MaxBatch

Erforderlich: Nein

Die maximale Anzahl von Instanzen, die parallel bereitgestellt werden dürfen.

MaxError

Erforderlich: Nein

Die maximale Anzahl von Instanzfehlern, die während der Bereitstellung zulässig sind.

TargetGroupNameList

Erforderlich: Nein

Die Liste der Zielgruppennamen für die Bereitstellung. Sie müssen die Zielgruppen bereits erstellt haben.

Zielgruppen stellen eine Reihe von Instanzen zur Verfügung, um bestimmte Anfragen zu bearbeiten. Wenn die Zielgruppe angegeben ist, werden Instanzen vor der Bereitstellung aus der Zielgruppe entfernt und nach der Bereitstellung wieder zur Zielgruppe hinzugefügt.

PreScript

Erforderlich: Nein

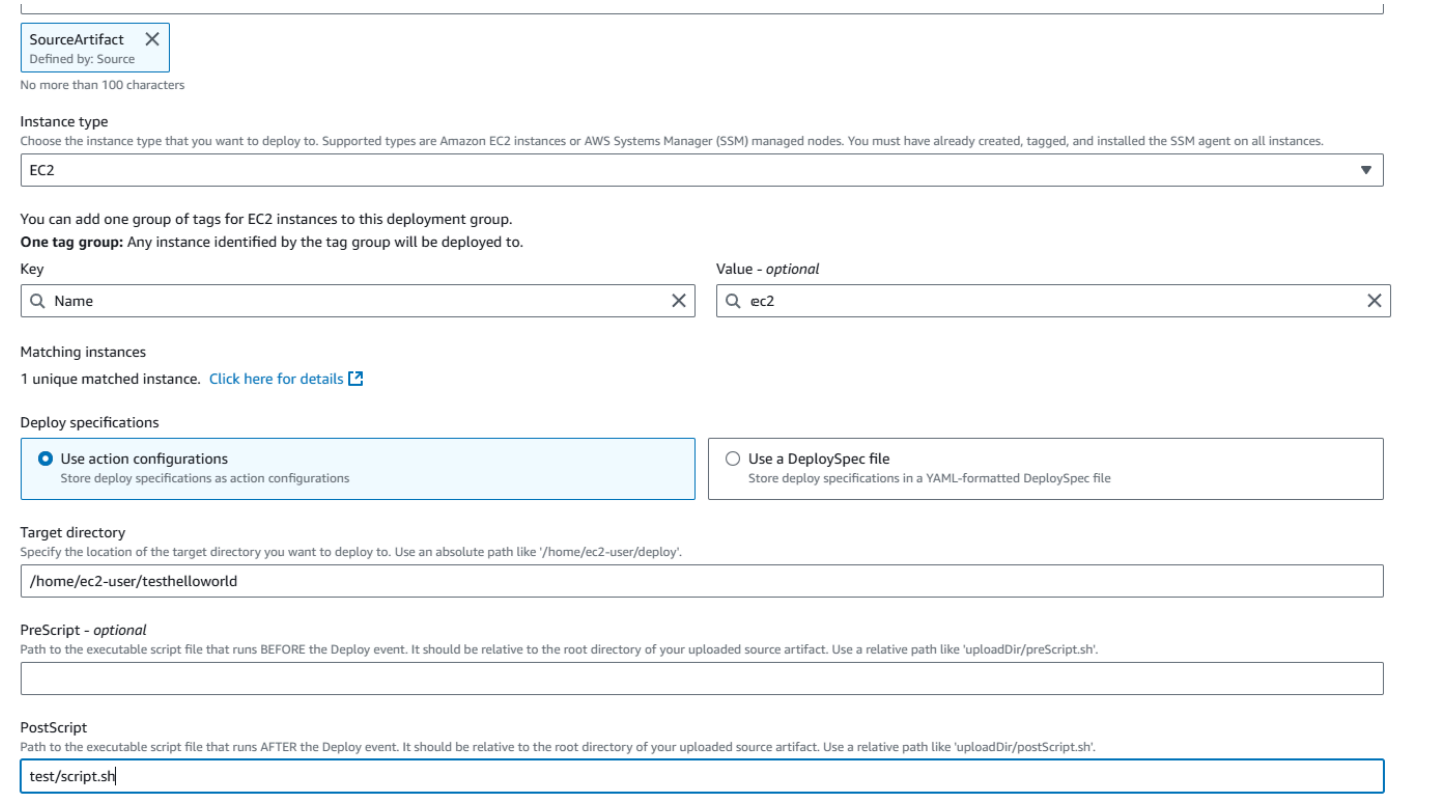
Das Skript, das vor der Bereitstellungsphase der Aktion ausgeführt werden soll.

PostScript

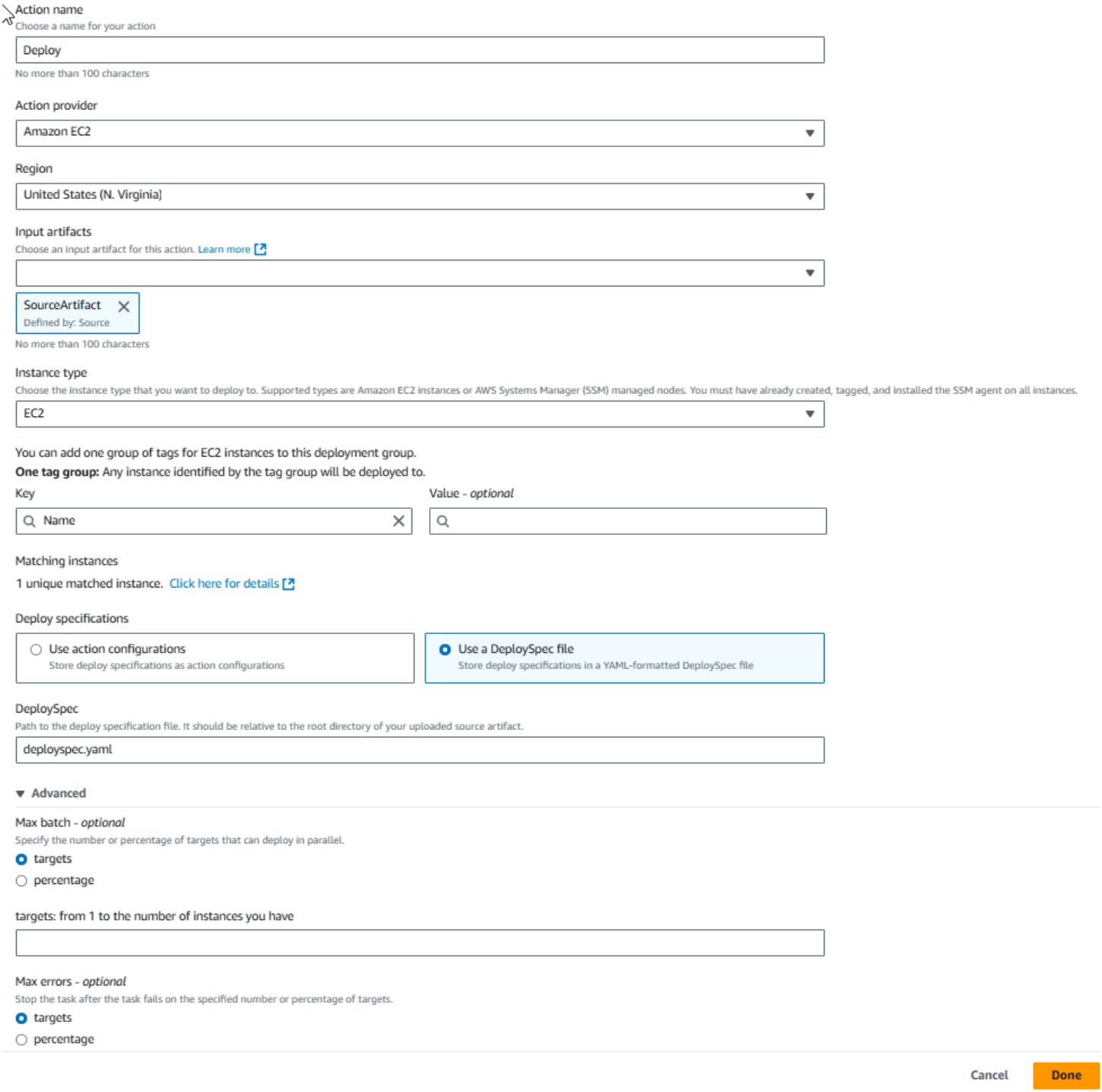
Erforderlich: Ja

Das Skript, das nach der Aktion Deploy-Phase ausgeführt werden soll.

Die folgende Abbildung zeigt ein Beispiel für die Seite „Bearbeiten“ für die Aktion, auf der Aktionskonfigurationen verwenden ausgewählt ist.



Die folgende Abbildung zeigt ein Beispiel für die Seite „Bearbeiten“ für die Aktion, auf der die Option DeploySpec Datei verwenden ausgewählt ist.



Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Die bereitgestellten Dateien, falls vorhanden, zur Unterstützung der Skriptaktionen während der Bereitstellung.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Richtlinienberechtigungen für die EC2-Bereitstellungsaktion

Wenn die Aktion CodePipeline ausgeführt wird, benötigt die CodePipeline Servicerolle die folgenden Berechtigungen, die entsprechend auf den Zugriff mit den geringsten Rechten beschränkt sind.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "StatementWithAllResource",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "elasticloadbalancing:DescribeTargetGroupAttributes",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "ssm:CancelCommand",
                "ssm:DescribeInstanceInformation",
                "ssm:ListCommandInvocations"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "StatementForLogs",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
```

```
                "arn:aws:logs:us-east-1:111122223333:log-group:/aws/codepipeline/
{{pipelineName}}:*"
            ]
        },
        {
            "Sid": "StatementForElasticloadbalancing",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:us-east-1:111122223333:targetgroup/
[[targetGroupName]]/*"
            ]
        },
        {
            "Sid": "StatementForSsmOnTaggedInstances",
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1:111122223333:instance/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/{{tagKey}}": "{{tagValue}}"
                }
            }
        },
        {
            "Sid": "StatementForSsmApprovedDocuments",
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1::document/AWS-RunPowerShellScript",
                "arn:aws:ssm:us-east-1::document/AWS-RunShellScript"
            ]
        }
    ]
```

```
}
```

Protokollieren Sie Gruppen für Ihre Pipeline in Protokollen CloudWatch

Wenn die Aktion CodePipeline ausgeführt wird, CodePipeline wird eine Protokollgruppe erstellt, die den Namen der Pipeline wie folgt verwendet. Auf diese Weise können Sie die Berechtigungen zum Protokollieren von Ressourcen anhand des Pipeline-Namens einschränken.

```
/aws/codepipeline/MyPipelineName
```

Die folgenden Berechtigungen für die Protokollierung sind in den obigen Updates für die Servicerolle enthalten.

- Protokolle: CreateLogGroup
- Protokolle: CreateLogStream
- Protokolle: PutLogEvents

Um Protokolle in der Konsole mithilfe der Dialogseite mit den Aktionsdetails anzuzeigen, muss der Konsolenrolle die Berechtigung zum Anzeigen von Protokollen hinzugefügt werden. Weitere Informationen finden Sie im Beispiel für eine Konsolenberechtigungsrichtlinie unterZum Anzeigen von Rechenprotokollen in der Konsole sind Berechtigungen erforderlich.

Berechtigungen für CloudWatch Protokolle nach der Dienstrollenrichtlinie

Wenn die Aktion CodePipeline ausgeführt wird, CodePipeline wird eine Protokollgruppe erstellt, die den Namen der Pipeline wie folgt verwendet. Auf diese Weise können Sie die Berechtigungen zum Protokollieren von Ressourcen anhand des Pipeline-Namens einschränken.

```
/aws/codepipeline/MyPipelineName
```

Um Protokolle in der Konsole mithilfe der Dialogseite mit den Aktionsdetails anzuzeigen, muss die Berechtigung zum Anzeigen von Protokollen zur Konsolenrolle hinzugefügt werden. Weitere Informationen finden Sie im Beispiel für eine Konsolenberechtigungsrichtlinie unterZum Anzeigen von Rechenprotokollen in der Konsole sind Berechtigungen erforderlich.

Referenz zur Spezifikationsdatei bereitstellen

Wenn die Aktion CodePipeline ausgeführt wird, können Sie eine Spezifikationsdatei angeben, um die Bereitstellung auf Ihren Instances zu konfigurieren. Die Deploy-Spezifikationsdatei gibt an, was installiert werden soll und welche Lifecycle-Event-Hooks als Reaktion auf Ereignisse im Bereitstellungszyklus ausgeführt werden sollen. Die Bereitstellungsspezifikationsdatei ist immer. YAML-formatted Die Bereitstellungsspezifikationsdatei wird verwendet, um:

- Ordnen Sie die Quelldateien in Ihrer Anwendungsrevision den entsprechenden Zielen auf der Instance zu.
- Geben Sie benutzerdefinierte Berechtigungen für bereitgestellte Dateien an.
- Legen Sie Scripts fest, die auf jeder Instance in verschiedenen Phasen des Bereitstellungsprozesses ausgeführt werden.

Die Bereitstellungsspezifikationsdatei unterstützt bestimmte Bereitstellungskonfigurationsparameter, die von CodeDeploy der AppSpec Datei unterstützt werden. Sie können Ihre vorhandene AppSpec Datei direkt verwenden. Alle nicht unterstützten Parameter werden ignoriert. Weitere Informationen zur AppSpec Datei in finden Sie in CodeDeploy der Referenz zur Anwendungsspezifikationsdatei im CodeDeploy Benutzerhandbuch.

Die Parameter für die Dateibereitstellung werden wie folgt angegeben.

- `files`— Die Bereitstellungsspezifikationsdatei bezeichnet das `source:` und `destination:` für die Bereitstellungsdateien.
- `scripts`— Die skriptgesteuerten Ereignisse für die Bereitstellung. Zwei Ereignisse werden unterstützt: `BeforeDeploy` und`AfterDeploy`.
- `hooks`— Die Lifecycle-Hooks für das Ereignis. Die folgenden Hooks werden unterstützt: `ApplicationStop` `BeforeInstall`,`AfterInstall`, `ApplicationStart`, und`ValidateService`.

> **Note**
>
> Der Hooks-Parameter ist aus AppSpec Kompatibilitätsgründen verfügbar CodeDeploy und nur in Version 0.0 (AppSpec Format) verfügbar. Für dieses Format CodePipeline werden die Ereignisse nach bestem Wissen und Gewissen zugeordnet.

In der Spezifikationsdatei müssen die richtigen YAML-Abstände verwendet werden. Andernfalls wird ein Fehler ausgegeben, wenn die Positionen und die Anzahl der Leerzeichen in einer Bereitstellungsspezifikationsdatei nicht korrekt sind. Weitere Informationen über Leerzeichen finden Sie in der YAML-Spezifikation.

Ein Beispiel für eine Deploy-Spezifikationsdatei finden Sie unten.

```
version: 0.1
files:
  - source: /index.html
    destination: /var/www/html/
scripts:
  BeforeDeploy:
    - location: scripts/install_dependencies
      timeout: 300
      runas: myuser
  AfterDeploy:
    - location: scripts/start_server
      timeout: 300
      runas: myuser
```

Informationen zum Anzeigen einer Aktionskonfiguration mit der angegebenen Bereitstellungsspezifikationsdatei finden Sie im Beispiel unter. Aktionsdeklaration mit Beispiel für eine Deploy-Spezifikation

Aktionsdeklaration

YAML

```
name: DeployEC2
actions:
- name: EC2
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: EC2
    version: '1'
  runOrder: 1
  configuration:
    InstanceTagKey: Name
    InstanceTagValue: my-instances
    InstanceType: EC2
```

```
    PostScript: "test/script.sh",
    TargetDirectory: "/home/ec2-user/deploy"
  outputArtifacts: []
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
```

JSON

```
{
    "name": "DeployEC2",
    "actions": [
        {
            "name": "EC2Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "EC2",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "InstanceTagKey": "Name",
                "InstanceTagValue": "my-instances",
                "InstanceType": "EC2",
                "PostScript": "test/script.sh",
                "TargetDirectory": "/home/ec2-user/deploy"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1"
        }
    ]
},
```

Aktionsdeklaration mit Beispiel für eine Deploy-Spezifikation

YAML

```
name: DeployEC2
actions:
- name: EC2
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: EC2
    version: '1'
  runOrder: 1
  configuration:
    DeploySpec: "deployspec.yaml"
    InstanceTagKey: Name
    InstanceTagValue: my-instances
    InstanceType: EC2
  outputArtifacts: []
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
```

JSON

```
{
    "name": "DeployEC2",
    "actions": [
        {
            "name": "EC2Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "EC2",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "DeploySpec": "deployspec.yaml",
                "InstanceTagKey": "Name",
                "InstanceTagValue": "my-instances",
                "InstanceType": "EC2"
            },
```

```
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Bereitstellung auf Amazon EC2-Instances mit CodePipeline— Dieses Tutorial führt Sie durch die Erstellung von EC2-Instances, in denen Sie eine Skriptdatei bereitstellen, sowie durch die Erstellung der Pipeline mithilfe der EC2-Aktion.
- Die EC2-Bereitstellungsaktion schlägt fehl und es wird die Fehlermeldung `Keine solche Datei` angezeigt— In diesem Thema wird die Problembehandlung bei Fehlern mit der EC2-Aktion beschrieben, die die Datei nicht gefunden hat.

Referenz zur Amazon ECR-Quellaktion

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 weiteren Quellartefakte zuzulassen. Weitere Informationen finden Sie unter Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-CodeDeploy.

Wenn Sie die Konsole verwenden, um Ihre Pipeline zu erstellen oder zu bearbeiten, CodePipeline wird eine EventBridge Regel erstellt, die Ihre Pipeline startet, wenn eine Änderung im Repository erfolgt.

> **Note**
>
> Für Amazon ECR, Amazon S3 oder CodeCommit Quellen können Sie auch eine Quellüberschreibung mithilfe eines Eingabe-Transformationseintrags erstellen, um das `revisionValue` In EventBridge für Ihr Pipeline-Ereignis zu verwenden, wobei das aus der Quellereignisvariablen für Ihren Objektschlüssel, Ihren Commit oder Ihre Bild-ID abgeleitet `revisionValue` wird. Weitere Informationen finden Sie in dem optionalen Schritt für die Eingabe der Eingabetransformation, der in den Verfahren unter Amazon ECR-Quellaktionen und Ressourcen EventBridgeHerstellen einer Verbindung zu Amazon S3-Quellaktionen mit einer für Ereignisse aktivierten Quelle, oder CodeCommit Quellaktionen und EventBridge enthalten ist.

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
- Berechtigungen für Servicerollen: Amazon ECR-Aktion
- Aktionserklärung (Amazon ECR-Beispiel)
- 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 Detail.json Image-Datei für Amazon blue/green ECS-Bereitstellungsaktionen.

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 Variablenreferenz.

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.

> **Note**
>
> Die `ImageTag` Ausgabevariable wird nicht ausgegeben, wenn die Quellrevision überschrieben wird.

ImageDigest

Der `sha256`-Digest des Image-Manifests.

ImageURI

Der URI für das Image.

Berechtigungen für Servicerollen: Amazon ECR-Aktion

Um Amazon ECR-Support zu erhalten, fügen Sie Ihrer Richtlinienerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "ecr:DescribeImages"
    ],
    "Resource": "resource_ARN"
},
```

Weitere Informationen zu dieser Aktion finden Sie unterReferenz zur Amazon ECR-Quellaktion.

Aktionserklärung (Amazon ECR-Beispiel)

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 Bereitstellung ECS-to-CodeDeploy— Dieses Tutorial enthält eine AppSpec Beispieldatei sowie eine CodeDeploy Beispielanwendung und eine Bereitstellungsgruppe zum Erstellen einer Pipeline mit einer CodeCommit Amazon ECR-Quelle, die auf Amazon ECS-Instances bereitgestellt wird.

ECRBuildAndPublishAktionsreferenz erstellen

Mit dieser Build-Aktion können Sie die Erstellung und Übertragung eines neuen Images automatisieren, wenn sich Ihre Quelle ändert. Diese Aktion basiert auf einem angegebenen Dockerfile-Speicherort und überträgt das Image. Diese Build-Aktion ist nicht mit der Amazon ECR-Quellaktion in identisch, die die Pipeline auslöst CodePipeline, wenn eine Änderung in Ihrem Amazon ECR-Quell-Repository erfolgt. Informationen zu dieser Aktion finden Sie unter. Referenz zur Amazon ECR-Quellaktion

Dies ist keine Quellaktion, die die Pipeline auslöst. Mit dieser Aktion wird ein Image erstellt und in Ihr Amazon ECR-Image-Repository übertragen.

Sie müssen bereits ein Amazon ECR-Repository erstellt und Ihrem Quellcode-Repository ein Dockerfile hinzugefügt haben, z. B. GitHub bevor Sie die Aktion zu Ihrer Pipeline hinzufügen.

> **Important**
>
> Diese Aktion verwendet CodePipeline Managed CodeBuild Compute, um Befehle in einer Build-Umgebung auszuführen. Für die Ausführung der Befehlsaktion fallen separate Gebühren in an AWS CodeBuild.

> **Note**
>
> Diese Aktion ist nur für Pipelines vom Typ V2 verfügbar.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Berechtigungen für Servicerollen: Aktion ECRBuildAndPublish
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Build`
- Eigentümer: `AWS`
- Anbieter: `ECRBuildAndPublish`
- Version: `1`

Konfigurationsparameter

ECRRepositoryName

Erforderlich: Ja

Der Name des Amazon ECR-Repositorys, in das das Bild übertragen wird.

DockerFilePath

Erforderlich: Nein

Der Speicherort des Dockerfiles, das zum Erstellen des Images verwendet wurde. Optional können Sie einen alternativen Dockerfile-Speicherort angeben, wenn er sich nicht auf der Stammebene befindet.

> **Note**
>
> Wenn kein Wert für angegeben `DockerFilePath` ist, wird standardmäßig die Stammebene des Quell-Repositorys verwendet.

ImageTags

Erforderlich: Nein

Die für das Bild verwendeten Tags. Sie können mehrere Tags als kommagetrennte Liste von Zeichenketten eingeben.

> **Note**
>
> Wenn kein Wert für `ImageTags` angegeben ist, wird standardmäßig der Wert `latest` verwendet.

RegistryType

Erforderlich: Nein

Gibt an, ob das Repository öffentlich oder privat ist. Gültige Werte sind `private | public`.

> **Note**
>
> Wenn kein Wert für `RegistryType` angegeben ist, wird standardmäßig der Wert `private` verwendet.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Das von der Quellaktion erzeugte Artefakt, das das Dockerfile enthält, das zum Erstellen des Images benötigt wird.

Ausgabeartefakte

- Anzahl der Artefakte: `0`

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 Variablenreferenz.

ECRImageDigestId

Der `sha256`-Digest des Image-Manifests.

ECRRepositoryName

Der Name des Amazon ECR-Repositorys, in das das Bild übertragen wurde.

Berechtigungen für Servicerollen: Aktion **`ECRBuildAndPublish`**

Für die Unterstützung der `ECRBuildAndPublish` Aktion fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

```
{
    "Statement": [
         {
            "Sid": "ECRRepositoryAllResourcePolicy",
            "Effect": "Allow",
            "Action": [
                "ecr:DescribeRepositories",
                "ecr:GetAuthorizationToken",
                "ecr-public:DescribeRepositories",
                "ecr-public:GetAuthorizationToken"
            ],
        "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchCheckLayerAvailability"
            ],
            "Resource": "PrivateECR_Resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr-public:GetAuthorizationToken",
                "ecr-public:DescribeRepositories",
                "ecr-public:InitiateLayerUpload",
                "ecr-public:UploadLayerPart",
                "ecr-public:CompleteLayerUpload",
                "ecr-public:PutImage",
                "ecr-public:BatchCheckLayerAvailability",
                "sts:GetServiceBearerToken"
            ],
            "Resource": "PublicECR_Resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sts:GetServiceBearerToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Fügen Sie Ihrer Servicerolle außerdem die folgenden Berechtigungen hinzu, falls sie nicht bereits für die `Commands` Aktion hinzugefügt wurden, um CloudWatch Protokolle einzusehen.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
```

```
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Beschränken Sie die Berechtigungen auf die Ebene der Pipeline-Ressourcen, indem Sie die ressourcenbasierten Berechtigungen in der Richtlinie zur Servicerolle verwenden.

Weitere Informationen zu dieser Aktion finden Sie unter. ECRBuildAndPublishAktionsreferenz erstellen

Aktionsdeklaration

YAML

```
name: ECRBuild
actionTypeId:
  category: Build
  owner: AWS
  provider: ECRBuildAndPublish
  version: '1'
runOrder: 1
configuration:
  ECRRepositoryName: actions/my-imagerepo
outputArtifacts: []
inputArtifacts:
- name: SourceArtifact
region: us-east-1
namespace: BuildVariables
```

JSON

```
{
    "name": "ECRBuild",
    "actionTypeId": {
        "category": "Build",
        "owner": "AWS",
        "provider": "ECRBuildAndPublish",
```

```
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {
        "ECRRepositoryName": "actions/my-imagerepo"
    },
    "outputArtifacts": [],
    "inputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "region": "us-east-1",
    "namespace": "BuildVariables"
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen Sie ein Docker-Image und übertragen Sie es auf Amazon ECR mit CodePipeline (Typ V2)— Dieses Tutorial enthält ein Beispiel für eine Docker-Datei und Anweisungen zum Erstellen einer Pipeline, die Ihr Image bei einer Änderung an Ihrem Quell-Repository an ECR überträgt und dann auf Amazon ECS bereitstellt.

Amazon Elastic Container Service und Referenz für Bereitstellungsaktionen in CodeDeploy Blaugrün

Sie können eine Pipeline konfigurieren AWS CodePipeline , die Containeranwendungen mithilfe einer blue/green Bereitstellung bereitstellt. In einer blue/green Bereitstellung können Sie neben der alten Version 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 fertige 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, z. B. ein CodeCommit Repository, verwendet, in dem Ihre Amazon ECS-Aufgabendefinition gespeichert ist.

Bevor Sie Ihre Pipeline erstellen, müssen Sie die Amazon ECS-Ressourcen, die Ressourcen sowie den CodeDeploy Load Balancer und die Zielgruppe bereits erstellt haben. Sie müssen das Image bereits mit Tags versehen 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 Aktion zur CodeDeploy blue/green Bereitstellung von Amazon ECS beschrieben CodePipeline. Referenzinformationen zu den standardmäßigen Bereitstellungsaktionen von Amazon ECS finden Sie unterReferenz zur Amazon Elastic Container Service-Bereitstellungsaktion. CodePipeline

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Berechtigungen für Servicerollen: CodeDeployToECS Aktion
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

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 Bereitstellungsgruppe, die für Amazon ECS-Aufgabensätze angegeben wurde, 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 Eingabe-Artefakts, das die Aufgabendefinitionsdatei für die Bereitstellungsaktion bereitstellt. Dies ist im Allgemeinen der Name des Ausgabeartefakts der Quellaktion. Wenn Sie die Konsole verwenden, lautet der Standardname für das Ausgabe-Artefakt 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 Ausgabe-Artefakt der Quellaktion. `SourceArtifact` `TaskDefinition`In der AppSpec Datei 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. im 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, der 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 in der Regel das Ausgabe-Artefakt 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 Images, das im Image-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, dann nach dem Image im Image-Repository, generiert dann dynamisch eine neue Revision der Aufgabendefinition und führt schließlich die AppSpec Befehle aus, um das Task-Set und den Container im Cluster bereitzustellen.

  Die `CodeDeployToECS` Aktion sucht nach einer `imageDetail.json` Datei, die die Image-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 Detail.json Image-Datei für Amazon blue/green ECS-Bereitstellungsaktionen.

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 Detail.json Bilddateien abgerufen wurde. Wenn Sie beispielsweise IMAGE1_NAME als Image1ContainerName Parameter festlegen, sollten Sie den Platzhalter <IMAGE1_NAME>als Wert des Bildfeldes in Ihrer Aufgabendefinitionsdatei angeben. In diesem Fall ersetzt die CodeDeployToECS Aktion den Platzhalter durch die <IMAGE1_NAME>tatsächliche Bild-URI, die aus dem Bild Detail.json in dem von Ihnen angegebenen Artefakt abgerufen wurde. Image1ArtifactName

Bei Aktualisierungen von Aufgabendefinitionen enthält die CodeDeploy `AppSpec.yaml` Datei die `TaskDefinition` Eigenschaft.

```
TaskDefinition: <TASK_DEFINITION>
```

Diese Eigenschaft wird durch die `CodeDeployToECS` Aktion aktualisiert, nachdem die neue Aufgabendefinition erstellt wurde.

<TASK_DEFINITION>Für den Wert des `TaskDefinition` Feldes muss der Platzhaltertext lauten. Die `CodeDeployToECS` Aktion ersetzt diesen Platzhalter durch den tatsächlichen ARN der dynamisch generierten Aufgabendefinition.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten für diesen Aktionstyp nicht.

Berechtigungen für Servicerollen: **`CodeDeployToECS`** Aktion

Für die `CodeDeployToECS` Aktion (blue/green Bereitstellungen) sind die folgenden Mindestberechtigungen erforderlich, um Pipelines mit einer Aktion CodeDeploy zur Amazon blue/green ECS-Bereitstellung zu erstellen.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCodeDeployDeploymentActions",
            "Action": [
                "codedeploy:CreateDeployment",
                "codedeploy:GetDeployment"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:deploymentgroup:
[[ApplicationName]]/*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "AllowCodeDeployApplicationActions",
            "Action": [
                "codedeploy:GetApplication",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:application:
[[ApplicationName]]",
                "arn:aws:codedeploy:*:111122223333:application:
[[ApplicationName]]/*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "AllowCodeDeployDeploymentConfigAccess",
            "Action": [
                "codedeploy:GetDeploymentConfig"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:deploymentconfig:*"
            ],
            "Effect": "Allow"
        },
        {
```

```
            "Sid": "AllowECSRegisterTaskDefinition",
            "Action": [
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "AllowPassRoleToECS",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/[[PassRoles]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ecs.amazonaws.com",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Sie können sich für die Verwendung der Tagging-Autorisierung in Amazon ECS entscheiden. Wenn Sie sich anmelden, müssen Sie die folgenden Berechtigungen gewähren:. `ecs:TagResource` Weitere Informationen zur Anmeldung und zur Feststellung, ob die Genehmigung erforderlich ist und ob die Tag-Autorisierung durchgesetzt wird, finden Sie im 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 Amazon ECS-Aufgabenausführung und IAM-Rollen für Aufgaben.

Sie können die Liste der Dienste unter dieser `iam:PassedToService` Bedingung auch erweitern`ecs-tasks.amazonaws.com`, wie im obigen Beispiel gezeigt.

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 Bereitstellung ECS-to-CodeDeploy— Dieses Tutorial führt Sie Schritt für Schritt durch die Erstellung der CodeDeploy 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, in der Ihr Docker-Image-Name, Ihr Containername, Ihr Amazon ECS-Servicename und Ihre Load Balancer-Konfiguration aufgeführt sind. Das Tutorial führt Sie dann durch die Erstellung der AppSpec Datei und der Pipeline für Ihre Bereitstellung.

> **Note**
>
> In diesem Thema und Tutorial wird die CodeDeploy/ECS blue/green Aktion für beschrieben CodePipeline. Informationen zu ECS-Standardaktionen finden Sie unter Tutorial: Continuous Deployment with CodePipeline. CodePipeline

- AWS CodeDeploy Benutzerhandbuch — Informationen zur Verwendung des Load Balancers, des Produktions-Listeners, der Zielgruppen und Ihrer Amazon ECS-Anwendung in einer blue/green 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-Aufgabensets finden Sie unter Was ist Amazon ECS?

Referenz zur Amazon Elastic Container Service-Bereitstellungsaktion

Sie können eine Amazon ECS-Aktion verwenden, um einen Amazon ECS-Service und eine Aufgabenliste 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 Version 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 Überarbeitungen.

Bevor Sie Ihre Pipeline erstellen, müssen Sie bereits die Amazon ECS-Ressourcen erstellt, das Image in Ihrem Image-Repository markiert und 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 Amazon ECS und CodeDeploy blue/green Bereitstellungsaktionen finden Sie unterAmazon Elastic Container Service und Referenz für Bereitstellungsaktionen in CodeDeploy Blaugrün. CodePipeline

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Berechtigungen für Servicerollen: Amazon ECS-Standardaktion
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

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 Image und das Tag. Sie verwenden diese Datei für ECS-Standardbereitstellungen. Weitere Informationen erhalten Sie unter Input artifacts (Eingabeartefakte) und imagedefinitions.json-Datei 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. Beispiel:

```
"DeploymentTimeout": "15"
```

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Die Aktion sucht im `imagedefinitions.json` Quelldatei-Repository für die Pipeline nach einer Datei. Ein Bilddefinitionsdokument ist eine JSON-Datei, die den Namen Ihres Amazon ECS-Containers sowie das Bild und den 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, bei der die Aktion nicht automatisiert ist. Informationen zur `imagedefinitions.json`-Datei finden Sie unter imagedefinitions.json-Datei 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.

Berechtigungen für Servicerollen: Amazon ECS-Standardaktion

Für Amazon ECS sind die folgenden Mindestberechtigungen erforderlich, um Pipelines mit einer Amazon ECS-Bereitstellungsaktion zu erstellen.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "TaskDefinitionPermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeTaskDefinition",
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "ECSServicePermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeServices",
                "ecs:UpdateService"
            ],
            "Resource": [
                "arn:aws:ecs:*:111122223333:service/[[clusters]]/*"
            ]
        },
        {
            "Sid": "ECSTagResource",
            "Effect": "Allow",
            "Action": [
```

```
                "ecs:TagResource"
            ],
            "Resource": [
                "arn:aws:ecs:*:111122223333:task-definition/
[[taskDefinitions]]:*"
            ],
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "RegisterTaskDefinition"
                    ]
                }
            }
        },
        {
            "Sid": "IamPassRolePermissions",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/[[passRoles]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ecs.amazonaws.com",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Sie können sich für die Verwendung der Tagging-Autorisierung in Amazon ECS entscheiden. Wenn Sie sich anmelden, müssen Sie die folgenden Berechtigungen gewähren:. `ecs:TagResource` Weitere Informationen zur Anmeldung und zur Feststellung, ob die Genehmigung erforderlich ist und ob die Tag-Autorisierung durchgesetzt wird, finden Sie im Zeitplan für die Tagging-Autorisierung im Amazon Elastic Container Service Developer Guide.

Sie müssen die `iam:PassRole` Berechtigungen hinzufügen, um IAM-Rollen für Aufgaben verwenden zu können. Weitere Informationen finden Sie unter IAM-Rolle für die Amazon ECS-Aufgabenausführung und IAM-Rollen für Aufgaben. Verwenden Sie den folgenden Richtlinientext.

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.

- Unter finden Tutorial: Erstellen Sie ein Docker-Image und übertragen Sie es auf Amazon ECR mit CodePipeline (Typ V2) Sie ein Tutorial, das Ihnen zeigt, wie Sie die ECRBuildandPublish Aktion verwenden, um ein Image zu übertragen und dann die ECS-Standardaktion für die Bereitstellung auf Amazon ECS zu verwenden.
- Tutorial: Continuous Deployment mit CodePipeline — Dieses Tutorial zeigt Ihnen, wie Sie ein Dockerfile erstellen, das Sie in einem Quelldatei-Repository wie z. B. speichern. 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 anschließend Ihre Pipeline mit einer Amazon ECS-Bereitstellungsaktion.

  > **Note**
  >
  > In diesem Thema und Tutorial wird die Amazon ECS-Standardbereitstellungsaktion für beschrieben CodePipeline. Informationen zu Amazon ECS für CodeDeploy blue/green Bereitstellungsaktionen finden Sie unterTutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-CodeDeploy. CodePipeline

- Amazon Elastic Container Service Developer Guide — Informationen zur Arbeit mit Docker-Images und -Containern, Amazon ECS-Services und -Clustern sowie Amazon ECS-Aufgabensets finden Sie unter Was ist Amazon ECS?

Referenz zur Amazon Elastic Kubernetes Service EKS-Bereitstellungsaktion

Sie können die `EKSDeploy` Aktion verwenden, um einen Amazon EKS-Dienst bereitzustellen. Für die Bereitstellung ist eine Kubernetes-Manifestdatei erforderlich, die zur Bereitstellung des Images CodePipeline verwendet wird.

Bevor Sie Ihre Pipeline erstellen, müssen Sie die Amazon EKS-Ressourcen bereits erstellt und das Image in Ihrem Image-Repository gespeichert haben. Optional können Sie VPC-Informationen für Ihren Cluster angeben.

Important

Diese Aktion verwendet CodePipeline Managed CodeBuild Compute, um Befehle in einer Build-Umgebung auszuführen. Für die Ausführung der Befehlsaktion fallen separate Gebühren in an AWS CodeBuild.

Note

Die Aktion „`EKSBereitstellen`“ ist nur für Pipelines vom Typ V2 verfügbar.

Die EKS-Aktion unterstützt sowohl öffentliche als auch private EKS-Cluster. Private Cluster sind der von EKS empfohlene Typ; beide Typen werden jedoch unterstützt.

Die EKS-Aktion wird für kontoübergreifende Aktionen unterstützt. Um eine kontoübergreifende EKS-Aktion hinzuzufügen, fügen Sie in der Aktionserklärung Daten `actionRoleArn` aus Ihrem Zielkonto hinzu.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Umgebungsvariablen

- Ausgabevariablen
- Berechtigungen für Richtlinien zur Servicerolle
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `EKS`
- Version: `1`

Konfigurationsparameter

ClusterName

Erforderlich: Ja

Der Amazon EKS-Cluster in Amazon EKS.

Optionen unter der Leitung

Die folgenden Optionen sind verfügbar, wenn Helm das ausgewählte Bereitstellungstool ist.

HelmReleaseName

Erforderlich: Ja (nur für den Typ Helm erforderlich)

Der Release-Name für Ihre Bereitstellung.

HelmChartLocation

Erforderlich: Ja (nur für den Typ Helm erforderlich)

Der Standort des Diagramms für Ihren Einsatz.

HelmValuesFiles

Erforderlich: Nein (optional nur für den Typ Helm)

Um Dateien mit Helm-Werten zu überschreiben, geben Sie die kommagetrennten Helm-Wertedateien in die Position des Helm-Diagramms ein.

Optionen unter Kubectl

Die folgenden Optionen sind verfügbar, wenn Kubectl das ausgewählte Bereitstellungstool ist.

ManifestFiles

Erforderlich: Ja (nur für den Typ Kubectl erforderlich)

Der Name Ihrer Manifestdatei, die Textdatei, die den Container-Namen Ihres Dienstes beschreibt, sowie das Bild und das Tag. Sie verwenden diese Datei, um Ihre Bild-URI und andere Informationen zu parametrisieren. Zu diesem Zweck können Sie eine Umgebungsvariable verwenden.

Sie speichern diese Datei im Quell-Repository für Ihre Pipeline.

Namespace

Erforderlich: Nein

Der Kubernetes-Namespace, der in `kubectl` unseren Befehlen verwendet werden soll. `helm`

Subnets

Erforderlich: Nein

Die Subnetze für die VPC für Ihren Cluster. Diese sind Teil derselben VPC, die mit Ihrem Cluster verbunden ist. Sie können auch Subnetze angeben, die noch nicht mit Ihrem Cluster verbunden sind, und sie hier angeben.

SecurityGroupIds

Erforderlich: Nein

Die Sicherheitsgruppen für die VPC für Ihren Cluster. Diese sind Teil derselben VPC, die mit Ihrem Cluster verbunden ist. Sie können auch Sicherheitsgruppen angeben, die noch nicht an Ihren Cluster angehängt sind, und sie hier angeben.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Die Aktion sucht im Quelldatei-Repository für die Pipeline nach der Kubernetes-Manifestdatei oder dem Helm-Diagramm. Wenn Sie Helm-Diagramme im TGZ-Format verwenden

möchten, die in einem S3-Bucket gespeichert sind, können Sie dies tun, indem Sie S3 Bucket/Key als Quellaktion konfigurieren. Der angegebene Objektschlüssel wäre beispielsweise. `my-chart-0.1.0.tgz`

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Umgebungsvariablen

Wird verwendet, um Variablen wie Bild-Repositorys oder Bild-Tags in Manifestdateien oder Helm-Diagrammwertdateien zu ersetzen.

Key (Schlüssel)

Der Schlüssel in einem Schlüssel-Wert-Umgebungsvariablenpaar, z. B. `$IMAGE_TAG`

Wert

Der Wert für das Schlüssel-Wert-Paar, z. B. `v1.0` Der Wert kann mit Ausgabevariablen aus Pipeline-Aktionen oder Pipeline-Variablen parametrisiert werden. Beispielsweise kann eine Pipeline eine ECRBuildAndPublish Aktion haben, die ein ECR-Bild mit erstellt`${codepipeline.PipelineExecutionId}`, und die EKS-Aktion kann dieses Bild `${codepipeline.PipelineExecutionId}` als Wert der Umgebungsvariablen verwenden.

Ausgabevariablen

EKSClusterName

Der Amazon EKS-Cluster in Amazon EKS.

Berechtigungen für Richtlinien zur Servicerolle

Um diese Aktion auszuführen, müssen die folgenden Berechtigungen in der Servicerollenrichtlinie Ihrer Pipeline verfügbar sein.

- EC2-Aktionen: Wenn die Aktion CodePipeline ausgeführt wird, sind EC2-Instance-Berechtigungen erforderlich. Beachten Sie, dass dies nicht mit der EC2-Instanzrolle identisch ist, die beim Erstellen Ihres EKS-Clusters erforderlich ist.

  Wenn Sie eine vorhandene Servicerolle verwenden, müssen Sie die folgenden Berechtigungen für die Servicerolle hinzufügen, um diese Aktion verwenden zu können.

  - ec2: CreateNetworkInterface
  - ec2: DescribeDhcpOptions
  - ec2: DescribeNetworkInterfaces
  - ec2: DeleteNetworkInterface
  - ec2: DescribeSubnets
  - ec2: DescribeSecurityGroups
  - ec2: DescribeVpcs
- EKS-Aktionen: Wenn die Aktion CodePipeline ausgeführt wird, sind EKS-Cluster-Berechtigungen erforderlich. Beachten Sie, dass dies nicht mit der IAM-EKS-Clusterrolle identisch ist, die beim Erstellen Ihres EKS-Clusters erforderlich ist.

  Wenn Sie eine vorhandene Servicerolle verwenden, müssen Sie die folgende Berechtigung für die Servicerolle hinzufügen, um diese Aktion verwenden zu können.

  - fragt: DescribeCluster
- Stream-Aktionen protokollieren: Wenn die Aktion CodePipeline ausgeführt CodePipeline wird, wird eine Protokollgruppe erstellt, die den Namen der Pipeline wie folgt verwendet. Auf diese Weise können Sie die Berechtigungen für die Protokollierung von Ressourcen anhand des Pipeline-Namens einschränken.

  ```
  /aws/codepipeline/MyPipelineName
  ```

  Wenn Sie eine vorhandene Servicerolle verwenden, müssen Sie die folgenden Berechtigungen für die Servicerolle hinzufügen, um diese Aktion verwenden zu können.

  - Protokolle: CreateLogGroup
  - Protokolle: CreateLogStream
  - Protokolle: PutLogEvents

Beschränken Sie in der Erklärung zur Dienstrollenrichtlinie die Berechtigungen auf die Ressourcenebene, wie im folgenden Beispiel gezeigt.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "arn:aws:eks:*:111122223333:cluster/YOUR_CLUSTER_NAME"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:111122223333:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME",
                "arn:aws:logs:*:111122223333:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
            ]
```

```
        }
    ]
}
```

Um Protokolle in der Konsole mithilfe der Dialogseite mit den Aktionsdetails anzuzeigen, muss der Konsolenrolle die Berechtigung zum Anzeigen von Protokollen hinzugefügt werden. Weitere Informationen finden Sie im Beispiel für eine Konsolenberechtigungsrichtlinie unterZum Anzeigen von Rechenprotokollen in der Konsole sind Berechtigungen erforderlich.

Hinzufügen der Servicerolle als Zugriffseintrag für Ihren Cluster

Nachdem die Berechtigungen in der Servicerollenrichtlinie Ihrer Pipeline verfügbar sind, konfigurieren Sie Ihre Cluster-Berechtigungen, indem Sie die CodePipeline Servicerolle als Zugriffseintrag für Ihren Cluster hinzufügen.

Sie können auch eine Aktionsrolle verwenden, die über die aktualisierten Berechtigungen verfügt. Weitere Informationen finden Sie im Tutorial-Beispiel unterSchritt 4: Erstellen Sie einen Zugriffseintrag für die CodePipeline Servicerolle.

Aktionsdeklaration

YAML

```
Name: DeployEKS
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: EKS
  Version: '1'
RunOrder: 2
Configuration:
  ClusterName: my-eks-cluster
  ManifestFiles: ManifestFile.json
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
```

JSON

```
{
```

```
    "Name": "DeployECS",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "EKS",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "ClusterName": "my-eks-cluster",
        "ManifestFiles": "ManifestFile.json"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Unter finden Sie Tutorial: Bereitstellung auf Amazon EKS mit CodePipeline ein Tutorial, das zeigt, wie Sie einen EKS-Cluster und eine Kubernetes-Manifestdatei erstellen, um die Aktion zu Ihrer Pipeline hinzuzufügen.

AWS Lambda Aktionsreferenz bereitstellen

Sie verwenden eine AWS Lambda Bereitstellungsaktion, um die Bereitstellung Ihres Anwendungscodes für Ihre serverlose Bereitstellung zu verwalten. Sie können eine Funktion bereitstellen und Bereitstellungsstrategien für die Bereitstellung des Datenverkehrs wie folgt verwenden:

- Kanarische und lineare Bereitstellungen zur Verkehrsverlagerung
- Bereitstellungen auf einmal

> **Note**
>
> Diese Aktion wird nur für Pipelines vom Typ V2 unterstützt.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Berechtigungen für die Lambda-Bereitstellungsaktion (Richtlinie) für die Aktion „Lambda-Bereitstellung“
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `Lambda`
- Version: `1`

Konfigurationsparameter

FunctionName

Erforderlich: Ja

Der Name der Funktion, die Sie in Lambda erstellt haben, z. B. `MyLambdaFunction`

Sie müssen bereits eine Version erstellt haben.

FunctionAlias

Erforderlich: Nein

Der Alias der Funktion, die Sie in Lambda erstellt haben und für die die Funktion bereitgestellt werden soll, z. B. `live` Der Alias muss existieren und eine Version dahinter haben, wenn die Aktionsausführung beginnt. (Es wird die Rollback-Zielversion sein.)

Wenn nicht angegeben, stellt die Aktion das Quellartefakt bereit `$LATEST` und erstellt eine neue Version. In diesem Anwendungsfall sind die Optionen für die Bereitstellungsstrategie und die Zielversion nicht verfügbar.

PublishedTargetVersion

Erforderlich: Nein

Die gewünschte Lambda-Funktionsversion, für die bereitgestellt werden soll. FunctionAlias Es können Variablen auf Pipeline- oder Aktionsebene sein, z. B. `#{variables.lambdaTargetVersion}` Die Version muss veröffentlicht werden, wenn die Aktionsausführung beginnt.

Erforderlich, wenn kein Eingabeartefakt bereitgestellt wird.

DeployStrategy

Erforderlich: Nein (Standardeinstellung ist`AllAtOnce`)

Legt die Rate fest, mit der die Lambda-Bereitstellungsaktion den Datenverkehr von der Originalversion der Lambda-Funktion auf die neue Version umleitet. FunctionAlias Die verfügbaren Bereitstellungsstrategien sind kanarisch oder linear. Zulässige Formate:

- `AllAtOnce` -

  Leitet den gesamten Datenverkehr auf einmal auf die aktualisierten Lambda-Funktionen um.

  Wenn nichts angegeben ist, ist der Standardwert `AllAtOnce`.

- `Canary10Percent5Minutes`- Verschiebt 10 Prozent des Datenverkehrs in der ersten Stufe. Die restlichen 90 Prozent werden fünf Minuten später bereitgestellt.

  Die Werte sowohl für den Prozentsatz als auch für die Minuten können geändert werden.

- `Linear10PercentEvery1Minute`- Verschiebt jede Minute 10 Prozent des Verkehrs, bis der gesamte Verkehr verlagert ist.

  Die Werte sowohl für den Prozentsatz als auch für die Minuten können geändert werden.

Die folgenden Überlegungen gelten für dieses Feld:

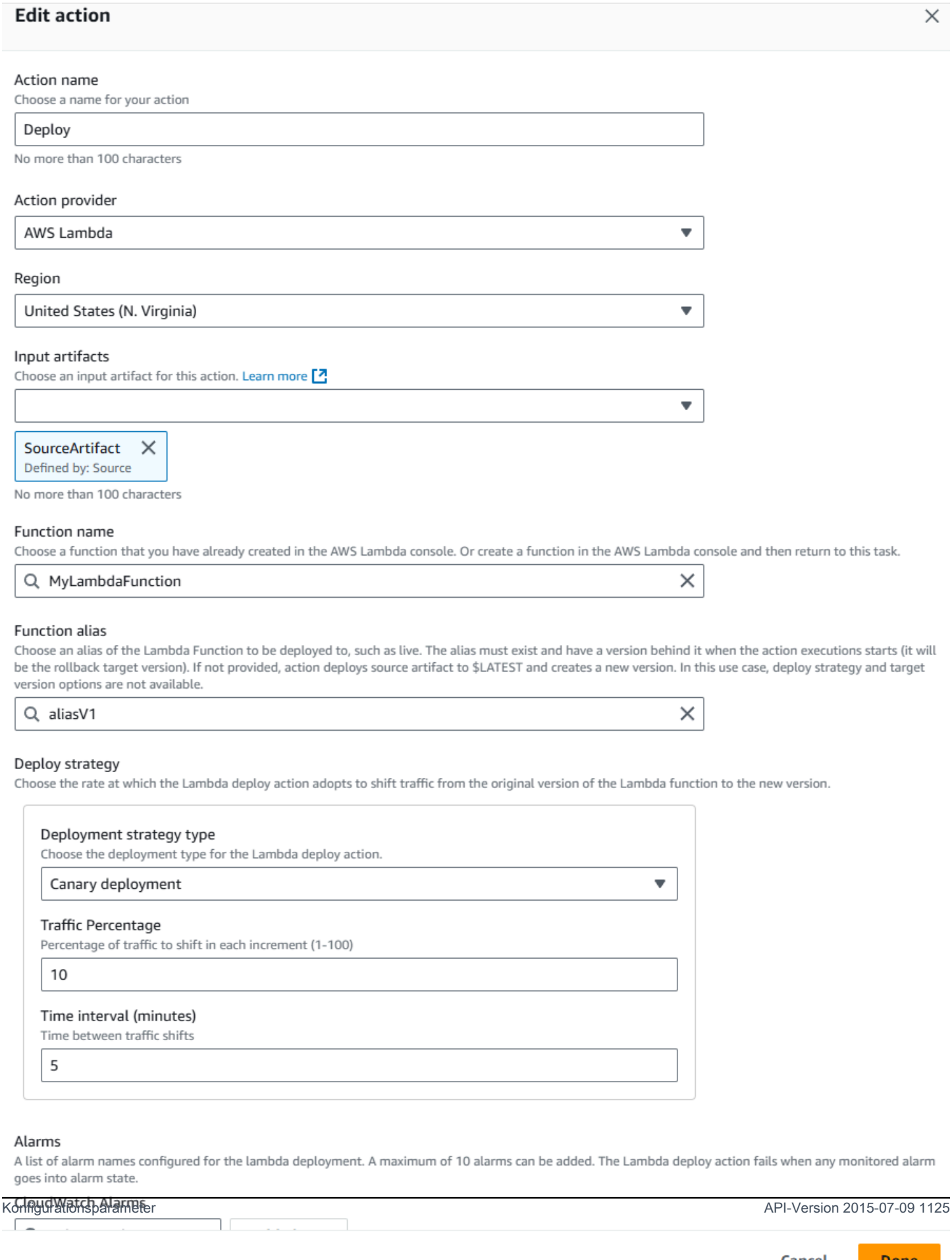
- Die maximale Wartezeit beträgt insgesamt 2 Tage.
- Nur verfügbar, FunctionAlias wenn angegeben.

Alarme

Erforderlich: Nein

Eine durch Kommas getrennte Liste von Alarmnamen, die für die Lambda-Bereitstellung konfiguriert wurden. Es können höchstens 10 Alarme hinzugefügt werden. Die Aktion schlägt fehl, wenn überwachte Alarme in den Status ALARM übergehen.

Die folgende Abbildung zeigt ein Beispiel für die Seite „Bearbeiten“ für die Aktion.


Edit action
Action name
Choose a name for your action
Deploy
No more than 100 characters
Action provider
AWS Lambda
Region
United States (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
Defined by: Source
No more than 100 characters
Function name
Choose a function that you have already created in the AWS Lambda console. Or create a function in the AWS Lambda console and then return to this task.
MyLambdaFunction
Function alias
Choose an alias of the Lambda Function to be deployed to, such as live. The alias must exist and have a version behind it when the action executions starts (it will be the rollback target version). If not provided, action deploys source artifact to $LATEST and creates a new version. In this use case, deploy strategy and target version options are not available.
aliasV1
Deploy strategy
Choose the rate at which the Lambda deploy action adopts to shift traffic from the original version of the Lambda function to the new version.
Deployment strategy type
Choose the deployment type for the Lambda deploy action.
Canary deployment
Traffic Percentage
Percentage of traffic to shift in each increment (1-100)
10
Time interval (minutes)
Time between traffic shifts
5
Alarms
A list of alarm names configured for the lambda deployment. A maximum of 10 alarms can be added. The Lambda deploy action fails when any monitored alarm goes into alarm state.
CloudWatch Alarms
Cancel
Done

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Die bereitgestellten Dateien, falls vorhanden, zur Unterstützung der Skriptaktionen während der Bereitstellung.

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. 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 Variablenreferenz.

FunctionVersion

Die neue Lambda-Funktionsversion, die bereitgestellt wurde.

Berechtigungen für die Lambda-Bereitstellungsaktion (Richtlinie) für die Aktion „Lambda-Bereitstellung“

Wenn die Aktion CodePipeline ausgeführt wird, benötigt die CodePipeline Servicerolle die folgenden Berechtigungen, die entsprechend auf den Zugriff mit den geringsten Rechten beschränkt sind.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "StatementForLambda",
            "Effect": "Allow",
            "Action": [
                "lambda:GetAlias",
                "lambda:GetFunctionConfiguration",
                "lambda:GetProvisionedConcurrencyConfig",
                "lambda:PublishVersion",
                "lambda:UpdateAlias",
                "lambda:UpdateFunctionCode"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:111122223333:function:
{{FunctionName}}",
                "arn:aws:lambda:us-east-1:111122223333:function:
{{FunctionName}}:*"
            ]
        },
        {
            "Sid": "StatementForCloudWatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": [
                "arn:aws:cloudwatch:us-east-1:111122223333:alarm:{{AlarmNames}}"
            ]
        },
        {
            "Sid": "StatementForLogs1",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:/us-east-1/
codepipeline/{{pipelineName}}",
                "arn:aws:logs:us-east-1:111122223333:log-group:/us-east-1/
codepipeline/{{pipelineName}}:*"
            ]
        },
        {
            "Sid": "StatementForLogs2",
            "Effect": "Allow",
            "Action": [
```

```
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:/us-east-1/
codepipeline/{{pipelineName}}:log-stream:*"
            ]
        }
    ]
}
```

Aktionsdeklaration

YAML

```
name: Deploy
actionTypeId:
  category: Deploy
  owner: AWS
  provider: Lambda
  version: '1'
runOrder: 1
configuration:
  DeployStrategy: Canary10Percent5Minutes
  FunctionAlias: aliasV1
  FunctionName: MyLambdaFunction
outputArtifacts: []
inputArtifacts:
- name: SourceArtifact
region: us-east-1
namespace: DeployVariables
```

JSON

```
{
    "name": "Deploy",
    "actionTypeId": {
        "category": "Deploy",
        "owner": "AWS",
        "provider": "Lambda",
        "version": "1"
    },
```

```
    "runOrder": 1,
    "configuration": {
        "DeployStrategy": "Canary10Percent5Minutes",
        "FunctionAlias": "aliasV1",
        "FunctionName": "MyLambdaFunction"
    },
    "outputArtifacts": [],
    "inputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "region": "us-east-1",
    "namespace": "DeployVariables"
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Bereitstellung von Lambda-Funktionen mit CodePipeline— Dieses Tutorial führt Sie durch die Erstellung einer Lambda-Beispielfunktion, in der Sie einen Alias und eine Version erstellen, die komprimierte Lambda-Funktion zu Ihrem Quellspeicherort hinzufügen und die Lambda-Aktion in Ihrer Pipeline ausführen.

Referenz zur Amazon S3-Bereitstellungsaktion

Sie verwenden eine Amazon S3-Bereitstellungsaktion, um Dateien für das Hosting oder die Archivierung statischer Websites in einem Amazon S3-Bucket bereitzustellen. Sie können angeben, ob die Bereitstellungsdateien vor dem Hochladen in Ihren Bucket extrahiert werden sollen.

> **Note**
>
> Dieses Referenzthema beschreibt die Amazon S3-Bereitstellungsaktion für den Fall, dass es CodePipeline sich bei der Bereitstellungsplattform um einen Amazon S3-Bucket handelt, der für das Hosting konfiguriert ist. Referenzinformationen zur Amazon S3-Quellaktion in CodePipeline finden Sie unterReferenz zur Amazon S3-Quellaktion.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Berechtigungen für Servicerollen: Aktion „S3-Bereitstellung“
- Beispielaktionskonfiguration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `S3`
- Version: `1`

Konfigurationsparameter

BucketName

Erforderlich: Ja

Der Name des Amazon S3-Buckets, in dem Dateien bereitgestellt werden sollen.

Extrahieren

Erforderlich: Ja

Wenn wahr, gibt dies an, dass Dateien vor dem Hochladen extrahiert werden sollen. Andernfalls bleiben die Anwendungsdateien für den Upload gezippt, z. B. im Fall einer gehosteten statischen Website. Falls falsch, `ObjectKey` ist das erforderlich.

ObjectKey

Bedingt. Erforderlich, wenn `Extract` = false

Der Name des Amazon S3-Objektschlüssels, der das Objekt im S3-Bucket eindeutig identifiziert.

KMSEncryptionKeyARN

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. Bei kontoübergreifenden Aktionen können Sie nur die Schlüssel-ID oder den Schlüssel-ARN verwenden, um den Schlüssel zu identifizieren. Cross-account Bei Aktionen wird die Rolle des anderen Kontos (AccountB) verwendet. Wenn Sie also die Schlüssel-ID angeben, wird der Schlüssel des anderen Kontos (accounB) verwendet.

> **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 vordefinierte ACL https://docs.aws.amazon.com/AmazonS3/latest/userguide/acl-overview.html#canned-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 requests/responses 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.

Berechtigungen für Servicerollen: Aktion „S3-Bereitstellung“

Für die Unterstützung von S3-Bereitstellungsaktionen fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl",
        "s3:GetBucketVersioning",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::[[s3DeployBuckets]]",
        "arn:aws:s3:::[[s3DeployBuckets]]/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

Für die Unterstützung von S3-Bereitstellungsaktionen müssen Sie Ihrer Richtlinienerklärung außerdem die folgenden Berechtigungen hinzufügen, wenn Ihre S3-Objekte über Tags verfügen:

```
"s3:GetObjectTagging",
"s3:GetObjectVersionTagging",
"s3:PutObjectTagging"
```

Beispielaktionskonfiguration

Im Folgenden finden Sie Beispiele für die Aktionskonfiguration.

Beispielkonfiguration, wenn **`Extract auf false gesetzt ist`**

Das folgende Beispiel zeigt die Standardaktionskonfiguration, wenn die Aktion mit dem Wert `Extract` auf erstellt wird`false`.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: S3
      Version: '1'
    RunOrder: 1
    Configuration:
      BucketName: website-bucket
```

```
      Extract: 'false'
      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": "false",
                "ObjectKey": "MyWebsite"
                },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

Beispielkonfiguration, wenn **`Extract auf true gesetzt ist`**

Das folgende Beispiel zeigt die Standardaktionskonfiguration, wenn die Aktion mit dem Wert `Extract` auf erstellt wird`true`.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: S3
      Version: '1'
    RunOrder: 1
    Configuration:
      BucketName: website-bucket
      Extract: 'true'
    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"
                },
```

```
            "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-Bereitstellungskonfiguration anzugeben.
- Referenz zur Amazon S3-Quellaktion— Diese Aktionsreferenz enthält Referenzinformationen und Beispiele für Amazon S3-Quellaktionen in CodePipeline.

Referenz zur Amazon S3-Quellaktion

Eine Quellaktion löst die Pipeline aus, wenn ein neues Objekt in den konfigurierten Bucket und Objektschlüssel hochgeladen wird.

In diesem Referenzthema wird die Amazon S3-Quellaktion beschrieben, bei CodePipeline der der Quellspeicherort ein Amazon S3-Bucket ist, der für die Versionierung konfiguriert ist. Referenzinformationen zur Amazon S3-Bereitstellungsaktion in finden Sie CodePipeline unterReferenz zur Amazon S3-Bereitstellungsaktion.

Bevor Sie die Pipeline verbinden, erstellen Sie einen Amazon S3-Quell-Bucket, aktivieren Sie die Versionierung und laden Sie Ihre Quelldateien als einzelne ZIP-Datei hoch.

Sie können einen neuen Bucket in Amazon S3 erstellen oder einen vorhandenen Amazon S3-Bucket auswählen, der als Quellspeicherort für Ihre Anwendungsdateien verwendet werden soll. Weitere Informationen zum Aktivieren der Versionierung finden Sie unter Verwenden der Versionierung im Amazon S3-Benutzerhandbuch.

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 entpackte Datei hochladen. Downstream-Aktionen, die eine ZIP-Datei erwarten, schlagen jedoch fehl.

Wenn Sie die Konsole verwenden, um Ihre Pipeline zu erstellen oder zu bearbeiten, CodePipeline erstellt eine EventBridge Regel, die Ihre Pipeline startet, wenn eine Änderung im Amazon S3-Quell-Bucket erfolgt.

Sie können auch eine Quellüberschreibung mithilfe eines Eingabe-Transformationseintrags erstellen, um das `revisionValue` In EventBridge für Ihr Pipeline-Ereignis zu verwenden, wobei `revisionValue` das von der Quellereignisvariablen für Ihren Objektschlüssel, Ihren Commit oder Ihre Bild-ID abgeleitet wird. Weitere Informationen finden Sie in dem optionalen Schritt für die Eingabe der Eingabetransformation, der in den nachfolgenden Verfahren enthalten istHerstellen einer Verbindung zu Amazon S3-Quellaktionen mit einer für Ereignisse aktivierten Quelle.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Berechtigungen für Servicerollen: S3-Quellaktion
- Aktionsdeklaration
- 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-Buckets, in dem Quelländerungen erkannt werden sollen.

S3ObjectKey

Erforderlich: Ja

Der Name des Amazon S3-Objektschlüssels, in dem Quelländerungen erkannt werden sollen.

AllowOverrideForS3ObjectKey

Erforderlich: Nein

`AllowOverrideForS3ObjectKey`steuert, ob Quellüberschreibungen von die bereits `S3ObjectKey` in der Quelle konfigurierte Aktion überschreiben `StartPipelineExecution` können. Weitere Informationen zu Quellüberschreibungen mit dem S3-Objektschlüssel finden Sie unter. Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung

> **Important**
>
> Wenn Sie diesen Parameter weglassen`AllowOverrideForS3ObjectKey`, können Sie den S3 ObjectKey in der Quellaktion CodePipeline standardmäßig 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 Quellrevisionsüberschreibungen außer Kraft gesetzt werden.

  > **Note**
  >
  > Wenn Sie beabsichtigen, allen CodePipeline Benutzern die Möglichkeit zu geben, den vorkonfigurierten S3-Objektschlüssel zu überschreiben, wenn Sie eine neue Pipeline-Ausführung starten, müssen Sie auf setzen. `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-Quell-Bucket nach Quelländerungen CodePipeline abgefragt wird. Wir empfehlen, stattdessen CloudWatch Events und CloudTrail to detect source changes zu verwenden. Weitere Informationen zur Konfiguration von CloudWatch Ereignissen finden Sie unter Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (CLI) oderMigrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail (CloudFormation Vorlage).

> **Important**
>
> Wenn Sie CloudWatch Ereignisse konfigurieren möchten, müssen Sie auf setzen, `PollForSourceChanges` `false` um doppelte Pipelineausführungen zu vermeiden.

Gültige Werte für diesen Parameter sind:

- `true`: Wenn diese Option gesetzt ist, wird Ihr Quellstandort nach Quellenänderungen CodePipeline abgefragt.

  > **Note**
  >
  > Wenn Sie es weglassen`PollForSourceChanges`, CodePipeline wird standardmäßig Ihr Quellstandort nach Quellenänderungen abgefragt. Dieses Verhalten ist das gleiche, als ob `PollForSourceChanges` enthalten wäre und auf `true` festgelegt würde.

- `false`: Wenn diese Option gesetzt ist, CodePipeline wird Ihr Quellstandort nicht nach Quellenänderungen abgefragt. Verwenden Sie diese Einstellung, wenn Sie beabsichtigen, eine CloudWatch Event-Regel zu konfigurieren, um Änderungen an der Quelle zu erkennen.

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-Aktion. Die Amazon 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 unterVariablenreferenz.

BucketName

Der Name des Amazon S3-Buckets bezieht sich auf die Quelländerung, 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-Objektschlüssels, der sich auf die Quelländerung 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.

Berechtigungen für Servicerollen: S3-Quellaktion

Für die Unterstützung von S3-Quellaktionen fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketVersioning",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::[[S3Bucket]]",
        "arn:aws:s3:::[[S3Bucket]]/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

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: amzn-s3-demo-source-bucket
      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": "amzn-s3-demo-source-bucket",
                "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 eine AppSpec Beispieldatei sowie eine CodeDeploy Beispielanwendung und eine Bereitstellungsgruppe. Verwenden Sie dieses Tutorial, um eine Pipeline mit einer Amazon S3-Quelle zu erstellen, die auf Amazon EC2-Instances bereitgestellt wird.

AWS AppConfig Aktionsreferenz bereitstellen

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 AWS AppConfig Bereitstellungsstrategie, die für die Bereitstellung verwendet werden soll.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Das Eingabeartefakt für die Bereitstellungsaktion.

Ausgabeartefakte

Nicht zutreffend.

Berechtigungen für Servicerollen: Aktion **AppConfig**

Wenn die Aktion CodePipeline ausgeführt wird, erfordert die CodePipeline Dienstrollenrichtlinie die folgenden Berechtigungen, die entsprechend auf die Ressourcenebene beschränkt sind, um den Zugriff mit den geringsten Rechten aufrechtzuerhalten.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "appconfig:StartDeployment",
                "appconfig:StopDeployment",
                "appconfig:GetDeployment"
            ],
            "Resource": [
                "arn:aws:appconfig:*:111122223333:application/[[Application]]",
                "arn:aws:appconfig:*:111122223333:application/[[Application]]/*",
                "arn:aws:appconfig:*:111122223333:deploymentstrategy/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

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.

CloudFormation Aktionsreferenz bereitstellen

Führt eine Operation auf einem 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 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 CloudFormation Aktionen, die mit Stacks und Änderungssätzen ausgeführt werden können, finden Sie unter dem `ActionMode` Parameter.

CodePipeline Ruft die CloudFormation `DescribeStackEvents` API auf, um eine Fehlermeldung für eine 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
- Berechtigungen für Servicerollen: CloudFormation Aktion
- 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 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. 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 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 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 CloudFormation gelöscht und anschließend ein neuer Stapel erstellt. Wenn sich der Stack nicht in einem ausgefallenen Zustand befindet, wird er CloudFormation aktualisiert.

  Der Stack befindet sich in einem fehlerhaften Zustand, wenn einer der folgenden Statustypen in 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.

Capabilities

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 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 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**
>
> 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 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 angezeigt wird. 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 CloudFormation Vorlagendatei, die den Vorlagentext enthält, hat eine Mindestlänge von 1 Byte und eine maximale Länge von 1 MB. Für CloudFormation Bereitstellungsaktionen in CodePipeline beträgt die maximale Größe des Eingabeartefakts immer 256 MB. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline und 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 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 angezeigt wird. 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 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 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 CloudFormation -Stack.

  Weitere Informationen zum Ausgabeabschnitt, den Sie für Ihre 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 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 Variablenreferenz. Ein Tutorial, das Ihnen zeigt, wie Sie eine Pipeline mit einer CloudFormation Bereitstellungsaktion in einer Pipeline erstellen, die CloudFormation Ausgabevariablen verwendet, finden Sie unterTutorial: Erstellen Sie eine Pipeline, die Variablen von verwendet AWS CloudFormation Aktionen zur Bereitstellung.

Berechtigungen für Servicerollen: CloudFormation Aktion

Wenn die Aktion CodePipeline ausgeführt wird, erfordert die CodePipeline Dienstrollenrichtlinie die folgenden Berechtigungen, die entsprechend auf den ARN der Pipeline-Ressource beschränkt sind, um den Zugriff mit den geringsten Rechten aufrechtzuerhalten. Fügen Sie Ihrer Richtlinienerklärung beispielsweise Folgendes hinzu:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCFNStackAccess",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStacks",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStackEvents",
                "cloudformation:GetTemplate",
                "cloudformation:DescribeChangeSet",
                "cloudformation:CreateChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:ExecuteChangeSet"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:111122223333:stack/
[[cfnDeployStackNames]]/*"
            ]
        },
        {
            "Sid": "ValidateTemplate",
            "Effect": "Allow",
            "Action": [
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIAMPassRole",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/[[cfnExecutionRoles]]"
            ],
```

```
            "Condition": {
                "StringEqualsIfExists": {
                    "iam:PassedToService": [
                        "cloudformation.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Beachten Sie, dass die `cloudformation:DescribeStackEvents` Genehmigung optional ist. Dadurch kann die 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.

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 Parameter. CodePipeline
- AWS CloudFormation API-Referenz — Der CreateStackParameter in der AWS CloudFormation API-Referenz beschreibt Stack-Parameter für CloudFormation Vorlagen.

CloudFormation StackSets Aktionsreferenz bereitstellen

CodePipeline bietet die Möglichkeit, CloudFormation StackSets Operationen als Teil Ihres CI/CD Prozesses auszuführen. Sie verwenden ein Stack-Set, um mithilfe einer einzigen CloudFormation Vorlage Stacks in AWS Konten in verschiedenen AWS Regionen zu erstellen. Alle in jedem Stack enthaltenen Ressourcen werden durch die 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 CloudFormation StackSets Konzepten für finden Sie unter StackSets Konzepte im AWS CloudFormation Benutzerhandbuch.

Sie integrieren Ihre Pipeline CloudFormation StackSets über zwei verschiedene 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 an bestimmten Instances. In der Konsole können Sie den CloudFormation Stack Set-Aktionsprovider auswählen, wenn Sie Ihre Pipeline erstellen oder bearbeiten.
- Die `CloudFormationStackInstances` Aktion verteilt Änderungen aus der `CloudFormationStackSet` Aktion auf bestimmte Instanzen, erstellt neue Stack-Instances und definiert Parameterüberschreibungen für angegebene Instanzen. In der Konsole können Sie den Aktionsprovider CloudFormation Stack Instances auswählen, wenn Sie eine vorhandene Pipeline bearbeiten.

Sie können diese Aktionen verwenden, um sie für AWS Zielkonten oder Organisationseinheiten-IDs von AWS Zielorganisationen bereitzustellen.

> **Note**
>
> Um Konten oder Organisationseinheiten-IDs für AWS Zielorganisationen bereitzustellen und das dienstverwaltete Berechtigungsmodell zu verwenden, müssen Sie den vertrauenswürdigen Zugriff zwischen AWS CloudFormation StackSets und AWS Organisationen aktivieren. Weitere Informationen findest du unter Vertrauenswürdigen Zugriff mit AWS CloudFormation Stacksets aktivieren.

Themen

- Wie CloudFormation StackSets Aktionen funktionieren
- Wie strukturiert man StackSets Aktionen in einer Pipeline
- CloudFormationStackSet-Aktion
- Die Aktion CloudFormationStackInstances
- Berechtigungen für Servicerollen: Aktion CloudFormationStackSet
- Berechtigungen für Servicerollen: CloudFormationStackInstances Aktion
- Berechtigungsmodelle für Stack-Set-Operationen
- Datentypen für Vorlagenparameter
- Weitere Informationen finden Sie auch unter

Wie CloudFormation StackSets Aktionen funktionieren

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 verteilt diese Änderungen auf angegebene Instanzen.

> **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 CloudFormationStackSet Aktion das Stack-Set, ohne dass Instanzen erstellt werden.
- Aktualisieren: Wenn die CloudFormationStackSet Aktion 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, erhalten alle verbleibenden Instanzen den Status VERALTET.

Sie können die CloudFormationStackSet Aktion verwenden, um das Stack-Set auf folgende Weise zu aktualisieren.

- Aktualisieren Sie die Vorlage auf einigen oder allen Instanzen.
- Aktualisieren Sie die Parameter auf einigen oder allen Instanzen.
- Aktualisieren Sie die Ausführungsrolle für das Stack-Set (diese muss der in der Administratorrolle angegebenen Ausführungsrolle entsprechen).
- Ändern Sie das Berechtigungsmodell (nur, wenn keine Instanzen erstellt wurden).
- Enable/Disable `AutoDeployment`wenn das `Service Managed` Stack-Set-Berechtigungsmodell
- Handeln Sie als delegierter Administrator in einem Mitgliedskonto, wenn das Stack-Set-Berechtigungsmodell verwendet wird`Service Managed`.
- Aktualisieren Sie die Administratorrolle.
- Aktualisieren Sie die Beschreibung im Stack-Set.
- Fügen Sie dem Stack-Set-Update Deployment-Ziele 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 darin enthaltenen Instanzen 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 der Ausführung der `CloudFormationStackSet` Aktion nur in einigen Instanzen aktualisiert wurden, wird der Rest markiert`OUTDATED`. In späteren Phasen der Pipeline werden die restlichen Instanzen im Stack in Wellen `CloudFormationStackInstances` aktualisiert, 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 gesetzt`OUTDATED`.

Während einer Bereitstellung können Stack-Instances auch einen Status anzeigen, der angibt, `OUTDATED` ob die Bereitstellung auf 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 in einer Teilmenge oder einer einzelnen Instanz bereitgestellt wird. Nachdem Sie Ihre Bereitstellung getestet und das generierte Stack-Set angesehen haben, fügen Sie dann 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 überprüfen Sie die Bereitstellung auf Ihrer ersten Instanz. Bei der ersten Stack-Set-Erstellung für ein 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 `CloudFormationStackInstances` als zweite Bereitstellungsaktion create/update Stack-Instances für die von Ihnen angegebenen Ziele hinzuzufügen. Bei der Erstellung von Stack-Instances für ein Konto, `Account-A` für das die `us-east-2` und `eu-central-1` Regions angegeben sind, werden beispielsweise die verbleibenden Stack-Instances erstellt und die ursprüngliche Instance wird 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 die Aktualisierung vorgesehen waren, aus der Bereitstellung entfernt und erhalten den Status VERALTET. Beispiel: Für ein Stack-Instance-Update für ein Konto, `Account-A` bei dem die `us-east-2` Region aus der Aktionskonfiguration entfernt wird, werden die verbleibenden Stack-Instances erstellt und die entfernte Instanz wird wie folgt auf OUTDATED 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 StackSets im AWS CloudFormation Benutzerhandbuch unter Bewährte Methoden für.

`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, in denen Operationen ausgeführt werden sollen, den Prozentsatz der Fehlertoleranz, bei dessen Überschreitung Stack-Operationen beendet werden, und die Anzahl der Konten, in denen Operationen auf 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 ohne eine Bereitstellung auf mindestens einer Stack-Instance nicht zulässig.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Beispiel CloudFormationStackSet Aktionskonfiguration

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 128 Zeichen oder weniger lang sein.

Description

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 in dem Format ein`"InputArtifactName::TemplateFileName"`, wie im folgenden Beispiel gezeigt.

```
SourceArtifact::template.txt
```

Parameters

Erforderlich: Nein

Eine Liste von Vorlagenparametern für Ihr Stack-Set, die während einer Bereitstellung aktualisiert werden.

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 Hinweise zu diesen Datentypen finden Sie unter. Datentypen für Vorlagenparameter

  Das folgende Beispiel zeigt einen Parameter, der `BucketName` mit dem Wert benannt ist`amzn-s3-demo-source-bucket`.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ```

  Das folgende Beispiel zeigt einen Eintrag mit mehreren Parametern:

```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ParameterKey=Asset1,ParameterValue=true
  ParameterKey=Asset2,ParameterValue=true
```

- Sie können den Speicherort der Datei eingeben, die eine Liste von Vorlagenparameterüberschreibungen enthält, die im Format eingegeben wurden`"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"
      }
  ]
  ```

Capabilities

Erforderlich: Nein

Gibt an, dass mit der Vorlage Ressourcen erstellt und aktualisiert werden können, abhängig von den Ressourcentypen in der Vorlage.

Sie müssen diese Eigenschaft verwenden, wenn Ihre Stack-Vorlage IAM-Ressourcen enthält oder Sie einen Stack direkt aus einer Vorlage erstellen, die Makros enthält. Damit die CloudFormation Aktion auf diese Weise erfolgreich ausgeführt werden kann, müssen Sie eine der folgenden Funktionen verwenden:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`

Sie können mehrere Funktionen angeben, indem Sie ein Komma und keine Leerzeichen zwischen den Funktionen verwenden. Das Beispiel in Beispiel CloudFormationStackSet 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 das Standardfeld verwendet. Weitere Informationen finden Sie unter Berechtigungsmodelle für Stack-Set-Operationen.

Folgende sind gültige Werte:

- `SELF_MANAGED`(Standard): Sie müssen Administrator- und Ausführungsrollen für die Bereitstellung auf Zielkonten erstellen.
- `SERVICE_MANAGED`: erstellt CloudFormation StackSets automatisch die IAM-Rollen, die für die Bereitstellung auf Konten erforderlich sind, die von AWS Organisationen verwaltet werden. Dazu muss ein Konto Mitglied einer Organisation sein.

> **Note**
>
> Dieser Parameter kann nur geändert werden, wenn im Stack-Set keine Stack-Instances vorhanden sind.

AdministrationRoleArn

> **Note**
>
> Da 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 für das SELF_MANAGED-Berechtigungsmodell optional 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 sowie 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, wird er auf gesetzt AWSCloudFormationStackSetAdministrationRole. Wenn Sie einen Rollennamen angeben ServiceManaged, dürfen Sie keinen Rollennamen definieren.

ExecutionRoleName

> **Note**
>
> Da 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 für das SELF_MANAGED-Berechtigungsmodell optional 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 werden. Der Name kann alphanumerische Zeichen sowie 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 Höchstlänge 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, wird er auf gesetzt`AWSCloudFormationStackSetExecutionRole`. Wenn Sie Service_Managed den Wert auf true setzen, dürfen Sie keinen Rollennamen definieren.

OrganizationsAutoDeployment

Erforderlich: Nein

> **Note**
>
> Dieser Parameter ist für das SERVICE_MANAGED-Berechtigungsmodell optional und wird nicht für das SELF_MANAGED-Berechtigungsmodell verwendet.

Beschreibt, ob Konten, die einer Zielorganisation oder Organisationseinheit (OU) hinzugefügt wurden, CloudFormation StackSets automatisch für AWS Organisationen bereitgestellt werden. Wenn `OrganizationsAutoDeployment` angegeben, geben Sie nicht `DeploymentTargets` und `Regions` an.

> **Note**
>
> Wenn keine Eingabe vorgesehen ist`OrganizationsAutoDeployment`, lautet der Standardwert`Disabled`.

Folgende sind gültige Werte:

- `Enabled`. Erforderlich: Nein.

  StackSets stellt automatisch zusätzliche Stack-Instances für AWS Organisationskonten bereit, die einer Zielorganisation oder Organisationseinheit (OU) in den angegebenen Regionen hinzugefügt werden. Wenn ein Konto aus einer Zielorganisation oder OU entfernt wird, werden Stack-Instances aus dem Konto in den angegebenen Regionen 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 OU entfernt wird.

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, in denen sich Stack-Set-Instanzen befinden sollten created/updated.

- Konten:

  Sie können Konten als Literalliste 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, in denen sich Stack-Set-Instanzen befinden sollen created/updated, eingegeben im Format. `InputArtifactName::AccountsFileName` Wenn Sie den Dateipfad verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat im JSON-Format vorliegen, 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 `accounts.txt` Fall, dass mehrere Konten aufgelistet werden:

```
[
    "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 literale 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 Stack-Set-Instanzen erstellt oder aktualisiert werden sollen. Wenn Sie den Dateipfad verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat im JSON-Format vorliegen, 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 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 Stack-Set-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, für die dieser Stack-Vorgang fehlschlagen kann, bevor der Vorgang in dieser Region CloudFormation beendet wird. Wenn der Vorgang in einer Region gestoppt wird, wird der Vorgang in nachfolgenden Regionen CloudFormation nicht versucht. Bei der Berechnung der Anzahl der Konten auf der Grundlage des angegebenen Prozentsatzes wird auf die nächste ganze Zahl 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 CloudFormation abgerundet. Wenn das Abrunden zu Null führen würde, wird die Zahl stattdessen auf Eins CloudFormation festgelegt. Sie verwenden diese Einstellung zwar, um den Höchstwert anzugeben, aber bei großen

Bereitstellungen kann die tatsächliche Anzahl der Konten, auf die gleichzeitig zugegriffen wird, aufgrund der Diensteinschränkung niedriger sein.

RegionConcurrencyType

Erforderlich: Nein

Sie können angeben, ob das Stack-Set AWS-Regionen sequentiell oder parallel verteilt werden soll, indem Sie den Bereitstellungsparameter Region Concurrency konfigurieren. Wenn die Option „Regionale Parallelität“ so angegeben ist, dass Stapel auf mehreren AWS-Regionen parallel verteilt werden, kann dies insgesamt zu kürzeren Bereitstellungszeiten führen.

- Parallel: Stack-Set-Bereitstellungen werden gleichzeitig durchgeführt, sofern die Bereitstellungsfehler einer Region eine bestimmte Ausfalltoleranz nicht überschreiten.
- Sequentiell: Stack-Set-Bereitstellungen werden nacheinander durchgeführt, sofern die Bereitstellungsfehler einer Region eine bestimmte Ausfalltoleranz nicht überschreiten. Die sequenzielle Bereitstellung ist die Standardauswahl.

ConcurrencyMode

Erforderlich: Nein

Im Parallelitätsmodus können Sie wählen, wie sich die Parallelitätsstufe bei Stack-Set-Vorgängen verhält, ob mit strikter oder weicher Ausfalltoleranz. Strikte Fehlertoleranz senkt die Bereitstellungsgeschwindigkeit bei fehlerhaften Stack-Set-Vorgängen, da die Parallelität bei jedem Fehler abnimmt. Bei Soft Failure Tolerance wird der Geschwindigkeit der Bereitstellung Priorität eingeräumt und gleichzeitig die Sicherheitsfunktionen genutzt. CloudFormation

- `STRICT_FAILURE_TOLERANCE`: Diese Option senkt dynamisch den Grad der Parallelität, um sicherzustellen, dass die Anzahl der ausgefallenen Konten niemals eine bestimmte Ausfalltoleranz überschreitet. Dies ist das Standardverhalten.
- `SOFT_FAILURE_TOLERANCE`: Diese Option entkoppelt die Ausfalltoleranz von der tatsächlichen Parallelität. Dadurch können Stack-Set-Operationen unabhängig von der Anzahl der Fehler auf einer bestimmten Parallelitätsstufe 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 festgelegt ist, stellen Sie sicher`DELEGATED_ADMIN`, dass die Pipeline-IAM-Rolle über die entsprechende Berechtigung verfügt`organizations:ListDelegatedAdministrators`. Andernfalls schlägt die Aktion bei der Ausführung fehl und es wird ein Fehler ähnlich dem folgenden angezeigt:`Account used is not a delegated administrator`.

- `SELF`: Bei der Stack-Set-Bereitstellung werden vom Dienst verwaltete Berechtigungen verwendet, wenn Sie beim Verwaltungskonto angemeldet sind.
- `DELEGATED_ADMIN`: Bei der Stack-Set-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 mehr Eingabeartefakte für Listen von Bereitstellungszielen, Konten und Parametern hinzufügen.

- Anzahl der Artefakte: `1 to 3`
- Beschreibung: Du kannst 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 Variablenreferenz.

Beispiel CloudFormationStackSet Aktionskonfiguration

Die folgenden Beispiele zeigen die Aktionskonfiguration für die CloudFormationStackSet Aktion.

Beispiel für das selbstverwaltete Berechtigungsmodell

Das folgende Beispiel zeigt eine CloudFormationStackSet Aktion, 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 dienstverwaltete Berechtigungsmodell

Das folgende Beispiel zeigt eine CloudFormationStackSet Aktion für das dienstverwaltete Berechtigungsmodell, bei dem die Option für die automatische Bereitstellung für AWS Organisationen 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"
}
```

Die Aktion CloudFormationStackInstances

Diese Aktion erstellt neue Instanzen und stellt Stack-Sets für angegebene 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 beispielsweise die Stack-Erstellung 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 Sie Folgendes verwenden`CloudFormationStackInstances`. Die Werte der Vorlagenparameter können auf der Ebene der Stack-Instance bei Vorgängen zum Erstellen oder Aktualisieren einer Stack-Set-Instanz 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 so lange gesetzt, `OUTDATED` bis die Änderungen für diese Instanz übernommen 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
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)

- Ausgabeartefakte
- Ausgabevariablen
- Beispielaktionskonfiguration

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 128 Zeichen oder weniger 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, in denen sich Stack-Set-Instanzen befinden sollten created/updated.

- Konten:

  Sie können Konten als Literalliste 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, in denen sich Stack-Set-Instanzen befinden sollen created/updated, eingegeben im Format. `InputArtifactName::AccountsFileName` Wenn Sie den Dateipfad verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat im JSON-Format vorliegen, 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 `accounts.txt` Fall, dass mehrere Konten aufgelistet werden:

    ```
    [
        "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 literale 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 Stack-Set-Instanzen erstellt oder aktualisiert werden sollen. Wenn Sie den Dateipfad verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat im JSON-Format vorliegen, 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: Ja

> **Note**
>
> Wenn dieser Parameter ausgewählt ist, müssen Sie auch auswählen DeploymentTargets.

Eine Liste von AWS Regionen, in denen Stack-Set-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 Hinweise zu diesen Datentypen finden Sie unter. Datentypen für Vorlagenparameter

  Das folgende Beispiel zeigt einen Parameter, der `BucketName` mit dem Wert benannt ist`amzn-s3-demo-source-bucket`.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ```

  Das folgende Beispiel zeigt einen Eintrag mit mehreren Parametern.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ParameterKey=Asset1,ParameterValue=true
  ParameterKey=Asset2,ParameterValue=true
  ```

- Sie können den Speicherort der Datei eingeben, die eine Liste von Vorlagenparameterüberschreibungen enthält, die im Format eingegeben wurden`InputArtifactName::ParameterOverridesFileName`, 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 CloudFormation beendet wird. Wenn der Vorgang in einer Region gestoppt wird, wird der Vorgang in nachfolgenden Regionen CloudFormation nicht versucht. Bei der Berechnung der Anzahl der Konten auf der Grundlage des angegebenen Prozentsatzes wird auf die nächste ganze Zahl CloudFormation abgerundet.

MaxConcurrentPercentage

Erforderlich: Nein

Der maximale Prozentsatz der Konten, auf denen 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 CloudFormation abgerundet. Wenn das Abrunden zu Null führen würde, wird die Zahl stattdessen auf Eins CloudFormation festgelegt. Sie geben zwar den Höchstwert an, aber bei großen Bereitstellungen kann die tatsächliche Anzahl der Konten, auf die gleichzeitig zugegriffen wird, aufgrund der Dienstdrosselung niedriger sein.

RegionConcurrencyType

Erforderlich: Nein

Sie können angeben, ob das Stack-Set AWS-Regionen sequentiell oder parallel verteilt werden soll, indem Sie den Bereitstellungsparameter Region Concurrency konfigurieren. Wenn die Option „Regionale Parallelität“ so angegeben ist, dass Stapel auf mehreren AWS-Regionen parallel verteilt werden, kann dies insgesamt zu kürzeren Bereitstellungszeiten führen.

- Parallel: Stack-Set-Bereitstellungen werden gleichzeitig durchgeführt, sofern die Bereitstellungsfehler einer Region eine bestimmte Ausfalltoleranz nicht überschreiten.

- Sequentiell: Stack-Set-Bereitstellungen werden nacheinander durchgeführt, sofern die Bereitstellungsfehler einer Region eine bestimmte Ausfalltoleranz nicht überschreiten. Die sequenzielle Bereitstellung ist die Standardauswahl.

ConcurrencyMode

Erforderlich: Nein

Im Parallelitätsmodus können Sie wählen, wie sich die Parallelitätsstufe bei Stack-Set-Vorgängen verhält, ob mit strikter oder weicher Ausfalltoleranz. Strikte Fehlertoleranz senkt die Bereitstellungsgeschwindigkeit bei fehlerhaften Stack-Set-Vorgängen, da die Parallelität bei jedem Fehler abnimmt. Bei Soft Failure Tolerance wird der Geschwindigkeit der Bereitstellung Priorität eingeräumt und gleichzeitig die Sicherheitsfunktionen genutzt. CloudFormation

- `STRICT_FAILURE_TOLERANCE`: Diese Option senkt dynamisch den Grad der Parallelität, um sicherzustellen, dass die Anzahl der ausgefallenen Konten niemals eine bestimmte Ausfalltoleranz überschreitet. Dies ist das Standardverhalten.
- `SOFT_FAILURE_TOLERANCE`: Diese Option entkoppelt die Ausfalltoleranz von der tatsächlichen Parallelität. Dadurch können Stack-Set-Operationen unabhängig von der Anzahl der Fehler auf einer bestimmten Parallelitätsstufe 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 festgelegt ist, stellen Sie sicher`DELEGATED_ADMIN`, dass die Pipeline-IAM-Rolle über die entsprechende Berechtigung verfügt`organizations:ListDelegatedAdministrators`. Andernfalls schlägt

die Aktion bei der Ausführung fehl und es wird ein Fehler ähnlich dem folgenden angezeigt:`Account used is not a delegated administrator`.

- `SELF`: Bei der Stack-Set-Bereitstellung werden vom Dienst verwaltete Berechtigungen verwendet, wenn Sie beim Verwaltungskonto angemeldet sind.
- `DELEGATED_ADMIN`: Bei der Stack-Set-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 die folgenden 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 Variablenreferenz.

Beispielaktionskonfiguration

Die folgenden Beispiele zeigen die Aktionskonfiguration für die CloudFormationStackInstances Aktion.

Beispiel für das selbstverwaltete Berechtigungsmodell

Das folgende Beispiel zeigt eine CloudFormationStackInstances Aktion, 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 dienstverwaltete Berechtigungsmodell

Das folgende Beispiel zeigt eine CloudFormationStackInstances Aktion für das dienstverwaltete Berechtigungsmodell, bei dem das Bereitstellungsziel die ID einer AWS Organisationseinheit einer Organisation ist. `ou-1111-1example`

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"
}
```

Berechtigungen für Servicerollen: Aktion **`CloudFormationStackSet`**

Für CloudFormation StackSets Aktionen sind die folgenden Mindestberechtigungen erforderlich.

Für die `CloudFormationStackSet` Aktion fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackSet",
        "cloudformation:UpdateStackSet",
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:DescribeStackSet",
        "cloudformation:ListStackInstances"
    ],
    "Resource": "resource_ARN"
},
```

Berechtigungen für Servicerollen: `CloudFormationStackInstances` Aktion

Für die `CloudFormationStackInstances` Aktion fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation"
    ],
    "Resource": "resource_ARN"
},
```

Berechtigungsmodelle für Stack-Set-Operationen

Da 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 dienstverwaltete Berechtigungen definieren.

Mit selbstverwalteten Berechtigungen erstellen Sie die beiden IAM-Rollen, die für Folgendes 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, für die Sie Stack-Set-Instanzen bereitstellen. StackSets Mithilfe dieses Berechtigungsmodells StackSets können Sie die Bereitstellung für jedes AWS Konto durchführen, für das der Benutzer über die Berechtigungen zum Erstellen einer IAM-Rolle verfügt. Weitere Informationen finden Sie im AWS CloudFormation Benutzerhandbuch unter Gewähren selbstverwalteter Berechtigungen.

> **Note**
>
> Da 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 Dienst verwalteten Berechtigungen können Sie Stack-Instances für Konten bereitstellen, die von AWS Organisationen verwaltet werden. Bei Verwendung dieses Berechtigungsmodells 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 Zukunft hinzugefügt werden. Weitere Informationen finden Sie im AWS CloudFormation Benutzerhandbuch unter „Vertrauenswürdigen Zugriff für AWS Organisationen aktivieren“.

Datentypen für Vorlagenparameter

Zu den Vorlagenparametern, die bei Stack-Set-Operationen verwendet werden, 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 in der Vorlage angegebene Standardwert AWS CloudFormation verwendet.
- Beispiel:

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

ParameterValue

- Beschreibung: Der dem Parameter zugeordnete Eingabewert.
- Beispiel:

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

UsePreviousValue

- Verwenden Sie bei 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"
```

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 VERALTET gesetzt, bis die Änderungen für diese Instanz übernommen 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 Best Practices 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 Parametern, die bei Stack-Set-Operationen verwendet werden, finden Sie in der AWS CloudFormation API-Referenz unter den folgenden CloudFormation Aktionen:

  - Die CreateStackSet Aktion erstellt ein Stack-Set.
  - Die UpdateStackSet Aktion aktualisiert das Stack-Set und die zugehörigen Stack-Instances in den angegebenen Konten und Regionen. Selbst wenn der durch das Aktualisieren des Stack-Sets erstellte Stack-Set-Vorgang fehlschlägt (ganz oder teilweise, unter oder über einer angegebenen Fehlertoleranz), wird das Stack-Set mit diesen Änderungen aktualisiert. Nachfolgende CreateStackInstances Aufrufe des angegebenen Stack-Sets verwenden das aktualisierte Stack-Set.
  - Die CreateStackInstances Aktion erstellt eine Stack-Instanz für alle angegebenen Regionen innerhalb aller angegebenen Konten in einem selbstverwalteten Berechtigungsmodell oder innerhalb aller angegebenen Bereitstellungsziele in einem dienstverwalteten Berechtigungsmodell. Sie können Parameter für die durch diese Aktion erstellten Instanzen überschreiben. Wenn die Instanzen bereits existieren, CreateStackInstances Aufrufe UpdateStackInstances mit denselben Eingabeparametern. Wenn Sie diese Aktion verwenden, um Instanzen zu erstellen, wird der Status anderer Stack-Instances dadurch nicht geändert.
  - Die UpdateStackInstances Aktion aktualisiert Stack-Instances mit dem Stack-Set für alle angegebenen Regionen innerhalb aller angegebenen Konten in einem selbstverwalteten

Berechtigungsmodell oder innerhalb aller angegebenen Bereitstellungsziele in einem dienstverwalteten Berechtigungsmodell. Sie können die Parameter für die durch diese Aktion aktualisierten Instanzen überschreiben. Wenn Sie diese Aktion verwenden, um eine Teilmenge von Instances zu aktualisieren, wird dadurch der Status anderer Stack-Instances nicht geändert.

- Die DescribeStackSetOperation Aktion gibt die Beschreibung der angegebenen Stack-Set-Operation zurück.
- Die DescribeStackSet Aktion gibt die Beschreibung des angegebenen Stack-Sets zurück.

AWS CodeBuild Aktionsreferenz erstellen und testen

Ermöglicht die Ausführung von Builds und Tests als Teil Ihrer Pipeline. Wenn Sie eine CodeBuild Build- oder Testaktion ausführen, werden die in der Buildspec angegebenen Befehle in einem Container ausgeführt. CodeBuild 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, wird im CodeBuild Build-Projekt angezeigt, dass der Quellanbieter lautet CodePipeline. Wenn Sie ein Build-Projekt in der CodeBuild Konsole erstellen, können Sie es nicht CodePipeline als Quellanbieter angeben, aber wenn Sie die Build-Aktion zu Ihrer Pipeline hinzufügen, wird die Quelle in der CodeBuild Konsole angepasst. Weitere Informationen finden Sie unter ProjectSource in der AWS CodeBuild -API-Referenz.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Berechtigungen für Servicerollen: CodeBuild Aktion
- Deklaration einer Aktion (CodeBuild Beispiel)
- Weitere Informationen finden Sie auch unter

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 der Aktion sein. CodeBuild sucht nach der Buildspec-Datei und führt die Buildspec-Befehle in dem Verzeichnis aus, das die entpackte Version dieses Artefakts enthält.

Dieser Parameter ist erforderlich, wenn für eine CodeBuild -Aktion mehrere Eingabeartefakte angegeben werden. 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.

BuildspecOverride

Erforderlich: Nein

Eine Inline-Buildspec-Definition oder Buildspec-Dateideklaration, die die zuletzt im Build-Projekt definierte Definition außer Kraft setzt, und zwar nur für diesen Build. Die im Projekt definierte Buildspec wird nicht geändert.

Wenn dieser Wert festgelegt ist, kann er einer der folgenden sein:

- Eine Inline-Buildspec-Definition. Weitere Informationen finden Sie in der Syntaxreferenz unter Buildspec-Syntax.
- Der Pfad zu einer alternativen Buildspec-Datei relativ zum Wert der integrierten `CODEBUILD_SRC_DIR` Umgebungsvariablen oder zum Pfad zu einem S3-Bucket. Der Bucket muss sich im selben AWS-Region wie das Build-Projekt befinden. Geben Sie die buildspec-Datei mit ihrem ARN an (z. B. `arn:aws:s3:::my-codebuild-sample2/buildspec.yml`). Wenn dieser Wert nicht angegeben oder auf eine leere Zeichenfolge gesetzt ist, muss der Quellcode eine Buildspec-Datei in seinem Stammverzeichnis enthalten. Weitere Informationen zum Hinzufügen eines Pfads finden Sie unter Buildspec-Dateiname und Speicherort.

> **Note**
>
> Da Sie mit dieser Eigenschaft die Build-Befehle ändern können, die im Container ausgeführt werden, sollten Sie beachten, dass ein IAM-Principal, der diese API aufrufen und diesen Parameter festlegen kann, die Standardeinstellungen außer Kraft setzen kann. Darüber hinaus empfehlen wir Ihnen, einen vertrauenswürdigen Buildspec-Speicherort wie eine Datei in Ihrem Quell-Repository oder einem Amazon S3-Bucket zu verwenden.

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 Deklaration einer Aktion (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 es sich bei diesem Wert um den Namen eines Parameters handeln, den Sie bereits im AWS Systems Manager Parameter Store gespeichert haben, bzw. um einen geheimen Schlüssel, den Sie bereits in AWS Secrets Manager gespeichert haben.

  > **Note**
  >
  > Wir raten dringend davon ab, Umgebungsvariablen zum Speichern vertraulicher Werte, insbesondere 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 die Konfiguration `namevalue`, und `type` für Ihre Umgebungsvariablen eingeben, insbesondere wenn die Umgebungsvariable die Syntax der CodePipeline Ausgabevariablen enthält, überschreiten Sie nicht die 1000-Zeichen-Grenze für das Wertefeld der Konfiguration. Ein Validierungsfehler wird zurückgegeben, wenn dieser Grenzwert überschritten wird.

Weitere Informationen finden Sie EnvironmentVariable in der AWS CodeBuild API-Referenz. 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.

ServiceRoleArnOverride

Erforderlich: Nein

Ein IAM-Rollen-ARN, der als Servicerolle für Builds CodeBuild verwendet wird, die durch diese Aktion ausgelöst werden. Er überschreibt die im Projekt konfigurierte Standard-Servicerolle. CodeBuild Wenn das CodeBuild Projekt nicht vorhanden ist, verwendet es seine Standard-Servicerolle.

Die angegebene Rolle muss über eine Vertrauensrichtlinie verfügen, die es dem `codebuild.amazonaws.com` Dienst ermöglicht, sie zu übernehmen. Sowohl die Identität, die zum Erstellen oder Aktualisieren der Pipeline verwendet wurde, als auch die Pipeline-Servicerolle müssen über die `iam:PassRole` Berechtigung für die angegebene Rolle verfügen. Eine Beispielrichtlinie finden Sie unter Berechtigungen für Servicerollen: CodeBuild Aktion.

Aus Sicherheitsgründen empfehlen wir, wenn ein CodeBuild Projekt von mehreren Pipelines gemeinsam genutzt wird, für jede Pipeline eine bereichsspezifische Rolle anzugeben, um dem Prinzip der geringsten Rechte zu folgen. Für Pipelines, die CodeConnections Quellaktionen verwenden, finden Sie unter ein Beispiel Fügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub Enterprise Server oder hinzu GitHub GitLab.com für die Festlegung des Geltungsbereichs der Verbindungsberechtigungen der Rolle zu einem bestimmten Repository.

> **Note**
>
> Die Rolle muss sich im selben AWS Konto wie das Projekt befinden. CodeBuild

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1 to 5`
- Beschreibung: CodeBuild sucht nach der Buildspec-Datei und führt die Buildspec-Befehle aus dem Verzeichnis des primären Quellartefakts aus. Wenn entweder eine einzelne Eingabequelle oder mehr als eine Eingabequelle für die CodeBuild Aktion angegeben wird, muss das einzelne Artefakt oder bei mehreren Eingabequellen das primäre Artefakt mithilfe des Aktionskonfigurationsparameters in festgelegt werden. `PrimarySource` 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 Buildspec-Datei definierten Artefakte für nachfolgende Aktionen in der Pipeline verfügbar zu machen. Wenn nur ein Ausgabe-Artefakt definiert ist, kann dieses Artefakt direkt im `artifacts` Abschnitt der Buildspec-Datei definiert werden. Wenn mehr als ein Ausgabe-Artefakt angegeben ist, müssen alle Artefakte, auf die verwiesen wird, in der Buildspec-Datei als sekundäre Artefakte definiert werden. Die Namen der ausgegebenen Artefakte CodePipeline müssen mit den Artefakt-Identifikatoren in der Buildspec-Datei 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 EnvironmentVariable im AWS CodeBuild API-Handbuch.

Weitere Informationen zur Verwendung von CodeBuild Umgebungsvariablen finden Sie in CodePipeline den Beispielen unterCodeBuild Ausgabevariablen für Aktionen. Eine Liste der Umgebungsvariablen, die Sie in verwenden können CodeBuild, finden Sie im AWS CodeBuild Benutzerhandbuch unter Umgebungsvariablen in Build-Umgebungen.

Berechtigungen für Servicerollen: CodeBuild Aktion

Wenn Sie CodeBuild Unterstützung benötigen, fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "codebuild:BatchGetBuilds",
                "codebuild:StartBuild",
                "codebuild:BatchGetBuildBatches",
                "codebuild:StartBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:*:111122223333:project/[[ProjectName]]"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Wenn Sie den `ServiceRoleArnOverride` Parameter verwenden, fügen Sie die folgende `iam:PassRole` Berechtigung hinzu, um die CodePipeline Übergabe der Rolle an zu ermöglichen CodeBuild:

```
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::111122223333:role/your-codebuild-override-role",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "codebuild.amazonaws.com"
```

```
        }
    }
}
```

Deklaration einer Aktion (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"}]'
      ServiceRoleArnOverride: 'arn:aws:iam::111122223333:role/my-scoped-codebuild-
role'
    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\"}]",
                "ServiceRoleArnOverride": "arn:aws:iam::111122223333:role/my-scoped-
codebuild-role"
            },
            "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 mit CodeBuild zum Testen von Code und Ausführen von Builds. Beispiele für Projekte mit mehreren Eingabe- und CodeBuild Ausgabeartefakten finden

Sie unter CodePipeline Integration mit CodeBuild und mehreren Eingabequellen und Beispiel für 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 eine Buildspec-Beispieldatei und eine Beispielanwendung zum Erstellen einer Pipeline mit einer GitHub Quelle, die eine Android-App mit und erstellt und testet. CodeBuild AWS Device Farm
- Referenz zur Build-Spezifikation für CodeBuild — Dieses Referenzthema enthält Definitionen und Beispiele zum Verständnis CodeBuild von Buildspec-Dateien. Eine Liste der Umgebungsvariablen, in denen Sie sie verwenden können CodeBuild, finden Sie im Benutzerhandbuch unter Umgebungsvariablen in Build-Umgebungen. AWS CodeBuild

AWS CodePipeline Aktionsreferenz aufrufen

Sie verwenden eine CodePipeline Aufrufaktion, um das Auslösen von Downstream-Pipeline-Ausführungen und die Übergabe von Pipeline-Variablen und Quellrevisionen zwischen Pipelines zu vereinfachen.

> **Note**
>
> Diese Aktion wird nur für Pipelines vom Typ V2 unterstützt.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Berechtigungen für die Aktion zum CodePipeline Aufrufen der Aktion „Dienst“
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Invoke`

- Eigentümer: `AWS`
- Anbieter: `CodePipeline`
- Version: `1`

Konfigurationsparameter

PipelineName

Erforderlich: Ja

Der Name der Pipeline, die bei der Ausführung die aktuelle Zielpipeline startet. Sie müssen die aufrufende Pipeline bereits erstellt haben. Die Aktion startet die `s3-pipeline-test` (Ziel-) Pipeline, wenn die angegebene (aufrufende) Pipeline eine Ausführung `my-s3-pipeline` startet.

SourceRevisions

Erforderlich: Nein

Die Quellrevisionen, die die Zielpipeline verwenden soll, wenn sie von der aufrufenden Pipeline gestartet wird. Beispielsweise stellt eine S3-Quellaktion Ausgabevariablen wie die S3-Versions-ID und den Objektschlüssel bereit. Sie können einen Revisionswert angeben, der verwendet werden soll, wenn die Pipeline aufgerufen wird.

Für die CLI geben Sie Quellrevisionen als serialisierte JSON-Zeichenfolge an. Weitere Informationen zur Verwendung von Quellrevisionsüberschreibungen finden Sie SourceRevisionOverride im CodePipeline API-Handbuch.

Das Mapping verwendet ein Zeichenfolgenformat, wie im folgenden Beispiel gezeigt:

```
[{"actionName":"Source","revisionType":"S3_OBJECT_VERSION_ID","revision
Value":"zq8mjNEXAMPLE"}]
```

Variablen

Erforderlich: Nein

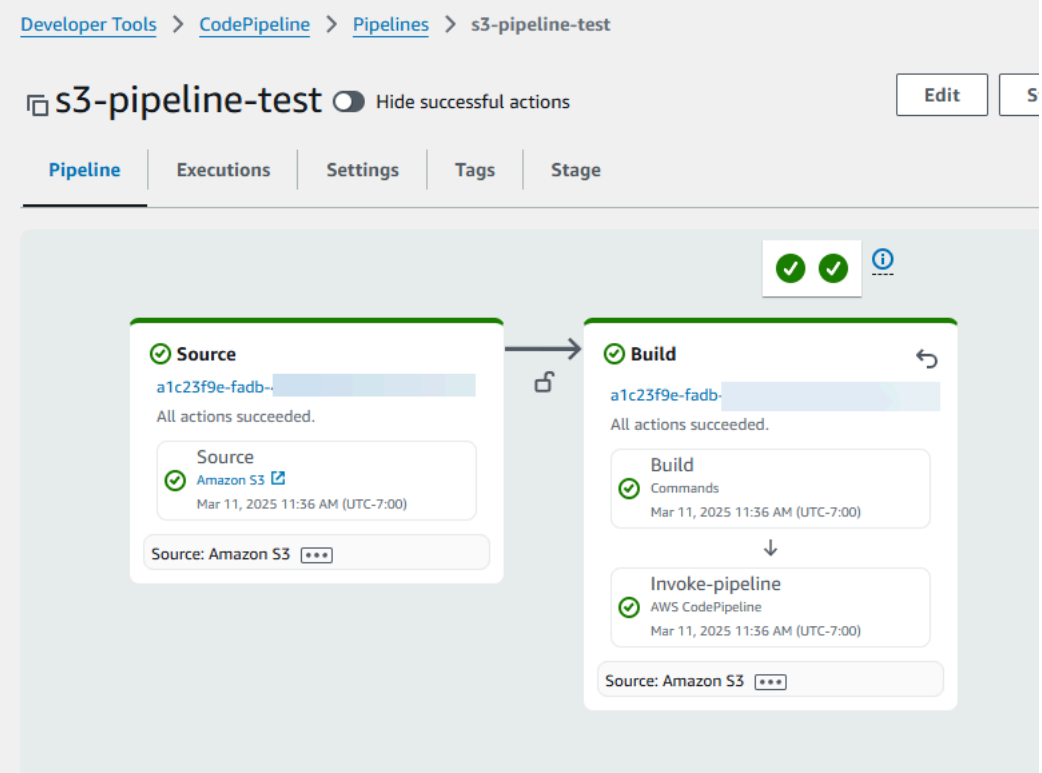
Die Namen und Werte der Variablen, die die Aktion unterstützen soll.

Für die CLI geben Sie Variablen als serialisierte JSON-Zeichenfolge an. Weitere Informationen zur Verwendung von Pipeline-Variablen finden Sie PipelineVariable im CodePipeline API-Handbuch.

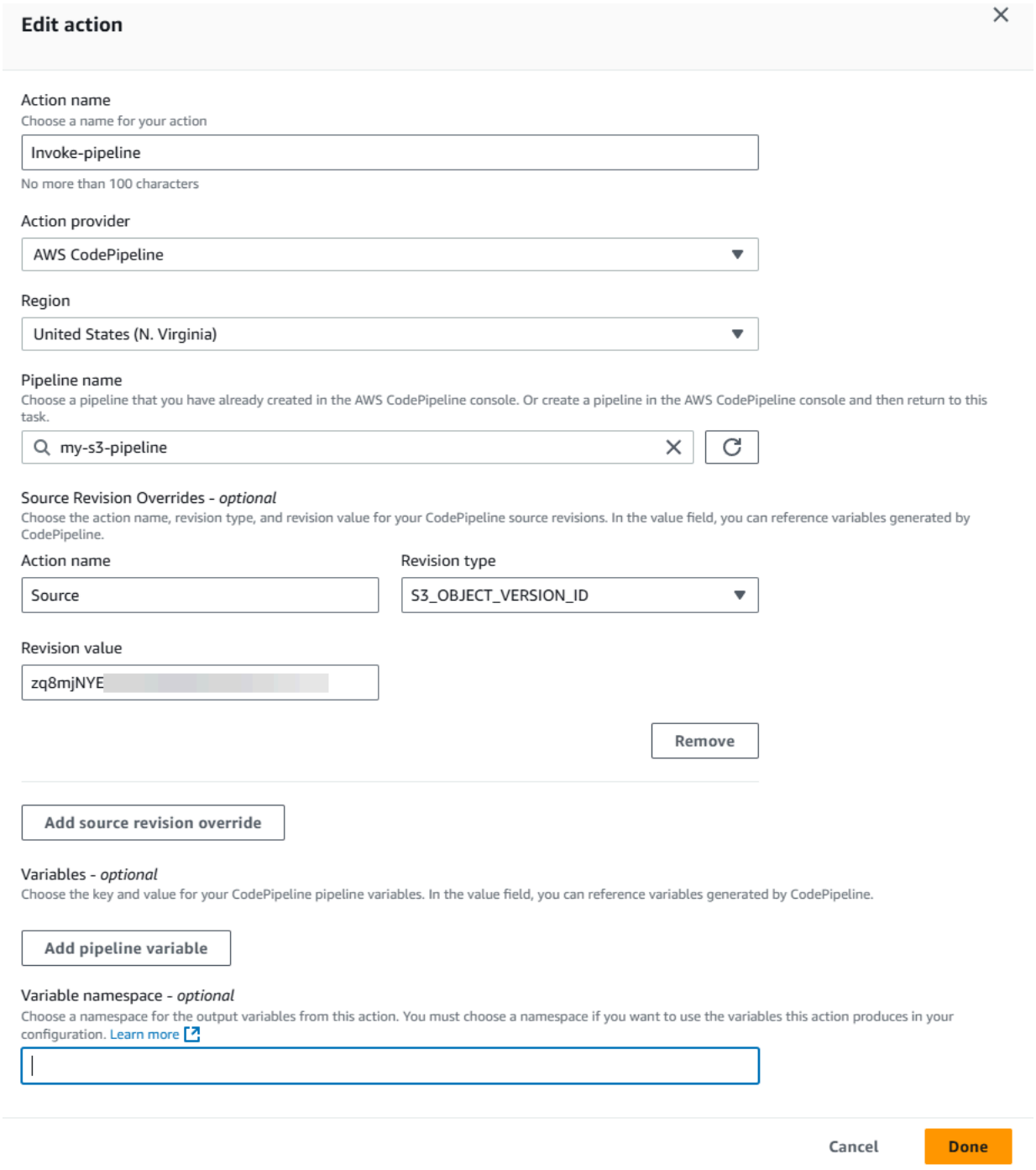
Das Mapping verwendet ein Zeichenkettenformat, wie im folgenden Beispiel gezeigt:

```
[{"name":"VAR1","value":"VALUE1"}]
```

Die folgende Abbildung zeigt ein Beispiel für die Aktion, die einer Pipeline in der Konsole hinzugefügt wurde.



Die folgende Abbildung zeigt ein Beispiel für die Seite „Bearbeiten“ für die Aktion. Im folgenden Beispiel ist für die Pipeline mit dem Namen `s3-pipeline-test` eine Aktion zum Aufrufen einer Pipeline konfiguriert, wie für die Konsole gezeigt. Die Aktion startet die `s3-pipeline-test` Pipeline, wenn die angegebene Pipeline eine Ausführung `my-s3-pipeline` abgeschlossen hat. Das Beispiel zeigt, dass die Quellversion für die Quellcodeüberschreibung der S3_OBJECT_VERSION_ID mit dem angegebenen Revisionswert von überschrieben wird. `zq8mjNYEexample`



Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0`
- Beschreibung: Eingabe-Artefakte sind für diesen Aktionstyp nicht gültig.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Berechtigungen für die Aktion zum CodePipeline Aufrufen der Aktion „Dienst“

Wenn die Aktion CodePipeline ausgeführt wird, erfordert die CodePipeline Servicerollenrichtlinie die entsprechende `codepipeline:StartPipelineExecution` Berechtigung, die entsprechend auf den ARN der Pipeline-Ressource heruntergeregelt ist, um den Zugriff mit den geringsten Rechten aufrechtzuerhalten.

```
{
           "Sid": "StatementForPipelineInvokeAction",
           "Effect": "Allow",
           "Action": "codepipeline:StartPipelineExecution",
           "Resource": [
               "arn:aws:codepipeline:{{region}}:{{AccountId}}:{{pipelineName}}"
           ]
       }
```

Aktionsdeklaration

YAML

```
name: Invoke-pipeline
actionTypeId:
  category: Invoke
  owner: AWS
  provider: CodePipeline
  version: '1'
runOrder: 2
configuration:
  PipelineName: my-s3-pipeline
  SourceRevisions:
 '[{"actionName":"Source","revisionType":"S3_OBJECT_VERSION_ID","revision
Value":"zq8mjNEXAMPLE"}]'
  Variables: '[{"name":"VAR1","value":"VALUE1"}]'
```

JSON

```
{
    "name": "Invoke-pipeline",
    "actionTypeId": {
        "category": "Invoke",
        "owner": "AWS",
        "provider": "CodePipeline",
        "version": "1"
    },
    "runOrder": 2,
    "configuration": {
        "PipelineName": "my-s3-pipeline",
        "SourceRevisions": "[{\"actionName\":\"Source\",\"revisionType\":
\"S3_OBJECT_VERSION_ID\",\"revisionValue\":\"zq8mjNEXAMPLE\"}]",
        "Variables": "[{\"name\":\"VAR1\",\"value\":\"VALUE1\"}]"
    }
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung— In diesem Abschnitt wird beschrieben, wie eine Pipeline mit Quellrevisionen manuell oder über den EventBridge Event-Input-Transformator gestartet wird.

CodeCommit Quellaktionsreferenz

Startet die Pipeline, wenn ein neuer Commit für das konfigurierte CodeCommit Repository und den Branch erfolgt.

Wenn Sie die Pipeline mithilfe der Konsole erstellen oder bearbeiten, CodePipeline wird eine EventBridge Regel erstellt, die Ihre Pipeline startet, wenn eine Änderung im Repository erfolgt.

> **Note**
>
> Für Amazon ECR, Amazon S3 oder CodeCommit Quellen können Sie auch eine Quellüberschreibung mithilfe eines Eingabe-Transformationseintrags erstellen, um das

`revisionValue` In EventBridge für Ihr Pipeline-Ereignis zu verwenden, wobei das von der Quellereignisvariablen für Ihren Objektschlüssel, Ihren Commit oder Ihre Bild-ID abgeleitet `revisionValue` wird. Weitere Informationen finden Sie in dem optionalen Schritt für die Eingabe der Eingabetransformation, der in den Verfahren unter Amazon ECR-Quellaktionen und Ressourcen EventBridgeHerstellen einer Verbindung zu Amazon S3-Quellaktionen mit einer für Ereignisse aktivierten Quelle, oder CodeCommit Quellaktionen und EventBridge enthalten ist.

Sie müssen bereits ein CodeCommit Repository erstellt haben, bevor Sie die Pipeline über eine CodeCommit Aktion verbinden.

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 eine ZIP-Datei mit einer flachen Kopie Ihres Commits ausgegeben wird.
- Vollständiger Klon — Konfiguriert die Quellaktion so, dass für nachfolgende Aktionen eine Git-URL-Referenz auf das Repository 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 Nicht-Aktionen zu übergeben, führt zu einem Fehler.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Berechtigungen für Servicerollen: CodeCommit Aktion
- 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 Quelländerungen CodePipeline abgefragt wird. Wir empfehlen, stattdessen CloudWatch Ereignisse zu verwenden, um Änderungen an der Quelle zu erkennen. Weitere Informationen zur Konfiguration von CloudWatch Ereignissen finden Sie unter Migrieren Sie Polling-Pipelines (CodeCommit Quelle) (CLI) oderMigrieren Sie Polling-Pipelines (CodeCommit Quelle) (CloudFormation Vorlage).

> **Important**
>
> Wenn Sie beabsichtigen, eine CloudWatch Ereignisregel zu konfigurieren, müssen Sie sie `PollForSourceChanges` auf festlegen, `false` um doppelte Pipelineausführungen zu vermeiden.

Gültige Werte für diesen Parameter sind:

- `true`: Falls gesetzt, CodePipeline fragt dein Repository nach Änderungen an der Quelle ab.

  > **Note**
  >
  > Wenn du es auslässt`PollForSourceChanges`, wird dein Projektarchiv CodePipeline standardmäßig nach Änderungen an der Quelle abgefragt. Dieses Verhalten ist das gleiche, als ob `PollForSourceChanges` enthalten wäre und auf `true` festgelegt würde.

- `false`: Wenn gesetzt, fragt dein Projektarchiv CodePipeline nicht nach Änderungen an der Quelle ab. Verwende diese Einstellung, wenn du beabsichtigst, eine CloudWatch Event-Regel zu konfigurieren, um Änderungen an der Quelle zu erkennen.

OutputArtifactFormat

Erforderlich: Nein

Das Ausgabe-Artefaktformat. 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 nachgeschalteten Aktionen verwendet werden.
> Wenn Sie diese Option wählen, müssen Sie die `codecommit:GitPull` Berechtigung zu Ihrer CodeBuild Servicerolle hinzufügen, wie unter gezeigtFügen Sie CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen hinzu. Sie müssen die `codecommit:GetRepository` Berechtigung auch Ihrer CodePipeline Servicerolle hinzufügen, wie unter gezeigtFügen Sie der CodePipeline Servicerolle Berechtigungen hinzu. Ein Tutorial, das Ihnen zeigt, wie Sie die Option „Vollständiger Klon“ 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 Commit-ID des Quellcodes wird CodePipeline als Quellversion 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 Variablenreferenz.

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 vorgenommen wurde, der die Pipeline ausgelöst hat.

BranchName

Der Name des Branches für das CodeCommit Repository, in dem die Quelländerung vorgenommen wurde.

AuthorDate

Das Datum im Zeitstempelformat, an dem der Commit erstellt wurde.

CommitterDate

Das Datum im Zeitstempelformat, an dem der Commit durchgeführt wurde.

Berechtigungen für Servicerollen: CodeCommit Aktion

Wenn die Aktion CodePipeline ausgeführt wird, erfordert die CodePipeline Servicerollenrichtlinie die folgenden Berechtigungen, die entsprechend auf den ARN der Pipeline-Ressource beschränkt sind, um den Zugriff mit den geringsten Rechten aufrechtzuerhalten. Fügen Sie Ihrer Richtlinienerklärung beispielsweise Folgendes hinzu:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:CancelUploadArchive",
                "codecommit:GetBranch",
                "codecommit:GetCommit",
                "codecommit:GetRepository",
                "codecommit:GetUploadArchiveStatus",
                "codecommit:UploadArchive"
            ],
            "Resource": [
                "arn:aws:codecommit:*:111122223333:[[codecommitRepostories]]"
            ]
        }
    ]
}
```

Beispielaktionskonfiguration

Beispiel für das Standardformat für Ausgabeartefakte

YAML

```
name: Source
actionTypeId:
  category: Source
  owner: AWS
  provider: CodeCommit
  version: '1'
runOrder: 1
configuration:
  BranchName: main
  PollForSourceChanges: 'false'
  RepositoryName: MyWebsite
outputArtifacts:
  - name: Artifact_MyWebsiteStack
inputArtifacts: []
region: us-west-2
namespace: SourceVariables
```

JSON

```
{
    "name": "Source",
    "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "provider": "CodeCommit",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {
        "BranchName": "main",
        "PollForSourceChanges": "false",
        "RepositoryName": "MyWebsite"
    },
    "outputArtifacts": [
        {
            "name": "Artifact_MyWebsiteStack"
        }
```

```
    ],
    "inputArtifacts": [],
    "region": "us-west-2",
    "namespace": "SourceVariables"
}
```

Beispiel für ein vollständiges 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: Eine einfache Pipeline (CodeCommit Repository) erstellen— Dieses Tutorial enthält eine AppSpec Beispieldatei sowie eine CodeDeploy Beispielanwendung und eine Bereitstellungsgruppe. Verwenden Sie dieses Tutorial, um eine Pipeline mit einer CodeCommit Quelle zu erstellen, die auf Amazon EC2-Instances bereitgestellt wird.

AWS CodeDeploy Aktionsreferenz bereitstellen

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 Amazon Elastic Container Service für CodeDeploy blue/green Bereitstellungsaktionen in CodePipeline finden Sie unterAmazon Elastic Container Service und Referenz für Bereitstellungsaktionen in CodeDeploy Blaugrün.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Berechtigungen für Servicerollen: AWS CodeDeploy Aktion
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `CodeDeploy`
- Version: `1`

Konfigurationsparameter

ApplicationName

Erforderlich: Ja

Der Name der Anwendung, in der Sie erstellt haben CodeDeploy.

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.

Berechtigungen für Servicerollen: AWS CodeDeploy Aktion

Wenn Sie AWS CodeDeploy Unterstützung benötigen, fügen Sie Ihrer Grundsatzerklärung Folgendes hinzu:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:CreateDeployment",
                "codedeploy:GetApplication",
                "codedeploy:GetDeployment",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:ListDeployments",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:GetDeploymentGroup"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:application:
[[codedeployApplications]]",
                "arn:aws:codedeploy:*:111122223333:deploymentgroup:
[[codedeployApplications]]/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:GetDeploymentConfig"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:deploymentconfig:
[[deploymentConfigs]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:ListDeploymentConfigs"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

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: Eine einfache Pipeline (CodeCommit Repository) erstellen— 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 und GitLab.com GitLab selbstverwaltete Aktionen

Quellaktionen für Verbindungen werden von unterstützt. AWS CodeConnections AWS 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 mit Code-Push- oder Pull-Request-Ereignistypen hinzufügen.

> **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 und GitLab.com 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 (via GitHub App) 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` Provider in der CLI)

> **Note**
>
> Sie können Verbindungen zu einem Repository herstellen GitLab, in dem Sie die Rolle „Besitzer“ haben. Anschließend kann die Verbindung mit dem Repository mit Ressourcen wie verwendet werden CodePipeline. Bei Repositorys in Gruppen müssen Sie nicht der Gruppenbesitzer sein.

- Self-managed Installation für GitLab (Enterprise Edition oder Community Edition) (über die Option für GitLab selbstverwaltete Anbieter in der CodePipeline Konsole oder den `GitLabSelfManaged` Anbieter 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. Detaillierte Anweisungen finden Sie unter Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu.

> **Note**
>
> Um Verbindungen in Europa (Mailand) nutzen zu können AWS-Region, müssen Sie:
>
> 1. Installieren Sie eine Region-specific App
> 2. Region aktivieren
>
> Diese Region-specific 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 deine Daten trotzdem mit unserem Service teilen, solange die Region-specific App installiert bleibt. Achte also darauf, die App zu deinstallieren, sobald du die Region deaktivierst. Weitere Informationen finden Sie unter Aktivieren einer Region.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Berechtigungen für Servicerollen: AWS CodeConnections action
- Aktionsdeklaration
- Installation der Installations-App und Herstellen einer Verbindung
- Weitere Informationen finden Sie auch unter

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 nachgeschalteten 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, GitHub Enterprise Server oder hinzu GitHub GitLab.com. 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 Variablenreferenz.

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.

Berechtigungen für Servicerollen: AWS CodeConnections action

Für AWS CodeConnections die Erstellung von Pipelines mit einer Quelle, die eine Verbindung verwendet, wie Bitbucket Cloud, ist die folgende Berechtigung erforderlich.

```
{
    "Effect": "Allow",
    "Action": [
        "codeconnections:UseConnection",
        "codestar-connections:UseConnection"
    ],
    "Resource": "resource_ARN"
```

```
},
```

> **Note**
>
> Beide Aktionen sind erforderlich. Operationen, die durch einen Anruf aufgerufen werden`codestar-connections:UseConnection`, während Operationen, die außerhalb des Anrufs aufgerufen werden. `codeconnections:UseConnection` Durch die Gewährung beider Aktionen wird sichergestellt, dass die Rolle für beide Codepfade funktioniert, unabhängig davon, ob der Ressourcen-ARN der Verbindung das `codestar-connections` Dienstpräfix `codeconnections` oder verwendet.

Weitere Informationen zu den IAM-Berechtigungen für Verbindungen finden Sie unter Referenz zu Verbindungsberechtigungen.

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 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 CloudFormation Vorlagenreferenz für die AWS CodeStar Connections-Ressource enthält Parameter und Beispiele für Verbindungen in 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.

Aktionsreferenz für Befehle

Mit der Aktion „Befehle“ können Sie Shell-Befehle in einer virtuellen Recheninstanz ausführen. Wenn Sie die Aktion ausführen, werden die in der Aktionskonfiguration angegebenen Befehle in einem separaten Container ausgeführt. Alle Artefakte, die als Eingabeartefakte für eine CodeBuild Aktion angegeben sind, sind in dem Container verfügbar, in dem die Befehle ausgeführt werden. Mit dieser Aktion können Sie Befehle angeben, ohne zuerst ein CodeBuild Projekt erstellen zu müssen. Weitere Informationen finden Sie unter ActionDeclaration und OutputArtifact in der AWS CodePipeline -API-Referenz.

> **Important**
>
> Diese Aktion verwendet CodePipeline Managed CodeBuild Compute, um Befehle in einer Build-Umgebung auszuführen. Für die Ausführung der Befehlsaktion fallen separate Gebühren in an AWS CodeBuild.

> **Note**
>
> Die Aktion Befehle ist nur für Pipelines vom Typ V2 verfügbar.

Themen

- Überlegungen zur Aktion „Befehle“
- Dienstrolle, Richtlinien, Berechtigungen
- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Umgebungsvariablen
- Berechtigungen für Servicerollen: Befehle, Aktion

- Aktionserklärung (Beispiel)
- Weitere Informationen finden Sie auch unter

Überlegungen zur Aktion „Befehle“

Die folgenden Überlegungen gelten für die Aktion Befehle.

- Die Befehlsaktion verwendet CodeBuild Ressourcen, die der CodeBuild Aktion ähneln, ermöglicht jedoch Shell-Umgebungsbefehle in einer virtuellen Recheninstanz, ohne dass ein Build-Projekt zugeordnet oder erstellt werden muss.

  > **Note**
  >
  > Für die Ausführung der Befehlsaktion fallen separate Gebühren an AWS CodeBuild.

- Da die Aktion „Befehle“ in CodeBuild Ressourcen CodePipeline verbraucht, werden die von der Aktion ausgeführten Builds den Build-Limits für Ihr Konto in CodeBuild zugeordnet. Builds, die mit der Aktion „Befehle“ ausgeführt werden, werden auf die für dieses Konto konfigurierten gleichzeitigen Build-Limits angerechnet.
- Das Timeout für Builds mit der Aktion „Befehle“ beträgt 55 Minuten, basierend auf CodeBuild Builds.
- Die Compute-Instance verwendet eine isolierte Build-Umgebung in CodeBuild.

  > **Note**
  >
  > Da die isolierte Build-Umgebung auf Kontoebene verwendet wird, kann eine Instanz für eine weitere Pipeline-Ausführung wiederverwendet werden.

- Alle Formate außer mehrzeiligen Formaten werden unterstützt. Bei der Befehlseingabe müssen Sie das einzeilige Format verwenden.
- Die Befehlsaktion wird für kontoübergreifende Aktionen unterstützt. Um eine Aktion für kontoübergreifende Befehle hinzuzufügen, fügen Sie in der Aktionsdeklaration die Aktion `actionRoleArn` aus Ihrem Zielkonto hinzu.
- Übernimmt für diese Aktion CodePipeline die Pipeline-Service-Rolle und verwendet diese Rolle, um den Zugriff auf Ressourcen zur Laufzeit zu ermöglichen. Es wird empfohlen, die Servicerolle so zu konfigurieren, dass die Berechtigungen auf die Aktionsebene beschränkt sind.

- Die der CodePipeline Servicerolle hinzugefügten Berechtigungen werden unter detailliert beschrieben. Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu
- Die zum Anzeigen von Protokollen in der Konsole erforderliche Berechtigung ist unter detailliert beschriebenZum Anzeigen von Rechenprotokollen in der Konsole sind Berechtigungen erforderlich.
- Im Gegensatz zu anderen Aktionen in CodePipeline legen Sie keine Felder in der Aktionskonfiguration fest. Sie legen die Felder für die Aktionskonfiguration außerhalb der Aktionskonfiguration fest.

Dienstrolle, Richtlinien, Berechtigungen

Wenn die Aktion CodePipeline ausgeführt wird, CodePipeline wird eine Protokollgruppe erstellt, die den Namen der Pipeline wie folgt verwendet. Auf diese Weise können Sie die Berechtigungen zum Protokollieren von Ressourcen anhand des Pipeline-Namens einschränken.

```
/aws/codepipeline/MyPipelineName
```

Wenn Sie eine vorhandene Servicerolle verwenden, müssen Sie die folgenden Berechtigungen für die Servicerolle hinzufügen, um die Aktion Befehle verwenden zu können.

- Protokolle: CreateLogGroup
- Protokolle: CreateLogStream
- Protokolle: PutLogEvents

Beschränken Sie in der Erklärung zur Servicerolle die Berechtigungen auf die Pipeline-Ebene, wie im folgenden Beispiel gezeigt.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME",
```

```
        "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
    ]
}
```

Um Protokolle in der Konsole mithilfe der Dialogseite mit den Aktionsdetails anzuzeigen, muss der Konsolenrolle die Berechtigung zum Anzeigen von Protokollen hinzugefügt werden. Weitere Informationen finden Sie im Beispiel für eine Konsolenberechtigungsrichtlinie unterZum Anzeigen von Rechenprotokollen in der Konsole sind Berechtigungen erforderlich.

Aktionstyp

- Kategorie: `Compute`
- Eigentümer: `AWS`
- Anbieter: `Commands`
- Version: `1`

Konfigurationsparameter

Befehle

Erforderlich: Ja

Sie können Shell-Befehle für die Ausführung der `Commands` Aktion angeben. In der Konsole werden Befehle in separaten Zeilen eingegeben. In der CLI werden Befehle als separate Zeichenfolgen eingegeben.

> **Note**
>
> Multi-line Formate werden nicht unterstützt und führen zu einer Fehlermeldung. Single-line Das Format muss für die Eingabe von Befehlen in das Feld Befehle verwendet werden.

> **Important**
>
> Die ComputeType Werte EnvironmentType und stimmen mit denen in überein CodeBuild. Wir unterstützen eine Teilmenge der verfügbaren Typen. Weitere Informationen finden Sie unter Erstellen von Umgebungs-Compute-Typen.

EnvironmentType

Erforderlich: Nein

Das Betriebssystemimage für die Build-Umgebung, die die Aktion „Befehle“ unterstützt. Die folgenden Werte sind für Build-Umgebungen gültig:

- LINUX_CONTAINER
- WINDOWS_SERVER_2022_CONTAINER

Die Auswahl für ermöglicht EnvironmentType dann den Berechnungstyp für dieses Betriebssystem im Feld. ComputeType Weitere Informationen zu den für diese Aktion verfügbaren CodeBuild Berechnungstypen finden Sie in der Referenz zu den Berechnungsmodi und -typen der Build-Umgebung im CodeBuild Benutzerhandbuch.

> **Note**
>
> Wenn nicht angegeben, wird die Berechnung für die Build-Umgebung standardmäßig wie folgt durchgeführt:
>
> - Berechnungstyp: BUILD_GENERAL1_SMALL
> - Umgebungstyp: LINUX_CONTAINER

ComputeType

Erforderlich: Nein

Basierend auf der Auswahl für EnvironmentType kann der Berechnungstyp angegeben werden. Die folgenden Werte sind für die Berechnung verfügbar. Beachten Sie jedoch, dass die verfügbaren Optionen je nach Betriebssystem variieren können.

- BUILD_GENERAL1_SMALL
- BUILD_GENERAL1_MEDIUM
- BUILD_GENERAL1_LARGE

> **Important**
>
> Einige Berechnungstypen sind mit bestimmten Umgebungstypen nicht kompatibel. Beispielsweise ist WINDOWS_SERVER_2022_CONTAINER nicht mit

BUILD_GENERAL1_SMALL kompatibel. Die Verwendung inkompatibler Kombinationen führt dazu, dass die Aktion fehlschlägt und ein Laufzeitfehler generiert wird.

Ausgabevariablen

Erforderlich: Nein

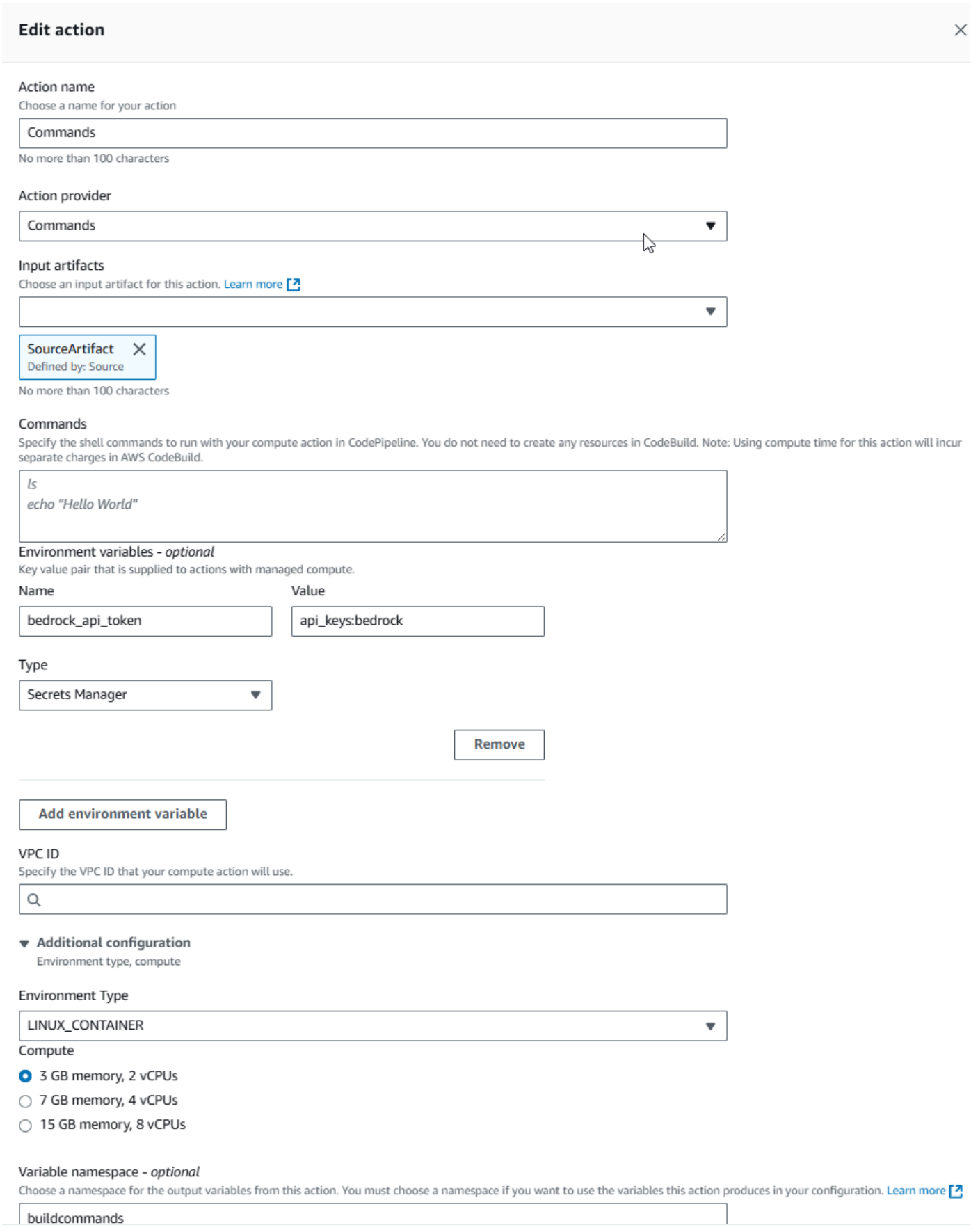
Geben Sie die Namen der Variablen in Ihrer Umgebung an, die Sie exportieren möchten. Eine Referenz zu CodeBuild Umgebungsvariablen finden Sie im CodeBuild Benutzerhandbuch unter Umgebungsvariablen in Build-Umgebungen.

Dateien

Erforderlich: Nein

Sie können Dateien, die Sie exportieren möchten, als Ausgabeartefakte für die Aktion bereitstellen.

Das unterstützte Format für Dateien ist dasselbe wie für CodeBuild Dateimuster. Geben Sie beispielsweise `**/` für alle Dateien ein. Weitere Informationen finden Sie in der Referenz zur Build-Spezifikation für CodeBuild im CodeBuild Benutzerhandbuch.


Edit action
Action name
Choose a name for your action
Commands
No more than 100 characters
Action provider
Commands
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
Defined by: Source
No more than 100 characters
Commands
Specify the shell commands to run with your compute action in CodePipeline. You do not need to create any resources in CodeBuild. Note: Using compute time for this action will incur separate charges in AWS CodeBuild.
ls
echo "Hello World"
Environment variables - optional
Key value pair that is supplied to actions with managed compute.
Name
Value
bedrock_api_token
api_keys:bedrock
Type
Secrets Manager
Remove
Add environment variable
VPC ID
Specify the VPC ID that your compute action will use.
Additional configuration
Environment type, compute
Environment Type
LINUX_CONTAINER
Compute
3 GB memory, 2 vCPUs
7 GB memory, 4 vCPUs
15 GB memory, 8 vCPUs
Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. Learn more
buildcommands
Cancel
Done

VpcId

Erforderlich: Nein

Die VPC-ID für Ihre Ressourcen.

Subnets

Erforderlich: Nein

Die Subnetze für die VPC. Dieses Feld wird benötigt, wenn Ihre Befehle eine Verbindung zu Ressourcen in einer VPC herstellen müssen.

SecurityGroupIds

Erforderlich: Nein

Die Sicherheitsgruppen für die VPC. Dieses Feld wird benötigt, wenn Ihre Befehle eine Verbindung zu Ressourcen in einer VPC herstellen müssen.

Im Folgenden finden Sie ein JSON-Beispiel für die Aktion mit den angezeigten Konfigurationsfeldern für Umgebung und Berechnungstyp sowie einer Beispielumgebungsvariablen.

```
{
           "name": "Commands1",
           "actionTypeId": {
             "category": "Compute",
             "owner": "AWS",
             "provider": "Commands",
             "version": "1"
           },
           "inputArtifacts": [
             {
               "name": "SourceArtifact"
             }
           ],
           "commands": [
             "ls",
             "echo hello",
             "echo $BEDROCK_TOKEN",
           ],
           "configuration": {
             "EnvironmentType": "LINUX_CONTAINER",
```

```
            "ComputeType": "BUILD_GENERAL1_MEDIUM"
          },
          "environmentVariables": [
            {
              "name": "BEDROCK_TOKEN",
              "value": "apiTokens:bedrockToken",
              "type": "SECRETS_MANAGER"
            }
          ],
          "runOrder": 1
        }
```

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1 to 10`

Ausgabeartefakte

- Anzahl der Artefakte: `0 to 1`

Umgebungsvariablen

Key (Schlüssel)

Der Schlüssel in einem Schlüssel-Wert-Umgebungsvariablenpaar, z. B. `BEDROCK_TOKEN`

Wert

Der Wert für das Schlüssel-Wert-Paar, z. B. `apiTokens:bedrockToken` Der Wert kann mit Ausgabevariablen aus Pipeline-Aktionen oder Pipeline-Variablen parametrisiert werden.

Wenn Sie den `SECRETS_MANAGER` Typ verwenden, muss dieser Wert der Name eines Geheimnisses sein, das Sie bereits in AWS Secrets Manager gespeichert haben.

Typ

Gibt die Art der Verwendung des Umgebungsvariablenwerts an. Der Wert kann entweder `PLAINTEXT` oder `SECRETS_MANAGER` sein. Wenn der Wert gleich ist`SECRETS_MANAGER`, geben Sie die Secrets-Referenz im `EnvironmentVariable` Wert an. Wenn keine Angabe gemacht wird, gilt der Standardwert `PLAINTEXT`.

> **Note**
>
> Wir raten dringend davon ab, Klartext-Umgebungsvariablen zum Speichern vertraulicher Werte, insbesondere AWS Anmeldeinformationen, zu verwenden. Wenn Sie die CodeBuild Konsole oder verwenden AWS CLI, werden Klartext-Umgebungsvariablen im Klartext angezeigt. Wenn es sich um vertrauliche Werte handelt, sollten Sie stattdessen den Typ `SECRETS_MANAGER` verwenden.

> **Note**
>
> Wenn Sie die Konfiguration`name`,`value`, und `type` für Ihre Umgebungsvariablen eingeben, insbesondere wenn die Umgebungsvariable die Syntax der CodePipeline Ausgabevariablen enthält, überschreiten Sie nicht die Grenze von 1000 Zeichen für das Wertefeld der Konfiguration. Ein Validierungsfehler wird zurückgegeben, wenn dieser Grenzwert überschritten wird.

Ein Beispiel für eine Aktionsdeklaration, die eine Umgebungsvariable zeigt, finden Sie unter. Konfigurationsparameter

> **Note**
>
> - Der `SECRETS_MANAGER` Typ wird nur für die Aktion „Befehle“ unterstützt.
> - Geheimnisse, auf die in der Aktion „Befehle“ verwiesen wird, werden in den Build-Protokollen ähnlich CodeBuild wie geschwärzt. Pipeline-Benutzer, die Bearbeitungszugriff auf die Pipeline haben, können jedoch potenziell immer noch auf diese geheimen Werte zugreifen, indem sie die Befehle ändern.
> - Um die verwenden zu können SecretsManager, müssen Sie Ihrer Pipeline-Servicerolle die folgenden Berechtigungen hinzufügen:
>
> ```
> {
>             "Effect": "Allow",
>             "Action": [
>                 "secretsmanager:GetSecretValue"
>             ],
>             "Resource": [
> ```

```
                "SECRET_ARN"
            ]
        }
```

Berechtigungen für Servicerollen: Befehle, Aktion

Um die Unterstützung von Befehlen zu erhalten, fügen Sie Ihrer Richtlinienerklärung Folgendes hinzu:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:iam::*:role/Service*",
                "arn:aws:iam::*:role/Service*"
            ]
        }
    ]
}
```

Aktionserklärung (Beispiel)

YAML

```
name: Commands_action
actionTypeId:
  category: Compute
  owner: AWS
  provider: Commands
  version: '1'
```

```
runOrder: 1
configuration: {}
commands:
- ls
- echo hello
- 'echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}'
outputArtifacts:
- name: BuildArtifact
  files:
  - **/
inputArtifacts:
- name: SourceArtifact
outputVariables:
- AWS_DEFAULT_REGION
region: us-east-1
namespace: compute
```

JSON

```
{
    "name": "Commands_action",
    "actionTypeId": {
        "category": "Compute",
        "owner": "AWS",
        "provider": "Commands",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {},
    "commands": [
        "ls",
        "echo hello",
        "echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}"
    ],
    "outputArtifacts": [
        {
            "name": "BuildArtifact",
            "files": [
                "**/"
            ]
        }
    ],
    "inputArtifacts": [
```

```
        {
            "name": "SourceArtifact"
        }
    ],
    "outputVariables": [
        "AWS_DEFAULT_REGION"
    ],
    "region": "us-east-1",
    "namespace": "compute"
}
```

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 Befehle mit Compute ausführt (Typ V2)— Dieses Tutorial enthält eine Beispielpipeline mit der Aktion „Befehle“.

AWS Device Farm Referenz zur Testaktion

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 Testframeworks, um Anwendungen auf bestimmten Geräten zu testen. Informationen zu den Typen von Testframeworks, die von der Device Farm-Aktion unterstützt werden, finden Sie unter Arbeiten mit Testtypen in AWS Device Farm.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Berechtigungen für Servicerollen: AWS Device Farm action
- 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-Projekt-ID.

Um Ihre Projekt-ID zu finden, wählen Sie in der Device 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 nachfolgenden URL`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

Befristet: Ja

Der Speicherort der Testspezifikationsdatei in Ihrem Eingabeartefakt. Dies ist für den Test im benutzerdefinierten Modus erforderlich.

DevicePoolArn

Erforderlich: Ja

Der ARN des Device Farm-Gerätepools.

Um die verfügbaren Gerätepool-ARNs für das Projekt abzurufen, einschließlich des ARN für Top-Geräte, verwenden Sie die AWS CLI, um den folgenden Befehl einzugeben:

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

TestType

Erforderlich: Ja

Gibt das unterstützte Testframework für Ihren Test an. Im Folgenden finden Sie eine Liste gültiger Werte für`TestType`:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTING
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTING
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- EINGEBAUTET_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 Leistungsdaten des Geräts wie CPU, FPS und Speicherleistung aufgezeichnet werden sollen.

RecordVideo

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob während des Tests ein Video aufgenommen werden soll.

RadioWifiEnabled

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob zu Wi-Fi 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 ausführen 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. Der Wert liegt 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 Durchführung 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 ein Timeout eintritt.

Breitengrad

Erforderlich: Nein

Der Breitengrad des Geräts, ausgedrückt in Grad im geographischen Koordinatensystem.

Längengrad

Erforderlich: Nein

Der Längengrad des Geräts, ausgedrückt in Grad des geographischen Koordinatensystems.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1`
- Beschreibung: Der Satz von Artefakten, der für die Testaktion zur Verfügung gestellt werden soll. Device Farm sucht nach den erstellten Anwendungs- und Testdefinitionen, die verwendet werden sollen.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten für diesen Aktionstyp nicht.

Berechtigungen für Servicerollen: AWS Device Farm action

Wenn die Aktion CodePipeline ausgeführt wird, erfordert die CodePipeline Servicerollenrichtlinie die folgenden Berechtigungen, die entsprechend auf den ARN der Pipeline-Ressource beschränkt sind, um den Zugriff mit den geringsten Rechten aufrechtzuerhalten. Fügen Sie Ihrer Richtlinienerklärung beispielsweise Folgendes hinzu:

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "devicefarm:ListProjects",
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
    ],
    "Resource": "resource_ARN"
},
```

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 weitere Beschreibung der von Device Farm unterstützten Testframeworks 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-Projekten helfen.
- Tutorial: Erstellen Sie eine Pipeline, die Ihre Android-App erstellt und testet mit AWS Device Farm— Dieses Tutorial enthält eine Beispiel-Buildspec-Datei 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-Quelle, die eine erstellte iOS-App mit Device Farm testet.

Referenz zur Elastic Beanstalk-Bereitstellungsaktion

Elastic Beanstalk ist eine interne Plattform, die für AWS die Bereitstellung und Skalierung von Webanwendungen verwendet wird. Sie verwenden eine Elastic Beanstalk-Aktion, um Anwendungscode in Ihrer Bereitstellungsumgebung bereitzustellen.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Berechtigungen für Servicerollen: Aktion ElasticBeanstalk bereitstellen
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `ElasticBeanstalk`
- Version: `1`

Konfigurationsparameter

ApplicationName

Erforderlich: Ja

Der Name der Anwendung, die Sie in Elastic Beanstalk erstellt haben.

EnvironmentName

Erforderlich: Ja

Der Name der Umgebung, die Sie in Elastic Beanstalk erstellt haben. Eine Umgebung ist eine Sammlung von AWS Ressourcen, auf denen eine Anwendungsversion ausgeführt wird. In jeder Umgebung wird jeweils nur eine Anwendungsversion ausgeführt. Sie können jedoch dieselbe Anwendungsversion oder verschiedene Anwendungsversionen in vielen Umgebungen gleichzeitig ausführen.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1`
- Beschreibung: Das Eingabeartefakt für die Aktion.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Berechtigungen für Servicerollen: Aktion **`ElasticBeanstalk`** bereitstellen

Für Elastic Beanstalk sind die folgenden Mindestberechtigungen erforderlich, um Pipelines mit einer `ElasticBeanstalk` Bereitstellungsaktion zu erstellen.

```
{
    "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 einschränken möchten. Weitere Informationen zum Erstellen einer Richtlinie, die den Zugriff nach den geringsten Rechten gewährt, finden Sie unter. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

Aktionsdeklaration

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: ElasticBeanstalk
      Version: '1'
    RunOrder: 1
    Configuration:
      ApplicationName: my-application
      EnvironmentName: my-environment
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
```

```
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "ElasticBeanstalk",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "ApplicationName": "my-application",
                "EnvironmentName": "my-environment"
            },
            "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.

- Bereitstellung einer Flask-Anwendung auf Elastic Beanstalk — Dieses Tutorial führt Sie anhand einer Flask-Beispielanwendung durch die Erstellung Ihrer Anwendungs- und Umgebungsressourcen in Elastic Beanstalk. Anschließend können Sie Ihre Pipeline mit einer Elastic Beanstalk-Bereitstellungsaktion erstellen, die Ihre Anwendung aus Ihrem Quell-Repository in Ihrer Elastic Beanstalk-Umgebung bereitstellt.

Referenz zum `InspectorScan` Aufrufen der Aktion durch Amazon Inspector

Amazon Inspector ist ein Schwachstellen-Management-Service, der Workloads automatisch erkennt und sie kontinuierlich auf Software-Schwachstellen und unbeabsichtigte Netzwerkbedrohungen überprüft. Die `InspectorScan` Aktion CodePipeline automatisiert die Erkennung und Behebung von Sicherheitslücken in Ihrem Open-Source-Code. Bei der Aktion handelt es sich um eine verwaltete Rechenaktion mit Funktionen für Sicherheitsscans. Du kannst sie InspectorScan mit Anwendungsquellcode in deinem Drittanbieter-Repository wie GitHub Bitbucket Cloud oder mit Bildern für Containeranwendungen verwenden. Bei deiner Aktion wirst du die von dir konfigurierten Sicherheitslücken und Warnmeldungen scannen und entsprechende Berichte erstellen.

> **Important**
>
> Diese Aktion verwendet CodePipeline Managed CodeBuild Compute, um Befehle in einer Build-Umgebung auszuführen. Für die Ausführung der Aktion fallen separate Gebühren in an AWS CodeBuild.

Themen

- ID des Aktionstyps
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Berechtigungen für Servicerollen: InspectorScan Aktion
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

ID des Aktionstyps

- Kategorie: `Invoke`
- Eigentümer: `AWS`
- Anbieter: `InspectorScan`

- Version: 1

Beispiel:

```
        {
            "Category": "Invoke",
            "Owner": "AWS",
            "Provider": "InspectorScan",
            "Version": "1"
        },
```

Konfigurationsparameter

InspectorRunMode

(Erforderlich) Die Zeichenfolge, die den Scanmodus angibt. Gültige Werte sind `SourceCodeScan | ECRImageScan`.

ECRRepositoryName

Der Name des Amazon ECR-Repositorys, in das das Bild übertragen wurde.

ImageTag

Das Tag, das für das Image verwendet wird.

Die Parameter für diese Aktion suchen nach dem Grad der Sicherheitslücke, den Sie angeben. Die folgenden Stufen für Schwellenwerte für Sicherheitslücken sind verfügbar:

CriticalThreshold

Die Anzahl der in Ihrer Quelle gefundenen Sicherheitslücken mit kritischem Schweregrad, bei deren CodePipeline Überschreitung die Aktion fehlschlagen sollte.

HighThreshold

Die Anzahl der in Ihrer Quelle gefundenen Sicherheitslücken mit hohem Schweregrad, bei deren CodePipeline Überschreitung die Aktion fehlschlagen sollte.

MediumThreshold

Die Anzahl der in Ihrer Quelle gefundenen Sicherheitslücken mit mittlerem Schweregrad, bei deren CodePipeline Überschreitung die Aktion fehlschlagen sollte.

LowThreshold

Die Anzahl der in Ihrer Quelle gefundenen Sicherheitslücken mit geringem Schweregrad, bei deren CodePipeline Überschreitung die Aktion fehlschlagen sollte.

Action name

Choose a name for your action

Scan

No more than 100 characters

Action provider

AWS InspectorScan

Region

US East (N. Virginia)

Input artifacts

Choose an input artifact for this action. Learn more

SourceArtifact

Defined by: Source

No more than 100 characters

Inspector Run mode

InspectorRunMode: SourceCodeScan or ECRImageScan.

Source Code Scan

SourceCodeScan: Scan the source code. Choose an input artifact for this run mode.

ECR Image Scan

ECRImageScan: Scan the ECR image.

Critical Threshold - *optional*

Threshold value for critical severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

High Threshold - *optional*

Threshold value for high severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Medium Threshold - *optional*

Threshold value for medium severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Low Threshold - *optional*

Threshold value for low severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Der Quellcode zum Scannen nach Sicherheitslücken. Wenn der Scan für ein ECR-Repository bestimmt ist, wird dieses Eingabeartefakt nicht benötigt.

Ausgabeartefakte

- Anzahl der Artefakte: 1
- Beschreibung: Informationen zur Sicherheitslücke Ihrer Quelle in Form einer Software Bill of Materials (SBOM) -Datei.

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 Variablenreferenz.

HighestScannedSeverity

Der vom Scan ausgegebene Schweregrad mit dem höchsten Schweregrad. Gültige Werte sind `medium | high | critical`.

Berechtigungen für Servicerollen: **`InspectorScan`** Aktion

Für die Unterstützung der `InspectorScan` Aktion fügen Sie Ihrer Richtlinienerklärung Folgendes hinzu:

```
{
        "Effect": "Allow",
        "Action": "inspector-scan:ScanSbom",
        "Resource": "*"
    },
    {
```

```
        "Effect": "Allow",
        "Action": [
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
        ],
        "Resource": "resource_ARN"
    },
```

Fügen Sie Ihrer Servicerolle außerdem die folgenden Berechtigungen hinzu, um CloudWatch Protokolle anzeigen zu können, sofern diese noch nicht für die Aktion Befehle hinzugefügt wurden.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Beschränken Sie die Berechtigungen auf die Ebene der Pipeline-Ressourcen, indem Sie die ressourcenbasierten Berechtigungen in der Richtlinienanweisung für die Servicerolle verwenden.

Aktionsdeklaration

YAML

```
name: Scan
actionTypeId:
  category: Invoke
  owner: AWS
  provider: InspectorScan
  version: '1'
runOrder: 1
configuration:
```

```
  InspectorRunMode: SourceCodeScan
outputArtifacts:
- name: output
inputArtifacts:
- name: SourceArtifact
region: us-east-1
```

JSON

```
{
                        "name": "Scan",
                        "actionTypeId": {
                            "category": "Invoke",
                            "owner": "AWS",
                            "provider": "InspectorScan",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "InspectorRunMode": "SourceCodeScan"
                        },
                        "outputArtifacts": [
                            {
                                "name": "output"
                            }
                        ],
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "region": "us-east-1"
                    },
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Weitere Informationen zu Amazon Inspector finden Sie im Amazon Inspector Inspector-Benutzerhandbuch.

AWS Lambda Aktionsreferenz aufrufen

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 Beispiel für ein Ereignis, das an eine Lambda-Aufruffunktion übergeben wird, 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 das JSON-Ereignis, das CodePipeline an Lambda gesendet wird, nicht, da dies dazu führen kann, dass Benutzeranmeldeinformationen in Logs protokolliert werden. CloudWatch Die CodePipeline Rolle verwendet ein JSON-Ereignis, um temporäre Anmeldeinformationen im Feld an Lambda zu übergeben. `artifactCredentials` 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 von der Lambda-Funktion als Eingabe verarbeitet werden kann.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0 to 5`
- Beschreibung: Der Satz von Artefakten, der der Lambda-Funktion zur Verfügung gestellt werden soll.

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 als Variablen alle Schlüssel-Wert-Paare, die im `outputVariables` Abschnitt der PutJobSuccessResult API-Anforderung enthalten sind.

Weitere Hinweise zu Variablen in CodePipeline finden Sie unter. Variablenreferenz

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 Positionen der Eingabe- und Ausgabeartefakte sowie 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 mit dem Job verknüpfte 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-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-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 Anmeldeinformationen für die AWS Sitzung, die für den Zugriff auf Eingabe- und Ausgabeartefakte im Amazon 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`: Die Zugriffsschlüssel-ID 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 CloudFormation -Artefakten für Pipelines finden Sie unter Verwenden von Parameter-Override-Funktionen mit CodePipeline Pipelines, Automatisieren der Bereitstellung von Lambda-based Anwendungen und Artifacts. AWS CloudFormation
- Ruft einen auf AWS Lambda Funktion in einer Pipeline in CodePipeline— Dieses Verfahren bietet ein Beispiel für eine Lambda-Funktion und zeigt Ihnen, wie Sie mithilfe der Konsole eine Pipeline mit einer Lambda-Aufrufaktion erstellen.

AWS OpsWorks Aktionsreferenz bereitstellen

Sie verwenden eine AWS OpsWorks Aktion zur Bereitstellung OpsWorks mithilfe Ihrer Pipeline.

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `OpsWorks`
- Version: `1`

Konfigurationsparameter

App

Erforderlich: Ja

Die OpsWorks App. Die App stellt den Code dar, den Sie bereitstellen und ausführen möchten.

Stack

Erforderlich: Ja

Der OpsWorks Stapel. Ein Stack ist ein Container für Ihre Anwendungsinfrastruktur.

Ebene

Erforderlich: Nein

Die OpsWorks Ebene. Eine Ebene spezifiziert die Konfiguration und die Ressourcen für eine Reihe von Instanzen.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1`
- Beschreibung: Dies ist das Eingabeartefakt für deine Aktion.

Ausgabeartefakte

- Anzahl der Artefakte: `0 to 1`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Berechtigungen für Servicerollen: AWS OpsWorks action

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"
},
```

Beispielaktionskonfiguration

YAML

```
Name: ActionName
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Version: 1
  Provider: OpsWorks
InputArtifacts:
  - Name: myInputArtifact
Configuration:
  Stack: my-stack
  App: my-app
```

JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "OpsWorks"
    },
    "InputArtifacts": [
        {
            "Name": "myInputArtifact"
        }
    ],
    "Configuration": {
```

```
        "Stack": "my-stack",
        "App": "my-app"
    }
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWS OpsWorks Benutzerhandbuch — Informationen zur Bereitstellung mit AWS OpsWorks finden Sie im AWS OpsWorks Benutzerhandbuch.

AWS Service Catalog Aktionsreferenz bereitstellen

Sie verwenden eine AWS Service Catalog Aktion, um Vorlagen mithilfe Ihrer Pipeline bereitzustellen. Dies sind Ressourcenvorlagen, die Sie in Service Catalog erstellt haben.

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `ServiceCatalog`
- Version: `1`

Konfigurationsparameter

TemplateFilePath

Erforderlich: Ja

Der Dateipfad für Ihre Ressourcenvorlage an Ihrem Quellspeicherort.

ProductVersionName

Erforderlich: Ja

Die Produktversion im Service Catalog.

ProductType

Erforderlich: Ja

Der Produkttyp im Service Catalog.

ProductId

Erforderlich: Ja

Die Produkt-ID im Servicekatalog.

ProductVersionDescription

Erforderlich: Nein

Die Beschreibung der Produktversion im Service Catalog.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1`
- Beschreibung: Dies ist das Eingabeartefakt für deine Aktion.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Berechtigungen für Servicerollen: Aktion „Servicekatalog“

Wenn Sie Service Catalog-Support benötigen, fügen Sie Ihrer Grundsatzerklärung 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"
}
```

Beispiele für Aktionskonfigurationen nach Typ der Konfigurationsdatei

Das folgende Beispiel zeigt eine gültige Konfiguration für eine Bereitstellungsaktion, die Service Catalog verwendet, für eine Pipeline, die in der Konsole ohne eine 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"
}
```

Beispielaktionskonfiguration

YAML

```
Name: ActionName
ActionTypeId:
  Category: Deploy
  Owner: AWS
```

```
  Version: 1
  Provider: ServiceCatalog
OutputArtifacts:
- Name: myOutputArtifact
Configuration:
  TemplateFilePath: S3_template.json
  ProductVersionName: devops S3 v2
  ProductType: CLOUD_FORMATION_TEMPLATE
  ProductVersionDescription: Product version description
  ProductId: prod-example123456
```

JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "ServiceCatalog"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "TemplateFilePath": "S3_template.json",
        "ProductVersionName": "devops S3 v2",
        "ProductType": "CLOUD_FORMATION_TEMPLATE",
        "ProductVersionDescription": "Product version description",
        "ProductId": "prod-example123456"
    }
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Service Catalog-Benutzerhandbuch — Informationen zu Ressourcen und Vorlagen in Service Catalog finden Sie im Service Catalog-Benutzerhandbuch.

- Tutorial: Erstellen Sie eine Pipeline, die im Service Catalog bereitgestellt wird— Dieses Tutorial zeigt Ihnen, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre Produktvorlage in Service Catalog bereitzustellen und Änderungen, die Sie in Ihrem Quell-Repository vorgenommen haben, zu übertragen.

AWS Step Functions Aktionsreferenz aufrufen

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
  ```

Input

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 Variablenreferenz.

StateMachineArn

Der ARN des Zustandsautomaten.

ExecutionArn

Der ARN der Ausführung des Zustandsautomaten. Nur Standard-Zustandsautomaten.

Berechtigungen für Servicerollen: **`StepFunctions`** Aktion

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"
},
```

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 mit 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 State-Machine-Ausführung 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 zur Regelstruktur

Dieser Abschnitt dient nur als Referenz für die Regelkonfiguration. Eine konzeptionelle Übersicht über die Pipeline-Struktur finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

Jeder Regelanbieter CodePipeline verwendet eine Reihe von erforderlichen und optionalen Konfigurationsfeldern in der Pipeline-Struktur. Dieser Abschnitt enthält die folgenden Referenzinformationen nach Regelanbietern:

- Gültige Werte für die `RuleType` Felder, die im Regelblock für die Pipelinestruktur enthalten sind, z. B. `Owner` und`Provider`.
- Beschreibungen und andere Referenzinformationen für die `Configuration` Parameter (erforderlich und optional), die im Abschnitt mit der Pipeline-Strukturregel enthalten sind.
- Gültiges Beispiel für die Konfiguration von JSON- und YAML-Regeln.

Referenzinformationen sind für die folgenden Regelanbieter verfügbar:

Themen

- CloudWatchAlarm
- CodeBuild Regel
- Befehle
- DeploymentWindow
- LambdaInvoke
- VariableCheck

CloudWatchAlarm

Wenn Sie eine Bedingung erstellen, können Sie die `CloudWatchAlarm` Regel hinzufügen. Dieser Abschnitt enthält eine Referenz für die Regelparameter. Weitere Informationen zu Regeln und Bedingungen finden Sie unterWie funktionieren die Bühnenbedingungen?.

Sie müssen bereits einen Alarm in Amazon CloudWatch als separate Ressource erstellt haben.

Themen

- Art der Regel

- Konfigurationsparameter
- Beispiel für eine Regelkonfiguration
- Weitere Informationen finden Sie auch unter

Art der Regel

- Kategorie: `Rule`
- Eigentümer: `AWS`
- Anbieter: `CloudWatchAlarm`
- Version: `1`

Konfigurationsparameter

AlarmName

Erforderlich: Ja

Der Name des CloudWatch Alarms. Dies ist eine separate Ressource, die in erstellt wurde CloudWatch.

AlarmStates

Erforderlich: Nein

Die gewünschten CloudWatch Alarmzustände, die von der Regel überwacht werden sollen. Gültige Werte sind ALARM, OK und INSUFFICIENT_DATA.

WaitTime

Erforderlich: Nein

Die Wartezeit in Minuten, bis eine Statusänderung möglich ist, bevor das Regelergebnis ausgeführt wird. Konfigurieren Sie beispielsweise 20 Minuten, um zu warten, bis sich der Alarmstatus auf OK ändert, bevor das Regelergebnis angewendet wird.

Beispiel für eine Regelkonfiguration

YAML

```
rules:
    - name: MyMonitorRule
      ruleTypeId:
        category: Rule
        owner: AWS
        provider: CloudWatchAlarm
        version: '1'
      configuration:
        AlarmName: CWAlarm
        WaitTime: '1'
      inputArtifacts: []
      region: us-east-1
```

JSON

```
                        "rules": [
                            {
                                "name": "MyMonitorRule",
                                "ruleTypeId": {
                                    "category": "Rule",
                                    "owner": "AWS",
                                    "provider": "CloudWatchAlarm",
                                    "version": "1"
                                },
                                "configuration": {
                                    "AlarmName": "CWAlarm",
                                    "WaitTime": "1"
                                },
                                "inputArtifacts": [],
                                "region": "us-east-1"
                            }
                        ]
                    }
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen können Ihnen bei der Arbeit mit dieser Regel helfen.

- Bedingungen bei Ausfall erstellen— Dieser Abschnitt enthält Schritte zum Erstellen eines Zustands bei Ausfall mit einer Alarmregel.

CodeBuild Regel

Wenn Sie eine Bedingung erstellen, können Sie die CodeBuild Regel hinzufügen. Dieser Abschnitt enthält eine Referenz zu den Regelparametern. Weitere Informationen zu Regeln und Bedingungen finden Sie unterWie funktionieren die Bühnenbedingungen?.

Sie können die CodeBuild Regel verwenden, um eine Bedingung zu erstellen, unter der die erfolgreiche Ausführung Ihres Build-Projekts die Regelkriterien erfüllt, z. B. wenn die Build-Ausführung für eine BeforeEntry-Bedingung erfolgreich war.

> **Note**
>
> Für BeforeEntry-Bedingungen, die mit dem Skip-Ergebnis konfiguriert sind, sind nur die folgenden Regeln verfügbar: und. `LambdaInvoke` `VariableCheck`

Themen

- Richtlinienberechtigungen für Servicerollen
- Regeltyp
- Konfigurationsparameter
- Beispiel für eine Regelkonfiguration
- Weitere Informationen finden Sie auch unter

Richtlinienberechtigungen für Servicerollen

Informationen zu den Berechtigungen für diese Regel finden Sie in der Richtlinienerklärung zu den CodePipeline Servicerollen wie folgt. Beschränken Sie die Berechtigungen auf die Ressourcenebene.

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource_ARN"
},
```

Regeltyp

- Kategorie: `Rule`
- 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 Buildspec-Datei und führt die Buildspec-Befehle in dem Verzeichnis aus, das die entpackte Version dieses Artefakts enthält.

Dieser Parameter ist erforderlich, wenn für eine CodeBuild -Aktion mehrere Eingabeartefakte angegeben werden. 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 Deklaration einer Aktion (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.

Beispiel für eine Regelkonfiguration

YAML

```
name: codebuild-rule
ruleTypeId:
  category: Rule
  owner: AWS
  provider: CodeBuild
  version: '1'
configuration:
  ProjectName: my-buildproject
  EnvironmentVariables: '[{"name":"VAR1","value":"variable","type":"PLAINTEXT"}]'
inputArtifacts:
- name: SourceArtifact
region: us-east-1
```

JSON

```
{
    "name": "codebuild-rule",
    "ruleTypeId": {
        "category": "Rule",
        "owner": "AWS",
        "provider": "CodeBuild",
        "version": "1"
```

```
    },
    "configuration": {
        "ProjectName": "my-buildproject"
    },
    "inputArtifacts": [
        {
            "name": "SourceArtifact",
            "EnvironmentVariables": "[{\"name\":\"VAR1\",\"value\":\"variable\",
\"type\":\"PLAINTEXT\"}]"
        }
    ],
    "region": "us-east-1"
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen können Ihnen bei der Arbeit mit dieser Regel helfen.

- Weitere Informationen zu Regeln und Bedingungen finden Sie unter RuleTypeIdBedingung und RuleExecutionim CodePipeline API-Leitfaden.

Befehle

Wenn Sie eine Bedingung erstellen, können Sie die `Commands` Regel hinzufügen. Dieser Abschnitt enthält eine Referenz zu den Regelparametern. Weitere Informationen zu Regeln und Bedingungen finden Sie unterWie funktionieren die Bühnenbedingungen?.

Sie können die `Commands` Regel verwenden, um eine Bedingung zu erstellen, unter der die erfolgreichen Befehle die Regelkriterien erfüllen, z. B. die Ausgabe und den Dateipfad für die Befehle, die bei einer BeforeEntry-Bedingung erfolgreich sind.

> **Note**
>
> Für BeforeEntry-Bedingungen, die mit dem Skip-Ergebnis konfiguriert sind, sind nur die folgenden Regeln verfügbar: und. `LambdaInvoke` `VariableCheck`

Themen

- Überlegungen zur Befehlsregel

- Richtlinienberechtigungen für die Servicerolle
- Art der Regel
- Konfigurationsparameter
- Beispiel für eine Regelkonfiguration
- Weitere Informationen finden Sie auch unter

Überlegungen zur Befehlsregel

Die folgenden Überlegungen gelten für die Befehlsregel.

- Die Befehlsregel verwendet CodeBuild Ressourcen, die der CodeBuild Aktion ähneln, erlaubt aber Befehle in der Shell-Umgebung in einer virtuellen Recheninstanz, ohne dass ein Build-Projekt zugeordnet oder erstellt werden muss.

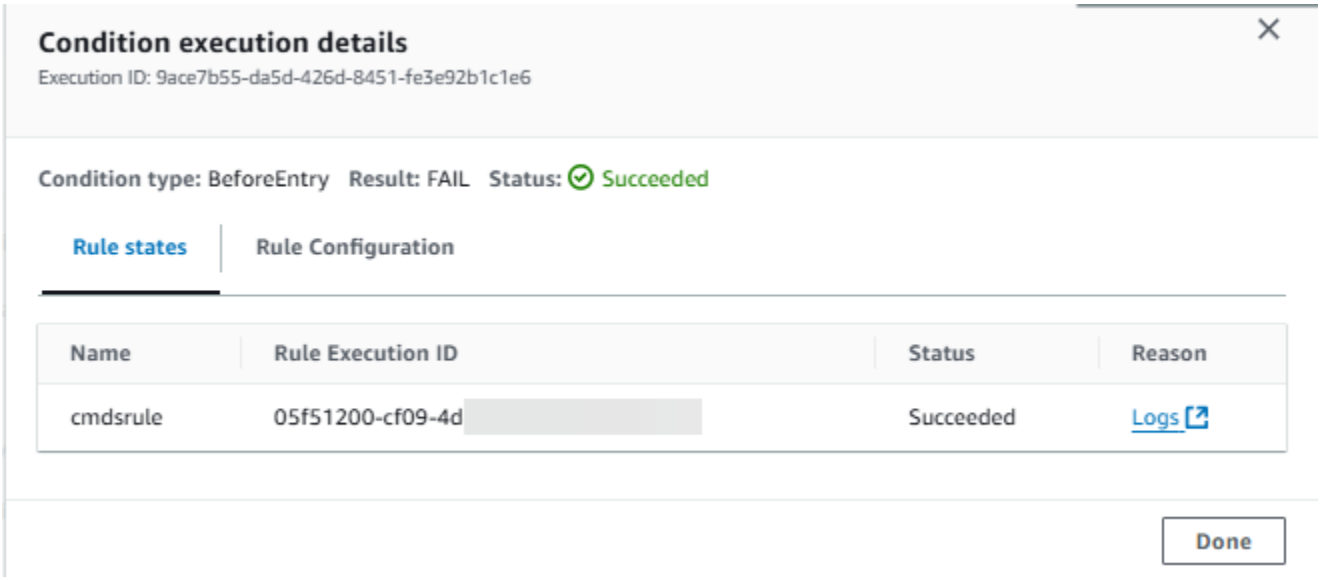
  > **Note**
  >
  > Für die Ausführung der Befehlsregel fallen separate Gebühren in an AWS CodeBuild.

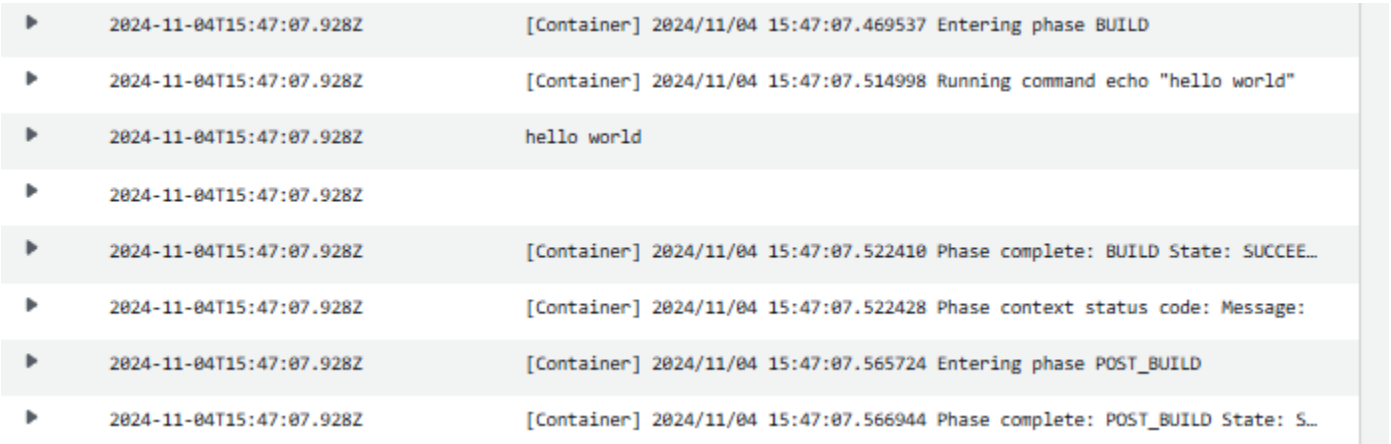
- Da die Befehlsregel in CodeBuild Ressourcen CodePipeline verbraucht, werden die von der Aktion ausgeführten Builds den Build-Limits für Ihr Konto in CodeBuild zugeordnet. Builds, die mit der Befehlsregel ausgeführt werden, werden auf die Limits für gleichzeitige Builds angerechnet, die für dieses Konto konfiguriert sind.
- Das Timeout für Builds mit der Befehlsregel beträgt 55 Minuten, je CodeBuild nach Builds.
- Die Recheninstanz verwendet eine isolierte Build-Umgebung in CodeBuild.

  > **Note**
  >
  > Da die isolierte Build-Umgebung auf Kontoebene verwendet wird, kann eine Instanz für eine andere Pipeline-Ausführung wiederverwendet werden.

- Alle Formate außer mehrzeiligen Formaten werden unterstützt. Bei der Eingabe von Befehlen müssen Sie das einzeilige Format verwenden.
- Für diese Regel CodePipeline wird die Pipelinedienst-Rolle übernommen und diese Rolle verwendet, um den Zugriff auf Ressourcen zur Laufzeit zu ermöglichen. Es wird empfohlen, die Servicerolle so zu konfigurieren, dass die Berechtigungen bis auf die Aktionsebene beschränkt sind.

- Die der CodePipeline Servicerolle hinzugefügten Berechtigungen werden unter detailliert beschrieben. Fügen Sie der CodePipeline Servicerolle Berechtigungen hinzu
- Die zum Anzeigen von Protokollen in der Konsole erforderlichen Berechtigungen finden Sie unterZum Anzeigen von Rechenprotokollen in der Konsole sind Berechtigungen erforderlich. Verwenden Sie in den folgenden Beispielbildschirmen den Link Logs, um Logs für eine erfolgreiche Befehlsregel in CloudWatch Logs aufzurufen.





- Im Gegensatz zu anderen Aktionen in CodePipeline legen Sie keine Felder in der Aktionskonfiguration fest. Sie legen die Felder für die Aktionskonfiguration außerhalb der Aktionskonfiguration fest.

Richtlinienberechtigungen für die Servicerolle

Wenn die Regel CodePipeline ausgeführt CodePipeline wird, wird eine Protokollgruppe mit dem Namen der Pipeline wie folgt erstellt. Auf diese Weise können Sie die Berechtigungen für die Protokollierung von Ressourcen mithilfe des Pipelinenamens einschränken.

```
/aws/codepipeline/MyPipelineName
```

Wenn Sie eine vorhandene Servicerolle verwenden, müssen Sie zur Verwendung der Aktion Befehle die folgenden Berechtigungen für die Servicerolle hinzufügen.

- Protokolle: CreateLogGroup
- Protokolle: CreateLogStream
- Protokolle: PutLogEvents

Beschränken Sie die Berechtigungen in der Richtlinie zur Servicerolle auf die Pipeline-Ebene, wie im folgenden Beispiel gezeigt.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

Um Protokolle in der Konsole mithilfe der Dialogseite mit den Aktionsdetails anzeigen zu können, muss der Konsolenrolle die Berechtigung zum Anzeigen von Protokollen hinzugefügt werden. Weitere Informationen finden Sie im Beispiel für die Richtlinie zu Konsolenberechtigungen unterZum Anzeigen von Rechenprotokollen in der Konsole sind Berechtigungen erforderlich.

Art der Regel

- Kategorie: `Rule`
- Eigentümer: `AWS`

- Anbieter: `Commands`
- Version: 1

Konfigurationsparameter

Befehle

Erforderlich: Ja

Sie können Shell-Befehle für die Ausführung der `Commands` Regel bereitstellen. In der Konsole werden Befehle in separaten Zeilen eingegeben. In der CLI werden Befehle als separate Zeichenfolgen eingegeben.

> **Note**
>
> Mehrzeilige Formate werden nicht unterstützt und führen zu einer Fehlermeldung. Für die Eingabe von Befehlen in das Befehlsfeld muss ein einzeiliges Format verwendet werden.

Die folgenden Details stellen die Standardrechnung dar, die für die Befehlsregel verwendet wird. Weitere Informationen finden Sie in der Referenz zu den Berechnungsmodi und Typen der Build-Umgebung im CodeBuild Benutzerhandbuch.

- CodeBuild Bild: aws/codebuild/amazonlinux 2-x86_64-standard:5.0
- Rechnertyp: Linux Small
- ComputeType-Wert der Umgebung: BUILD_ _SMALL GENERAL1
- Wert des Umgebungstyps: LINUX_CONTAINER

Beispiel für eine Regelkonfiguration

YAML

```
result: FAIL
rules:
- name: CommandsRule
  ruleTypeId:
    category: Rule
    owner: AWS
```

```
    provider: Commands
    version: '1'
  configuration: {}
  commands:
  - ls
  - printenv
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
```

JSON

```
{
    "result": "FAIL",
    "rules": [
        {
            "name": "CommandsRule",
            "ruleTypeId": {
                "category": "Rule",
                "owner": "AWS",
                "provider": "Commands",
                "version": "1"
            },
            "configuration": {},
            "commands": [
                "ls",
                "printenv"
            ],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1"
        }
    ]
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen können Ihnen bei der Arbeit mit dieser Regel helfen.

- Weitere Informationen zu Regeln und Bedingungen finden Sie unter RuleTypeIdBedingung und RuleExecutionim CodePipeline API-Leitfaden.

DeploymentWindow

Wenn Sie eine Bedingung erstellen, können Sie die `DeploymentWindow` Regel hinzufügen. Dieser Abschnitt enthält eine Referenz für die Regelparameter. Weitere Informationen zu Regeln und Bedingungen finden Sie unterWie funktionieren die Bühnenbedingungen?.

Themen

- Typ der Regel
- Konfigurationsparameter
- Beispiel für eine Regelkonfiguration
- Weitere Informationen finden Sie auch unter

Typ der Regel

- Kategorie: `Rule`
- Eigentümer: `AWS`
- Anbieter: `DeploymentWindow`
- Version: `1`

Konfigurationsparameter

Cron

Erforderlich: Ja

Der Ausdruck, der die Tage und Uhrzeiten definiert, an denen die Bereitstellung zulässig ist. Cron-Ausdrücke bestehen aus 6 erforderlichen Feldern und einem optionalen Feld, das durch Leerzeichen getrennt ist. Mit den Cron-Ausdrucksfeldern können Sie ein Zeitplanmuster mit einem Cron-Ausdruck wie folgt angeben.

Feldname	Zulässige Werte	Zulässige Sonderzeichen
Sekunden	N/A	*
Minuten	0-59	, - * /
Stunden	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Monat	1-12 oder JAN-DEC	, - * /
Day-of-Week	1-7 oder SUN-SAT	, - * ? / L #
Jahr (optional)	leer, 1970-2199	, - * /

- Das Zeichen '*' wird verwendet, um alle Werte anzugeben. Beispielsweise bedeutet „*“ im Minutenfeld „jede Minute“.
- Das '?' Das Zeichen ist für die Felder „Tag des Monats“ und „Wochentag“ zulässig. Es wird verwendet, um „keinen bestimmten Wert“ anzugeben. Dies ist nützlich, wenn Sie in einem der beiden Felder etwas angeben müssen, in dem anderen jedoch nicht.
- Das Zeichen '-' wird verwendet, um Bereiche anzugeben. Beispielsweise bedeutet „10-12" im Stundenfeld „die Stunden 10, 11 und 12".
- Das Zeichen ',' wird verwendet, um zusätzliche Werte anzugeben. Beispielsweise bedeutet „MO, MI, FR“ im Feld „Wochentag“ „die Tage Montag, Mittwoch und Freitag“.
- Das Zeichen '/' wird verwendet, um Inkremente anzugeben. Beispielsweise bedeutet "0/15" im Sekundenfeld „die Sekunden 0, 15, 30 und 45". Und "5/15" im Sekundenfeld bedeutet „die Sekunden 5, 20, 35 und 50". Die Angabe von '*' vor dem '/' entspricht der Angabe von 0. Dies ist der Wert, mit dem begonnen werden soll.
- Das Zeichen 'L' ist für die Felder „Tag des Monats“ und „Wochentag“ zulässig. Dieses Zeichen ist die Abkürzung für „letztes“, hat aber in jedem der beiden Felder eine andere Bedeutung. Beispielsweise bedeutet der Wert „L“ im Feld „Tag des Monats“ „der letzte Tag des Monats“ — Tag 31 für Januar, Tag 28 für Februar in Jahren, die keine Schaltjahre sind. Wenn er alleine im Feld Wochentag verwendet wird, bedeutet er einfach „7“ oder „SAT“. Wenn es jedoch im Feld Wochentag nach einem anderen Wert verwendet wird, bedeutet dies „der letzte <specified_day>Tag des Monats“ — zum Beispiel bedeutet „6L“ „der letzte Freitag des Monats“.

Sie können auch einen Abstand zum letzten Tag des Monats angeben, z. B. "L-3", was den drittletzten Tag des Kalendermonats bedeuten würde.

- Das Zeichen 'W' ist für das Feld „Tag des Monats“ zulässig. Dieses Zeichen wird verwendet, um den Wochentag (Monday-Friday) anzugeben, der dem angegebenen Tag am nächsten liegt. Wenn Sie beispielsweise „15W“ als Wert für das Feld „Tag des Monats“ angeben, lautet die Bedeutung: „der Wochentag, der dem 15. des Monats am nächsten ist“. Wenn also der 15. ein Samstag ist, wird der Trigger am Freitag, dem 14., ausgelöst. Wenn der 15. ein Sonntag ist, wird der Trigger am Montag, den 16. ausgelöst. Wenn der 15. ein Dienstag ist, wird er am Dienstag, den 15. ausgelöst.
- Die Zeichen 'L' und 'W' können für den Ausdruck für den Monatstag auch kombiniert werden, um 'LW' zu ergeben, was übersetzt „letzter Wochentag des Monats“ bedeutet.
- Das Zeichen '#' ist für das Feld für den Wochentag zulässig. Dieses Zeichen wird verwendet, um den „n-ten“ <specified_day>Tag des Monats anzugeben. Beispielsweise bedeutet der Wert „6 #3“ im Feld „Wochentag“ den dritten Freitag des Monats (Tag 6 = Freitag und "#3" = der dritte Freitag im Monat).
- Bei den zulässigen Zeichen und den Namen der Monate und Wochentage wird nicht zwischen Groß- und Kleinschreibung unterschieden.

TimeZone

Erforderlich: Nein

Die Zeitzone für das Bereitstellungsfenster. Der reguläre Ausdruck entspricht Mustern in den folgenden Formaten:

- Region/City Format. Der Wert entspricht einer Zeitzone im Format Region/City oder Region/City _City. Zum Beispiel `America/New_York` oder `Europe/Berlin`.
- UTC-Format. Der Wert entspricht der Zeichenfolge UTC, optional gefolgt von einem Offset im Format + HH:MM oder -HH:MM. Beispiel: `UTC`, `UTC+05:30` oder `UTC-03:00`. Dies ist das Standardformat, wenn der Parameter nicht anders festgelegt ist.
- Abkürzungsformat. Der Wert entspricht einer 3- bis 5-stelligen Abkürzung für eine Zeitzone. Zum Beispiel `EST` oder `IST`.

  Eine Tabelle mit gültigen TimeZone ID-Werten finden Sie unter https://docs.oracle.com/middleware/1221/wcs/tag-ref/MISC/TimeZones.html. Beachten Sie, dass bestimmte Abkürzungen doppelt auftreten, z. B. CST für Central Standard Time, China Standard Time und Cuba Standard Time.

Beispiel für eine Regelkonfiguration

YAML

```
- name: MyDeploymentRule
  ruleTypeId:
    category: Rule
    owner: AWS
    provider: DeploymentWindow
    version: '1'
  configuration:
    Cron: 0 0 9-17 ? * MON-FRI *
    TimeZone: PST
  inputArtifacts: []
  region: us-east-1
```

JSON

```
[
                    {
                        "name": "MyDeploymentRule",
                        "ruleTypeId": {
                            "category": "Rule",
                            "owner": "AWS",
                            "provider": "DeploymentWindow",
                            "version": "1"
                        },
                        "configuration": {
                            "Cron": "0 0 9-17 ? * MON-FRI *",
                            "TimeZone": "PST"
                        },
                        "inputArtifacts": [],
                        "region": "us-east-1"
                    }
                ]
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen können Ihnen bei der Arbeit mit dieser Regel helfen.

- Bedingungen bei Erfolg erstellen— Dieser Abschnitt enthält Schritte zum Erstellen der Bedingung „Bei Erfolg“ mit einer Regel für das Bereitstellungsfenster.
- Weitere Informationen zu Regeln und Bedingungen finden Sie unter RuleTypeId Bedingung und RuleExecution im CodePipeline API-Handbuch.

LambdaInvoke

Wenn Sie eine Bedingung erstellen, können Sie die `LambdaInvoke` Regel hinzufügen. Dieser Abschnitt enthält eine Referenz für die Regelparameter. Weitere Informationen zu Regeln und Bedingungen finden Sie unterWie funktionieren die Bühnenbedingungen?.

Sie müssen bereits eine Funktion in Lambda als separate Ressource erstellt haben.

Themen

- Regeltyp
- Konfigurationsparameter
- Beispiel für eine Regelkonfiguration
- Weitere Informationen finden Sie auch unter

Regeltyp

- Kategorie: `Rule`
- Eigentümer: `AWS`
- Anbieter: `LambdaInvoke`
- Version: `1`

Konfigurationsparameter

FunctionName

Erforderlich: Ja

Der Name der Lambda-Funktion.

UserParameters

Erforderlich: Nein

Dies sind Parameter, die als Eingabe für die Funktion im Schlüssel-Wert-Paarformat bereitgestellt werden.

Beispiel für eine Regelkonfiguration

YAML

```
- name: MyLambdaRule
  ruleTypeId:
    category: Rule
    owner: AWS
    provider: LambdaInvoke
    version: '1'
  configuration:
    FunctionName: my-function
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
```

JSON

```
[
    {
        "name": "MyLambdaRule",
        "ruleTypeId": {
            "category": "Rule",
            "owner": "AWS",
            "provider": "LambdaInvoke",
            "version": "1"
        },
        "configuration": {
            "FunctionName": "my-function"
        },
        "inputArtifacts": [
            {
                "name": "SourceArtifact"
            }
        ],
        "region": "us-east-1"
    }
]
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen können Ihnen bei der Arbeit mit dieser Regel helfen.

- Bedingungen bei Ausfall erstellen— Dieser Abschnitt enthält Schritte zum Erstellen eines Zustands bei Ausfall mit einer Alarmregel.

VariableCheck

Wenn Sie eine Bedingung erstellen, können Sie die `VariableCheck` Regel hinzufügen. Dieser Abschnitt enthält eine Referenz zu den Regelparametern. Weitere Informationen zu Regeln und Bedingungen finden Sie unterWie funktionieren die Bühnenbedingungen?.

Sie können die `VariableCheck` Regel verwenden, um eine Bedingung zu erstellen, bei der die Ausgabevariable mit einem angegebenen Ausdruck verglichen wird. Die Regel besteht die Prüfung, wenn der Variablenwert die Regelkriterien erfüllt, z. B. wenn der Wert gleich oder größer als eine angegebene Ausgabevariable ist.

Themen

- Art der Regel
- Konfigurationsparameter
- Beispiel für eine Regelkonfiguration
- Weitere Informationen finden Sie auch unter

Art der Regel

- Kategorie: `Rule`
- Eigentümer: `AWS`
- Anbieter: `VariableCheck`
- Version: `1`

Konfigurationsparameter

Operator

Erforderlich: Ja

Der Operator, der angibt, welche Operation für die Variablenprüfung ausgeführt werden soll.

Im folgenden Beispiel wird die Ausgabevariable für den Repository-Namen daraufhin überprüft, ob sie gleich ist`MyDemoRepo`.

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

Die folgenden Operatoren stehen zur Verfügung, um einen Ausdruck wie folgt zu erstellen.

- Gleich — Wählen Sie diesen Operator, um zu überprüfen, ob die Variable dem Zeichenkettenwert entspricht.

  CLI-Parameter: `EQ`
- Enthält — Wählen Sie diesen Operator, um zu überprüfen, ob die Variable den Zeichenkettenwert als Teilstring enthält.

  CLI-Parameter: `CONTAINS`
- Entspricht — Wählen Sie diesen Operator, um zu überprüfen, ob die Variable einem bestimmten Regex-Ausdruck als Zeichenkettenwert entspricht.

  Alle regulären Ausdrücke CodePipeline entsprechen der Java-Regex-Syntax. Eine umfassende Beschreibung der Java Regex-Syntax und ihrer Konstrukte finden Sie unter java.util.regex.Pattern.

  CLI-Parameter: `MATCHES`
- Entspricht nicht — Wählen Sie diesen Operator, um zu überprüfen, ob die Variable nicht dem Zeichenkettenwert entspricht.

  CLI-Parameter: `NE`

Variable

Erforderlich: Ja

Die zu überprüfenden Pipeline-Variablen.

Wert

Erforderlich: Ja

Der Wert für den Ausdruck, gegen den geprüft werden soll.

Im folgenden Beispiel wird die Ausgabevariable für den Repository-Namen daraufhin überprüft, ob sie gleich ist`MyDemoRepo`.

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

Im folgenden JSON-Beispiel werden zwei separate Regeln definiert, eine für eine `EQ` (equals) - Anweisung, die das Repository überprüft, und den Branch-Namen, formatiert als # {. SourceVariables RepositoryName} und eine davon überprüft `CONTAINS` die als # {formatierte Ausgabevariable der Commit-Nachricht. SourceVariables CommitMessage} gegen den angegebenen Wert „update“.

```
"beforeEntry": {
                  "conditions": [
                      {
                          "result": "FAIL",
                          "rules": [
                              {
                                  "name": "MyVarCheckRule",
                                  "ruleTypeId": {
                                      "category": "Rule",
                                      "owner": "AWS",
                                      "provider": "VariableCheck",
                                      "version": "1"
                                  },
                                  "configuration": {
                                      "Operator": "EQ",
                                      "Value": "MyDemoRepo",
                                      "Variable": "#{SourceVariables.RepositoryName}"
                                  },
                                  "inputArtifacts": [],
                                  "region": "us-east-1"
                              },
                              {
                                  "name": "MyVarCheckRuleContains",
                                  "ruleTypeId": {
                                      "category": "Rule",
```

```
                                    "owner": "AWS",
                                    "provider": "VariableCheck",
                                    "version": "1"
                                },
                                "configuration": {
                                    "Operator": "CONTAINS",
                                    "Value": "update",
                                    "Variable": "#{SourceVariables.CommitMessage}"
                                },
                                "inputArtifacts": [],
                                "region": "us-east-1"
                            }
                        ]
                    }
                ]
            }
        }
    ],
```

Beispiel für eine Regelkonfiguration

YAML

```
- name: MyVariableCheck
  ruleTypeId:
    category: Rule
    owner: AWS
    provider: VariableCheck
    version: '1'
  configuration:
    Variable: "#{SourceVariables.RepositoryName}"
    Value: MyDemoRepo
    Operator: EQ
  inputArtifacts: []
  region: us-west-2
```

JSON

```
"rules": [
    {
        "name": "MyVariableCheck",
        "ruleTypeId": {
            "category": "Rule",
```

```
            "owner": "AWS",
            "provider": "VariableCheck",
            "version": "1"
        },
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
        "inputArtifacts": [],
        "region": "us-west-2"
    }
]
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen können Ihnen bei der Arbeit mit dieser Regel helfen.

- Tutorial: Erstellen Sie eine Variablenprüfregel für eine Pipeline als Eingabebedingung— Dieser Abschnitt enthält ein Tutorial mit Schritten zum Erstellen einer Bedingung bei der Eingabe mit einer Regel zur Überprüfung von Variablen.
- Variablenreferenz— Dieser Abschnitt enthält Referenzinformationen und Beispiele für Pipeline-Variablen.
- Weitere Informationen zu Regeln und Bedingungen finden Sie unter RuleTypeIdBedingung und RuleExecutionim CodePipeline API-Leitfaden.

Referenz zum Integrationsmodell

Es gibt mehrere vorgefertigte Integrationen für Dienste von Drittanbietern, mit denen bestehende Kundentools in den Pipeline-Release-Prozess integriert werden können. Partner oder Drittanbieter verwenden ein Integrationsmodell, um Aktionstypen zur Verwendung in zu implementieren. CodePipeline

Verwenden Sie diese Referenz, wenn Sie Aktionstypen planen oder mit ihnen arbeiten, die mit einem unterstützten Integrationsmodell in verwaltet werden CodePipeline.

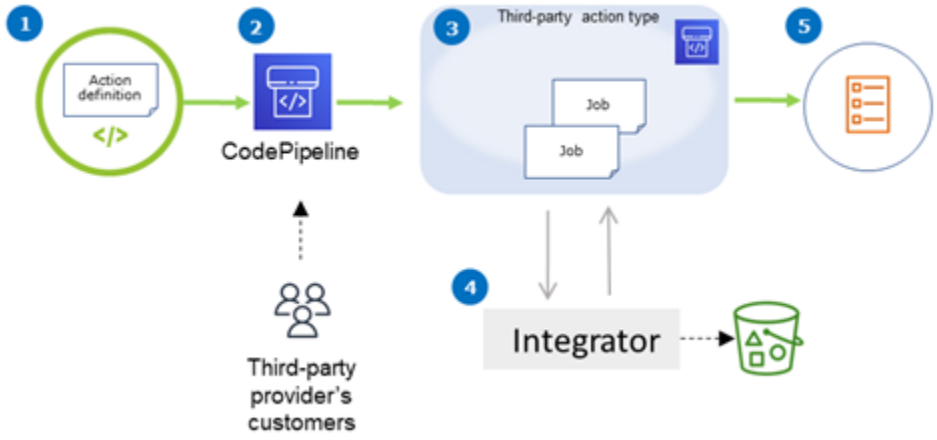
Um Ihren Drittanbieter-Aktionstyp als Partnerintegration mit zu zertifizieren CodePipeline, verweisen Sie auf das AWS Partnernetzwerk (APN). Diese Informationen sind eine Ergänzung zur Referenz. AWS CLI

Themen

- Wie funktionieren Aktionstypen von Drittanbietern mit dem Integrator
- Konzepte
- Unterstützte Integrationsmodelle
- Lambda-Integrationsmodell
- Integrationsmodell für Berufstätige

Wie funktionieren Aktionstypen von Drittanbietern mit dem Integrator

Sie können Aktionstypen von Drittanbietern zu Kunden-Pipelines hinzufügen, um Aufgaben im Zusammenhang mit Kundenressourcen abzuschließen. 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 die Jobs werden in die Warteschlange gestellt. CodePipeline Wenn der Job fertig ist CodePipeline, wird eine Job-Anfrage gesendet.
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 eine success/failure Ausgabe mit einem Auftragsergebnis und einem Fortsetzungstoken zurück (der Job-Worker verwendet success/failure APIs oder die Lambda-Funktion sendet die success/failure Ausgabe).

Hinweise 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 gekennzeichnet. `Provider` `Version` Beispiel:

```
        {

            "Category": "Deploy",
            "Owner": "AWS",
            "Provider": "CodeDeploy",
            "Version": "1"
        },
```

Alle Aktionen desselben Typs haben dieselbe Implementierung.

Action

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 input/output Artefakte 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 Aktionsausführungskonfiguration, die den Integrationstyp definiert, der von einem Aktionstyp verwendet wird. Gültige Werte sind `JobWorker` und `Lambda`.

Integration

Beschreibt eine Software, die von einem Integrator zur Implementierung eines Aktionstyps ausgeführt wird. CodePipeline unterstützt zwei Integrationstypen, die den beiden unterstützten Action Engines `JobWorker` und `Lambda` entsprechen.

Integrator

Die Person, der die Implementierung eines Aktionstyps gehört.

Aufgabe

Ein Stück Arbeit mit Pipeline und Kundenkontext zur Ausführung einer Integration. Eine Aktionsausführung besteht aus einem oder mehreren Aufträgen.

Beruf: Arbeiter

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 zu arbeiten. CodePipeline Das Lambda-Integrationsmodell verwendet eine Lambda-Funktion, um Jobanfragen zu verarbeiten, wenn Ihre Aktion ausgeführt wird.
- Jobworker-Integrationsmodell: Das Job Worker-Integrationsmodell ist das zuvor verwendete Modell für Integrationen von Drittanbietern. Das Jobworker-Integrationsmodell verwendet einen Job Worker, der so konfiguriert ist, dass er die CodePipeline APIs kontaktiert, um Jobanfragen zu verarbeiten, wenn Ihre Aktion ausgeführt wird.

Zum Vergleich werden in der folgenden Tabelle die Funktionen der beiden Modelle beschrieben:

	Lambda-Integrationsmodell	Integrationsmodell für Berufstätige
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 stattdessen, bis die nächste Jobanfrage eingegangen ist.	Der Integrator schreibt die Integration als Jobworker, der ständig nach verfügbaren Jobs in den Pipelines des Kunden fragt. Der Jobworker führt dann den Job aus und sendet das Auftragsergebnis mithilfe von APIs an CodePipeline ihn zurück. CodePipeline
Infrastruktur	AWS Lambda	Stellen Sie Job Worker-Code in der Infrastruktur des Integrators bereit, z. B. in Amazon EC2-Instances.

	Lambda-Integrationsmodell	Integrationsmodell für Berufstätige
Aufwand für die Entwicklung	Die Integration enthält nur die Geschäftslogik.	Die Integration muss nicht nur die Geschäftslogik enthalten, sondern auch mit CodePipeline APIs interagieren.
Aufwand der Operation	Geringerer operativer Aufwand, da die Infrastruktur nur aus AWS Ressourcen besteht.	Höherer operativer Aufwand, da der Mitarbeiter 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 dessen Abschluss ein Ergebnis zurückgeben müssen. Wir empfehlen nicht, dass der Integrator weiter darauf wartet, 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 eingeht, um den Job zu überprüfen, bis er abgeschlossen ist.	Jobs (hours/days) mit sehr langer Laufzeit können mit diesem Modell aufrechterhalten werden.

Lambda-Integrationsmodell

Das unterstützte Lambda-Integrationsmodell umfasst die Erstellung der Lambda-Funktion und die Definition der Ausgabe für den Aktionstyp des 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. Angesichts des Kontextes des Kunden und der Pipeline möchten Sie beispielsweise möglicherweise mit einer Erweiterung Ihres Service für den Kunden beginnen.

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 zu einem Job, die eine Integration verwendet, um den Job abzuschließen.
  - Enthält eine Karte mit den folgenden Elementen:
    - `actionConfiguration`:
      - Die Konfigurationsdaten für die Aktion. Die Felder für die Aktionskonfiguration sind eine Zuordnung von Schlüssel-Wert-Paaren, die Ihr Kunde zur Eingabe von Werten benötigt. 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 von Informationen über die Ausgabe einer Aktion.
  - Inhalt: Liste des Datentyps`Artifact`, optional vorhanden
- `actionCredentials`:
  - Stellt ein Objekt mit AWS Sitzungsanmeldeinformationen dar. Diese Anmeldeinformationen sind 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 verfügen außerdem über dieselben Berechtigungen wie die angegebene Vorlage für Richtlinienerklärungen in der Definitionsdatei für den Aktionstyp.
  - 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 Art des Objekts kann dies eine Commit-ID (GitHub) oder eine Revisions-ID (Amazon S3) sein.
    - Typ: Zeichenfolge, optional vorhanden
  - `location`:
    - Die Position eines Artefakts.
    - Inhalt: Typ des Datentyps`ArtifactLocation`, optional vorhanden
- `ArtifactLocation`:
  - `type`:
    - Die Art des Artefakts an der Position.
    - 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 Jobworker-Ressource des Integrators muss bei Erfolg, Misserfolg oder Fortsetzung 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 abgeschlossen ist.
  - `Continue`: Zeigt an, dass ein Job erfolgreich war 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 abgeschlossen ist.

- `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 an CodePipeline zurückgegeben wird.
  - Optional. Erforderlich, wenn das `Fail` Ergebnis
  - Muss Null sein, `result` wenn `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 `Success` oder `result` ist`Fail`.
  - Eigenschaften:
    - `State`: Ein Hash des Zustands, der übergeben werden soll.
- `status`: Status der Aktionsausführung.
  - Optional.
  - Eigenschaften:
    - `ExternalExecutionId`: Eine optionale externe Ausführungs-ID oder Commit-ID, die dem Job zugeordnet werden soll.

    - `Summary`: Eine optionale Zusammenfassung dessen, was passiert ist. In Fehlerszenarien wird dies die Fehlermeldung, die dem Benutzer angezeigt wird.
- `outputVariables`: Eine Reihe von key/value Paaren, die an die nächste Aktionsausführung übergeben werden.
    - Optional.
    - Muss Null sein, wenn `Continue` oder `result` ist`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 des Ergebnisses Ihrer Lambda-Funktion. Es ist eine Möglichkeit, den Jobstatus an den Job zu übergeben CodePipeline und darauf hinzuweisen, dass der Job fortgesetzt werden muss. Wenn beispielsweise ein Integrator einen Build für den Kunden auf seiner Ressource gestartet hat, wartet er nicht darauf, dass der Build abgeschlossen ist, sondern weist darauf hin, CodePipeline dass kein Endergebnis vorliegt, 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 laufen muss, können Sie Fortsetzungstoken verwenden.

Das CodePipeline Team ruft den Integrator nach 30 Sekunden mit demselben `continuation` Token in seiner Nutzlast auf, damit es überprüfen kann, ob der Vorgang abgeschlossen ist. Wenn der Build abgeschlossen ist, gibt der Integrator das success/fail Endergebnis 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 die Berechtigungen zu geben, ihn mithilfe des Dienstprinzips aufzurufen:. CodePipeline `codepipeline.amazonaws.com` Sie können Berechtigungen mithilfe der Befehlszeile CloudFormation oder über die Befehlszeile hinzufügen. Ein Beispiel finden Sie unter Mit Aktionstypen arbeiten.

Integrationsmodell für Berufstätige

Nachdem Sie Ihren allgemeinen Arbeitsablauf entworfen haben, können Sie Ihren Job Worker erstellen. Obwohl die Besonderheiten der Drittanbieter-Aktion bestimmen, was für den Job-Worker benötigt wird, bieten die meisten Auftragsarbeiter für Drittanbieter-Aktionen die folgenden Funktionen:

- Abfrage nach Aufträgen von CodePipeline Using. `PollForThirdPartyJobs`
- Bestätigen von Aufträgen und Zurückgeben der Ergebnisse zur CodePipeline Verwendung von `AcknowledgeThirdPartyJob``PutThirdPartyJobSuccessResult`, und. `PutThirdPartyJobFailureResult`
- Artefakte werden abgerufen, and/or indem Artefakte für die Pipeline in den Amazon S3-Bucket eingegeben werden. Um Artefakte aus dem Amazon S3-Bucket herunterzuladen, müssen Sie einen Amazon S3-Client erstellen, der Signature Version 4-Signing (Sig V4) verwendet. Sig V4 ist erforderlich für AWS KMS.

  Um Artefakte in den Amazon S3-Bucket hochzuladen, müssen Sie auch die Amazon `PutObject` S3-Anfrage so konfigurieren, dass sie die Verschlüsselung über 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 Mitarbeiter 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 den, 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 Specifying the AWS Key Management Service in Amazon S3 Using the AWS SDKs. Weitere Informationen zum Amazon S3-Bucket für finden Sie CodePipeline unterCodePipeline Konzepte.

Wählen und Konfigurieren einer Strategie für die Berechtigungsverwaltung Ihres Auftragsworkers

Um einen Job Worker für Ihre Drittanbieter-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 Job Worker benötigen, hinzuzufügen, indem Sie Amazon EC2-Instances mit einer Instance-Rolle AWS Identity and Access Management (IAM) erstellen, mit der Sie die Ressourcen, die Sie für Ihre Integration benötigen, problemlos 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. Informationen finden Sie unter Amazon EC2 — Virtual Server Hosting. Informationen zum Einrichten einer Amazon EC2-Instance finden Sie unter Erste Schritte mit Amazon EC2 Linux-Instances.

Eine weitere Strategie, die Sie in Betracht ziehen sollten, ist die Verwendung von Identity Federation mit IAM, um Ihr vorhandenes System und Ihre Ressourcen für Identitätsanbieter zu integrieren. Diese Strategie ist nützlich, wenn Sie bereits über einen Anbieter für Unternehmensidentitäten verfügen oder bereits so konfiguriert sind, dass Benutzer unterstützt werden, die Web-Identitätsanbieter verwenden. Mit Identity Federation 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 Anleitung unter Eine Rolle für einen externen Identitätsanbieter (Verbund) erstellen im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter Eine Rolle für einen IAM-Benutzer erstellen 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 zur Verwendung mit Ihrem externen Job-Mitarbeiter erstellen könnten. Diese Richtlinie dient lediglich als Beispiel und wird unverändert bereitgestellt.

JSON

```
{
  "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 Eine Pipeline, Phasen und Aktionen erstellen.

AWS CodePipeline Jobworker für Containeraktionen, wie z. B. eine Amazon ECR-Quellaktion oder Amazon ECS-Bereitstellungsaktionen, verwenden Definitionsdateien, um den Bild-URI und den Container-Namen der Aufgabendefinition zuzuordnen. Jede Definitionsdatei ist eine JSON-formatted Datei, die vom Aktionsanbieter wie folgt verwendet wird:

- Amazon ECS-Standardbereitstellungen erfordern eine `imagedefinitions.json` Datei als Eingabe für die Bereitstellungsaktion. Ein Tutorial, das die Amazon ECS-Standardbereitstellungsaktion in verwendet CodePipeline, finden Sie unterTutorial: Amazon ECS-Standardbereitstellung mit CodePipeline. Ein weiteres Beispiel-Tutorial, das die Amazon ECS-Standardbereitstellungsaktion CodePipeline zusammen mit der ECRBuildAndPublish Aktion verwendet, finden Sie unterTutorial: Erstellen Sie ein Docker-Image und übertragen Sie es auf Amazon ECR mit CodePipeline (Typ V2).
- Amazon blue/green ECS-Bereitstellungen benötigen eine `imageDetail.json` Datei als Eingabe für die Bereitstellungsaktion. Ein Tutorial mit einer blue/green Beispielbereitstellung finden Sie unterTutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-CodeDeploy.
  - Amazon ECR-Quellaktionen generieren eine `imageDetail.json`-Datei, die als Ausgabe aus der Quellaktion bereitgestellt wird.

Themen

- imagedefinitions.json-Datei für Amazon ECS-Standardbereitstellungsaktionen
- Detail.json Image-Datei für Amazon blue/green ECS-Bereitstellungsaktionen

imagedefinitions.json-Datei für Amazon ECS-Standardbereitstellungsaktionen

Ein Image-Definitionsdokument ist eine JSON-Datei, die den Namen Ihres Amazon ECS-Containers sowie das Bild und den Tag beschreibt. Wenn Sie containerbasierte Anwendungen bereitstellen, müssen Sie eine Image-Definitionsdatei generieren, um dem CodePipeline Jobworker den Amazon

ECS-Container und die Bildidentifikation zur Verfügung zu stellen, die er aus dem Image-Repository abrufen kann, z. B. Amazon ECR.

> **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 kodiert sein. UTF-8
- 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, wie z. B. Ihr CodeCommit Repository, hochgeladen oder als eingebautes 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.

Key (Schlüssel)	Value (Wert)
Name	*container_name*
imageUri	*imageUri*

> **Note**
>
> Das Namensfeld wird für den Namen des Container-Images verwendet, was den Namen für das Docker-Image bedeutet.

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:

    a. Wenn Ihre Pipeline so erstellt wurde, dass die Build-Phase übersprungen wird, 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-Bucket handelt, denken Sie daran, die JSON-Datei zu komprimieren.

b. Wenn Ihre Pipeline über eine Build-Phase verfügt, fügen Sie Ihrer Buildspec-Datei einen Befehl hinzu, der die Image-Definitionsdatei während der Build-Phase in Ihrem 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.

Eine schrittweise Anleitung zum Erstellen einer Pipeline, die Amazon ECS als Bereitstellungsanbieter verwendet, finden Sie unter Tutorial: Continuous Deployment with. CodePipeline

Detail.json Image-Datei für Amazon blue/green ECS-Bereitstellungsaktionen

Ein `imageDetail.json` Dokument ist eine JSON-Datei, die Ihre Amazon ECS-Image-URI beschreibt. Wenn Sie containerbasierte Anwendungen für eine blue/green Bereitstellung bereitstellen, müssen Sie die `imageDetail.json` Datei generieren, um Amazon ECS und dem CodeDeploy Job-Worker die Bildidentifikation zur Verfügung zu stellen, die sie aus dem Image-Repository abrufen müssen, 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 Referenz für Bereitstellungsaktionen in CodeDeploy Blaugrün

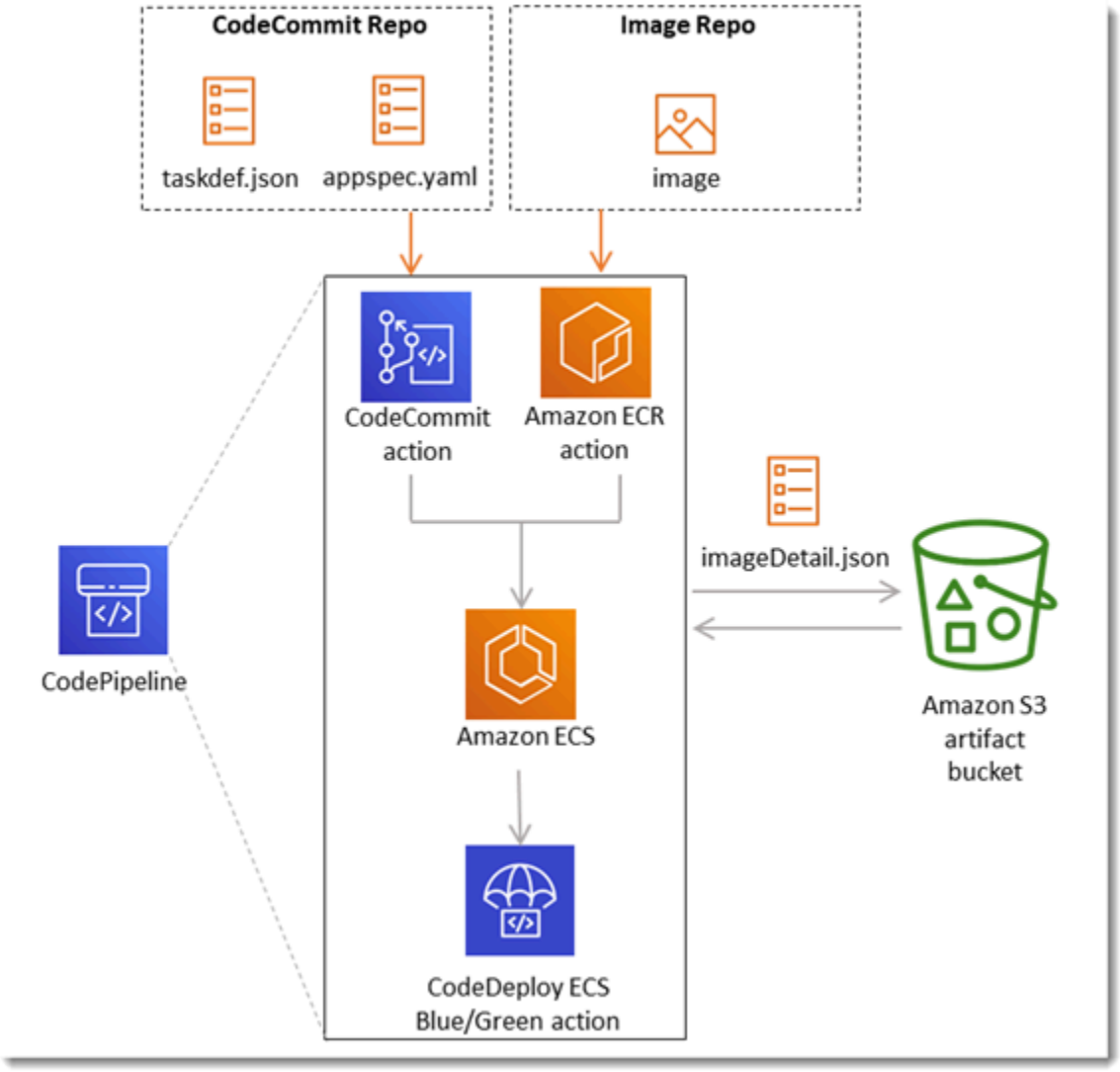
Sie müssen die Datei `imageDetail.json` als Quell- oder Build-Artefakt erstellen, damit sie ein Eingabeartefakt für die Bereitstellungsaktion ist. Sie können eine dieser Methoden verwenden, um die `imageDetail.json`-Datei in der Pipeline bereitzustellen:

- Fügen Sie die `imageDetail.json` Datei in Ihren Quellspeicherort ein, sodass sie in der Pipeline als Eingabe für Ihre Amazon blue/green ECS-Bereitstellungsaktion bereitgestellt wird.

  > **Note**
  >
  > Wenn Ihr Quell-Repository ein Amazon S3-Bucket ist, 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-Quellstufe umfasst, finden Sie unter. Tutorial: Erstellen Sie eine Pipeline mit einer Amazon ECR-Quelle und Bereitstellung ECS-to-CodeDeploy



Die `imageDetail.json`-Datei stellt den Abbild-URI bereit. Er muss mit dem Schlüssel-Wert-Paar konstruiert werden.

Key (Schlüssel)	Value (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 an das Bild-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 Containernamen wie folgt der Amazon ECS-Aufgabendefinition 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 im Rahmen der Planung der containerbasierten blue/green Anwendungsbereitstellung für Ihre Pipeline gegebenenfalls die Quellphase und die Buildphase.
2. Wählen Sie eine der folgenden Optionen:

   a. Wenn Ihre Pipeline die Build-Phase 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 Buildspec-Datei einen Befehl hinzu, der die Image-Definitionsdatei während der Build-Phase in Ihrem 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
      ```

Variablenreferenz

Dieser Abschnitt dient nur als Referenz. Informationen zum Erstellen von Variablen finden Sie unter Arbeiten mit Variablen.

Mithilfe von Variablen können Sie Ihre Pipeline-Aktionen mit Werten konfigurieren, die zum Zeitpunkt der Pipeline-Ausführung oder der Aktionsausführung festgelegt 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 den Wert nicht direkt ein, wenn Sie geheime Parameter übergeben. Der Wert wird als Klartext gerendert und ist daher lesbar. Verwenden Sie aus Sicherheitsgründen keinen Klartext mit Geheimnissen. Wir empfehlen dringend, diese Option AWS Secrets Manager zum Speichern von Geheimnissen zu verwenden.

Beispiele für die Verwendung von Variablen finden Sie Schritt für Schritt:

- 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
- Ein Tutorial mit einer Lambda-Aktion, die Variablen aus einer Upstream-Aktion (CodeCommit) verwendet und Ausgabevariablen generiert, finden Sie unter. Tutorial: Variablen mit Lambda-Aufrufaktionen verwenden
- Ein Tutorial mit einer CloudFormation Aktion, die auf Stack-Ausgabevariablen aus einer CloudFormation Upstream-Aktion verweist, finden Sie unterTutorial: Erstellen Sie eine Pipeline, die Variablen von verwendet AWS CloudFormation Aktionen zur Bereitstellung.
- 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 als Variablen alle Umgebungsvariablen, die als Teil des Builds exportiert wurden. 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 Felder der Aktionskonfiguration eingeben, überschreiten Sie nicht die Grenze von 1000 Zeichen für die Konfigurationsfelder. Ein Validierungsfehler wird zurückgegeben, wenn dieser Grenzwert überschritten wird.

Themen

- Konzepte
- Anwendungsfälle für Variablen
- Konfigurieren von Variablen
- Variablenauflösung
- Regeln für Variablen
- Für Pipeline-Aktionen verfügbare Variablen

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, diese Variablen verfügbar zu machen:

- 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 die durch eine Aktion erzeugten Variablen sehen, 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. Um zu sehen, welche Variablen jeder Aktionstyp erzeugt, lesen Sie die 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 können.

- Variablen auf Pipeline-Ebene richten sich an CodePipeline Kunden, die jedes Mal dieselbe Pipeline verwenden möchten, wobei die Eingaben für die Aktionskonfiguration geringfügig variieren. Jeder Entwickler, der eine Pipeline startet, fügt den Variablenwert in der Benutzeroberfläche hinzu, wenn die Pipeline gestartet wird. Bei 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 einzige Pipeline verwenden, um Ressourcen in mehreren Build- und Bereitstellungsumgebungen bereitzustellen. Bei einer Pipeline mit einem CodeCommit Repository

kann beispielsweise die Bereitstellung von einem bestimmten Zweig und einer bestimmten Zielbereitstellungsumgebung aus mit CodeBuild CodeDeploy Parametern auf Pipeline-Ebene erfolgen.

> **Note**
>
> Für Amazon ECR, Amazon S3 oder CodeCommit Quellen können Sie auch eine Quellüberschreibung mithilfe eines Eingabe-Transformationseintrags erstellen, um das `revisionValue` In EventBridge für Ihr Pipeline-Ereignis zu verwenden, wobei das aus der Quellereignisvariablen für Ihren Objektschlüssel, Ihren Commit oder Ihre Bild-ID abgeleitet `revisionValue` wird. Weitere Informationen finden Sie in dem optionalen Schritt für die Eingabe der Eingabetransformation, der in den Verfahren unter Amazon ECR-Quellaktionen und Ressourcen EventBridgeHerstellen einer Verbindung zu Amazon S3-Quellaktionen mit einer für Ereignisse aktivierten Quelle, oder CodeCommit Quellaktionen und EventBridge enthalten ist.

Konfigurieren von Variablen

Sie können Variablen auf Pipelineebene 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 Überschreibungen für alle erforderlichen Variablen angeben, wenn Sie eine Pipeline starten. Andernfalls schlägt die Pipeline-Ausführung mit einem Validierungsfehler fehl.

Sie stellen Variablen auf Pipeline-Ebene mithilfe des Variablenattributs in der Pipeline-Struktur bereit. Im folgenden Beispiel `Variable1` hat die Variable einen Wert von`Value1`.

```
"variables": [
     {
         "name": "Variable1",
         "defaultValue": "Value1",
         "description": "description"
     }
 ]
```

Ein Beispiel für die JSON-Struktur der Pipeline finden Sie unterEine Pipeline, Phasen und Aktionen erstellen.

Ein Tutorial mit einer Variablen auf Pipeline-Ebene, die zum Zeitpunkt der Pipeline-Ausführung übergeben wird, finden Sie unter. Tutorial: Variablen auf Pipeline-Ebene verwenden

Beachten Sie, dass die Verwendung von Variablen auf Pipeline-Ebene in Quellaktionen aller Art 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 den `namespace` Parameter der CodeCommit Quellaktion 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.

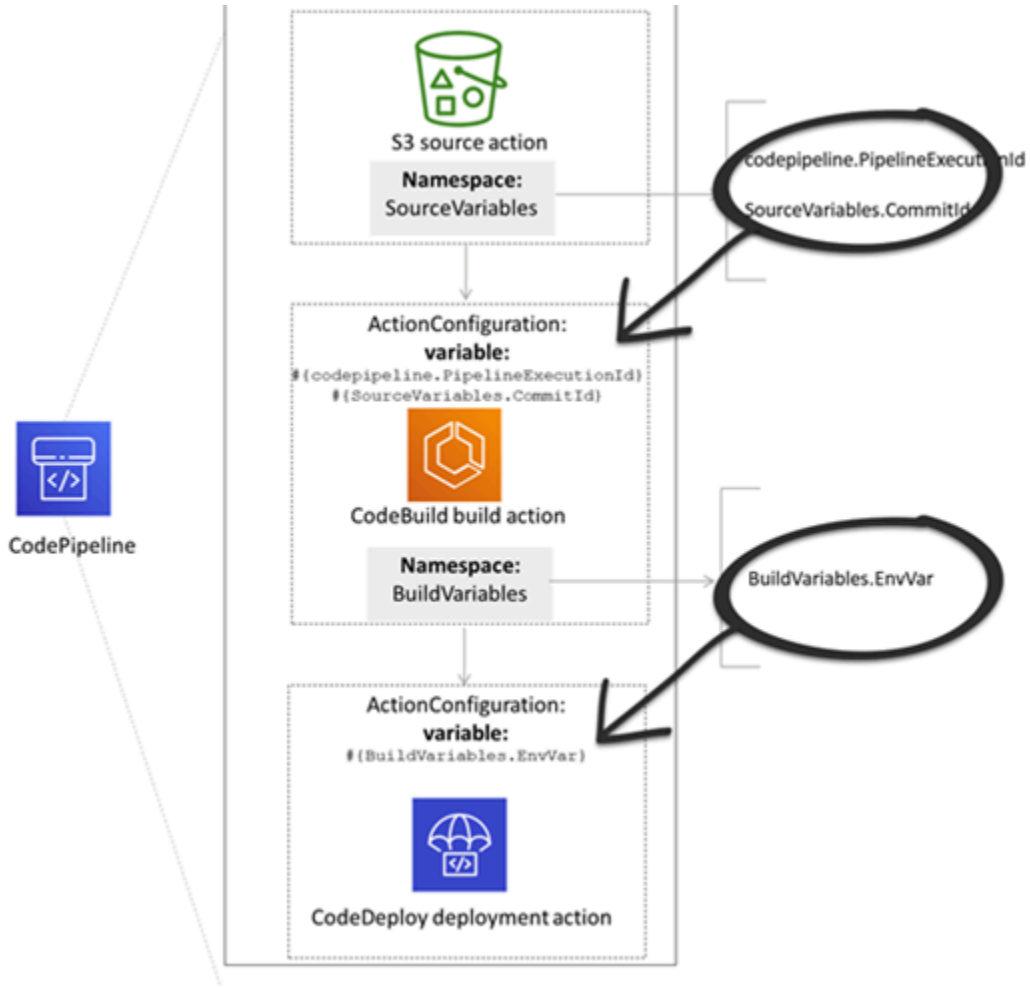
In diesem Beispiel zeigt das Konfigurationsfeld der Build-Aktion Umgebungsvariablen an, die bei Aktionsausführung aktualisiert werden. Das Beispiel gibt den Namespace und die Variable für die Ausführungs-ID mit `#{codepipeline.PipelineExecutionId}` sowie den Namespace und die Variable für die Commit-ID 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 Aktion alle Variablenreferenzen auf, die in der Konfiguration vorhanden waren, bevor die Aktionsausführung initiiert 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.

Schritt-für-Schritt-Verfahren zum Verwalten von Variablen finden Sie unter Arbeiten mit Variablen.

Aktionen mit definierten Variablenschlüsseln

Im Gegensatz zu einem Namespace, den Sie auswählen können, verwenden die folgenden Aktionen variable Schlüssel, die nicht bearbeitet werden können. Für den Amazon S3-Aktionsanbieter sind beispielsweise nur die `VersionId` Variablenschlüssel `ETag` und verfügbar.

Jede Ausführung hat auch eine Reihe von CodePipeline-generated Pipeline-Variablen, die Daten über die Ausführung enthalten, wie z. B. die Pipeline-Release-ID. Diese Variablen können von jeder Aktion in der Pipeline verbraucht werden.

Themen

- CodePipeline Ausführungs-ID-Variable
- Ausgabevariablen für Amazon ECR-Aktionen
- CloudFormation StackSets Aktionsausgabevariablen
- CodeCommit Aktionsausgabevariablen
- CodeStarSourceConnection Aktionsausgabevariablen
- GitHub Aktion, Ausgabevariablen GitHub (über die OAuth-App), Aktion)
- Ausgabevariablen für S3-Aktionen

CodePipeline Ausführungs-ID-Variable

CodePipeline Variable mit Ausführungs-ID

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 Variablensyntax
`ImageDigest`	sha256:EXAMPLE1122334455	`#{SourceVariables.ImageDigest}`
`ImageTag`	brandneue	`#{SourceVariables.ImageTag}`
`ImageURI`	11111 EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest	`#{SourceVariables.ImageURI}`
`RegistryId`	EXAMPLE12233	`#{SourceVariables.RegistryId}`
`RepositoryName`	my-image-repo	`#{SourceVariables.RepositoryName}`

CloudFormation StackSets Aktionsausgabevariablen

CloudFormation StackSets Variablen

Variablenschlüssel	Beispielwert	Beispiel für Variablensyntax
`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 Variablensyntax
`AuthorDate`	2019-10-29 T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	Entwicklung	`#{SourceVariables.BranchName}`
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Behebung eines Fehlers (maximale Größe 100 KB)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	2019-10-29 T03:32:21Z	`#{SourceVariables.CommitterDate}`
`RepositoryName`	mein CodeCommitRepo	`#{SourceVariables.RepositoryName}`

CodeStarSourceConnection Aktionsausgabevariablen

Die verfügbaren Ausgabevariablen hängen vom Triggertyp ab (Push, Pull Request oder Tag). Die in der folgenden Tabelle als Alle markierten Variablen werden für jeden Triggertyp erzeugt. Variablen, die mit einem bestimmten Triggertyp gekennzeichnet sind, werden nur erzeugt, wenn die Pipeline-Ausführung durch diesen Trigger gestartet wird.

`CodeStarSourceConnection`Variablen (Bitbucket Cloud GitHub, GitHub Enterprise Repository und GitLab.com)

Variablenschlüssel	Beispielwert	Beispiel für eine Variablensyntax	Auslösertyp
`AuthorDate`	2019-10-29 T03:32:21Z	`#{SourceVariables.AuthorDate}`	Alle
`AuthorDisplayName`	Jane Doe	`#{SourceVariables.AuthorDisplayName}`	Alle
`AuthorEmail`	jane.doe@example.com	`#{SourceVariables.AuthorEmail}`	Alle
`AuthorId`	Jane Doe	`#{SourceVariables.AuthorId}`	Alle
`BranchName`	Entwicklung	`#{SourceVariables.BranchName}`	Drücken, Markieren
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`	Alle

Variablenschlüssel	Beispielwert	Beispiel für eine Variablensyntax	Auslösertyp
`CommitMessage`	Behebung eines Fehlers (maximale Größe 100 KB)	`#{SourceVariables.CommitMessage}`	Alle
`ConnectionArn`	arn:aws:codestar-connections:region ::connection/ *account-id connection-id*	`#{SourceVariables.ConnectionArn}`	Alle
`DestinationBranchName`	Haupt	`#{SourceVariables.DestinationBranchName}`	Pull-Anforderung
`FullRepositoryName`	username/GitHubRepo	`#{SourceVariables.FullRepositoryName}`	Alle
`ProviderType`	GitHub	`#{SourceVariables.ProviderType}`	Alle
`PullRequestId`	1	`#{SourceVariables.PullRequestId}`	Pull-Anforderung
`PullRequestTitle`	Neues Feature hinzufügen	`#{SourceVariables.PullRequestTitle}`	Pull-Anforderung

Variablenschlüssel	Beispielwert	Beispiel für eine Variablensyntax	Auslösertyp
SourceBranchName	Feature-Branch	#{SourceVariables.SourceBranchName}	Pull-Anforderung
TagName	v1.0.0	#{SourceVariables.TagName}	Markierung

GitHub Aktion, Ausgabevariablen GitHub (über die OAuth-App), Aktion)

GitHub Variablen (GitHub (über die OAuth-App), Aktion)

Variablenschlüssel	Beispielwert	Beispiel für eine Variablensyntax
AuthorDate	2019-10-29 T03:32:21Z	#{SourceVariables.AuthorDate}
BranchName	Haupt	#{SourceVariables.BranchName}
CommitId	exampleb01f91b31	#{SourceVariables.CommitId}
CommitMessage	Behebung eines Fehlers (maximale Größe 100 KB)	#{SourceVariables.CommitMessage}
CommitterDate	29.10.2019 T03:32:21Z	#{SourceVariables.CommitterDate}
CommitUrl		#{SourceVariables.CommitUrl}

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`RepositoryName`	mein GitHubRepo	`#{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 benutzerkonfigurierten Variablenschlüsseln

Für CodeBuild CloudFormation, und Lambda-Aktionen werden die Variablenschlüssel vom Benutzer konfiguriert.

Themen

- CloudFormation Ausgabevariablen für Aktionen
- CodeBuild Ausgabevariablen für Aktionen
- Ausgabevariablen für Lambda-Aktionen

CloudFormation Ausgabevariablen für Aktionen

CloudFormation variables

Variablenschlüssel	Beispiel für eine Variablensyntax
Für CloudFormation Aktionen werden Variablen aus beliebige n Werten generiert, die im `Outputs` Abschnitt einer Stack-	`#{DeployVariables.StackName}`

Variablenschlüssel	Beispiel für eine Variablensyntax
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 Stack-Erstellung, Stack-Akt ualisierungen 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 unterCloudFormation Aktionsreferenz bereitstellen. Ein Tutorial, das Ihnen zeigt, wie Sie eine Pipeline mit einer CloudFormation Bereitstellungsaktion in einer Pipeline erstellen, die CloudFormation Ausgabevariablen verwendet , finden Sie unterTutorial: Erstellen Sie eine Pipeline, die Variablen von verwendet AWS CloudFormation Aktionen zur Bereitstellung.	

CodeBuild Ausgabevariablen für Aktionen

CodeBuild Variablen

Variablenschlüssel	Beispiel für eine Variablensyntax
Für CodeBuild Aktionen werden Variablen aus Werten generiert, die von exportierten Umgebungsvariablen generiert wurden. Richten Sie eine CodeBuild Umgebungsvariable ein, indem Sie Ihre CodeBuild Aktion in der Buildspec bearbeite n CodePipeline oder die Umgebungsvariable zur Buildspec hinzufügen.	`#{BuildVariables.EnvVar}`

Variablenschlüssel	Beispiel für eine Variablensyntax
Fügen Sie Ihrer CodeBuild Buildspec Anweisungen hinzu, um die Umgebungsvariable im Abschnitt „Exportierte Variablen“ hinzuzufügen. Siehe env/exported -variables im Benutzerh andbuch. AWS CodeBuild	

Ausgabevariablen 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-Anforderung enthalten sind. Ein Tutorial mit einer Lambda-Aktion, die Variablen aus einer Upstream-Aktion (CodeCommit) verwendet und Ausgabeva riablen generiert, finden Sie unter. Tutorial: Variablen mit Lambda-Aufrufaktionen verwenden	`#{TestVariables.te stRunId}`

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. Zum Beispiel:

- `*.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 eine Pipeline haben, die Polling verwendet, um auf Quelländerungen zu reagieren, können Sie sie so aktualisieren, dass sie die empfohlene Erkennungsmethode verwendet. Ein Migrationsleitfaden mit Anweisungen zur Aktualisierung Ihrer Polling-Pipelines zur Verwendung der empfohlenen ereignisbasierten Änderungserkennungsmethode finden Sie unter. Migrieren Sie Polling-Pipelines, um ereignisbasierte Änderungserkennung zu verwenden

Aktualisieren Sie eine Quellaktion GitHub (über die OAuth-App) auf eine Quellaktion GitHub (über die GitHub App)

AWS CodePipeline In gibt es zwei unterstützte Versionen der GitHub Quellaktion:

- Empfohlen: Die Aktion GitHub (via GitHub App) verwendet eine GitHub App-basierte Authentifizierung, die von einer CodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen Ressource unterstützt wird. Sie installiert eine AWS CodeStar Connections-Anwendung in Ihrer GitHub Organisation, sodass Sie den Zugriff darin verwalten können. GitHub
- Nicht empfohlen: Die Aktion GitHub (über die OAuth-App) 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 (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 im Hinweis unterCodeStarSourceConnection für Bitbucket Cloud GitHub, GitHub Enterprise Server und GitLab.com GitLab selbstverwaltete Aktionen.

Die Verwendung der Aktion GitHub (über GitHub App) anstelle der Aktion (über die GitHub OAuth-App) bietet einige wichtige Vorteile:

- Bei Verbindungen sind CodePipeline keine OAuth-Apps oder persönlichen Zugriffstoken mehr erforderlich, um auf dein Repository zuzugreifen. Wenn du eine Verbindung herstellst, installierst du eine GitHub App, die die Authentifizierung für dein GitHub Repository verwaltet und Berechtigungen auf Organisationsebene gewährt. Du musst OAuth-Tokens als Benutzer autorisieren, um auf das Repository zuzugreifen. Weitere Hinweise zum OAuth-based GitHub

Zugriff im Gegensatz zum Zugriff finden Sie App-based GitHub unter. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps

- Wenn Sie Aktionen GitHub (per GitHub App) in der CLI oder verwalten CloudFormation, müssen Sie Ihr persönliches Zugriffstoken nicht mehr als geheim 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 und GitLab.com GitLab selbstverwaltete Aktionen.
- Wenn du eine Verbindungsressource erstellst, die du zusammen mit deiner Aktion GitHub (per GitHub App) verwenden möchtest CodePipeline, kannst du dieselbe Verbindungsressource verwenden, um andere unterstützte Dienste, wie CodeGuru Reviewer, mit deinem Repository zu verknüpfen.
- In GitHub (via GitHub App) kannst du Repositorys klonen, um in nachfolgenden CodeBuild Aktionen auf Git-Metadaten zuzugreifen, während du in GitHub (über die OAuth-App) nur die Quelle herunterladen kannst.
- Ein Administrator installiert die App für die Repositorys Ihrer Organisation. Sie müssen OAuth-Token nicht 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 Scoped-Down-Richtlinie für die Verwendung von Verbindungen mit einem bestimmten Repository.

Sie können die Schritte in diesem Thema verwenden, um Ihre Quellaktion GitHub (über die OAuth-App) zu löschen und eine Quellaktion GitHub (über die GitHub App) aus der Konsole hinzuzufügen. CodePipeline

Themen

- Schritt 1: Ersetzen Sie Ihre Aktion (über die OAuth-App) GitHub
- Schritt 2: Erstellen Sie eine Verbindung zu GitHub
- Schritt 3: Speichern Sie Ihre GitHub Quellaktion

Schritt 1: Ersetzen Sie Ihre Aktion (über die OAuth-App) GitHub

Verwenden Sie die Pipeline-Bearbeitungsseite, um Ihre Aktion (über die OAuth-App) durch eine GitHub Aktion GitHub (über GitHub die App) zu ersetzen.

Um Ihre Aktion (über die OAuth-App) zu ersetzen GitHub

1. Loggen Sie sich in die CodePipeline Konsole ein.
2. Wählen Sie Ihre Pipeline aus und klicken Sie auf Bearbeiten. Wählen Sie auf Ihrer Quellstufe Stufe bearbeiten aus. Es wird eine Meldung angezeigt, in der empfohlen wird, Ihre Aktion zu aktualisieren.
3. Wählen Sie unter Aktionsanbieter die Option GitHub (über GitHub App) aus.
4. Führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie unter Verbindung die Option Verbinden mit aus, falls Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben GitHub. Fahren Sie mit Schritt 2 fort: Erstellen Sie eine Verbindung zu GitHub.
   - Wenn Sie bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Verbindung aus. Fahren Sie mit Schritt 3: Speichern Sie die Quellaktion für Ihre Verbindung fort.

Schritt 2: Erstellen Sie eine Verbindung zu GitHub

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die GitHub Seite Verbinden mit angezeigt.

Um eine Verbindung herzustellen zu GitHub

1. Unter GitHub Verbindungseinstellungen wird Ihr Verbindungsname im Feld Verbindungsname angezeigt.

   Wählen Sie unter GitHub Apps eine App-Installation aus oder wählen Sie Eine 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, um fortzufahren.
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 Verbinden mit 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 Folgendes ein, `owner-name/repository-name` wie in diesem Beispiel gezeigt:

```
my-account/my-repository
```

2. Wählen Sie im Ausgabe-Artefaktformat 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 nachgeschalteten Aktionen verwendet werden.

     Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter gezeigt. Fügen Sie CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub Enterprise Server oder hinzu GitHub GitLab.com Ein Tutorial, das Ihnen zeigt, wie Sie die Option „Vollständiger Klon“ verwenden, finden Sie unterTutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

3. Unter Ausgabeartefakte können Sie den Namen des Ausgabe-Artefakts für diese Aktion beibehalten, z. B. `SourceArtifact` Wählen Sie Fertig, um die Seite „Aktion 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, Stufen, Aktionen und Webhooks, die ein AWS Konto in jeder AWS Region haben kann.

Diese Kontingente gelten pro Region und können erhöht werden. 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)	CloudFormation Bereitstellungsaktion: 3 Tage CodeDeploy und CodeDeploy ECS (blue/green) -Bereitstellungsaktionen: 5 Tage AWS Lambda Aktion aufrufen: 24 Stunden **Note** Während die Aktion ausgeführt wird, kontaktiert Lambda CodePipeline regelmäßig, um einen Status zu erhalten. 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 der Status der

Ressource	Standard
	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 bis zu 86400 Minuten (60 Tage) mit einem Mindestwert von 5 Minuten konfiguriert werden. Weitere Informationen

Ressource	Standard
	finden Sie unter ActionDeclaration in der CodePipeline -API-Referenz. Wenn konfiguriert, wird dieses Timeout für die Aktion angewende t. Andernfalls wird die Standarde instellung auf Kontoebene verwendet. Alle anderen Aktionen; 1 Stunde **Note** Das Zeitlimit für Amazon ECS-Berei tstellungsaktionen 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 Änderungs erkennung 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 Erkennung von Änderungen.[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 Sie Polling-Pipelines (CodeCommit oder Amazon S3-Quelle) (Konsole).
- Informationen zum Aktualisieren einer Amazon S3 S3-Quellaktion finden Sie unterMigrieren Sie Polling-Pipelines (CodeCommit oder Amazon S3-Quelle) (Konsole).
- Informationen zum Aktualisieren einer GitHub Quellaktion finden Sie unterMigrieren Sie Polling-Pipelines zu Webhooks (über die OAuth-AppGitHub) (Quellaktionen) (Konsole).

Die folgenden Kontingente AWS CodePipeline gelten für die Verfügbarkeit in der Region, die Benennungsbeschränkungen und die zulässige Größe von Artefakten. 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	USA Ost (Nord-Virginia) USA Ost (Ohio) USA West (Nordkalifornien) USA West (Oregon) Kanada (Zentral) Europa (Frankfurt) Europa (Zürich) * Europa (Stockholm) Europa (Mailand) * Europa (Spanien) * Europa (Irland) Europa (London) Europa (Paris) Israel (Tel Aviv) * Afrika (Kapstadt) * Asien-Pazifik (Hongkong) * Asien-Pazifik (Mumbai) Asien-Pazifik (Hyderabad) * Asien-Pazifik (Tokio)

	Asien-Pazifik (Seoul) Asien-Pazifik (Osaka) Asien-Pazifik (Singapur) Asien-Pazifik (Sydney) Asien-Pazifik (Jakarta) * Asien-Pazifik (Melbourne) * Asien-Pazifik (Malaysia) * Asien-Pazifik (Neuseeland) * Südamerika (São Paulo) Naher Osten (Bahrain) * Naher Osten (VAE) * AWS GovCloud (US-West) AWS GovCloud (US-East)
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	Partneraktionsnamen unterliegen den gleichen Namenskonventionen und Einschränkungen wie andere Aktionsnamen in CodePipel ine. 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 Bauaktion: 36 Stunden Testaktion: 8 Stunden Benutzerdefinierte Aktionen: 24 Stunden Step Functions rufen Aktion auf: 7 Tage Timeout für die Aktion „Befehle“ beim Erstellen: 55 Minuten
Maximallänge des Aktionskonfigurationsschlüssels (z. B. die CodeBuild-Konfigurationsschlüssel `ProjectName` , `PrimarySource` und `EnvironmentVariables`)	50 Zeichen
Maximale Länge des Aktionskonfigurationswerts (z. B. sollte der Wert der `RepositoryName` Konfiguration in der CodeCommit Aktionskonfiguration weniger als 1000 Zeichen lang sein): `"RepositoryName": "my-repo-name-less-than-1000-characters"`)	1 000 Zeichen

Maximale Anzahl von Aktionen pro Pipeline	1000
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 paralleler Aktionen in einer Phase	100
Maximalanzahl der sequenziellen Aktionen in einer Phase	100
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 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 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 CloudFormation Anbieter wird die `ParameterOverrides` Eigenschaft verwendet, um ein JSON-Objekt zu speichern, das Werte für die 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-Artefakte, die für jede Aktion zulässig sind, finden Sie unter Gültige Eingabe- und Ausgabeartefakte für jeden Aktionstyp
Anzahl der Monate, in denen Informationen zum Verlauf der Pipeline-Ausführung aufbewahrt werden	12

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	Jede Kombination aus Unicode-Buchstaben, Zahlen, Leerzeichen und zulässigen Zeichen mit einer Länge UTF-8 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	Jede Kombination aus Unicode-Buchstaben, Zahlen, Leerzeichen und zulässigen Zeichen mit einer Länge UTF-8 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: GitHub Quellaktionen (über die OAuth-App)

Dieser Anhang enthält Informationen zur Aktion GitHub (über die OAuth-App) in. CodePipeline

> **Note**
>
> Wir empfehlen zwar nicht, die Aktion GitHub (über die OAuth-App) zu verwenden, aber bestehende Pipelines mit der Aktion GitHub (über die OAuth-App) funktionieren weiterhin ohne Auswirkungen. Bei einer Pipeline mit einer Aktion GitHub (über die OAuth-App) werden Tokens CodePipeline verwendet OAuth-based , um eine Verbindung zu deinem Repository herzustellen. GitHub Im Gegensatz dazu verwendet die GitHub Aktion (per GitHub App) eine Verbindungsressource, um AWS Ressourcen mit deinem 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 GitHub (über die OAuth-App) auf eine Quellaktion GitHub (über die GitHub App). Weitere Informationen zum OAuth-based GitHub Zugriff im Gegensatz zum App-basierten GitHub Zugriff finden Sie unterhttps://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps.

CodePipeline Verwendet für die Integration GitHub eine GitHub OAuth-Anwendung für Ihre Pipeline. CodePipelineverwendet Webhooks, um die Änderungserkennung für Ihre Pipeline mit der Quellaktion GitHub (über die OAuth-App) zu verwalten.

> **Note**
>
> Wenn Sie eine Quellaktion GitHub (über die GitHub App) in konfigurieren CloudFormation, geben Sie keine GitHub Token-Informationen an und fügen auch keine Webhook-Ressource hinzu. Sie konfigurieren eine Verbindungsressource, wie in AWS::CodeStarConnections: :Connection im CloudFormation Benutzerhandbuch gezeigt.

Diese Referenz enthält die folgenden Abschnitte für die Aktion GitHub (über die OAuth-App):

- Informationen zum Hinzufügen einer Quellaktion und eines GitHub Webhooks (über die OAuth-App) zu einer Pipeline finden Sie unter. Hinzufügen einer GitHub Quellaktion (über die OAuth-App)

- Hinweise zu den Konfigurationsparametern und YAML/JSON Beispielausschnitten für eine Quellaktion GitHub (über die OAuth-App) finden Sie unter. GitHub Referenz zur Quellaktion (über die OAuth-App)

> **Important**
>
> Verwenden Sie beim Erstellen von CodePipeline Webhooks nicht Ihre eigenen Anmeldeinformationen und verwenden Sie nicht dasselbe geheime Token für mehrere Webhooks wieder. Generieren Sie für optimale Sicherheit ein eindeutiges geheimes Token für jeden Webhook, den Sie erstellen. Das geheime Token ist eine beliebige Zeichenfolge, die Sie angeben und mit der die GitHub Webhook-Payloads berechnet und signiert werden, an die gesendeten Webhook-Payloads CodePipeline, um die Integrität und Authentizität der Webhook-Payloads zu schützen. Die Verwendung Ihrer eigenen Anmeldeinformationen oder die Wiederverwendung desselben Tokens in mehreren Webhooks kann zu Sicherheitslücken führen.

> **Note**
>
> Wenn ein geheimes Token bereitgestellt wurde, wird es in der Antwort redigiert.

Themen

- Hinzufügen einer GitHub Quellaktion (über die OAuth-App)
- GitHub Referenz zur Quellaktion (über die OAuth-App)

Hinzufügen einer GitHub Quellaktion (über die OAuth-App)

Sie fügen GitHub (über die OAuth-App) Quellaktionen hinzu, indem Sie: CodePipeline

- Verwenden Sie den Assistenten zum Erstellen einer Pipeline in der CodePipeline Konsole (Erstellen Sie eine benutzerdefinierte Pipeline (Konsole)) oder auf der Aktionsseite „Bearbeiten“, um die GitHub Anbieteroption 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 unterGitHub Referenz zur Quellaktion (über die OAuth-App), um die Aktion zu erstellen, wie unter gezeigtErstellen einer Pipeline (CLI).
- Deaktivierung regelmäßiger Überprüfungen und manuelles Erstellen der Änderungserkennung, da bei der Methode zur Änderungserkennung standardmäßig die Pipeline durch Abfragen der Quelle gestartet wird. Sie migrieren Ihre Polling-Pipeline für Aktionen GitHub (über die OAuth-App) zu Webhooks.

GitHub Referenz zur Quellaktion (über die OAuth-App)

> **Note**
>
> Wir empfehlen zwar nicht, die Aktion GitHub (über die OAuth-App) zu verwenden, aber bestehende Pipelines mit der Aktion GitHub (über die OAuth-App) funktionieren weiterhin ohne Auswirkungen. Bei einer Pipeline mit einer Quellaktion GitHub (über die OAuth-App) werden Tokens CodePipeline verwendet OAuth-based , um eine Verbindung zu deinem Repository herzustellen. GitHub Im Gegensatz dazu verwendet die neue GitHub Aktion (per GitHub App) eine Verbindungsressource, um AWS Ressourcen mit deinem 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 GitHub (über die OAuth-App) auf eine Quellaktion GitHub (über die GitHub App).

Löst die Pipeline aus, wenn ein neuer Commit für das konfigurierte GitHub Repository und den Branch vorgenommen wird.

CodePipeline Verwendet für die 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.

Wenn Sie den Zugriff auf CodePipeline Repositorys einschränken möchten, erstellen Sie ein GitHub Konto und gewähren Sie dem Konto nur Zugriff auf die Repositorys, mit denen Sie eine Integration

durchführen möchten. CodePipeline Verwenden Sie dieses Konto, wenn Sie die Verwendung von GitHub Repositorys für Quellphasen in Pipelines konfigurieren CodePipeline .

Weitere Informationen finden Sie in der GitHub Entwicklerdokumentation auf der GitHub Website.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Aktionserklärung (GitHub Beispiel)
- Verbindung zu GitHub (OAuth) herstellen
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Source`
- Eigentümer: `ThirdParty`
- Anbieter: `GitHub`
- Version: `1`

Konfigurationsparameter

Eigentümer

Erforderlich: Ja

Der Name des GitHub Benutzers oder der Organisation, dem/der 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.

OAuthToken

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 ein OAuth-Token, um die GitHub Verbindung zu registrieren.
- Wenn Sie die verwenden, AWS CLI um die Pipeline zu erstellen, 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 CloudFormation Vorlage verwenden, um die Pipeline zu erstellen, müssen Sie das Token zunächst als Geheimnis 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 Quelländerungen CodePipeline abgefragt wird. 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 GitHub (über die OAuth-App) (Quellaktionen) (CLI) oder Aktualisieren Sie die Pipelines für Push-Ereignisse GitHub (über die OAuth-App) (Quellaktionen) (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 es dein Repository nach Änderungen an der Quelle ab.

  > **Note**
  >
  > Wenn du es auslässt`PollForSourceChanges`, wird dein Projektarchiv CodePipeline standardmäßig nach Änderungen an der Quelle abgefragt. Dieses Verhalten ist das gleiche, als ob `PollForSourceChanges` auf `true` festgelegt würde.

- `False`: Wenn gesetzt, fragt dein Projektarchiv CodePipeline nicht nach Änderungen an der Quelle 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 Commit-ID des Quellcodes wird CodePipeline als Quellversion 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 unterVariablenreferenz.

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 vorgenommen wurde, der die Pipeline ausgelöst hat.

BranchName

Der Name des Branches für das GitHub Repository, in dem die Quelländerung vorgenommen wurde.

AuthorDate

Das Datum im Zeitstempelformat, an dem der Commit erstellt wurde.

CommitterDate

Das Datum im Zeitstempelformat, an dem der Commit durchgeführt wurde.

Aktionserklärung (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 zu GitHub (OAuth) herstellen

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 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 Schnipsel für die Ressource in. CloudFormation
- Ressourcenreferenz für das AWS CloudFormation Benutzerhandbuch AWS::CodeStar:: GitHubRepository — Dazu gehören Felddefinitionen, Beispiele und Ausschnitte für die Ressource in. CloudFormation

- Tutorial: Erstellen Sie eine Pipeline, die Ihre Android-App erstellt und testet mit AWS Device Farm— Dieses Tutorial enthält eine Beispiel-Buildspec-Datei und eine Beispielanwendung zum Erstellen einer Pipeline mit einer Quelle. GitHub Es erstellt und testet eine Android-App mit und. CodeBuild AWS Device Farm

AWS CodePipeline Dokumenthistorie des Benutzerhandbuches

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: 23. Mai 2025

Änderung	Beschreibung	Datum
Die Dokumentation zum Verhalten des Pipeline-Trigger-Filters wurde korrigiert	Die Dokumentation für Beispiele für Triggerfilter wurde korrigiert, um die Beziehung zwischen Ein- und Ausschlüssen genau zu beschreiben. Ausschlüsse haben bei Verzweigungs- und Dateipfadfiltern Vorrang vor Einschlüssen. Weitere Informationen finden Sie unter Beispiele für Triggerfilter.	16. März 2026
Support für Azure hinzugefügt DevOps	Support für die Konfiguration von Verbindungen für AWS Ressourcen zur Interaktion mit Azure hinzugefügt DevOps. Weitere Informationen finden Sie unter Verbindungen für Azure DevOps.	5. August 2025
Die Dokumentation für die EC2-Deploy-Action-Deploy-	Es wurden neue Informati onen zur Verwendung der Bereitstellungsspezifikatio	16. Mai 2025

Spezifikationsdatei wurde aktualisiert.	nsdatei für die EC2-Berei tstellungsaktion mit aktualisi erten Beispielen und Konfigura tionsoptionen hinzugefügt. Weitere Informationen finden Sie auf der Aktionsreferenzsei te unter Amazon EC2 EC2-Aktionsreferenz.	
Dokumentation für die Lambda-Bereitstellungsaktion hinzugefügt	Unterstützung für die neue Lambda-Bereitstellungsaktio n mit aktualisierten Beispiele n und Konfigurationsopti onen hinzugefügt. Weitere Informationen finden Sie auf der Aktionsreferenzseite unter Aktionsreferenz AWS Lambda bereitstellen.	16. Mai 2025
Fügen Sie der Aktion „Befehle“ Felder für den Berechnun gstyp, den Umgebungstyp und den Typ der Umgebungs variablen hinzu	Es wurden Informationen zu neuen Feldern hinzugefügt, mit denen Sie eine Build-Umg ebung und einen Berechnun gstyp für die Aktion Befehle angeben können. Ein neues Typfeld für Umgebungs variablen wurde auch für die Aktion Befehle hinzugefügt. Weitere Informationen finden Sie auf der Aktionsreferenzsei te unter CodePipeline Befehle > Aktion.	8. Mai 2025

Neues Thema zur Dokumentation der Standardrichtlinie für Servicerollen	Es wurden Informationen zur Mindestrichtlinie für Servicerollen hinzugefügt und die Tabelle mit Links zu zusätzlichen Dienstrollenberechtigungen für jede Aktion in aktualisiert CodePipeline. Weitere Informationen finden Sie auf der Referenzseite für neue Regeln unter CodePipeline Service Role Policy.	26. März 2025
Neue `CodePipeline` Aufruf-Aktion	Es wurden Informationen zur neuen `CodePipeline` Aufrufaktion hinzugefügt. Weitere Informationen finden Sie auf der Aktionsreferenzseite unter Aktionsreferenz CodePipeline aufrufen.	14. März 2025
Neue `CodeBuild` Regel	Es wurden Informationen über die neue `CodeBuild` Regel hinzugefügt, mit der Build-Projekte als Regel für Stufenbedingungen ausgeführt werden können. Die Referenzseite für neue Regeln finden Sie unter CodeBuild Regelreferenz.	14. März 2025

Verbindungen teilen in AWS CodeConnections	Für Pipelines mit Verbindungen können Sie Verbindungen verwenden AWS CodeConnections, die für die gemeinsame Nutzung zwischen AWS-Konten konfiguriert sind. Sie können gemeinsam genutzte Verbindungen in AWS Resource Access Manager konfigurieren. Weitere Informationen finden Sie unter Verwenden einer Verbindung, die gemeinsam mit einer anderen Person genutzt wird AWS-Konto.	6. März 2025

EventBridge Eingabe-Transformationseintrag für Quellversionen	Für Pipelines mit CodeCommit Amazon ECR- und Amazon S3 S3-Quellen können Sie den EventBridge Eingabe-Transformationseintrag für Quellrevisionen verwenden, wobei der von der Quellereignisvariablen für Ihren Objektschlüssel (S3), Commit () oder Image-ID (ECRCodeCommit) abgeleitet `revisionValue` wird. Weitere Informationen finden Sie unter Amazon ECR-Quellaktionen und EventBridge -Ressourcen, Herstellen einer Verbindung zu Amazon S3 S3-Quellaktionen mit einer für Ereignisse aktivierten CodeCommit Quelle und Quellaktionen und EventBridge.	03. März 2025
Spezifikationsüberschreibung für die Aktion erstellen AWS CodeBuild	Sie können die Build-Spezifikation für die CodeBuild Aktion überschreiben und stattdessen Befehle direkt eingeben. Weitere Informationen finden Sie auf der Aktionsreferenzseite unter Aktionsreferenz zum AWS CodeBuild Erstellen und Testen. Weitere Informationen finden Sie unter Erstellen einer Pipeline, Phasen und Aktionen.	03. März 2025

Neue EC2 Bereitstellungsaktion	Es wurden Informationen zur neuen EC2 Bereitstellungsaktion hinzugefügt. Weitere Informationen finden Sie auf der Aktionsreferenzseite unter Amazon EC2 EC2-Aktionsreferenz. Ein Tutorial finden Sie unter Tutorial: Deploy to EC2-Instances with CodePipeline.	21. Februar 2025
Neue EKS Bereitstellungsaktion	Es wurden Informationen zur neuen EKS Bereitstellungsaktion hinzugefügt. Weitere Informationen finden Sie auf der Aktionsreferenzseite unter EKS Deploy Action. Ein Tutorial finden Sie unter Tutorial: In Amazon EKS bereitstellen mit CodePipeline.	20. Februar 2025
Neue CloudWatch Metriken und Dimensionen für CodePipeline	Metriken für Pipelines wurden in CloudWatch Metriken hinzugefügt. Siehe CodePipeline CloudWatch Metriken.	13. Februar 2025
Neue Commands Regel	Es wurden Informationen über die neue Commands Regel hinzugefügt, mit der Shell-Befehle als Regel für Stufenbedingungen ausgeführt werden können. Weitere Informationen finden Sie auf der Referenzseite für neue Regeln unter Befehlsregelreferenz.	17. Dezember 2024

Erweiterte Beispiele und Referenzinformationen für Trigger	Beschreibungen für Pull-Request-Ereignisfilter nach Anbieter wurden unter Pull-Request-Ereignisse für Trigger nach Anbieter hinzugefügt. Es wurden erweiterte Beispiele für Trigger mit detaillierteren Informationen zu Ein- und Ausschlüssen in Push-Ereignissen und Pull-Request-Ereignissen in Pull-Request-Ereignissen für Trigger nach Anbieter hinzugefügt. Zusätzliche JSON-Referenzinformationen zu Ein- und Ausschlüssen in Triggern wurden hinzugefügt.	17. Dezember 2024
Neue Aktionen im Aktionskatalog	Sie können jetzt die `InspectorScan` Aktionen `ECRBuildAndPublish` und verwenden. Weitere Informationen finden Sie auf den Referenzseiten für InspectorScanAktionen ECRBuildAndPublishund Aktionen.	22. November 2024

Neue automatische Konfiguration für Phase-Wiederholungsversuche bei einem Fehler	Sie können eine Phase so konfigurieren, dass eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in der Phase automatisch wiederholt werden. Weitere Informationen finden Sie unter Konfigurieren einer Phase für die automatische Wiederholung bei einem Fehler.	15. Oktober 2024
Neues `Skip` Ergebnis für die Teilnahmebedingungen	Es wurden Informationen über das `Skip` Ergebnis hinzugefügt, die für die Teilnahmebedingungen verfügbar sind. Sie können die `VariableCheck` und `LambdaInvoke` -Regeln mit dieser Konfiguration verwenden. Weitere Informationen finden Sie unter Erstellen von Teilnahmebedingungen mit Skip-Ergebnis. Eine Liste der Überlegungen zu Bedingungen, bei denen Ergebnisse übersprungen werden, finden Sie unter Überlegungen zu Ergebnissen, die für Stufenbedingungen konfiguriert wurden.	15. Oktober 2024

Neue Konsolenschritte zum Erstellen von Pipelines aus statischen Vorlagen	In der CodePipeline Konsole können Sie einen Assistenten zur Erstellung neuer Pipelines verwenden, um aus mehreren statischen Vorlagen auszuwählen, in CloudFormation denen Pipeline-Ressourcen generiert werden sollen. Weitere Informationen finden Sie unter Erstellen einer Pipeline aus statischen Vorlagen.	9. Oktober 2024
Neue `Commands` Aktion	Es wurden Informationen über die neue `Commands` Aktion hinzugefügt, mit der Shell-Befehle als Aktion in Ihrer Pipeline ausgeführt werden können. Weitere Informationen finden Sie auf der Referenzseite für neue Aktionen unter Befehlsaktion und zu den Berechtigungen für Servicerollen unter Berechtigungen zur CodePipeline Servicerolle hinzufügen. Ein Tutorial finden Sie unter Tutorial: Eine Pipeline erstellen, die Befehle mit Compute ausführt.	3. Oktober 2024

Neue `VariableCheck` Regel für Bedingungen	Es wurden Informationen zur `VariableCheck` Regel für Stufenbedingungen hinzugefügt. Die Referenzseite für neue Regeln finden Sie unter VariableCheck. Ein Tutorial finden Sie unter Tutorial: Eine Regel zur Überprüfung von Variablen für eine Pipeline als Eingabebedingung erstellen.	27. September 2024
Aktualisierungen der Verbindungen für GitHub	Es wurden Informationen zur Verwendung des GitHub Benutzerzugriffstokens bei Verbindungen zu GitHub (GitHub V2-Aktionen) hinzugefügt. Benutzerzugriffstoken werden bei CodeBuild Projekten verwendet. Siehe GitHub Verbindungen und Tutorial: Verwenden Sie den vollständigen Klon mit einer GitHub Pipeline-Quelle.	16. September 2024
Aktualisierungen zur Umstrukturierung der Pipeline-JSON-Referenz und des Inhaltsverzeichnisses des Leitfadens	Der Leitfaden wurde neu strukturiert, einschließlich einiger Änderungen im Abschnittstitel, um die Benutzerfreundlichkeit der Referenz- und Aufgabenbereiche zu verbessern.	16. August 2024

Aktualisierung des Felds „Geheimes Token“ in den Antworten für `PutWebhook` und `ListWebhooks` Aktionen	Das geheime Token-Feld für die `ListWebhooks` Aktionen `PutWebhook` und wurde aktualisiert. Wenn ein geheimes Token angegeben wurde, wird es in der Antwort geschwärzt. Weitere Informationen finden Sie in den Hinweisen zu Anhang A: Quellaktionen für GitHub Version 1. Entsprechende Aktualisierungen finden Sie im CodePipeline API-Leitfaden unter PutWebhookund ListWebhooks.	6. August 2024
Es wurden neue Inhalte für die Bedingungen und Regeln der Phase hinzugefügt	Sie können jetzt Stufenbedingungen und Regeln für Pipelines vom Typ V2 konfigurieren. Weitere Informationen finden Sie unter Konzepte, Wie funktionieren Stufenbedingungen? , und Bedingungen für eine Phase konfigurieren. Es wurde ein Referenzkapitel für Regeln hinzugefügt, das Referenzinformationen enthält. Weitere Informationen finden Sie CodePipelinein der Regelreferenz.	30. Juli 2024

Neue Referenzinformationen für Pipeline-Typen und verwandte Funktionen wurden hinzugefügt	Für die Bewertung der Kosten für die Umstellung auf Pipelines vom Typ V2 ist ein neues Analyseskript verfügbar. Weitere Informationen finden Sie unter Welcher Pipeline-Typ ist der richtige für mich? . Es wurde eine Referenztabelle hinzugefügt, die Links zu allen CodePipeline Servicedokumenten nach Funktionen enthält. Weitere Informationen finden Sie in der CodePipeline Funktionsreferenz.	11. Juli 2024
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

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 CloudFormation StackSets Aktionsreferenz und Aktionsreferenz AWS Step Functions.	27. März 2024
Aktualisierungen für verwaltete 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 CloudFormation Aktion in hinzugefügt CodePipeline	CloudFormation In Aktionsfehlermeldungen können jetzt Details zu Ressourcen angezeigt werden, bei denen Fehler aufgetreten sind. Weitere Informationen finden Sie in der 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-Ausführung 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.	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 für verwaltete 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 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 Eine Pipeline starten, Tutorial: Eine Pipeline erstellen mit 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 CloudFormation StackSets Bereitstellungsaktionen	Das neue Tutorial, Tutorial: Eine Pipeline erstellen, die CloudFormation StackSets als Bereitstellungsanbieter verwendet wird, enthält Schritte CloudFormation StackSets zum Erstellen und Aktualisieren Ihrer Stack-Sets und Stack-Instances mit Ihrer Pipeline. Das Referenzschema CloudFormation StackSets Aktionsstruktur wurde ebenfalls hinzugefügt.	30. Dezember 2020
Neue unterstützte Region Asien-Pazifik (Hongkong)	CodePipeline ist jetzt im asiatisch-pazifischen Raum (Hongkong) erhältlich. 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 über eine bestehende Pipeline mit einer GitHub Quellaktion verfügen, finden Sie weitere Informationen unter Aktualisieren einer Quellaktion GitHub (per OAuth App) auf eine Quellaktion GitHub (über GitHub App).	30. September 2020

Die CodeBuild Aktion unterstützt das Aktivieren von Batch-Builds 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 Invoke-Aktionen	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 konfigurierbar (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 eine Pipeline-Ausführung in CodePipeline anhalten. Sie können entweder angeben, dass die Ausführung nach dem Abschluss von in Arbeit befindlichen Aktionen beendet wird, oder Sie können angeben, dass die Ausführung sofort angehalten und in Arbeit befindliche Aktionen beendet werden sollen. Siehe Pipeline-Ausführungen anhalten und Pipeline-Ausführung in CodePipeline beenden.	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 Konfiguration von Aktionen 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 die Kontrolle über das Speichern ACLs und den Cache für Amazon S3 S3-Bereitstellungsaktionen an	Sie können jetzt vorgefert igte ACL- und Cache-Kon trolloptionen angeben, wenn Sie eine Amazon S3 S3-Bereit stellungsaktion 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 Themen zum Erstellen einer Pipeline, zum Bearbeiten einer Pipeline und zur CodePipeline-Pipeline-Strukturreferenz 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 Thema Referenz der 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 Thema Referenz der 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-CodeDeploy Bereitstellungsaktionen	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 ECS-to-CodeDeploy Bereitstellung erstellen, enthält Schritte zur Verwendung der Konsole zum Erstellen einer Pipeline, die in einem Image-Repository gespeicherte Containeranwendungen in einem Amazon ECS-Cluster mit CodeDeploy Datenverkehrs-Routing bereitstellt. Die Themen zum Erstellen einer Pipeline und zur CodePipeline -Pipeline-Strukturreferenz 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 CloudFormation zum Hinzufügen einer Aktion, die sich in einer anderen Region als Ihrer Pipeline befindet. Die Themen zum Erstellen einer Pipeline, zum Bearbeiten einer Pipeline und zur CodePipeline -Pipeline-Strukturreferenz 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 ist jetzt integrierbar mit 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.

Änderungen	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

Änderungen	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 Erkennung von Änderungen. 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. Eine Pipeline, Phasen und Aktionen erstellen und 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

Änderungen	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 CodePipelineEine Pipeline, Phasen und Aktionen erstellen, 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 Erkennung von Änderungen. Das Tutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen 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. Eine Pipeline, Phasen und Aktionen erstellenund Eine Pipeline bearbeiten in CodePipeline wurden ebenfalls aktualisiert. Weitere Informationen finden Sie unter Starten Sie eine Pipeline in CodePipeline.	11. Oktober 2017

Änderungen	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 Event-Regel ein, um E-Mail-Benachrichtigungen über Änderungen des Pipeline-Status zu erhalten hinzugefügt. Weitere Informationen finden Sie unter Überwachung von CodePipeline Ereignissen.	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

Änderungen	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, Berechtigungsreferenz 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 Step 4: Beta in Step 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

Änderungen	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 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

Änderungen	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. Eine Pipeline, Phasen und Aktionen erstellen Sie können das serverlose Anwendungsmodell jetzt CodePipeline zusammen mit 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

Änderungen	Beschreibung	Datum geändert
Aktualisierte Themen	AWS CloudFormation kann jetzt als Bereitstellungsanbieter in Pipelines ausgewählt werden, sodass Sie im Rahmen einer Pipeline-Ausführung Maßnahmen für 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. Eine Pipeline, Phasen und Aktionen erstellen 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) erhältlich. 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	Eine Pipeline, Phasen und Aktionen erstellen 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

Änderungen	Beschreibung	Datum geändert
Aktualisiertes Thema	Der Abschnitt Hinzufügen einer manuellen Genehmigungsaktion zu einer Phase 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 Überwachen 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 Hinzufügen einer manuellen Genehmigungsaktion zu einer Phase 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

Änderungen	Beschreibung	Datum geändert
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
Neues Thema	Ein neues Thema mit dem Titel 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 Eine Pipeline, Phasen und Aktionen erstellen Authentifizierung und Zugriffsk ontrolle, und, CodePipeline Referenz zur Pipeline-StrukturProdukt- und Serviceintegrationen mit CodePipel ine, 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 OpsWorks ist derzeit nur in der Region USA Ost (Nord-Virginia) (us-east-1) verfügbar.	2. Juni 2016
Neue und aktualisi erte Themen	Es wurde ein neues Thema mit dem Titel Tutorial: Eine einfache Pipeline (CodeCommit Repository) erstellen 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

Änderungen	Beschreibung	Datum geändert
Neues Thema	Es wurde ein neues Thema mit dem Titel Ruft einen auf AWS Lambda Funktion in einer Pipeline in CodePipeline hinzugefügt. Dieses Thema enthält AWS Lambda Beispielf unktionen und Schritte zum Hinzufügen von Lambda-Fu nktionen zu Pipelines.	27. Januar 2016
Aktualisiertes Thema	Zu den ressourcenbasierten Richtlinien für Authentif izierung 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

Änderungen	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 einer anderen verwendet werden AWS Konto hinzugefügt. Zu Authentifizierung und Zugriffskontrolle wurde ein neuer Abschnitt hinzugefügt.	25. August 2015
Aktualisiertes Thema	Das Thema Erstellen Sie eine benutzerdefinierte Aktion und fügen Sie sie hinzu in CodePipeline wurde aktualisiert, 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 CodePipeline aktualisiert .	6. August 2015
Neues Thema	Das Thema Problembehebung CodePipeline wurde hinzugefügt. Aktualisierte Schritte wurden für IAM-Rollen und Jenkins hinzugefügt. Tutorial: Erstellen einer vierstufigen 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
Aktualisiertes Thema	In Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) wurde eine temporäre Umgebung für Probleme beim Herunterladen von Beispieldateien hinzugefügt.	17. Juli 2015

Änderungen	Beschreibung	Datum geändert
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 Benutzerh andbuchs CodePipeline .	9. Juli 2015

CodePipeline Feature-Referenz

Dieser Abschnitt ist eine allgemeine Referenz für Funktionen in der CodePipeline Dokumentation. Eine konzeptionelle Übersicht über die Pipeline-Struktur finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

Diese Tabelle wird regelmäßig mit Referenzinformationen zu Funktionen aktualisiert.

Datum der Tabellenaktualisierung	Funktion	CodePipeline Benutzerleitfaden	CodePipeline API-Leitfaden	CloudFormation Referenz	AWS CDK Referenz — L1-Konstrukte
15. Oktober 2024	Automatischer Wiederholungsversuch bei Stufenausfall	Konfigurieren Sie eine Phase für die automatische Wiederholung bei einem Fehler	RetryConfiguration		
15. Oktober 2024	Ergebnis für BeforeEntry-Bedingungen überspringen	Eingabebedingungen mit Ergebnis und `VariableCheck` Regel überspringen (Konsole) erstellen	FailureConditions		
03. Oktober 2024	Befehlsaktion mit neuer Compute-Kategorie	Aktionsreferenz für Befehle	ActionDeclaration		

Datum der Tabellenaktualisierung	Funktion	CodePipeline Benutzerleitfaden	CodePipeline API-Leitfaden	CloudFormation Referenz	AWS CDK Referenz — L1-Konstrukte
28. August 202	Typ BeforeEntry für Phasenbedingungen	Bedingungen für eine Phase konfigurieren	BeforeEntryConditions	AWS::CodePipeline::Pipeline::BeforeEntryConditions	BeforeEntry
28. August 202	Typ OnSuccess für Stufenbedingungen	Bedingungen für eine Phase konfigurieren	SuccessConditions	AWS::CodePipeline::Pipeline::SuccessBedingungen	onSuccess
26. April 2024	Stage-Rollbacks und OnFailure-Typ für Stufenbedingungen	Konfiguration des Stage-Rollbacks	RollbackStage	onFailure	onFailure
08. Februar 2024	Ausführungsmodi PARALLEL und QUEUED	Den Pipeline-Ausführungsmodus festlegen oder ändern	PipelineExecution	ExecutionMode	Ausführungsmodus
17. November 2	Überschreibungen der Quelle	Starten Sie eine Pipeline mit einer Quellrevisionsüberschreibung	SourceRevisionOverride	N/A	N/A

Datum der Tabellenaktualisierung	Funktion	CodePipeline Benutzerleitfaden	CodePipeline API-Leitfaden	CloudFormation Referenz	AWS CDK Referenz — L1-Konstrukte
24. Oktober 202	Typen von Pipelines	Welcher Pipeline-Typ ist der richtige für mich?	PipelineDeclaration	PipelineType	Aufzählung PipelineType
24. Oktober 202	Variablen auf Pipeline-Ebene	Tutorial: Variablen auf Pipeline-Ebene verwenden	PipelineVariable	AWS::CodePipeline::Pipeline::PipelineVariableDeclaration	Variablen auf Pipeline-Ebene
24. Oktober 202	Triggert und filtert Dateipfade/ Branches/ Pull-Anfragen	Automatisieren Sie das Starten von Pipelines mithilfe von Triggern und Filtern Tutorial: Filtere nach Branch-Namen für Pull-Requests, um deine Pipeline zu starten (Typ V2)	ActionDeclaration	AWS::CodePipeline::Pipeline::PipelineTriggerDeclaration	Auslöser

Datum der Tabellenaktualisierung	Funktion	CodePipeline Benutzerleitfaden	CodePipeline API-Leitfaden	CloudFormation Referenz	AWS CDK Referenz — L1-Konstrukte
17. Oktober 202	Versuchen Sie es erneut mit einer Phase	Konfiguration des Wiederholungsversuchs für eine fehlgeschlagene Phase oder für fehlgeschlagene Aktionen	ActionDeclaration	N/A	N/A

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.