Leitfaden

AWS CodePipeline

API-Version 2015-07-09

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS CodePipeline: Leitfaden

Copyright © 2024 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 ist CodePipeline? 1
Kontinuierliche Bereitstellung und kontinuierliche Integration 1
Was kann ich mit machen CodePipeline? 2
Ein kurzer Blick auf CodePipeline 3
Was sind die ersten Schritte mit CodePipeline? 4
Konzepte 4
Pipelines 5
Phasen 5
Aktionen 5
Pipeline-Ausführungen 6
Stufenausführungen 7
Aktionsausführungen 7
Aktionstypen 8
Übergänge 8
-Artefakte 9
Quell-Revisionen 9
Auslöser 10
Variablen 10
DevOps Pipeline-Beispiel 11
So funktionieren Pipeline-Ausführungen 13
So werden Pipeline-Ausführungen gestartet 14
Pipeline-Ausführungen beenden 14
So werden Ausführungen im SUPERSEDED-Modus verarbeitet 18
So werden Ausführungen im QUEUED-Modus verarbeitet 19
So werden Ausführungen im PARALLEL-Modus verarbeitet 21
Verwalten des Pipeline-Flows 21
Eingabe- und Ausgabe-Artefakte 24
Pipeline-Typen 27
Welche Art von Pipeline ist für mich die richtige? 28
Erste Schritte 29
Schritt 1: Erstellen Sie einen AWS-Konto Benutzer mit Administratorrechten 29
So melden Sie sich für ein AWS-Konto an 29
Einen Administratorbenutzer erstellen 30

Schritt 2: Wenden Sie eine verwaltete Richtlinie für den Administratorzugriff auf an CodePipeline ... 31
Schritt 3: Installieren Sie den AWS CLI ... 33
Schritt 4: Öffnen Sie die Konsole für CodePipeline ... 34
Nächste Schritte ... 34
Produkt- und Service-Integrationen ... 35
Integrationen mit CodePipeline Aktionstypen ... 35
Quellaktions-Integrationen ... 35
Erstellen von Aktionsintegrationen ... 43
Testen von Aktionsintegrationen ... 45
Bereitstellungsaktions-Integrationen ... 47
Integration von Genehmigungsaktionen in Amazon Simple Notification Service ... 54
Aufrufen von Aktionsintegrationen ... 54
Allgemeine Integrationen mit CodePipeline ... 56
Beispiele aus der Community ... 59
Blog-Posts ... 59
Tutorials ... 64
Tutorial: Verwende Git-Tags, um deine Pipeline zu starten ... 65
Voraussetzungen ... 66
Schritt 1: Öffne CloudShell und klone dein Repository ... 67
Schritt 2: Eine Pipeline zum Auslösen auf Git-Tags erstellen ... 67
Schritt 3: Markiere deine Commits zur Veröffentlichung ... 71
Schritt 4: Änderungen veröffentlichen und Logs einsehen ... 73
Tutorial: Filtern nach Verzweigungsnamen für Pull-Anforderungen zum Starten Ihrer Pipeline ... 73
Voraussetzungen ... 73
Schritt 1: Erstellen einer Pipeline, um bei der Pull-Anforderung für bestimmte Verzweigungen zu beginnen ... 74
Schritt 2: Erstellen und Zusammenführen einer Pull-Anforderung in GitHub.com, um Ihre Pipeline-Ausführungen zu starten ... 77
Tutorial: Verwenden Sie Variablen auf Pipeline-Ebene ... 78
Voraussetzungen ... 78
Schritt 1: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt ... 79
Schritt 2: Änderungen veröffentlichen und Protokolle anzeigen ... 83
Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) ... 83
Erstellen eines S3-Buckets ... 84

Erstellen Sie Windows Server Amazon EC2 EC2-Instances und installieren Sie den Agenten CodeDeploy ... 86
Erstellen Sie eine Anwendung in CodeDeploy ... 89
Erstellen Ihrer ersten Pipeline ... 91
Hinzufügen einer weiteren Phase ... 94
Deaktivieren und Aktivieren von Übergängen zwischen Phasen ... 101
Bereinigen von Ressourcen ... 103
Tutorial: Erstellen Sie eine einfache Pipeline (CodeCommit Repository) ... 103
Erstellen Sie ein CodeCommit Repository ... 104
Herunterladen und Push-Übertragen Ihres Codes sowie Durchführen eines Commits für Ihren Code ... 105
Erstellen Sie eine Amazon EC2 EC2-Linux-Instance und installieren Sie den Agenten CodeDeploy ... 108
Erstellen Sie eine Anwendung in CodeDeploy ... 111
Erstellen Ihrer ersten Pipeline ... 112
Aktualisiere den Code in deinem CodeCommit Repository ... 115
Bereinigen von Ressourcen ... 117
Weitere Informationen ... 118
Tutorial: Erstellen einer vierstufigen Pipeline ... 118
Erfüllen der Voraussetzungen ... 119
Erstellen Sie eine Pipeline ... 124
Hinzufügen weiterer Phasen ... 126
Bereinigen von Ressourcen ... 129
Tutorial: Einrichten einer CloudWatch Events-Regel, um E-Mail-Benachrichtigungen zu Änderungen am Pipelinestatus zu empfangen ... 130
Einrichten einer E-Mail-Benachrichtigung mit Amazon SNS ... 131
Erstellen Sie eine CloudWatch Events-Benachrichtigungsregel für CodePipeline ... 133
Bereinigen von Ressourcen ... 134
Tutorial: Erstellen und Testen einer Android-App mit AWS Device Farm ... 135
Konfigurieren Sie CodePipeline für die Verwendung Ihrer Device Farm-Tests ... 136
Tutorial: Testen Sie eine iOS-App mit AWS Device Farm ... 141
Für CodePipeline die Verwendung Ihrer Device Farm Farm-Tests konfigurieren (Beispiel Amazon S3) ... 142
Tutorial: Erstellen Sie eine Pipeline, die in Service Catalog bereitgestellt wird ... 147
Option 1: Bereitstellung in Service Catalog ohne Konfigurationsdatei ... 148
Option 2: Bereitstellung in Service Catalog mithilfe einer Konfigurationsdatei ... 153

Tutorial: Erstellen einer Pipeline mit AWS CloudFormation 158
Beispiel 1: Erstellen einer AWS CodeCommit-Pipeline mit AWS CloudFormation 158
Beispiel 2: Erstellen einer Amazon-S3-Pipeline mitAWS CloudFormation 161
Tutorial: Erstellen einer Pipeline, die Variablen aus AWS CloudFormation-Bereitstellungsaktionen verwendet 165
Voraussetzungen: Erstellen Sie eine AWS CloudFormation Servicerolle und ein CodeCommit Repository 166
Schritt 1: Herunterladen, Bearbeiten und Hochladen der AWS CloudFormation-Beispielvorlage 166
Schritt 2: Erstellen der Pipeline 168
Schritt 3: Hinzufügen einer AWS CloudFormation-Bereitstellungsaktion zum Erstellen des Änderungssatzes 170
Schritt 4: Hinzufügen einer manuellen Genehmigungsaktion 172
Schritt 5: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Änderungssatz auszuführen 172
Schritt 6: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Stack zu löschen 173
Tutorial: Amazon-ECS-Standardbereitstellung mit CodePipeline 174
Voraussetzungen 174
Schritt 1: Hinzufügen einer Build-Spezifikationsdatei zu Ihrem Quell-Repository 177
Schritt 2: Erstellen einer durchgängigen Bereitstellungs-Pipeline 179
Schritt 3: Hinzufügen von Amazon-ECR-Berechtigungen zur CodeBuild Rolle 181
Schritt 4: Testen Ihrer Pipeline 182
Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy 182
Voraussetzungen 184
Schritt 1: Image erstellen und in ein Amazon ECR-Repository übertragen 185
Schritt 2: Erstellen Sie Aufgabendefinitions- und AppSpec Quelldateien und übertragen Sie sie in ein Repository CodeCommit 186
Schritt 3: Erstellen Ihres Application Load Balancer und der Zielgruppen 190
Schritt 4: Erstellen Sie Ihren Amazon ECS-Cluster und -Service 193
Schritt 5: Erstellen Ihrer CodeDeploy-Anwendung und Bereitstellungsgruppe (ECS-Datenverarbeitungsplattform) 196
Schritt 6: Erstellen Ihrer Pipeline 197
Schritt 7: Vornehmen einer Änderung an Ihrer Pipeline und Überprüfen der Bereitstellung ... 202
Tutorial: Erstellen einer Pipeline zum Bereitstellen eines Amazon Alexa-Skills 202

Voraussetzungen 203
Schritt 1: Erstellen eines Alexa Developer Services-LWA-Sicherheitsprofils 203
Schritt 2: Erstellen Sie Alexa-Skill-Quelldateien und übertragen Sie sie in Ihr CodeCommit Repository 203
Schritt 3: Verwenden von ASK-CLI-Befehlen zum Erstellen eines Aktualisierungs-Tokens ... 206
Schritt 4: Erstellen Ihrer Pipeline 206
Schritt 5: Vornehmen einer Änderung an einer beliebigen Quelldatei und Überprüfen der Bereitstellung 209
Tutorial: Erstellen Sie eine Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet 209
Option 1: Statische Website-Dateien auf Amazon S3 bereitstellen 210
Option 2: Bereitstellen von erstellten Archivdateien aus einem S3-Quell-Bucket in Amazon S3 215
Tutorial: Veröffentlichen von Anwendungen in der AWS Serverless Application Repository 221
Bevor Sie beginnen 222
Schritt 1: Erstellen einer buildspec.yml-Datei 222
Schritt 2: Erstellen und Konfigurieren Ihrer Pipeline 223
Schritt 3: Bereitstellen der Veröffentlichungsanwendung 225
Schritt 4: Erstellen der Veröffentlichungsaktion 226
Tutorial: Verwenden von Variablen mit Lambda-Aufrufaktionen 226
Voraussetzungen 227
Schritt 1: Erstellen einer Lambda-Funktion 227
Schritt 2: Fügen Sie Ihrer Pipeline eine Lambda-Aufrufaktion und eine manuelle Genehmigungsaktion hinzu 230
Tutorial: Verwenden einerAWS Step FunctionsAufrufen von Aktion 232
Voraussetzung: Erstellen oder Auswählen einer einfachen Pipeline 233
Schritt 1: Erstellen des Beispielzustandsautomaten 233
Schritt 2: Aufrufen Step Functions -Aufrufaktion zu Ihrer Pipeline hinzufügen 234
Tutorial: Eine Pipeline erstellen, die AppConfig als Bereitstellungsanbieter verwendet wird 235
Voraussetzungen 236
Schritt 1: Erstellen Sie Ihre Ressourcen AWS AppConfig 236
Schritt 2: Laden Sie Dateien in Ihren S3-Quell-Bucket hoch 237
Schritt 3: Erstellen Ihrer Pipeline 237
Schritt 4: Nehmen Sie eine Änderung an einer beliebigen Quelldatei vor und überprüfen Sie die Bereitstellung 239
Tutorial: Verwenden Sie den vollständigen Klon mit einer GitHub Pipeline-Quelle 239
Voraussetzungen 240

Schritt 1: Erstellen Sie eine README-Datei ... 240
Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt ... 241
Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um Verbindungen zu verwenden ... 245
Schritt 4: Repository-Befehle in der Build-Ausgabe anzeigen ... 245
Tutorial: Verwenden Sie den vollständigen Klon mit einer CodeCommit Pipeline-Quelle ... 245
Voraussetzungen ... 246
Schritt 1: Erstellen Sie eine README-Datei ... 246
Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt ... 247
Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um das Repository zu klonen ... 250
Schritt 4: Repository-Befehle in der Build-Ausgabe anzeigen ... 250
Tutorial: Erstellen Sie eine Pipeline mit AWS CloudFormation StackSets Bereitstellungsaktionen ... 250
Voraussetzungen ... 251
Schritt 1: Laden Sie die AWS CloudFormation Beispielvorlage und die Parameterdatei hoch ... 252
Schritt 2: Erstellen der Pipeline ... 168
Schritt 3: Erste Bereitstellung anzeigen ... 257
Schritt 4: Fügen Sie eine CloudFormationStackInstances Aktion hinzu ... 257
Schritt 5: Sehen Sie sich die Stackset-Ressourcen für Ihre Bereitstellung an ... 258
Schritt 6: Nehmen Sie ein Update für Ihr Stack-Set vor ... 259
Bewährte Methoden und Anwendungsfälle ... 260
Beispiele für die Verwendung von CodePipeline ... 260
Verwenden von CodePipeline mit Amazon S3 AWS CodeCommit, und AWS CodeDeploy .. 261
Verwenden von CodePipeline mit Aktionsanbietern von Drittanbietern (GitHub und Jenkins) ... 261
Verwenden von CodePipeline mit AWS CodeStar zum Erstellen einer Pipeline in einem Codeprojekt ... 262
Verwenden von CodePipeline zum Kompilieren, Erstellen und Testen von Code mit CodeBuild ... 262
Verwenden von CodePipeline mit Amazon ECS für die kontinuierliche Bereitstellung von containerbasierten Anwendungen in der Cloud ... 263
Verwenden von CodePipeline mit Elastic Beanstalk für die kontinuierliche Bereitstellung von Webanwendungen in der Cloud ... 263

Verwenden von CodePipeline mit AWS Lambda für die kontinuierliche Bereitstellung von Lambda-basierten und Serverless-Anwendungen ... 263
Verwenden von CodePipeline mit - AWS CloudFormation Vorlagen für die kontinuierliche Bereitstellung in der Cloud ... 263
Markieren von Ressourcen ... 264
CodePipeline Mit Amazon VPC verwenden ... 266
Verfügbarkeit ... 266
Erstellen eines VPC-Endpunkts für CodePipeline ... 267
Fehlerbehebung für Ihre VPC-Einrichtung ... 268
Arbeiten mit Pipelines ... 269
Starten einer Pipeline in CodePipeline ... 270
Quellaktionen und Methoden zur Erkennung von Änderungen ... 271
Manuelles Starten einer Pipeline ... 273
Starten einer Pipeline nach einem Zeitplan ... 275
Starten einer Pipeline mit einer Quellrevisionsüberschreibung ... 278
Pipeline-Ausführung anhalten ... 280
Pipeline-Ausführung beenden (Konsole) ... 281
Stoppen Sie eine eingehende Ausführung (Konsole) ... 285
Pipeline-Ausführung anhalten (CLI) ... 286
Stoppen einer eingehenden Ausführung (CLI) ... 287
Erstellen Sie eine Pipeline ... 288
Erstellen einer Pipeline (Konsole) ... 289
Erstellen einer Pipeline (CLI) ... 302
Amazon-ECR-Quellaktionen und EventBridge ... 309
Amazon S3-Quellaktionen und EventBridge ... 319
Bitbucket-Cloud-Verbindungen ... 340
CodeCommit -Quellaktionen und EventBridge ... 347
GitHub Verbindungen ... 362
GitHub Enterprise Server-Verbindungen ... 369
GitLab.com-Verbindungen ... 378
Verbindungen für GitLab selbstverwaltete ... 386
Bearbeiten einer Pipeline ... 394
Bearbeiten einer Pipeline (Konsole) ... 395
Bearbeiten einer Pipeline (AWS CLI) ... 399
Pipelines und Details anzeigen ... 404
Pipelines anzeigen (Konsole) ... 404

Aktionsdetails in einer Pipeline (Konsole) anzeigen ... 409
Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen ... 412
Anzeigen von der Pipeline-Details und des Verlaufs (CLI) ... 413
Pipeline löschen ... 414
Löschen einer Pipeline (Konsole) ... 414
Löschen einer Pipeline (CLI) ... 414
Erstellen einer Pipeline, die Ressourcen aus einem anderen Konto verwendet ... 416
Voraussetzung: Erstellen eines AWS KMS-Verschlüsselungsschlüssels ... 418
Schritt 1: Einrichten von Kontorichtlinien und Rollen ... 419
Schritt 2: Bearbeiten der Pipeline ... 428
Migrieren Sie Abfrage-Pipelines, um die ereignisbasierte Änderungserkennung zu nutzen ... 431
Wie migriert man Polling-Pipelines ... 432
Polling-Pipelines in Ihrem Konto anzeigen ... 433
Migrieren Sie Abfrage-Pipelines mit einer Quelle CodeCommit ... 438
Migrieren Sie Abfrage-Pipelines mit einer für Ereignisse aktivierten S3-Quelle ... 459
Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail ... 487
Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Verbindungen ... 522
Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Webhooks ... 526
CodePipeline-Servicerolle erstellen ... 544
Erstellen der CodePipeline-Servicerolle (Konsole) ... 544
Erstellen der CodePipeline-Servicerolle (CLI) ... 545
Markieren einer Pipeline ... 549
Markieren von Pipelines (Konsole) ... 549
Markieren von Pipelines (CLI) ... 551
Erstellen einer Benachrichtigungsregel ... 553
Arbeiten mit Auslösern ... 558
Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen ... 558
Überlegungen zu Auslöserfiltern ... 561
Beispiele für Auslöserfilter ... 561
Filtern nach Push-Ereignissen (Konsole) ... 563
Filtern nach Pull-Anforderungen (Konsole) ... 564
Auslöserfilterung in der Pipeline-JSON (CLI) ... 566
Auslöserfilterung in AWS CloudFormation Vorlagen ... 570
Verwalten von Ausführungen ... 572
Ausführungen anzeigen ... 572

Ausführungsverlauf einer Pipeline anzeigen (Konsole) ... 572
Anzeige des Ausführungsstatus (Konsole) ... 573
Eine eingehende Ausführung anzeigen (Konsole) ... 575
Anzeigen von Revisionen der Pipeline-Ausführungsquelle (Konsole) ... 576
Anzeigen von Aktionsausführungen (Konsole) ... 578
Anzeigen von Informationen zu Aktionsartefakten und zum Artefaktspeicher (Konsole) ... 579
Anzeigen von der Pipeline-Details und des Verlaufs (CLI) ... 579
Festlegen des Pipeline-Ausführungsmodus ... 591
Festlegen des Pipeline-Ausführungsmodus (Konsole) ... 592
Festlegen des Pipeline-Ausführungsmodus (CLI) ... 592
Wiederholen Sie eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase ... 596
Versuchen Sie es erneut mit einer fehlgeschlagenen Phase (Konsole) ... 597
Eine fehlgeschlagene Phase erneut versuchen (CLI) ... 598
Verwenden von Aktionen ... 602
Arbeiten mit Aktionstypen ... 602
Beantragen Sie einen Aktionstyp ... 604
Hinzufügen eines verfügbaren Aktionstyps zu einer Pipeline (Konsole) ... 610
Einen Aktionstyp anzeigen ... 612
Einen Aktionstyp aktualisieren ... 614
Erstellen einer benutzerdefinierten Aktion für eine Pipeline ... 616
Erstellen einer benutzerdefinierten Aktion ... 618
Erstellen eines Auftragsworkers für Ihre benutzerdefinierte Aktion ... 622
Hinzufügen einer benutzerdefinierten Aktion zu einer Pipeline ... 630
Markieren einer benutzerdefinierten Aktion in CodePipeline ... 633
Hinzufügen von Tags zu benutzerdefinierten Aktionen ... 634
Anzeigen von Tags für eine benutzerdefinierte Aktion ... 634
Bearbeiten von Tags für eine benutzerdefinierte Aktion ... 635
Entfernen von Tags aus einer benutzerdefinierten Aktion ... 635
Aufrufen einer Lambda-Funktion in einer Pipeline ... 636
Schritt 1: Erstellen einer Pipeline ... 638
Schritt 2: Erstellen einer Lambda-Funktion ... 639
Schritt 3: Fügen Sie die Lambda-Funktion zu einer Pipeline in derCodePipeline Konsole hinzu ... 644
Schritt 4: Testen der Pipeline mit der Lambda-Funktion ... 645
Schritt 5: Nächste Schritte ... 645
JSON-Beispielereignis ... 646

Weitere Beispielfunktionen ... 648
Wiederholen Sie eine fehlgeschlagene Aktion in einer Phase ... 661
Wiederholen fehlgeschlagener Aktionen (Konsole) ... 662
Wiederholen fehlgeschlagener Aktionen (CLI) ... 664
Verwalten von Genehmigungsaktionen in Pipelines ... 666
Konfigurationsoptionen für manuelle Genehmigungsaktionen ... 667
Setup- und Workflow-Übersicht für Genehmigungsaktionen ... 668
Erteilen Sie einem IAM-Benutzer Genehmigungsberechtigungen in CodePipeline ... 669
Gewähren Sie Amazon SNS SNS-Berechtigungen für eine Servicerolle ... 672
Hinzufügen einer manuellen Genehmigungsaktion ... 674
Erlauben oder Ablehnen einer Genehmigungsaktion ... 679
JSON-Datenformat für manuelle Genehmigungsbenachrichtigungen ... 683
Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline ... 684
Verwalten von regionsübergreifenden Aktionen in einer Pipeline (Konsole) ... 686
Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline (CLI) ... 689
Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline (AWS CloudFormation) ... 695
Arbeiten mit Variablen ... 697
Konfigurieren von Aktionen für Variablen ... 698
Anzeigen von Ausgabevariablen ... 703
Beispiel: Variablen in manuellen Genehmigungen verwenden ... 706
Beispiel: Verwenden Sie eine BranchName Variable mit CodeBuild Umgebungsvariablen ... 706
Arbeiten mit Phasenübergängen ... 709
Deaktivieren oder Aktivieren von Übergängen (Konsole) ... 709
Deaktivieren oder Aktivieren von Übergängen (CLI) ... 711
Überwachung von Pipelines ... 714
Überwachen von CodePipeline Ereignissen ... 715
Detailtypen ... 717
Ereignisse auf Pipeline-Ebene ... 719
Ereignisse auf Stufenebene ... 728
Ereignisse auf Aktionsebene ... 732
Erstellen einer Regel, die eine Benachrichtigung über ein Pipeline-Ereignis sendet ... 740
Platzhalter-Bucket-Referenz für Ereignisse ... 745
Platzhalter-Bucket-Namen für Ereignisse nach Region ... 746
Protokollieren von API-Aufrufen mit AWS CloudTrail ... 749
CodePipeline-Informationen in CloudTrail ... 749
Grundlagen zu CodePipeline-Protokolldateieinträgen ... 750

Fehlerbehebung 753
Pipeline-Fehler: Eine mit AWS Elastic Beanstalk konfigurierte Pipeline gibt eine Fehlermeldung zurück: „Bereitstellung fehlgeschlagen. Die bereitgestellte Rolle verfügt nicht über ausreichende Berechtigungen: Service:AmazonElasticLoadBalancing“ 754
Bereitstellungsfehler: Eine Pipeline, die mit einer AWS Elastic Beanstalk Bereitstellungsaktion konfiguriert ist, hängt, anstatt fehlzuschlagen, wenn die Berechtigung „DescribeEvents“ fehlt ... 755
Pipeline-Fehler: Eine Quellaktion gibt die Meldung mit unzureichenden Berechtigungen zurück: „Kann nicht auf das CodeCommit Repository zugreifen`repository-name`. Überprüfen Sie, ob die Pipeline-IAM-Rolle über ausreichende Berechtigungen für den Zugriff auf das Repository verfügt.“ 755
Pipeline-Fehler: Ein Jenkins-Build oder eine Testaktion werden über eine lange Zeit ausgeführt und schlagen dann aufgrund fehlender Anmeldeinformationen oder Berechtigungen fehl. 756
Pipeline-Fehler: Eine Pipeline, die in einer AWS Region mit einem Bucket erstellt wurde, der in einer anderen AWS Region erstellt wurde, gibt ein „InternalError“ mit dem Code „JobFailed“ zurück 757
Bereitstellungsfehler: Eine ZIP-Datei mit einer WAR-Datei wird erfolgreich für AWS Elastic Beanstalk bereitgestellt, die Anwendungs-URL meldet jedoch den Fehler „404 nicht gefunden“. 755
Pipeline-Artefakt-Ordnernamen werden gekürzt 758
Hinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder GitLab.com 758
Hinzufügen von CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen 760
Pipeline-Fehler: Eine Bereitstellung mit der CodeDeployToECS-Aktion gibt eine Fehlermeldung zurück: „Ausnahme beim Versuch, die Artefaktdatei der Aufgabendefinition zu lesen aus: <Name des Quellartefakts>“ 761
GitHub Quellaktion der Version 1: Die Repository-Liste zeigt verschiedene Repositorys 761
GitHub Quellaktion der Version 2: Verbindung für ein Repository kann nicht hergestellt werden 762
Amazon S3 error: CodePipeline service role <ARN> erhält eine S3-Zugriffsverweigerung für den S3-Bucket <BucketName> 762
Pipelines mit einer Amazon S3-, Amazon-ECR- oder CodeCommit -Quelle werden nicht mehr automatisch gestartet 765
Verbindungsfehler beim Herstellen einer Verbindung mit GitHub: „Ein Problem ist aufgetreten, stellen Sie sicher, dass Cookies in Ihrem Browser aktiviert sind“ oder „Ein Organisationsbesitzer muss die GitHub App installieren“ 766

Fehler, wenn die CloudFormationStackInstances Aktionen CloudFormationStackSet oder in einer Region nicht verfügbar sind ... 767
Pipelines, deren Ausführungsmodus in den Modus QUEUED oder PARALLEL geändert wurde, schlagen fehl, wenn das Ausführungslimit erreicht ist ... 768
Pipelines im PARALLEL-Modus haben eine veraltete Pipeline-Definition, wenn sie beim Wechsel in den Modus QUEUED oder SUPERSEDED bearbeitet werden ... 768
Pipelines, die aus dem PARALLEL-Modus geändert wurden, zeigen einen vorherigen Ausführungsmodus an ... 768
Pipelines mit Verbindungen, die Auslöserfilterung nach Dateipfaden verwenden, beginnen möglicherweise nicht bei der Erstellung von Verzweigungen ... 769
Pipelines mit Verbindungen, die Auslöserfilterung nach Dateipfaden verwenden, werden möglicherweise nicht gestartet, wenn das Dateilimit erreicht ist ... 769
Benötigen Sie Hilfe für ein anderes Problem? ... 770
Sicherheit ... 771
Datenschutz ... 772
Richtlinie für den Datenverkehr zwischen Netzwerken ... 773
Verschlüsselung im Ruhezustand ... 774
Verschlüsselung während der Übertragung ... 774
Verwaltung von Verschlüsselungsschlüsseln ... 774
Konfigurieren der serverseitigen Verschlüsselung für Artefakte, die in Amazon S3 für gespeichert sind CodePipeline ... 774
Verwenden von AWS Secrets Manager zum Nachverfolgen von Datenbankpasswörtern oder API-Schlüsseln von Drittanbietern ... 778
Identity and Access Management ... 778
Zielgruppe ... 779
Authentifizierung mit Identitäten ... 780
Verwalten des Zugriffs mit Richtlinien ... 783
Funktionsweise AWS CodePipeline von mit IAM ... 785
Beispiele für identitätsbasierte Richtlinien ... 792
Beispiele für eine ressourcenbasierte Richtlinie ... 830
Fehlerbehebung ... 831
CodePipeline -Berechtigungsreferenz ... 834
Verwalten der CodePipeline Servicerolle ... 844
Vorfallreaktion ... 856
Compliance-Validierung ... 857
Ausfallsicherheit ... 858

Sicherheit der Infrastruktur ... 859
Bewährte Methoden für die Gewährleistung der Sicherheit ... 859
Befehlszeilenreferenz ... 861
Referenz der Pipeline-Struktur ... 862
Gültige Aktionstypen und Anbieter in CodePipeline ... 862
Anforderungen an die Pipeline- und Phasenstruktur in CodePipeline ... 867
Anforderungen an die Aktionsstruktur in CodePipeline ... 870
Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp ... 876
Standardeinstellungen für den Parameter PollForSourceChanges ... 878
Konfigurationsdetails nach Anbietertyp ... 880
Referenz der Aktionsstruktur ... 882
Amazon ECR ... 883
Aktionstyp ... 883
Konfigurationsparameter ... 884
Input artifacts (Eingabeartefakte) ... 884
Ausgabeartefakte ... 884
Ausgabevariablen ... 884
Aktionsdeklaration (Beispiel für Amazon ECR) ... 885
Weitere Informationen finden Sie auch unter ... 886
Amazon ECS undCodeDeploy Blaugrün ... 887
Aktionstyp ... 888
Konfigurationsparameter ... 888
Input artifacts (Eingabeartefakte) ... 890
Ausgabeartefakte ... 891
Aktionsdeklaration ... 891
Weitere Informationen finden Sie auch unter ... 892
Amazon Elastic Container Service ... 893
Aktionstyp ... 894
Konfigurationsparameter ... 894
Input artifacts (Eingabeartefakte) ... 895
Ausgabeartefakte ... 896
Aktionsdeklaration ... 896
Weitere Informationen finden Sie auch unter ... 897
Amazon S3 S3-Aktionen ... 897
Aktionstyp ... 898
Konfigurationsparameter ... 898

Input artifacts (Eingabeartefakte) 900
Ausgabeartefakte 900
Beispielaktionskonfiguration 900
Weitere Informationen finden Sie auch unter 903
Quellaktion für Amazon S3 903
Aktionstyp 905
Konfigurationsparameter 905
Input artifacts (Eingabeartefakte) 906
Ausgabeartefakte 906
Ausgabevariablen 906
Aktionsdeklaration 907
Weitere Informationen finden Sie auch unter 908
AWSAppConfig 908
Aktionstyp 909
Konfigurationsparameter 909
Input artifacts (Eingabeartefakte) 910
Ausgabeartefakte 910
Beispielaktionskonfiguration 910
Weitere Informationen finden Sie auch unter 911
AWS CloudFormation 912
Aktionstyp 912
Konfigurationsparameter 913
Input artifacts (Eingabeartefakte) 918
Ausgabeartefakte 918
Ausgabevariablen 918
Aktionsdeklaration 919
Weitere Informationen finden Sie auch unter 920
AWS CloudFormation StackSets 921
Wie funktionieren Aktionen AWS CloudFormation StackSets 922
Wie strukturiert man StackSets Aktionen in einer Pipeline 924
`CloudFormationStackSet`-Aktion 926
CloudFormationStackInstances-Aktion 939
Berechtigungsmodelle für Stack-Set-Operationen 949
Datentypen von Vorlagenparametern 950
Weitere Informationen finden Sie auch unter 920
AWS CodeBuild 952

Aktionstyp ... 952
Konfigurationsparameter ... 953
Input artifacts (Eingabeartefakte) ... 955
Ausgabeartefakte ... 955
Ausgabevariablen ... 956
Aktionsdeklaration (CodeBuild-Beispiel) ... 956
Weitere Informationen finden Sie auch unter ... 958
AWS CodeCommit ... 958
Aktionstyp ... 959
Konfigurationsparameter ... 959
Input artifacts (Eingabeartefakte) ... 961
Ausgabeartefakte ... 961
Ausgabevariablen ... 961
Beispielaktionskonfiguration ... 963
Weitere Informationen finden Sie auch unter ... 965
AWS CodeDeploy ... 965
Aktionstyp ... 966
Konfigurationsparameter ... 966
Input artifacts (Eingabeartefakte) ... 966
Ausgabeartefakte ... 967
Aktionsdeklaration ... 967
Weitere Informationen finden Sie auch unter ... 968
CodeStarSourceConnection für Bitbucket Cloud GitHub-, GitHub Enterprise Server-, GitLab.com- und GitLab selbstverwaltete Aktionen ... 969
Aktionstyp ... 972
Konfigurationsparameter ... 973
Input artifacts (Eingabeartefakte) ... 974
Ausgabeartefakte ... 974
Ausgabevariablen ... 975
Aktionsdeklaration ... 976
Installieren der Installations-App und Erstellen einer Verbindung ... 977
Weitere Informationen finden Sie auch unter ... 978
AWS Device Farm ... 978
Aktionstyp ... 979
Konfigurationsparameter ... 979
Input artifacts (Eingabeartefakte) ... 983

Ausgabeartefakte ... 984
Aktionsdeklaration ... 984
Weitere Informationen finden Sie auch unter ... 985
AWS Lambda ... 986
Aktionstyp ... 986
Konfigurationsparameter ... 987
Input artifacts (Eingabeartefakte) ... 987
Ausgabeartefakte ... 987
Ausgabevariablen ... 987
Beispielaktionskonfiguration ... 987
JSON-Beispielereignis ... 989
Weitere Informationen finden Sie auch unter ... 991
Snyk ... 992
ID des Aktionstyps ... 992
Input artifacts (Eingabeartefakte) ... 993
Ausgabeartefakte ... 993
Weitere Informationen finden Sie auch unter ... 993
AWS Step Functions ... 993
Aktionstyp ... 994
Konfigurationsparameter ... 994
Input artifacts (Eingabeartefakte) ... 995
Ausgabeartefakte ... 995
Ausgabevariablen ... 996
Beispielaktionskonfiguration ... 996
Behavior ... 999
Weitere Informationen finden Sie auch unter ... 911
Integration-Subtyreferenz ... 1002
So funktionieren Aktionstypen von Drittanbietern mit dem Integrator ... 1002
Konzepte ... 1003
Unterstützte Integrationmodelle ... 1005
Lambda-Integrationmodell ... 1006
Aktualisieren Sie Ihre Lambda-Funktion, um die Eingabe von zu verarbeiten CodePipeline 1007
Geben Sie die Ergebnisse aus der Lambda-Funktion zurück CodePipeline ... 1011
Verwenden Sie Fortsetzungstoken, um auf Ergebnisse eines asynchronen Prozesses zu warten ... 1013

Geben Sie CodePipeline die Berechtigungen zum Aufrufen der Integrator-Lambda-Funktion zur Laufzeit an ... 1014
Modell zur Integration von Arbeitnehmern ... 1014
Wählen und Konfigurieren einer Strategie für die Berechtigungsverwaltung Ihres Auftragsworkers ... 1015
Referenz zu Abbild-Definitionsdateien ... 1017
imagedefinitions.json-Datei für Amazon ECS-SStandardbereitstellungsaktionen ... 1017
imageDetail.json-Datei für Amazon ECS-Blau/Grün-Bereitstellungsaktionen ... 1020
Variablen ... 1025
Konzepte ... 1026
Variablen ... 1026
Namespaces ... 1027
Anwendungsfälle für Variablen ... 1028
Konfigurieren von Variablen ... 1029
Konfigurieren von Variablen auf Pipeline-Ebene ... 1029
Konfigurieren von Variablen auf Aktionsebene ... 1030
Variablenauflösung ... 1032
Regeln für Variablen ... 1033
Für Pipeline-Aktionen verfügbare Variablen ... 1034
Aktionen mit definierten Variablenschlüsseln ... 1034
Aktionen mit benutzerdefinierten Variablenschlüsseln ... 1038
Arbeiten mit Glob-Mustern in der Syntax ... 1041
Aktivieren von Pipelines für die empfohlenen Methode zur Änderungserkennung ... 1043
Aktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2 ... 1044
Schritt 1: Ersetzen Sie Ihre GitHub Aktion für Version 1 ... 1045
Schritt 2: Stellen Sie eine Verbindung her zu GitHub ... 1046
Schritt 3: Speichern Sie Ihre GitHub Quellaktion ... 1047
Kontingente ... 1049
Anhang A: Quellaktionen der GitHub Version 1 ... 1067
Hinzufügen einer GitHub Version 1-Quellaktion ... 1068
GitHub Referenz zur Quellaktionsstruktur der Version 1 ... 1068
Aktionstyp ... 1069
Konfigurationsparameter ... 1070
Input artifacts (Eingabeartefakte) ... 1071
Ausgabeartefakte ... 1072

Ausgabevariablen 1072
Aktionsdeklaration (GitHub-Beispiel) 1073
Herstellen einer Verbindung mit GitHub (OAuth) 1074
Weitere Informationen finden Sie auch unter 1075
Dokumentverlauf 1076
Frühere Aktualisierungen 1103
AWS-Glossar 1115
..... mcxvi

Was ist AWS CodePipeline?

AWS CodePipeline ist ein kontinuierlicher Bereitstellungsservice, mit dem Sie die für die Freigabe Ihrer Software erforderlichen Schritte entwickeln, visualisieren und automatisieren können. Sie können die verschiedenen Phasen eines Software-Release-Prozesses schnell modellieren und konfigurieren. CodePipeline automatisiert die Schritte, die erforderlich sind, um Ihre Softwareänderungen kontinuierlich freizugeben. Weitere Informationen zu Preisen für finden Sie unter CodePipeline-Preise.

Themen

- Kontinuierliche Bereitstellung und kontinuierliche Integration
- Was kann ich mit machen CodePipeline?
- Ein kurzer Blick auf CodePipeline
- Was sind die ersten Schritte mit CodePipeline?
- CodePipeline -Konzepte
- DevOps Pipeline-Beispiel
- So funktionieren Pipeline-Ausführungen
- Eingabe- und Ausgabe-Artefakte
- Pipeline-Typen
- Welche Art von Pipeline ist für mich die richtige?

Kontinuierliche Bereitstellung und kontinuierliche Integration

CodePipeline ist ein kontinuierlicher Bereitstellungsservice, der das Erstellen, Testen und Bereitstellen 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.

Die kontinuierliche Integration ist eine Softwareentwicklungspraxis, bei der Mitglieder eines Teams ein Versionskontrollsystem verwenden und ihre Arbeit häufig an demselben Standort integrieren, z. B. an einem Hauptzweig. 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 Durchführen einer kontinuierlichen Integration und kontinuierlichen Bereitstellung auf AWS: Beschleunigung der 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 mit machen CodePipeline?

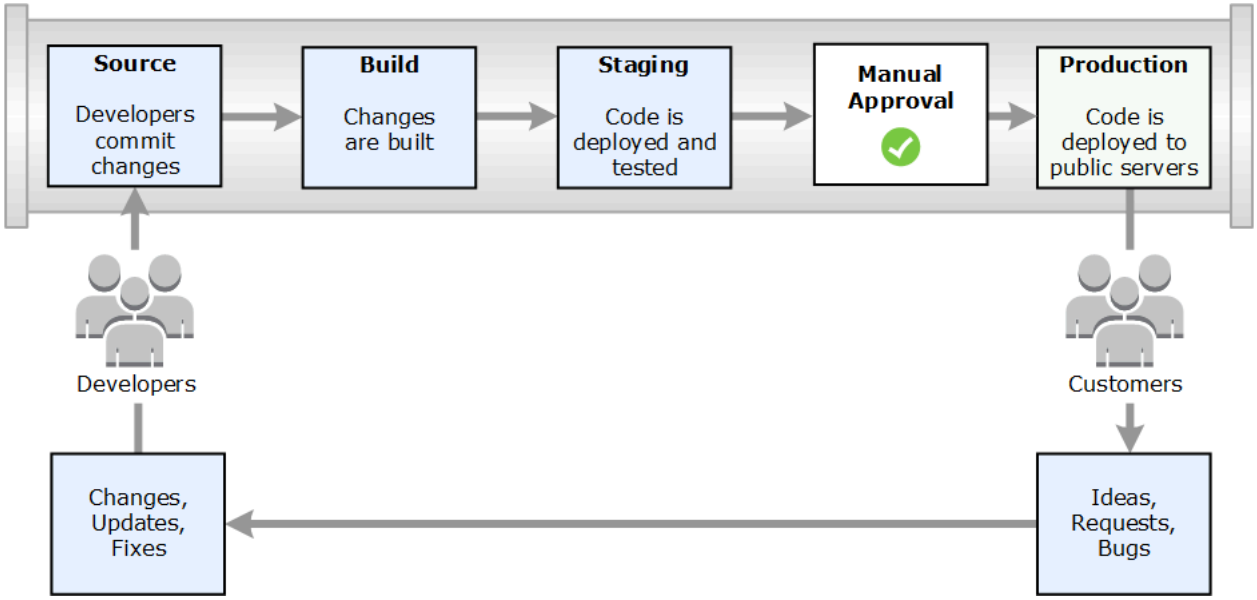
Sie können verwenden, CodePipeline um Ihre Anwendungen automatisch in der Cloud zu erstellen, zu testen und bereitzustellen. Insbesondere können Sie Folgendes:

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

- Fortschritt auf einen Blick anzeigen: Sie können den Echtzeitstatus Ihrer Pipelines überprüfen, die Details aller Warnungen überprüfen, fehlgeschlagene Phasen oder Aktionen wiederholen, Details zu den Quellrevisionen anzeigen, die in der letzten Pipeline-Ausführung in jeder Phase verwendet wurden, und eine 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. Führt auf dem Staging-Server weitere Tests CodePipeline aus, z. B. Integrations- oder Lasttests. Nach erfolgreichem Abschluss dieser Tests und nachdem eine manuelle Genehmigungsaktion, die der Pipeline hinzugefügt wurde, genehmigt wurde, CodePipeline stellt den getesteten und genehmigten Code für Produktions-Instances bereit.

CodePipeline kann Anwendungen auf EC2-Instances mithilfe von CodeDeploy, AWS Elastic Beanstalkoder bereitstellenAWS OpsWorks Stacks. CodePipeline kann auch containerbasierte

Anwendungen auf -Services mithilfe von Amazon ECS bereitstellen. Entwickler können die Integrationspunkte, die mit bereitgestellt werden, auch verwenden, CodePipeline um andere Tools oder Services hinzuzufügen, einschließlich Build-Services, Testanbietern oder anderen Bereitstellungszielen oder Systemen.

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

Was sind die ersten Schritte mit CodePipeline?

So beginnen Sie mit CodePipeline:

1. Erfahren Sie, wie CodePipeline funktioniert, indem Sie den CodePipeline -Konzepte Abschnitt lesen.
2. Bereiten Sie sich auf die Verwendung vor, CodePipeline indem Sie die Schritte unter befolgenErste Schritte mit CodePipeline.
3. Experiment ierenSie mit , CodePipeline indem Sie die Schritte in den CodePipeline - TutorialsTutorials befolgen.
4. Verwenden Sie CodePipeline für Ihre neuen oder vorhandenen Projekte, indem Sie die Schritte unter befolgenErstellen einer Pipeline in CodePipeline.

CodePipeline -Konzepte

Das Modellieren und Konfigurieren des automatisierten Freigabeprozesses ist einfacher, wenn Sie die Konzepte und Begriffe verstehen, die in AWS CodePipeline verwendet werden. Im Folgenden finden Sie einige Konzepte, die Sie kennen sollten, wenn Sie CodePipeline verwenden.

Ein Beispiel für eine DevOps Pipeline finden Sie unter DevOps Pipeline-Beispiel.

Die folgenden Begriffe werden in verwendet CodePipeline:

Themen

- Pipelines
- Phasen
- Aktionen
- Pipeline-Ausführungen
- Stufenausführungen

- Aktionsausführungen
- Aktionstypen
- Übergänge
- -Artefakte
- Quell-Revisionen
- Auslöser
- Variablen

Pipelines

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

Phasen

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

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 kann eine Bereitstellungsstufe eine Bereitstellungsaktion enthalten, die Code für einen Datenverarbeitungsservice wie Amazon EC2 oder bereitstelltAWS Lambda.

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

Aktionen können in Reihe 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 belegte Stufe zu betreten, wird auf eine eingehende Ausführung verwiesen. Eine eingehende Ausführung kann weiterhin fehlschlagen, ersetzt oder manuell gestoppt werden. Weitere Informationen zur Funktionsweise eingehender Ausführungen finden Sie unter Funktionsweise eingehender Ausführungen.

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

Weitere Informationen finden Sie unter PipelineExecution.

Angehaltene Ausführungen

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

Es gibt zwei Möglichkeiten, eine Pipeline-Ausführung anzuhalten:

- Anhalten und warten
- Anhalten und beenden

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

Fehlgeschlagene Ausführungen

Wenn eine Ausführung fehlschlägt, wird sie angehalten und durchläuft die Pipeline nicht vollständig. Ihr Status ist `FAILED` und die Phase wird freigeschaltet. Eine neuere Ausführung kann in die

entsperrte Phase eintreten und diese sperren. Sie können eine fehlgeschlagene Ausführung wiederholen, wenn diese nicht ersetzt wurde oder nicht wiederholbar ist.

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 Standardausführungsmodus.

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

Weitere Informationen zu Ausführungsmodi und gesperrten Stufen finden Sie unter So werden Ausführungen im SUPERSEDED-Modus verarbeitet.

Stufenausführungen

Eine Stufenausführung ist der Prozess, bei dem alle Aktionen innerhalb einer Stufe abgeschlossen werden. Informationen zur Funktionsweise einer Stufenausführung und Informationen zu gesperrten Stufen finden Sie unter So werden Ausführungen im SUPERSEDED-Modus verarbeitet.

Gültige Status für Stufen sind `InProgress`, `Stopping`, `StoppedSucceeded`, und `Failed`. Sie können eine fehlgeschlagene Phase wiederholen, es sei denn, die fehlgeschlagene Phase ist nicht wiederholbar. Weitere Informationen finden Sie unter StageExecution.

Aktionsausführungen

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

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

Aktionstypen

Aktionstypen sind vorkonfigurierte Aktionen, die in zur Auswahl verfügbar sind CodePipeline. Der Aktionstyp wird von Eigentümer, Anbieter, Version und Kategorie definiert. Der Aktionstyp stellt benutzerdefinierte Parameter bereit, mit denen die Aktionsaufgaben in einer Pipeline abgeschlossen werden.

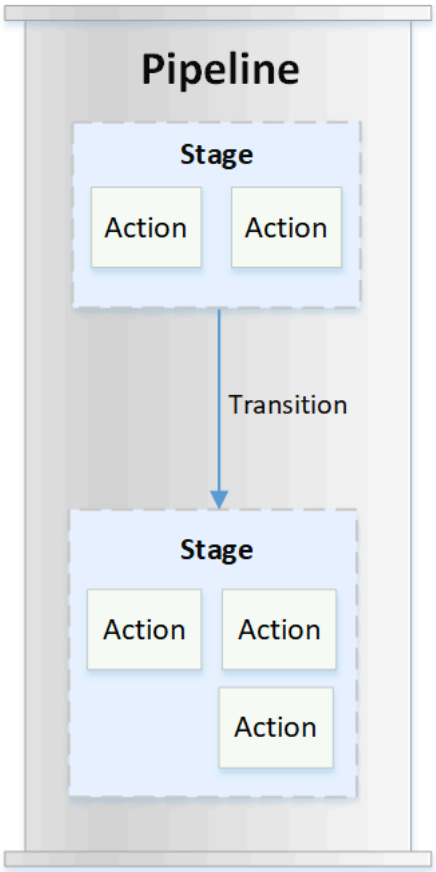
Informationen zu den Produkten AWS-Services und Services von und Drittanbietern, die Sie je nach Aktionstyp in Ihre Pipeline integrieren können, finden Sie unter Integrationen mit CodePipeline Aktionstypen.

Informationen zu den Integrationsmodellen, die für Aktionstypen in unterstützt werden CodePipeline, finden Sie unter Integration-Subtyreferenz.

Informationen darüber, wie Drittanbieter Aktionstypen in einrichten und verwalten können CodePipeline, finden Sie unter Arbeiten mit Aktionstypen.

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.



-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 übergeben die Ausgabe an eine andere Aktion zur weiteren Verarbeitung mithilfe des Pipeline-Artefakts bucket. CodePipeline copies Artefakte in den Artefaktspeicher, wo die Aktion sie aufnimmt. 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. Eine Ausführung verarbeitet

Quellrevisionen. Für CodeCommit - GitHub und -Repositorys ist dies der Commit. Bei S3-Buckets oder -Aktionen ist dies die Objektversion.

Sie können eine Pipeline-Ausführung mit einer Quellrevision starten, z. B. einem Commit, die Sie angeben. Die Ausführung verarbeitet die angegebene Revision und überschreibt die für die Ausführung verwendete Revision. Weitere Informationen finden Sie unter Starten einer Pipeline mit einer Quellrevisionsüberschreibung.

Auslöser

Auslöser sind Ereignisse, die Ihre Pipeline starten. Einige Auslöser, z. B. das manuelle Starten einer Pipeline, sind für alle Quellaktionsanbieter in einer Pipeline verfügbar. Bestimmte Auslöser hängen vom Quellanbieter für eine Pipeline ab. Beispielsweise müssen CloudWatch Ereignisse mit Ereignisressourcen von Amazon konfiguriert werden, CloudWatch für die der Pipeline-ARN als Ziel in der Ereignisregel hinzugefügt wurde. Amazon CloudWatch Events ist die empfohlenen Auslöser für die automatische Änderungserkennung für Pipelines mit einer - CodeCommit oder S3-Quellaktion. Webhooks sind eine Art von Auslöser, der für Repository-Ereignisse von Drittanbietern konfiguriert ist. WebhookV2 ist beispielsweise ein Auslösertyp, mit dem Git-Tags verwendet werden können, um Pipelines mit externen Quellanbietern wie GitHub.com, GitHub Enterprise Server, GitLab.com, GitLab selbstverwaltet oder Bitbucket Cloud zu starten. In der Pipeline-Konfiguration können Sie einen Filter für Auslöser wie Push- oder Pull-Anfrage angeben. Sie können Code-Push-Ereignisse nach Git-Tags, Verzweigungen oder Dateipfaden filtern. Sie können Pull-Anforderungsereignisse nach Ereignissen (geöffnet, aktualisiert, geschlossen), Verzweigungen oder Dateipfaden filtern.

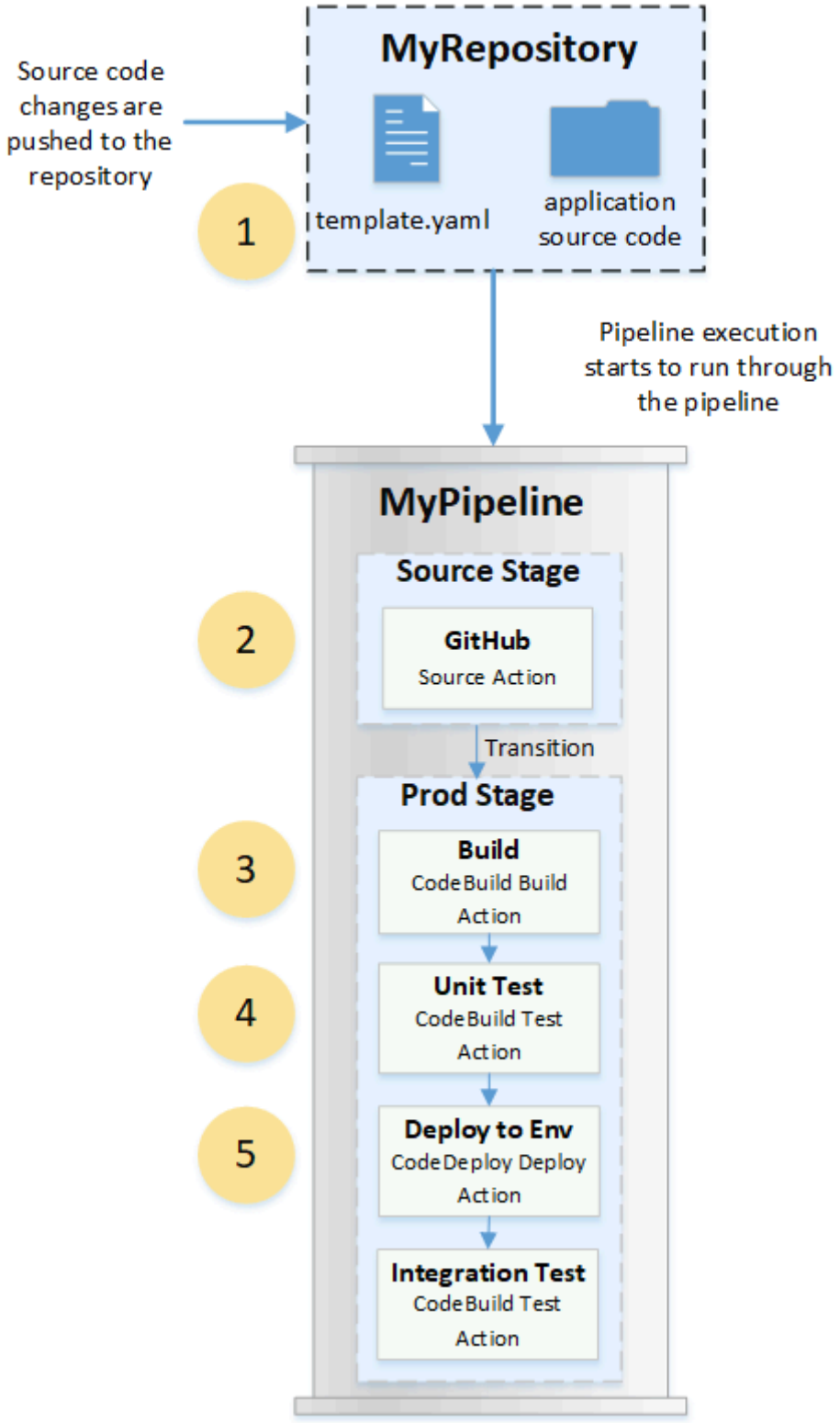
Weitere Informationen zu Auslösern finden Sie unter Starten einer Pipeline in CodePipeline. Ein Tutorial, das Sie durch die Verwendung von Git-Tags als Auslöser für Ihre Pipeline führt, finden Sie unter Tutorial: Verwende Git-Tags, um deine Pipeline zu starten.

Variablen

Eine Variable ist ein Wert, der verwendet werden kann, um Aktionen in Ihrer Pipeline dynamisch zu konfigurieren. Variablen können entweder auf Pipeline-Ebene deklariert oder durch Aktionen in der Pipeline ausgegeben werden. Variable Werte werden zum Zeitpunkt der Pipeline-Ausführung aufgelöst und können im Ausführungsverlauf angezeigt werden. Für Variablen, die auf Pipeline-Ebene deklariert sind, 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 Variablen.

DevOps Pipeline-Beispiel

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



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

1. Der Anwendungsquellcode wird in einem Repository verwaltet, das als GitHub Quellaktion in der Pipeline konfiguriert ist. Wenn Entwickler per Push in das Repository übertragen werden, CodePipeline erkennt die übertragene Änderung und eine Pipeline-Ausführung beginnt in der Quellphase .
2. Die GitHub Quellaktion wird erfolgreich abgeschlossen (d. h. die neuesten Änderungen wurden heruntergeladen und im Artefakt-Bucket gespeichert, der für diese Ausführung einzigartig ist). Die von der GitHub Quellaktion erzeugten Ausgabeartefakte, bei denen es sich um die Anwendungsdateien aus dem Repository handelt, werden dann als Eingabeartefakte verwendet, die von den Aktionen in der nächsten Phase verarbeitet werden sollen.
3. Die Pipeline-Ausführung wechselt von der Quellphase zur Produktionsphase. Die erste Aktion in der Prod-Phase führt ein Build-Projekt aus, das in erstellt CodeBuild und als Build-Aktion in der Pipeline 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 Testprojekt für Einheiten, das in erstellt CodeBuild und als Testaktion in der Pipeline konfiguriert wurde.
5. Der komponentengetestete Code wird als Nächstes von einer Bereitstellungsaktion in der Produktionsphase bearbeitet, die die Anwendung zu einer Produktionsumgebung bereitstellt. Nachdem die Bereitstellungsaktion erfolgreich abgeschlossen wurde, ist die letzte Aktion in der -Phase ein Integrationstestprojekt, das in erstellt CodeBuild und als Testaktion in der Pipeline 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. CodePipeline verfolgt jede Pipeline-Ausführung, die beginnt, wenn eine Pipeline manuell gestartet wird oder eine Änderung am Quellcode vorgenommen wird. CodePipeline verwendet die folgenden Ausführungsmodi, um die Art und Weise zu behandeln, wie jede Ausführung die Pipeline durchläuft.

- SUPERSEDED-Modus: Eine neuere Ausführung kann eine ältere überfordern. Dies ist die Standardeinstellung.
- QUEUED-Modus: Ausführungen werden nacheinander in der Reihenfolge verarbeitet, in der sie in die Warteschlange gestellt werden. Dazu ist der Pipeline-Typ V2 erforderlich.
- PARALLEL-Modus: Im PARALLEL-Modus werden Ausführungen gleichzeitig und unabhängig voneinander ausgeführt. Ausführungen warten nicht, bis andere Ausführungen abgeschlossen sind, bevor sie gestartet oder abgeschlossen werden. Dazu ist der Pipeline-Typ V2 erforderlich.

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 von Ihnen geplante Amazon CloudWatch Events-Regel auslösen. Wenn beispielsweise eine Quellcodeänderung per Push zu einem Repository übertragen wird, das als Quellaktion der Pipeline konfiguriert ist, erkennt die Pipeline die Änderung und startet eine Ausführung.

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

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:

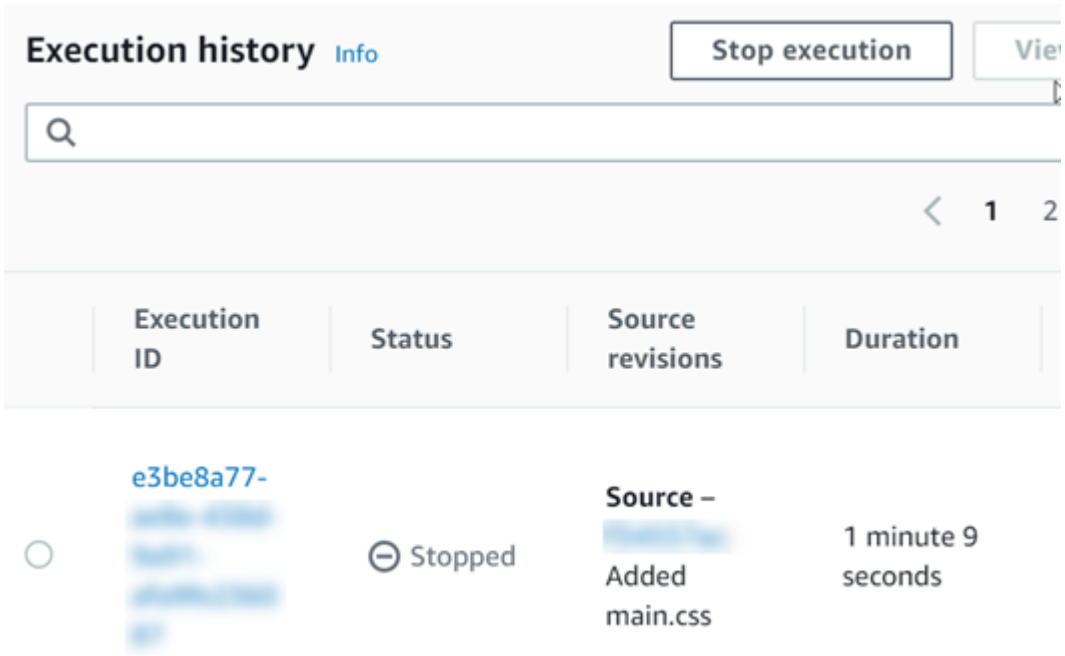
Option 1: Anhalten und warten

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.

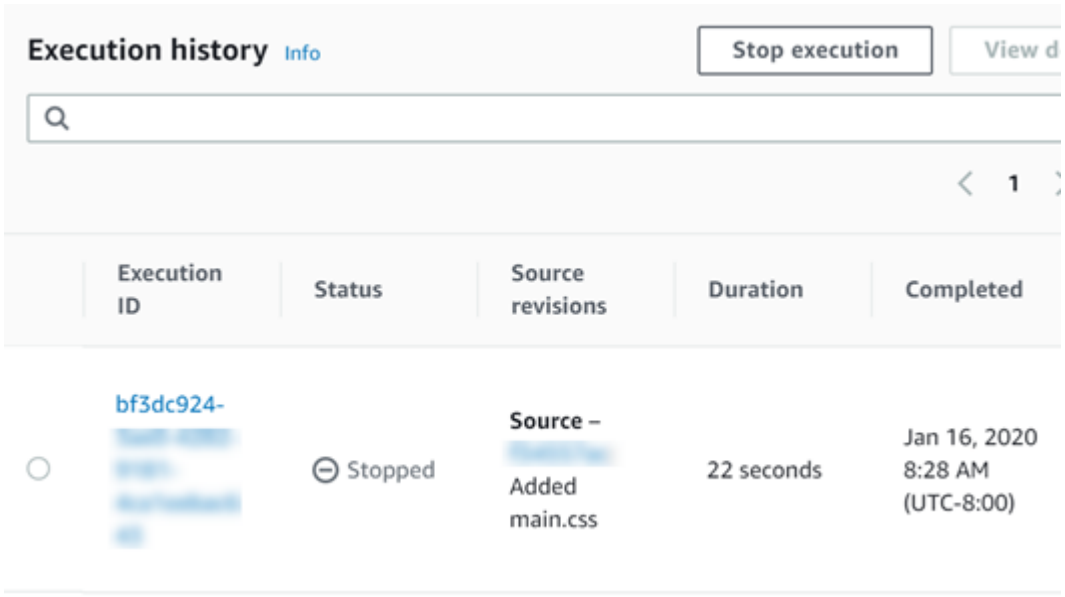
   In der Verlaufsansicht wird der Ausführungsstatus Stopped (Beendet) angezeigt, nachdem die laufende Aktion abgeschlossen ist.



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 out-of-sequence Aufgaben in Ihrer Pipeline vermeidet. Wenn eine Aktion in abgebrochen wird CodePipeline, setzt der Aktionsanbieter alle Aufgaben im Zusammenhang mit der Aktion fort. Im Fall einer AWS CloudFormation-Aktion wird die Bereitstellungsaktion in der Pipeline beendet, aber die Stapelaktualisierung wird möglicherweise fortgesetzt und führt zu einer fehlgeschlagenen Aktualisierung.

Als Beispiel für abgebrochene Aktionen, die zu out-of-sequence Aufgaben führen können, wird die Aktion in abgebrochen, aber in Amazon S3 fortgesetzt, wenn Sie eine große Datei (1 GB) über eine S3-Bereitstellungsaktion bereitstellen und die Aktion beenden und abbrechen CodePipeline. Amazon S3 erhält keine Anweisungen 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.

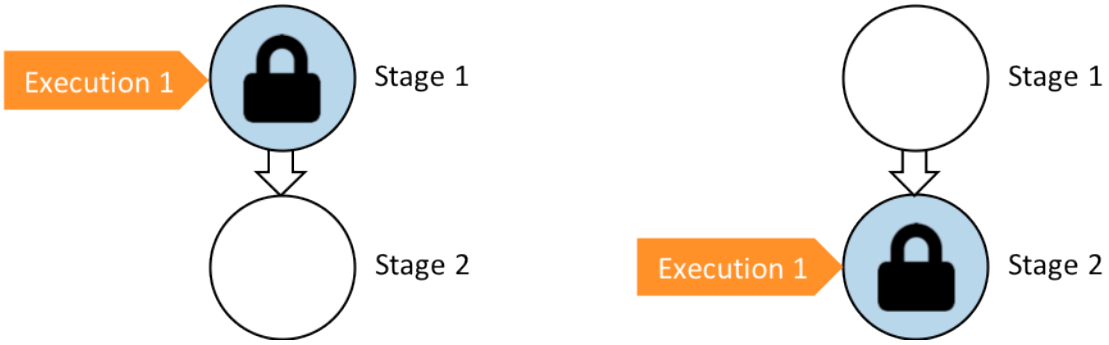
Sie können die Option zum Anhalten und Beenden verwenden, wenn Sie eine benutzerdefinierte Aktion haben. Sie können beispielsweise eine benutzerdefinierte Aktion mit Arbeit abbrechen, die nicht abgeschlossen werden muss, bevor Sie eine neue Ausführung für einen Bugfix starten.

So 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.

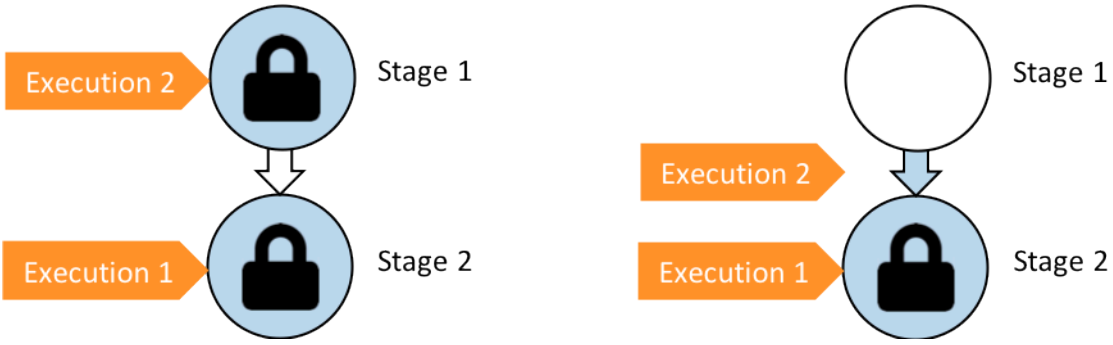


Vor: Stage 1 is locked as Execution 1 enters. Nach: Stage 2 is locked as Execution 1 enters.

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.

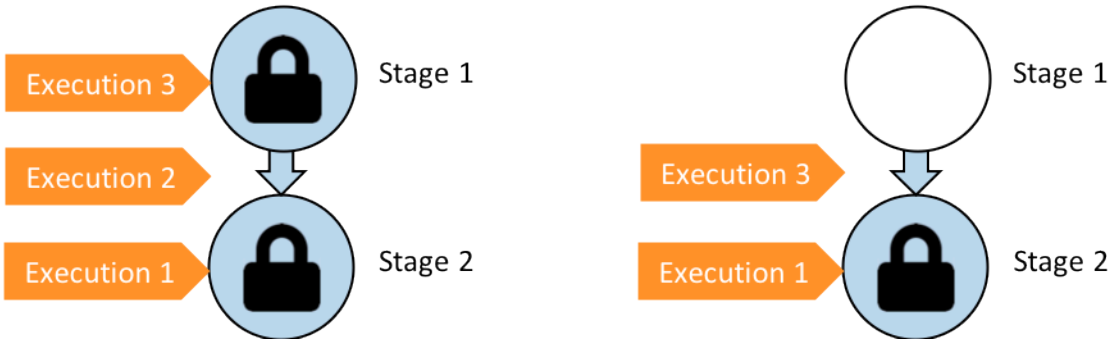
> **Note**
>
> Bevor Sie eine Ausführung beenden, empfehlen wir Ihnen, den Übergang vor der Phase zu deaktivieren. Auf diese Weise wird die Phase aufgrund der angehaltenen Ausführung entsperrt und akzeptiert keine nachfolgende Pipeline-Ausführung.



Vor: Stage 2 is locked as Execution 1 enters. Nach: 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.

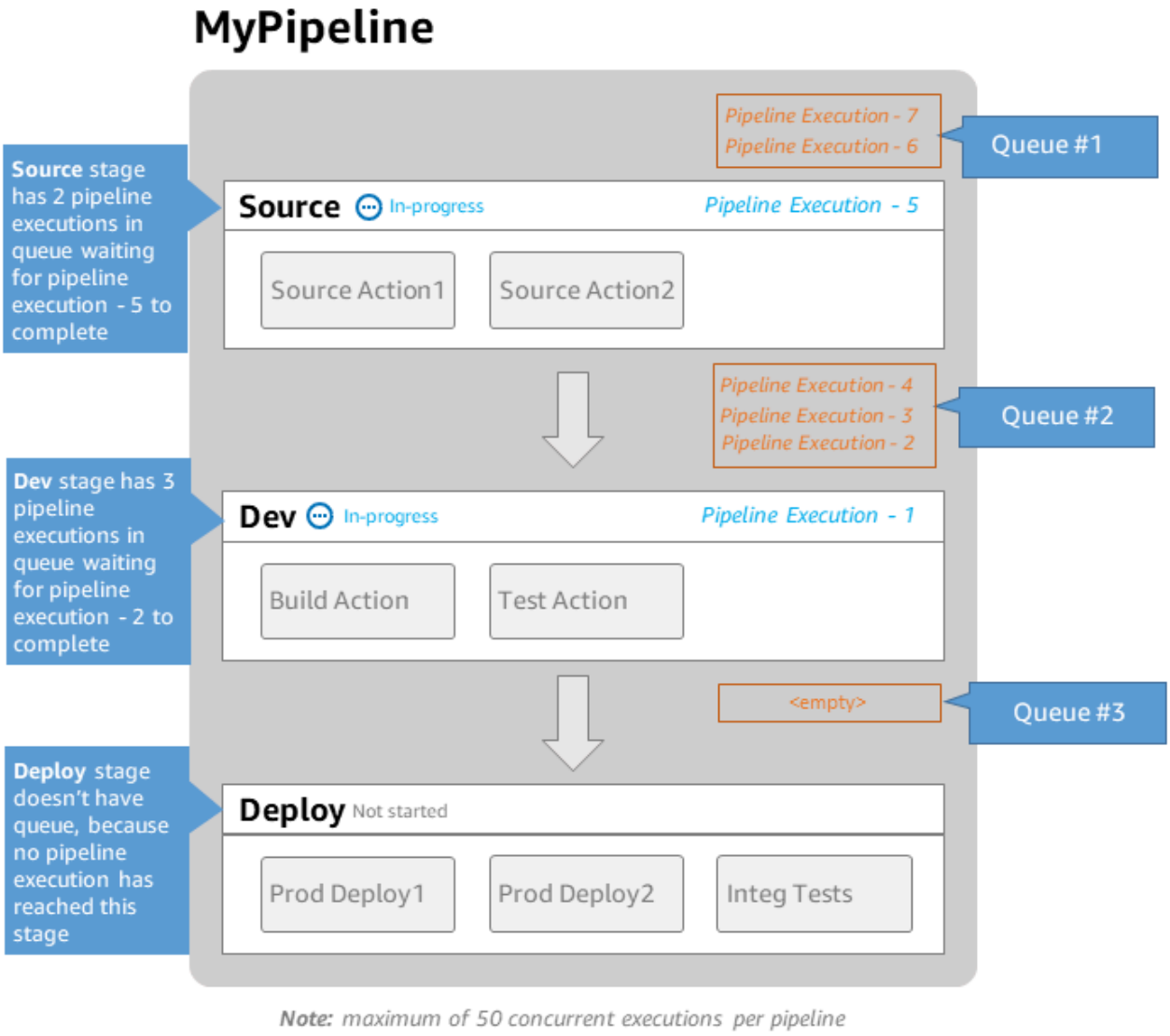
So werden Ausführungen im QUEUED-Modus verarbeitet

Bei Pipelines im QUEUED-Modus werden Phasen gesperrt, wenn eine Ausführung verarbeitet wird. Warteausführungen überschreiben jedoch keine bereits gestarteten Ausführungen.

Warteausführungen sammeln sich an den Eintrittspunkten zu gesperrten Phasen in der Reihenfolge an, in der sie die Stufe erreichen, und bilden eine Warteschlange mit wartenden Ausführungen. Im QUEUED-Modus können Sie mehrere Warteschlangen in derselben Pipeline haben. Wenn

eine Ausführung in der Warteschlange in eine Stufe eintritt, wird die Stufe gesperrt und es können keine anderen Ausführungen eingehen. Dieses Verhalten bleibt dasselbe wie im SUPERSEDED-Modus. Wenn die Ausführung die Stufe beendet hat, wird die Stufe entsperrt und ist für die nächste Ausführung bereit.

Das folgende Diagramm zeigt, wie Phasen in einer Pipeline-Prozessausführung im QUEUED-Modus ausgeführt werden. Während die Quellphase beispielsweise Ausführung 5 verarbeitet, bilden die Ausführungen für 6 und 7 die Warteschlange 1 und warten am Stufeneinstiegspunkt. Die nächste Ausführung in der Warteschlange wird nach dem Entsperren der Stufe verarbeitet.



Weitere Informationen zu Kontingenten mit Ausführungsmodi finden Sie unter Kontingente in AWS CodePipeline.

So werden Ausführungen im PARALLEL-Modus verarbeitet

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

Verwenden Sie den PARALLEL-Modus in Entwicklungsumgebungen, in denen jedes Feature seinen eigenen Feature-Zweig hat und auf Zielen bereitgestellt wird, die nicht von anderen Benutzern gemeinsam genutzt werden.

Weitere Informationen zu Kontingenten mit Ausführungsmodi finden Sie unter Kontingente in AWS CodePipeline.

Verwalten des Pipeline-Flows

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 gelangen. Die Pipeline-Ausführung, die darauf wartet, in eine Phase überzugehen, in der der Übergang deaktiviert ist, wird als eingehende Ausführung bezeichnet. Nachdem Sie den Übergang aktiviert haben, wechselt eine eingehende Ausführung in die Stufe 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 Berechtigung 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

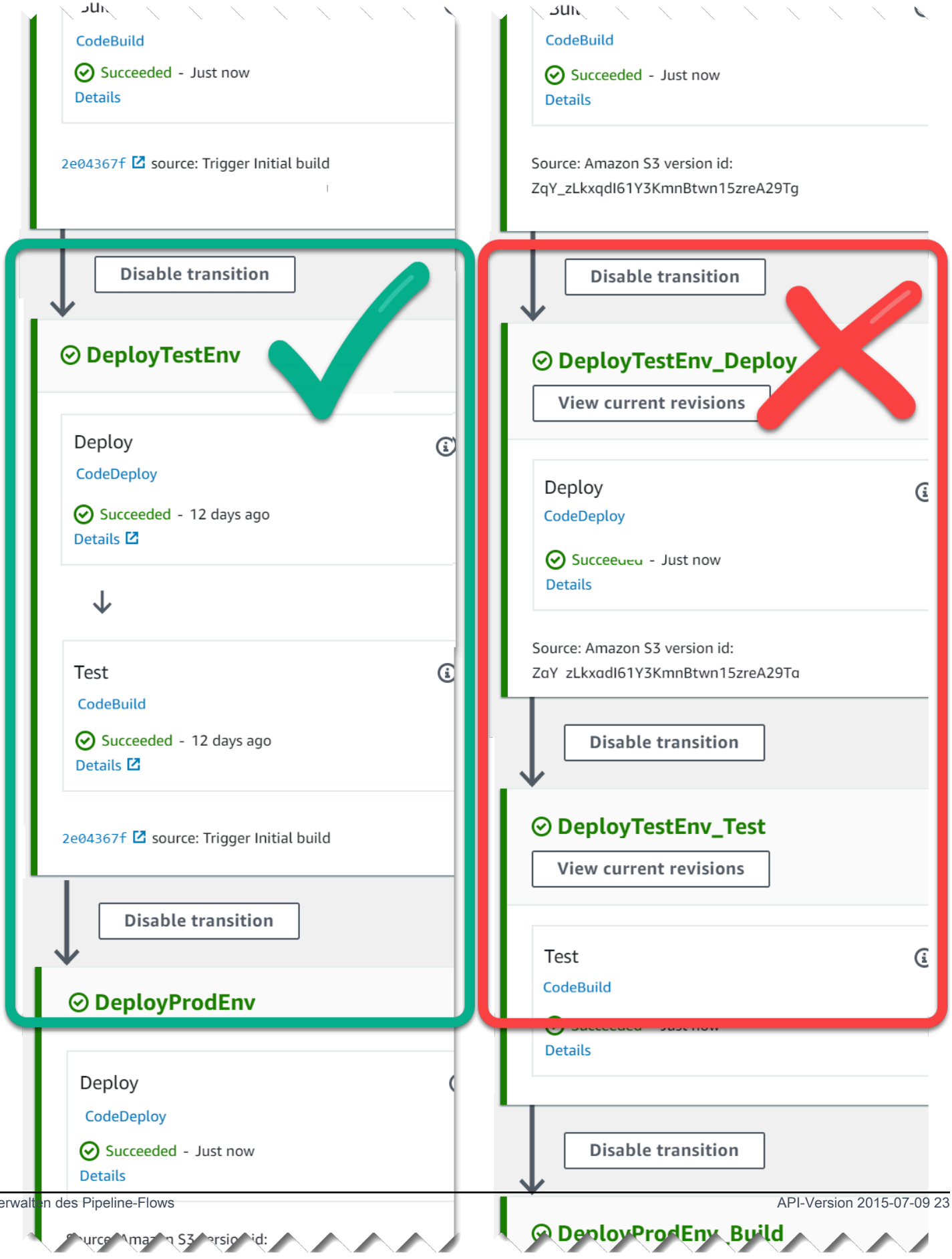
Genehmigungsaktion wird auf die gleiche Weise wie eine fehlgeschlagene Aktion verarbeitet.

- 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. Hierdurch wird die Ausführung fortgesetzt (fehlgeschlagene Aktionen werden wiederholt und, wenn erfolgreich, in der Phase/Pipeline fortgesetzt).
  - Eine andere Ausführung tritt in die fehlgeschlagene Phase ein und ersetzt die fehlgeschlagene Ausführung. In diesem Fall kann die fehlgeschlagene Ausführung nicht wiederholt werden.

Empfohlene Pipeline-Struktur

Wenn Sie den Fluss einer Codeänderung durch die Pipeline festlegen, sollten Sie verwandte Aktionen innerhalb einer Phase gruppieren, sodass die Aktionen dieselbe Ausführung verarbeiten, wenn die Phase gesperrt wird. Sie können eine Stufe für jede Anwendungsumgebung, AWS-Regionoder Availability Zone usw. 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).

Funktionsweise eingehender Ausführungen

Eine eingehende Ausführung ist eine Ausführung, die darauf wartet, dass eine nicht verfügbare Phase, ein Übergang oder eine Aktion verfügbar wird, bevor sie fortfährt. Die nächste Phase, der nächste Übergang oder die nächste Aktion sind möglicherweise nicht verfügbar, da:

- Eine andere Ausführung hat bereits in die nächste Phase aufgenommen und sie gesperrt.
- Der Übergang in die nächste Phase ist deaktiviert.

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

Bei eingehenden Ausführungen sind die folgenden Überlegungen zu berücksichtigen:

- Sobald die Aktion, der Übergang oder die gesperrte Stufe verfügbar ist, geht die laufende eingehende Ausführung in die Stufe über und fährt mit der Pipeline fort.
- Während die eingehende Ausführung wartet, kann sie manuell gestoppt werden. Eine eingehende Ausführung kann den `Failed` Status `InProgress`, `Stopped`oder haben.
- Wenn eine eingehende Ausführung gestoppt wurde oder fehlgeschlagen ist, kann sie nicht erneut versucht werden, da keine fehlgeschlagenen Aktionen wiederholt werden können. Wenn eine eingehende Ausführung gestoppt wurde und der Übergang aktiviert ist, wird die gestoppte eingehende Ausführung nicht in die -Phase fortgesetzt.

Sie können eine eingehende Ausführung anzeigen oder beenden.

Eingabe- und Ausgabe-Artefakte

CodePipeline lässt sich in Entwicklungstools integrieren, um nach Codeänderungen zu suchen und dann in allen Phasen des kontinuierlichen Bereitstellungsprozesses zu erstellen und bereitzustellen. Artefakte sind die Dateien, die von Aktionen in der Pipeline verarbeitet werden, z. B. Dateien oder Ordner mit Anwendungscode, Indexseitendateien, Skripts usw. Das Amazon S3-Quellaktionsartefakt

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

Aktionen verwenden Eingabe- und Ausgabeartefakte, die in dem Amazon S3-Artefakt-Bucket gespeichert sind, den Sie beim Erstellen der pipeline. CodePipeline zips ausgewählt haben, und überträgt die Dateien für Eingabe- oder Ausgabeartefakte, je nach Aktionstyp in der Stufe.

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

Beispielsweise:

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

Wenn Sie eine Aktion erstellen oder bearbeiten, legen Sie das Eingabe- und Ausgabe-Artefakt oder die Eingabe- und Ausgabe-Artefakte für die Aktion fest. Für eine zweistufige Pipeline mit einer Quelle

und Bereitstellungsstufe 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 erstellt einen Amazon S3-Bucket im selben AWS-Konto und AWS-Region speichert Elemente für alle Pipelines. Jedes Mal, wenn Sie die Konsole verwenden, um eine weitere Pipeline in dieser Region zu erstellen, CodePipeline erstellt einen Ordner für diese Pipeline im Bucket. Dieser Ordner wird zum Speichern von Artefakten für Ihre Pipeline verwendet, während der automatisierte Veröffentlichungsprozess ausgeführt wird. Dieser Bucket hat den Namen codepipeline-*Region*-*12345EXAMPLE*, wobei *Region* die AWS-Region ist, in der Sie die Pipeline erstellt haben, und *12345EXAMPLE* eine zufällige Zahl mit 12 Ziffern ist, die sicherstellt, dass der Bucket-Name eindeutig ist.

  > **Note**
  >
  > Wenn Sie bereits über einen Bucket verfügen, der mit codepipeline-region beginnt – in der Region, in der Sie die Pipeline erstellen – CodePipeline verwendet diesen als Standard-Bucket. Es folgt auch der lexikografischen Reihenfolge. Beispiel: codepipeline-region-abcexample wird vor codepipeline-region-defexample ausgewählt.

  CodePipeline kürzt Artefaktnamen, was dazu führen kann, dass einige Bucket-Namen ähnlich erscheinen. Obwohl der Artefaktname gekürzt zu sein scheint, CodePipeline weist dem Artefakt-Bucket auf eine Weise zu, die nicht von Artefakten mit gekürzten Namen betroffen ist. 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 über einen Artefakt-Bucket in der Pipeline AWS-Konto und verfügen und Sie müssen über einen Artefakt-Bucket pro Region verfügenAWS-Region, 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 im selben AWS-Konto und in derselben AWS-Region wie die Pipeline befindet. Sie können dies tun, wenn Sie Bedenken haben, die für Ihr Konto zulässigen Limits für 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 -AWSAnbieter in einer Region, die sich von Ihrer Pipeline unterscheidet), müssen Sie einen Artefakt-Bucket für jede weitere Region bereitstellen, in der Sie eine Aktion ausführen möchten.

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

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

  Ein Artefakt kann von mehr als einer Aktion bearbeitet werden.

Pipeline-Typen

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

- Pipelines vom Typ V1 haben eine JSON-Struktur, die Standardparameter für Pipeline, Phase und Aktionsebene enthält.
- Pipelines vom Typ V2 haben dieselbe Struktur wie ein V1-Typ sowie zusätzliche Parameter für die Release-Sicherheit und Auslöserkonfiguration.

Weitere Informationen zu Preisen für finden Sie unter CodePipeline-Preise.

Auf der CodePipeline Referenz zur Pipeline-Struktur Seite finden Sie Details zu Parametern in den einzelnen Pipeline-Typen. Informationen darüber, welche Art von Pipeline ausgewählt werden soll, finden Sie unter Welche Art von Pipeline ist für mich die richtige?.

Welche Art von Pipeline ist für mich die richtige?

Der Pipeline-Typ wird durch den Satz von Merkmalen und Features 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.

	V1-Typ	V2-Typ
Merkmale		
Anwendungsfälle	• Standardbereitstellungen	• Bereitstellungen mit Konfiguration durch Übergabe von Variablen auf Pipeline-Ebene zur Laufzeit • Bereitstellungen, bei denen Pipelines so konfiguriert sind, dass sie auf Git-Tags beginnen
Variablen auf Aktionsebene	Unterstützt	Unterstützt
Variablen auf Pipeline-Ebene	Nicht unterstützt	Unterstützt
Überschreibungen von Quellrevisionen	Nicht unterstützt	Unterstützt
Auslöser und Filtern von Git-Tags, Pull-Anforderungen , Verzweigungen oder Dateipfaden	Nicht unterstützt	Unterstützt
Ausführungsmodi PARALLEL und QUEUED	Nicht unterstützt	Unterstützt

Weitere Informationen zu Preisen für finden Sie unter CodePipeline-Preise.

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

- Schritt 1: Erstellen Sie einen AWS-Konto Benutzer mit Administratorrechten
- Schritt 2: Wenden Sie eine verwaltete Richtlinie für den Administratorzugriff auf an CodePipeline
- Schritt 3: Installieren Sie den AWS CLI
- Schritt 4: Öffnen Sie die Konsole für CodePipeline
- Nächste Schritte

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

So melden Sie sich für ein AWS-Konto an

Wenn Sie kein AWS-Konto haben, führen Sie die folgenden Schritte zum Erstellen durch.

Anmeldung für ein AWS-Konto

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

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

   Wenn Sie sich für ein AWS-Konto anmelden, wird ein Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen eine Bestätigungs-E-Mail, sobald die Anmeldung abgeschlossen ist. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Einen Administratorbenutzer erstellen

Wenn Sie sich für AWS-Konto registriert haben, sichern Sie Root-Benutzer des AWS-Kontos, aktivieren Sie AWS IAM Identity Center erstellen Sie einen Administratorbenutzer, damit Sie nicht den Root-Benutzer für alltägliche Aufgaben verwenden.

Schützen Ihres Root-Benutzer des AWS-Kontos

1. Melden Sie sich bei AWS Management Console als Kontobesitzer an, indem Sie Stammbenutzer auswählen und Ihre AWS-Konto-E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

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

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

Erstellen eines Administratorbenutzers

1. IAM Identity Center aktivieren.

   Eine genaue Anleitung finden Sie unter Aktivierung von AWS IAM Identity Center im AWS IAM Identity Center-Benutzerhandbuch.

2. Gewähren Sie im IAM Identity Center einem Administratorbenutzer Administratorzugriff.

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

Als Administratorbenutzer anmelden

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

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

Schritt 2: Wenden Sie eine verwaltete Richtlinie für den Administratorzugriff auf an 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 gewährenAWS-Service.

Note

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

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

- Benutzer und Gruppen in AWS IAM Identity Center:

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

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

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

- IAM-Benutzer:

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

Schritt 3: Installieren Sie den 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.

So installieren und konfigurieren Sie die AWS CLI

1. Laden Sie die AWS CLI auf den lokalen Computer herunter und installieren Sie. Auf diese Weise können Sie über die CodePipeline Befehlszeile mit interagieren. Weitere Informationen finden Sie unter Erste Schritte mit der AWS-Befehlszeilenschnittstelle.

   > **Note**
   >
   > CodePipeline funktioniert nur mit den AWS CLI Versionen 1.7.38 und höher. Führen Sie den Befehl aus, um festzustellenAWS 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 die AWS CLI mit dem Befehl configure 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 verwenden möchten. CodePipeline Wenn Sie den Standard-Regionennamen eingeben sollen, geben Sie den Namen Region an, in der Sie die Pipeline erstellen (z. B. `us-east-2`). Wenn Sie nach dem standardmäßigen Ausgabeformat gefragt werden, geben Sie `json` an. 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 Regionen und Endpunkten CodePipeline, für die verfügbar sind, finden Sie unter AWS CodePipelineEndpunkte und Kontingente.

Schritt 4: Öffnen Sie die Konsole für CodePipeline

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

Nächste Schritte

Sie haben die Voraussetzungen erfüllt. Sie können mit der Verwendung beginnen CodePipeline. Informationen zum Einstieg in die Arbeit mit CodePipeline finden Sie unter CodePipeline -Tutorials.

Produkt- und Serviceintegrationen mit CodePipeline

Standardmäßig AWS CodePipeline ist in eine Reihe von - AWS-Services und -Partnerprodukten und -services integriert. Verwenden Sie die Informationen in den folgenden Abschnitten, um für die Integration in die von Ihnen verwendeten Produkte und Services CodePipeline zu konfigurieren.

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 Aktionstyp organisiert.

Themen

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

Quellaktions-Integrationen

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

Themen

- Amazon-ECR-Quellaktionen
- Amazon S3-Quellaktionen

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

Amazon-ECR-Quellaktionen

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

Weitere Informationen:

- Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter . Amazon ECR
- Erstellen einer Pipeline in CodePipeline
- Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy

Amazon 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-Bucket als Quellaktion für Ihren Code verwendet wird.

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

Weitere Informationen:

- Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter . Quellaktion für Amazon S3
- Schritt 1: Erstellen eines S3-Buckets für Ihre Anwendung
- Erstellen einer Pipeline (CLI)

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

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

Verbindungen (`CodeStarSourceConnection`-Aktionen) werden verwendet, um auf Ihre Bitbucket-Cloud, GitHub, GitHub Enterprise Server, GitLab.com oder GitLab Ihr selbstverwaltetes Repository zuzugreifen.

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

Bitbucket-Cloud	Sie können so konfigurieren CodePipeline , dass ein Bitbucket-Cloud-Repository als Quelle für Ihren Code verwendet wird. Sie müssen zuvor ein Bitbucket-Konto und mindestens ein Bitbucket-Cloud-Repository erstellt haben. Sie können eine Quellaktion für Ihr Bitbucket-Cloud-Repository hinzufügen, indem Sie entweder eine Pipeline erstellen oder eine vorhandene bearbeiten. **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 Sie eine Verbindung erstellen, installieren Sie die AWS CodeStar App mit Ihrem Code-Repository eines Drittanbieters und verknüpfen sie dann mit Ihrer Verbindung. Verwenden Sie für Bitbucket Cloud die Bitbucket-Option in der Konsole oder die `-CodestarSourceConnection` Aktion in der CLI. Siehe Bitbucket-Cloud-Verbindungen. Sie können die Option Vollständiger Klon für diese Aktion verwenden, um auf die Git-Metadaten des Repositorys zu verweisen, sodass Downstream-Aktionen Git-Befehle direkt ausführen können. Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden. Weitere Informationen: • Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen. • Ein Tutorial „Erste Schritte“, das eine Pipeline mit einer Bitbucket-Cloud-Quelle erstellt, finden Sie 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 vorhandene bearbeiten. Sie können Ressourcen, die als Verbindungen bezeichnet werden, einrichten, damit Ihre Pipelines auf Code-Repositorys von Drittanbietern zugreifen können. Wenn Sie eine Verbindung erstellen, installieren Sie die AWS CodeStar App mit Ihrem Code-Repository eines Drittanbieters und verknüpfen sie dann mit Ihrer Verbindung.

	Verwenden Sie die GitHub (Version 2)-Anbieteroption in der Konsole oder die `-CodestarSourceConnection` Aktion in der CLI. Siehe GitHub Verbindungen. Sie können die Option Vollständiger Klon für diese Aktion verwenden, um auf die Git-Metadaten des Repositorys zu verweisen, sodass Downstream-Aktionen Git-Befehle direkt ausführen können. Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden. Weitere Informationen: • Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter . CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen • Ein Tutorial, das Ihnen zeigt, wie Sie eine Verbindung zu einem GitHub Repository herstellen und die Option Vollständiger Klon verwenden, finden Sie unter Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden. • Die aktuelle GitHub Aktion ist die Version 2-Quellaktion für GitHub. Die Version 1 GitHub action wird mit OAuth-Token-Authentifizierung verwaltet . Obwohl wir nicht empfehlen, die Aktion GitHub Version 1 zu verwenden , funktionieren vorhandene Pipelines mit der Aktion GitHub Version 1 weiterhin ohne Auswirkungen. Sie können jetzt eine CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen Quellaktion in Ihrer Pipeline verwenden, die Ihre GitHub Quellaktion mit GitHub Apps verwaltet. Wenn Sie über eine Pipeline verfügen, die die Version 1 GitHub action verwendet , lesen Sie die Schritte zum Aktualisieren, um eine Version 2 GitHub action in zu verwendenAktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2.
GitHub Enterprise Server	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 vorhandene 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 erstellen, installieren Sie die AWS CodeStar App mit Ihrem Code-Repository eines Drittanbieters und verknüpfen sie dann mit Ihrer Verbindung.

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

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

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

Weitere Informationen:

- Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter . CodeStarSourceConn ection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen
- Ein Tutorial, das Ihnen zeigt, wie Sie eine Verbindung zu einem GitHub Repository herstellen und die Option Vollständiger Klon 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 vorhandene bearbeiten. Verwenden Sie die GitLab Anbieteroption in der Konsole oder die `CodestarSourceConnection` Aktion mit dem `GitLab` Anbieter in der CLI. Siehe GitLab.com-Verbindungen. Weitere Informationen: • Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter . CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen
GitLab Selbstverwaltet	Sie können so konfigurieren CodePipeline , dass eine GitLab selbstverwaltete Installation als Quelle für Ihren Code verwendet wird. Sie müssen zuvor ein GitLab Konto und ein Abonnement für die selbstverwaltete GitLab (Enterprise Edition oder Community Edition) erstellt haben. Sie können eine Quellaktion für Ihr GitLab selbstverwaltetes Repository hinzufügen, indem Sie entweder eine Pipeline erstellen oder eine vorhandene bearbeiten. Sie können Ressourcen, die als Verbindungen bezeichnet werden, einrichten, damit Ihre Pipelines auf Code-Repositorys von Drittanbietern zugreifen können. Wenn Sie eine Verbindung erstellen, installieren Sie die AWS CodeStar App mit Ihrem Code-Repository eines Drittanbieters und verknüpfen sie dann mit Ihrer Verbindung. Verwenden Sie die Option GitLab Selbstverwalteter Anbieter in der -Konsole oder die -`CodestarSourceConnection` Aktion in der CLI. Siehe Verbindungen für GitLab selbstverwaltete. Sie können die Option Vollständiger Klon für diese Aktion verwenden, um auf die Git-Metadaten des Repositorys zu verweisen, sodass Downstream-

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

Weitere Informationen:

- Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter . CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen
- Die Schritte zum Erstellen einer Verbindung mit diesem Anbietertyp finden Sie unter Verbindungen für GitLab selbstverwaltete.

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 eine Verzweigung 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 vorhandene bearbeiten.

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

Weitere Informationen:

- Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter CodeCommit.
- Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)
- CodePipeline verwendet Amazon CloudWatch Events, um Änderungen an CodeCommit Repositorys zu erkennen, die als Quelle für eine Pipeline verwendet werden. Für jede Quellaktion gibt es eine entsprechende Regel. Diese Ereignisregel startet Ihre Pipeline, wenn eine Änderung im Repository auftritt. Siehe Allgemeine Integrationen mit CodePipeline.

GitHub (Version 1) Quellaktionen

Die Aktion der GitHub Version 1 wird mit OAuth Apps verwaltet. In verfügbaren Regionen können Sie auch eine CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen Quellaktion in Ihrer Pipeline verwenden, die Ihre GitHub Quellaktion mit GitHub Apps verwaltet. Wenn Sie über eine Pipeline verfügen, die die Aktion der GitHub Version 1 verwendet, lesen Sie die Schritte zum Aktualisieren, um eine Aktion der GitHub Version 2 in zu verwendenAktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2.

> **Note**
>
> Obwohl wir nicht empfehlen, die Aktion der GitHub Version 1 zu verwenden, funktionieren vorhandene Pipelines mit der Aktion der GitHub Version 1 weiterhin ohne Auswirkungen.

Weitere Informationen:

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

Erstellen von Aktionsintegrationen

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

Themen

- CodeBuild -Build-Aktionen
- CloudBees Erstellen von Aktionen
- Jenkins-Build-Aktionen
- TeamCity Erstellen von Aktionen

CodeBuild -Build-Aktionen

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

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

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

Weitere Informationen:

- Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter AWS CodeBuild.
- Was ist CodeBuild?
- CodeBuild – Vollständig verwalteter Build-Service

CloudBees Erstellen von Aktionen

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

Weitere Informationen:

- re:INVENT 2017: Cloud First mit AWS

Jenkins-Build-Aktionen

Sie können so konfigurieren CodePipeline , dass Jenkins CI verwendet wird, 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 auf Jenkins wird pro Projekt konfiguriert. Sie müssen das CodePipeline Plugin für Jenkins auf jeder Jenkins-Instance installieren, die Sie mit verwenden möchten CodePipeline. Sie müssen auch den CodePipeline Zugriff auf das Jenkins-Projekt konfigurieren. Schützen Sie Ihr Jenkins-Projekt, indem Sie es so konfigurieren, dass es ausschließlich HTTPS/SSL-Verbindungen akzeptiert. Wenn Ihr Jenkins-Projekt auf einer Amazon EC2-Instance installiert ist, sollten Sie Ihre AWS Anmeldeinformationen angeben, indem Sie 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 Erstellen von Aktionen

Sie können so konfigurieren CodePipeline , dass es verwendet, TeamCity 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 organisiert und können Ihnen bei der Konfiguration von für CodePipeline die Integration mit den folgenden Testaktionsanbietern helfen.

Themen

- CodeBuild Testaktionen
- AWS Device Farm Testaktionen
- Testaktionen für Ghost Inspector
- Testaktionen für Micro Focus StormRunner Load

CodeBuild Testaktionen

CodeBuild ist ein vollständig verwalteter Build-Service in der Cloud. CodeBuild kompiliert Ihren Quellcode, führt Einheitentests durch und erzeugt Artefakte, die bereitgestellt werden können.

Sie können CodeBuild einer Pipeline als Testaktion hinzufügen. Weitere Informationen finden Sie in der CodePipeline-Referenz für Aktionskonfigurationen für AWS CodeBuild.

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

Weitere Informationen:

- Informationen zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter AWS CodeBuild.
- Was ist CodeBuild?

AWS Device Farm Testaktionen

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 , dass es verwendet, AWS Device Farm um Ihren Code in einer oder mehreren Aktionen in einer Pipeline zu 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-Protokolle, pixel-to-pixel Screenshots und Leistungsdaten enthält, wird aktualisiert, sobald die Tests abgeschlossen sind. AWS Device Farm unterstützt das Testen nativer und hybrider Android-, iOS- und Fire OS-Apps, einschließlich derjenigen, die mit PhoneGap, microSD, 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 zum Anzeigen von Konfigurationsparametern und einem JSON/YAML-Beispielausschnitt finden Sie unter AWS Device Farm.

- Was ist AWS Device Farm?
- Verwenden von AWS Device Farm in einer CodePipeline Testphase

Testaktionen für Ghost Inspector

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

Testaktionen für Micro Focus StormRunner Load

Sie können so konfigurieren CodePipeline , dass Micro Focus StormRunner Load in einer oder mehreren Aktionen in einer Pipeline verwendet wird.

Weitere Informationen:

- Micro-Focus- StormRunner Load-Dokumentation für die Integration in CodePipeline

Bereitstellungsaktions-Integrationen

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

Themen

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

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

Amazon S3-Bereitstellungsaktionen

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

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

Weitere Informationen:

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

AWS AppConfig Aktionen bereitstellen

AWS AppConfig ist eine Funktion von AWS Systems Manager zum Erstellen, Verwalten und schnellen Bereitstellen von Anwendungskonfigurationen. Sie können AppConfig mit Anwendungen verwenden, die auf EC2-Instances AWS Lambda, Containern, mobilen Anwendungen oder IoT-Geräten gehostet werden.

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für AWSAppConfig
- Tutorial: Erstellen einer Pipeline mit AWS AppConfig als Bereitstellungsanbieter

AWS CloudFormation Aktionen bereitstellen

AWS CloudFormation bietet Entwicklern und Systemadministratoren eine einfache Möglichkeit, eine Sammlung verwandter AWS Ressourcen zu erstellen und zu verwalten, indem Vorlagen verwendet

werden, um diese Ressourcen 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 zum Ausführen Ihrer Anwendung erforderlich sind.

Das AWS Serverless Application Model (AWS SAM) erweitert , AWS CloudFormation um eine vereinfachte Möglichkeit zum Definieren und Bereitstellen von Serverless-Anwendungen zu bieten. AWS SAM unterstützt Amazon API Gateway-APIs , AWS Lambda-Funktionen und Amazon DynamoDB-Tabellen. Sie können CodePipeline mit AWS CloudFormation und dem AWS SAM verwenden, um Ihre Serverless-Anwendungen kontinuierlich bereitzustellen.

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

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für AWS CloudFormation
- Kontinuierliche Bereitstellung mit CodePipeline – Erfahren Sie, wie Sie verwenden, CodePipeline um einen fortlaufenden Bereitstellungs-Workflow für zu erstellen AWS CloudFormation.
- Automatisierung der Bereitstellung von Lambda-basierten Anwendungen – Erfahren Sie, wie Sie das AWS Serverless Application Model und verwenden, AWS CloudFormation um einen kontinuierlichen Bereitstellungs-Workflow für Ihre Lambda-basierte Anwendung zu erstellen.

AWS CloudFormation StackSets Aktionen bereitstellen

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

> **Note**
>
> Die `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und sind in den Regionen Asien-Pazifik (Hongkong), Europa (Zürich), Europa (Mailand), Afrika (Kapstadt)

und Naher Osten (Bahrain) nicht verfügbar. Informationen zum Verweisen auf andere verfügbare Aktionen finden Sie unter Produkt- und Serviceintegrationen mit CodePipeline.

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

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

- CloudFormationStackSet
- CloudFormationStackInstances

Weitere Informationen:

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

Amazon-ECS-Bereitstellungsaktionen

Amazon ECS ist ein hoch skalierbarer, hochleistungsfähiger 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 Codeänderung in Ihrem Quellcodeverwaltungs-Repository veranlasst Ihre Pipeline, ein neues Docker-Image zu erstellen, es in Ihre Container-Registry zu verschieben und dann das aktualisierte Image in Amazon ECS bereitzustellen. Sie können auch die ECS-Anbieteraktion (Blau/Grün) in verwenden CodePipeline , um Datenverkehr mit an Amazon ECS weiterzuleiten und bereitzustellen CodeDeploy.

Weitere Informationen:

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

Elastic Beanstalk-Bereitstellungsaktionen

Elastic Beanstalk ist ein Service für die Bereitstellung und Skalierung von Webanwendungen und -services, die mit Java, .NET, PHP, Node.js, Python, Ruby, Go und Docker auf vertrauten Servern wie Apache, Nginx, Passenger und IIS entwickelt wurden. Sie können CodePipeline für die Verwendung von Elastic Beanstalk zur Bereitstellung Ihres Codes konfigurieren. Sie können die Elastic Beanstalk-Anwendung und -Umgebung erstellen, um sie in einer Bereitstellungsaktion in einer Phase zu verwenden, 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. Informationen zum Verweisen auf andere verfügbare Aktionen finden Sie unter Produkt- und Serviceintegrationen mit CodePipeline.

Weitere Informationen:

- Erste Schritte mit Elastic Beanstalk
- Erstellen einer Pipeline in CodePipeline

AWS OpsWorks Aktionen bereitstellen

AWS OpsWorks ist ein Konfigurationsverwaltungsservice, mit dem Sie Anwendungen aller Formen und Größen mit Chef konfigurieren und betreiben können. Mit können AWS OpsWorks Stacks Sie die Architektur der Anwendung und die Spezifikation jeder Komponente einschließlich Paketinstallation, Softwarekonfiguration und Ressourcen wie Speicher definieren. Sie können so konfigurieren CodePipeline , dass es verwendet, AWS OpsWorks Stacks um Ihren Code in Verbindung mit benutzerdefinierten Chef-Cookbooks und Anwendungen in bereitzustellen AWS OpsWorks.

- Benutzerdefinierte Chef-Cookbooks – AWS OpsWorks verwendet Chef-Cookbooks, um Aufgaben wie das Installieren und Konfigurieren von Paketen und das Bereitstellen von Anwendungen zu erledigen.
- Anwendungen – Eine AWS OpsWorks Anwendung besteht aus Code, den Sie auf einem Anwendungsserver ausführen möchten. Der Anwendungscode wird in einem Repository gespeichert, z. B. in einem Amazon S3-Bucket.

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

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

Weitere Informationen:

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

Bereitstellungsaktionen für Service Catalog

Service Catalog ermöglicht es Organisationen, Kataloge von -Produkten zu erstellen und zu verwalten, die für die Verwendung in genehmigt sind AWS.

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

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

Weitere Informationen:

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

Amazon Alexa-Bereitstellungsaktionen

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

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong) und Europa (Mailand) nicht verfügbar. Informationen zur Verwendung anderer Bereitstellungsaktionen, die in dieser Region verfügbar sind, finden Sie unter Bereitstellungsaktions-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

CodeDeploy koordiniert Anwendungsbereitstellungen auf Amazon EC2-Instances, On-Premises-Instances oder beidem. Sie können so konfigurieren CodePipeline , dass CodeDeploy für die Bereitstellung Ihres Codes verwendet wird. Sie können die CodeDeploy Anwendung, Bereitstellung und Bereitstellungsgruppe erstellen, die Sie in einer Bereitstellungsaktion in einer Phase verwenden können, entweder bevor Sie die Pipeline erstellen oder wenn Sie den Assistenten zum Erstellen von Pipelines verwenden.

Weitere Informationen:

- Schritt 3: Erstellen Sie eine Anwendung in CodeDeploy
- Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)

XebiaLabs Aktionen bereitstellen

Sie können CodePipeline so konfigurieren, dass XebiaLabs für die Bereitstellung Ihres Codes in einer oder mehreren Aktionen in einer Pipeline verwendet wird.

Weitere Informationen:

- Verwenden von XL Deploy mit CodePipeline

Integration von Genehmigungsaktionen in Amazon Simple Notification Service

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

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

Weitere Informationen:

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

Aufrufen von Aktionsintegrationen

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

Themen

- Lambda-Aufrufaktionen
- Snyk-Aufrufaktionen
- Step Functions ruft Aktionen auf

Lambda-Aufrufaktionen

Mit Lambda können Sie Code ausführen, ohne Server bereitstellen oder verwalten zu müssen. Sie können so konfigurieren CodePipeline , dass Lambda-Funktionen verwendet werden, um Ihren

Pipelines Flexibilität und Funktionalität zu verleihen. Sie können die Lambda-Funktion erstellen, die als Aktion in einer Phase hinzugefügt werden soll, entweder bevor Sie die Pipeline erstellen oder wenn Sie den Assistenten zum Erstellen einer Pipeline verwenden.

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für AWS Lambda
- Aufrufen einer AWS Lambda-Funktion in einer Pipeline in CodePipeline

Snyk-Aufrufaktionen

Sie können so konfigurieren CodePipeline , dass Ihre Open-Source-Umgebungen sicher bleiben, indem Sie Sicherheitsschwachstellen erkennen und beheben und Abhängigkeiten in Ihrem Anwendungscode und Ihren Container-Images aktualisieren. Sie können auch die Snyk-Aktion in verwenden CodePipeline , um Sicherheitstestkontrollen in Ihrer Pipeline zu automatisieren.

Weitere Informationen:

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

Step Functions ruft Aktionen auf

Mit Step Functions können Sie Zustandsautomaten erstellen und konfigurieren. Sie können so konfigurieren CodePipeline , dass Step Functions-Aufrufaktionen verwendet werden, um Zustandsautomaten-Ausführungen auszulösen.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong) und Europa (Mailand) nicht verfügbar. Informationen zum Verweisen auf andere verfügbare Aktionen finden Sie unter Produkt- und Serviceintegrationen mit CodePipeline.

Weitere Informationen:

- CodePipeline Referenz zur Aktionskonfiguration für AWS Step Functions
- Tutorial: Verwenden einerAWS Step FunctionsAufrufen einer Aktion 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 Änderungen in Ihrem auf der AWS-Services Grundlage von Regeln erkennt, die Sie definieren, und eine Aktion in einer oder mehreren aufruft AWS-Services , die im Falle einer Änderung angegeben wird. • Starten Sie eine Pipeline-Ausführung automatisch, wenn sich etwas ändert – Sie können CodePipeline als Ziel in Regeln konfigurieren, die in Amazon eingerichtet sind EventBridge. Dadurch werden Pipelines automatisch gestartet, wenn es Änderungen in einem anderen Service gibt. Weitere Informationen: • Was ist Amazon EventBridge? • Starten einer Pipeline in CodePipeline. • CodeCommit -Quellaktionen und EventBridge • Erhalten Sie Benachrichtigungen, wenn sich ein Pipeline-Status ändert – Sie können EventBridge Regeln einrichten, um Änderunge n des Ausführungsstatus für eine Pipeline, Phase oder Aktion zu erkennen und darauf zu reagieren. Weitere Informationen: • Überwachen von CodePipeline Ereignissen • Tutorial: Einrichten einer CloudWatch Events-Regel, um E-Mail-Be nachrichtigungen zu Änderungen am Pipelinestatus zu empfangen
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-Instance eine - AWS Cloud9 Entwicklungsumgebung. Weitere Informationen finden Sie im AWS Cloud9 -Benutzerhandbuch. Weitere Informationen: • Einrichten von AWS Cloud9
AWS CloudTrail	CloudTrail erfasst AWS API-Aufrufe und zugehörige Ereignisse, die von oder im Namen eines AWS Kontos getätigt wurden, und stellt Protokoll dateien in einem von Ihnen angegebenen Amazon S3-Bucket bereit. 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. Standardmäßig CodePipeline verwendet , AWS KMS um Artefakte für Pipelines zu verschlüsseln, die in Amazon 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 der Pipeline hinzufügen und Kontorichtlinien und -rollen einrichten, um den kontoübergreifenden Zugriff zu ermöglichen. Weitere Informationen finden Sie unter Erstellen einer Pipeline in CodePipeline, die Ressourcen aus einem anderen AWS-Konto verwendet. • Um eine Pipeline aus einem AWS Konto zu erstellen, das einen - AWS CloudFormation Stack auf einem anderen 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 auf einem anderen 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 AWS standardmäßigen 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 der serverseitigen Verschlüsselung für Artefakte, die in Amazon S3 für gespeichert sind CodePipeline. Für ein können AWS KMS key Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN verwenden.

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

Beispiele aus der Community

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

> **Note**
>
> Diese Links dienen nur zu Informationszwecken und sollten nicht als umfassende Liste oder als Empfehlung für den Inhalt der Beispiele betrachtet werden. AWS ist nicht für den Inhalt oder die Genauigkeit externer Inhalte verantwortlich.

Themen

- Integrationsbeispiele: Blog-Beiträge

Integrationsbeispiele: Blog-Beiträge

- Nachverfolgen des AWS CodePipeline Build-Status aus dem Git-Repository eines Drittanbieters

  Erfahren Sie, wie Sie Ressourcen einrichten, die Ihren Pipeline- und Build-Aktionsstatus in Ihrem Drittanbieter-Repository anzeigen, sodass der Entwickler den Status verfolgen kann, ohne den Kontext zu wechseln.

  Veröffentlicht im März 2021
- CI/CD mit AWS CodeCommit, AWS CodeBuildAWS CodeDeploy und vervollständigen AWS CodePipeline

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

Veröffentlicht September 2020

- So stellen Sie von GitHub auf Amazon EC2 bereit mit CodePipeline

  Erfahren Sie, wie Sie CodePipeline von Grund auf einrichten, um Entwicklungs-, Test- und Produktionszweige in separaten Bereitstellungsgruppen bereitzustellen. Erfahren Sie, wie Sie IAM-Rollen, den CodeDeploy Agenten und CodeDeployzusammen mit verwenden und konfigurieren CodePipeline.

  Veröffentlicht im April 2020

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

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

  Veröffentlicht im März 2020

- Implementieren DevSecOps mit CodePipeline

  Erfahren Sie, wie Sie eine CI/CD-Pipeline in verwenden CodePipeline , um präventive und detektivische Sicherheitskontrollen zu automatisieren. In diesem Beitrag wird beschrieben, wie Sie eine Pipeline verwenden, um eine einfache Sicherheitsgruppe zu erstellen und Sicherheitsprüfungen während der Quell-, Test- und Produktionsphase durchzuführen, um die Sicherheitslage Ihrer AWS Konten zu verbessern.

  Veröffentlichung März 2017

- Kontinuierliche Bereitstellung in Amazon ECS mit CodePipeline, , CodeBuildAmazon ECR und AWS CloudFormation

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

  - Laden Sie eine AWS CloudFormation Beispielvorlage und Anweisungen für die Verwendung herunter, um Ihre eigene Pipeline für die kontinuierliche Bereitstellung aus dem ECS-Referenzarchitektur: Continuous Deployment Repo auf zu erstellen GitHub.

Veröffentlicht im Januar 2017

- Kontinuierliche Bereitstellung für serverlose Anwendungen

  Erfahren Sie, wie Sie eine Sammlung von verwenden AWS-Services , um eine Pipeline für die kontinuierliche Bereitstellung für Ihre Serverless-Anwendungen zu erstellen. Sie verwenden das Serverless Application Model (SAM), um die Anwendung und ihre Ressourcen zu definieren und Ihre 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 CodePipeline und Dynatrace

  Erfahren Sie, wie Sie Dynatrace-Überwachungslösungen verwenden CodePipeline, um Pipelines in zu skalieren, Testausführungen automatisch zu analysieren, bevor Code festgeschrieben wird, und optimale Vorlaufzeiten einzuhalten.

Veröffentlichung November 2016

- Erstellen einer Pipeline für AWS Elastic Beanstalk in CodePipeline Verwendung von AWS CloudFormation und CodeCommit

  Erfahren Sie, wie Sie die kontinuierliche Bereitstellung in einer CodePipeline Pipeline für eine Anwendung in implementieren AWS Elastic Beanstalk. Alle AWS Ressourcen werden automatisch über eine AWS CloudFormation -Vorlage bereitgestellt. Diese Anleitung enthält auch CodeCommit und AWS Identity and Access Management (IAM).

Veröffentlicht im Mai 2016

- Automatisieren von CodeCommit und CodePipeline in AWS CloudFormation

  Verwenden Sie AWS CloudFormation , um die Bereitstellung von AWS Ressourcen für eine kontinuierliche Bereitstellungs-Pipeline zu automatisieren, die CodeCommit CodePipeline, CodeDeployund verwendet AWS Identity and Access Management.

Veröffentlicht im April 2016

- Erstellen einer kontoübergreifenden Pipeline in AWS CodePipeline

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

Veröffentlichung März 2016

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

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

  Veröffentlichung März 2016

- Erstellen einer Pipeline mit der AWS CodePipeline Konsole

  Erfahren Sie, wie Sie mit der - AWS CodePipeline Konsole eine zweistufige Pipeline in einer Anleitung erstellen, die auf der basiert AWS CodePipeline Tutorial: Erstellen einer vierstufigen Pipeline.

  Veröffentlichung März 2016

- Mocking von AWS CodePipeline Pipelines mit AWS Lambda

  Erfahren Sie, wie Sie eine Lambda-Funktion aufrufen, mit der Sie die Aktionen und Phasen eines CodePipeline Softwarebereitstellungsprozesses nach Ihrem Entwurf visualisieren können, bevor die Pipeline betriebsbereit ist. Während Sie Ihre Pipeline-Struktur entwerfen, können Sie die Lambda-Funktion verwenden, um zu testen, ob Ihre Pipeline erfolgreich abgeschlossen wird.

  Veröffentlichung Februar 2016

- Ausführen von AWS Lambda Funktionen in CodePipeline mithilfe von AWS CloudFormation

  Erfahren Sie, wie Sie einen - AWS CloudFormation Stack erstellen, der alle Ressourcen bereitstellt, die in der Benutzerhandbuchaufgabe AWS verwendet werdenAufrufen einer AWS Lambda-Funktion in einer Pipeline in CodePipeline.

  Veröffentlichung Februar 2016

- Bereitstellen von benutzerdefinierten CodePipeline Aktionen in AWS CloudFormation

  Erfahren Sie, wie Sie mit benutzerdefinierte Aktionen in AWS CloudFormation bereitstellen CodePipeline.

Veröffentlicht im Januar 2016

- Bereitstellung CodePipeline mit AWS CloudFormation

  Erfahren Sie, wie Sie eine grundlegende Continuous-Delivery-Pipeline in CodePipeline mithilfe von bereitstellen AWS CloudFormation.

  Erschienen: Dezember 2015

- Bereitstellen von CodePipeline zu AWS OpsWorks mithilfe einer benutzerdefinierten Aktion und AWS Lambda

  Erfahren Sie, wie Sie Ihre Pipeline und die - AWS Lambda Funktion für die Bereitstellung in AWS OpsWorks mithilfe von konfigurieren CodePipeline.

  Veröffentlichung Juli 2015

CodePipeline -Tutorials

Nach Abschluss der Schritte in Erste Schritte mit CodePipeline können Sie eines der AWS CodePipeline-Tutorials in diesem Benutzerhandbuch versuchen:

Ich möchte den Assistenten verwenden, um eine Pipeline zu erstellen, die verwendet, CodeDeploy um eine Beispielanwendung aus einem Amazon S3-Bucket auf Amazon EC2-Instances bereitzustellen, auf denen Amazon Linux ausgeführt wird. Nachdem ich meine Pipeline mit dem Assistenten erstellt haben, möchte ich eine dritte Stufe hinzufügen.	Siehe Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket).
Ich möchte eine zweistufige Pipeline erstellen, die verwendet, CodeDeploy um eine Beispiela nwendung aus einem CodeCommit Repository auf einer Amazon EC2-Instance bereitzustellen, auf der Amazon Linux ausgeführt wird.	Siehe Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository).
Ich möchte der Pipeline mit drei Stufen, die ich im ersten Tutorial erstellt habe, eine Build-Stufe hinzufügen. Die neue Stufe verwendet Jenkins für die Erstellung meiner Anwendung.	Siehe Tutorial: Erstellen einer vierstufigen Pipeline.
Ich möchte eine CloudWatch Ereignisregel einrichten, die Benachrichtigungen sendet, wenn es Änderungen am Ausführungsstatus meiner Pipeline, Phase oder Aktion gibt.	Siehe Tutorial: Einrichten einer CloudWatch Events-Regel, um E-Mail-Benachrichtigungen zu Änderungen am Pipelinestatus zu empfangen.
Ich möchte eine Pipeline mit einer GitHub Quelle erstellen, die eine Android-App mit CodeBuild und erstellt und testetAWS Device Farm.	Siehe Tutorial: Erstellen einer Pipeline, die Ihre Android-App mit erstellt und testet AWS Device Farm.

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

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

- Erstellen einer Pipeline, die im -Benutzerhandbuch verwendet CodeBuild AWS CodeBuild
- Verwenden von CodePipeline mit AWS OpsWorks Stacks im AWS OpsWorks -Benutzerhandbuch
- Kontinuierliche Bereitstellung mit CodePipeline im AWS CloudFormation -Benutzerhandbuch
- Erste Schritte mit Elastic Beanstalk im -AWS Elastic BeanstalkEntwicklerhandbuch
- Einrichten einer Pipeline für die kontinuierliche Bereitstellung mit CodePipeline

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

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

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

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

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

Themen

- Voraussetzungen
- Schritt 1: Öffne CloudShell und klone dein Repository
- Schritt 2: Eine Pipeline zum Auslösen auf Git-Tags erstellen
- Schritt 3: Markiere deine Commits zur Veröffentlichung
- Schritt 4: Änderungen veröffentlichen und Logs einsehen

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 verwendenAWS Management Console, um eine Verbindung herzustellen, werden Sie aufgefordert, sich mit Ihren GitHub Anmeldeinformationen anzumelden.

Schritt 1: Öffne CloudShell und klone dein Repository

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

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

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

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

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

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

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

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

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

So erstellen Sie mit dem Assistenten eine Pipeline

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

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

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

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

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

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

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

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

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

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

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

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

Wählen Sie Weiter.

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

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

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

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

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

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

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

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

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

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

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

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

i. Wählen Sie Weiter.

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

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

Schritt 3: Markiere deine Commits zur Veröffentlichung

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

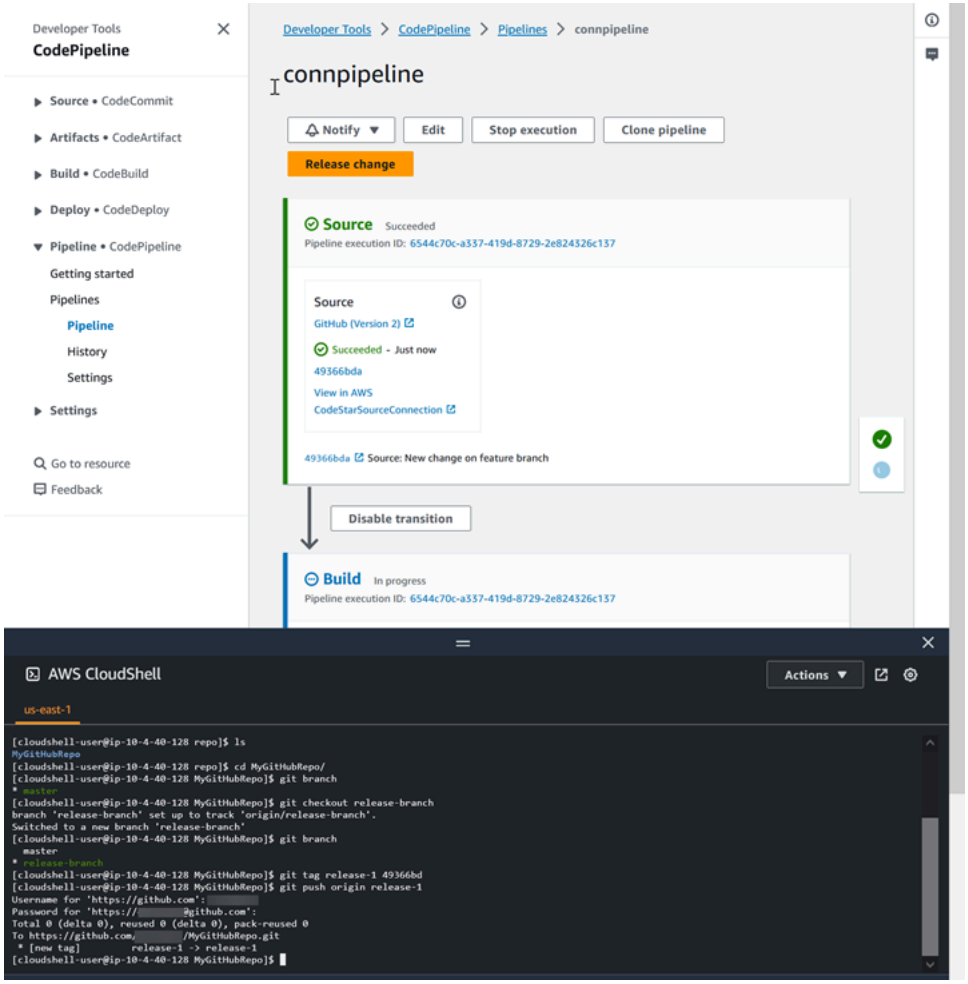
1. Verweisen Sie auf die kopierten Commit-IDs, die Sie taggen möchten. Um die Commits in den einzelnen Branches anzuzeigen, geben Sie im CloudShell Terminal den folgenden Befehl ein, um die Commit-IDs zu erfassen, die Sie taggen möchten:

```
git log
```

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

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

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



Schritt 4: Änderungen veröffentlichen und Logs einsehen

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

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

Tutorial: Filtern nach Verzweigungsnamen für Pull-Anforderungen zum Starten Ihrer Pipeline

In diesem Tutorial erstellen Sie eine Pipeline, die eine Verbindung zu Ihrem GitHub.com-Repository herstellt, in dem die Quellaktion so konfiguriert ist, dass Ihre Pipeline mit einer Auslöserkonfiguration gestartet wird, die nach Pull-Anforderungen filtert. Wenn ein bestimmtes Pull-Anforderungsereignis für eine bestimmte Verzweigung eintritt, wird Ihre Pipeline gestartet. Dieses Beispiel zeigt Ihnen, wie Sie eine Pipeline erstellen, die das Filtern nach Verzweigungsnamen ermöglicht. Weitere Informationen zum Arbeiten mit Auslösern finden Sie unter Auslöserfilterung in der Pipeline-JSON (CLI). Weitere Informationen zum Filtern mit Regex-Mustern im Glob-Format finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

Dieses Tutorial stellt über den `CodeStarSourceConnection` Aktionstyp eine Verbindung zu GitHub.com her.

Themen

- Voraussetzungen
- Schritt 1: Erstellen einer Pipeline, um bei der Pull-Anforderung für bestimmte Verzweigungen zu beginnen
- Schritt 2: Erstellen und Zusammenführen einer Pull-Anforderung in GitHub.com, um Ihre Pipeline-Ausführungen zu starten

Voraussetzungen

Sie benötigen Folgendes, um starten zu können:

- Erstellen Sie ein GitHub.com-Repository mit Ihrem GitHub.com-Konto.
- Halten Sie Ihre GitHub Anmeldeinformationen bereit. Wenn Sie AWS Management Console eine Verbindung über die einrichten, werden Sie aufgefordert, sich mit Ihren - GitHub Anmeldeinformationen anzumelden.

Schritt 1: Erstellen einer Pipeline, um bei der Pull-Anforderung für bestimmte Verzweigungen zu beginnen

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

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

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

6. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie unter Artifact store (Artefaktspeicher) die Option Default location (Standardstandort) aus, um den

Standard-Artefakt-Speicherort für die Pipeline in der entsprechenden Region zu verwenden, beispielsweise mit dem Amazon S3-Artefakt-Bucket als Standard.

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

Wählen Sie Next (Weiter).

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

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

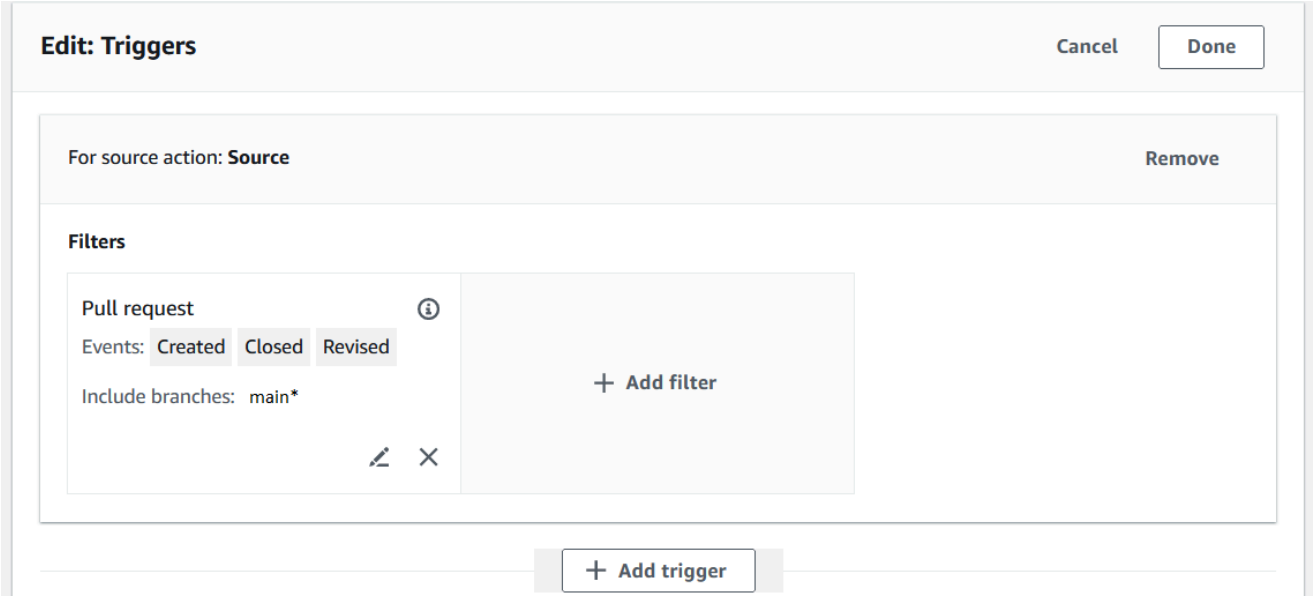
   b. Wählen Sie unter Verbindung eine vorhandene Verbindung aus oder erstellen Sie eine neue. Informationen zum Erstellen oder Verwalten einer Verbindung für Ihre GitHub Quellaktion finden Sie unter GitHub Verbindungen.

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

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

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

      Geben Sie unter Verzweigungen im Feld Einschließen ein`main*`.



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

Wählen Sie Weiter aus.

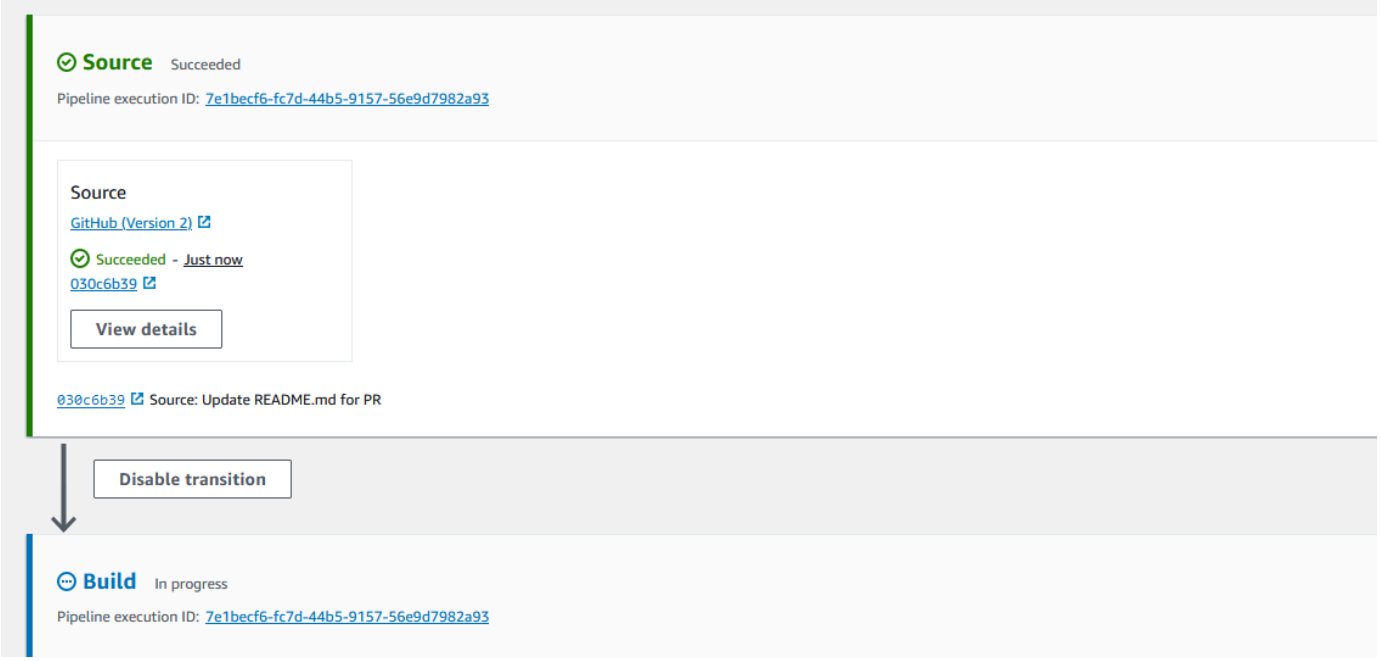
8. Wählen Sie unter Build-Stufe hinzufügen unter Build-Anbieter die Option ausAWS CodeBuild. Belassen Sie unter Region als Standardeinstellung die Pipeline-Region. Wählen oder erstellen Sie das Build-Projekt wie unter beschriebenTutorial: Verwende Git-Tags, um deine Pipeline zu starten. Diese Aktion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

Schritt 2: Erstellen und Zusammenführen einer Pull-Anforderung in GitHub.com, um Ihre Pipeline-Ausführungen zu starten

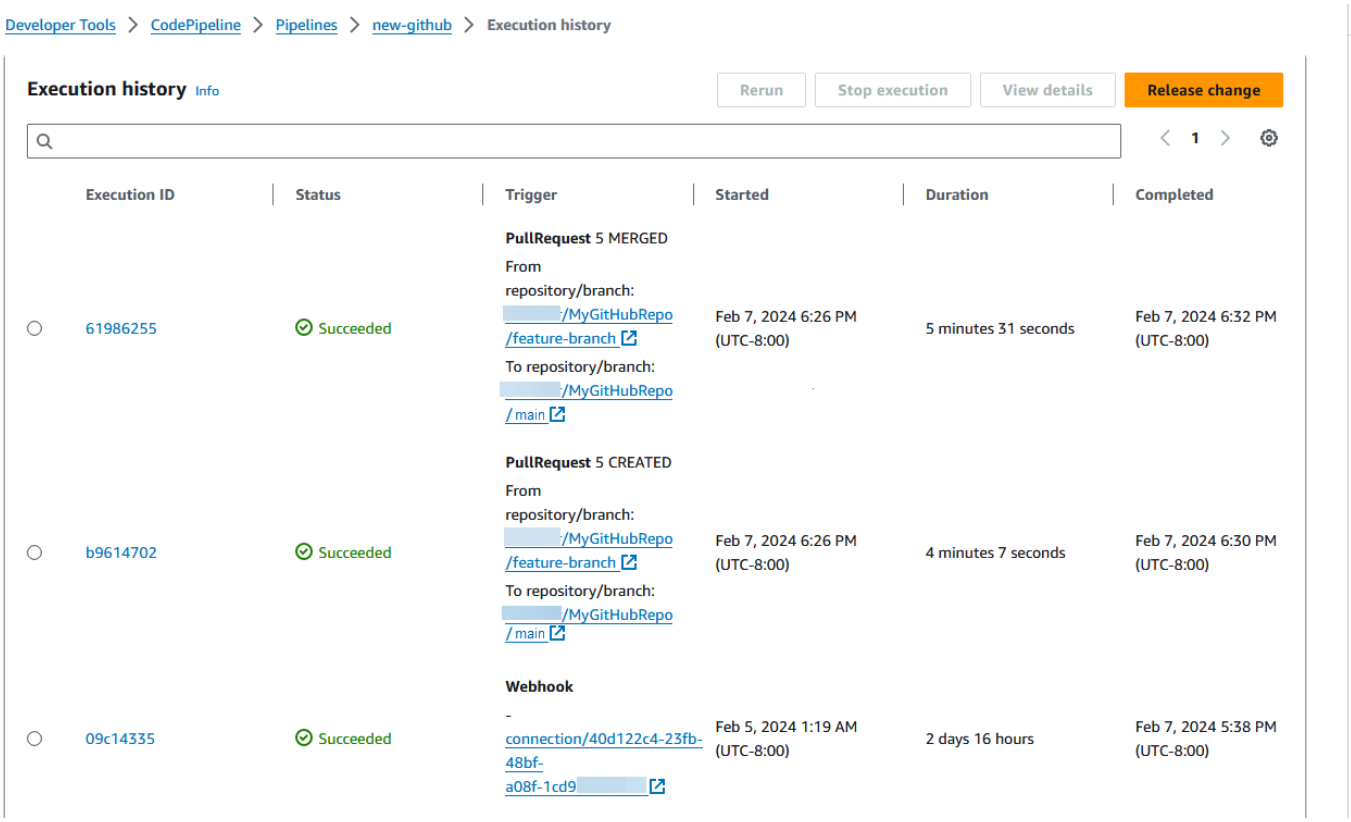
In diesem Abschnitt erstellen Sie eine Pull-Anforderung und führen sie zusammen. Dadurch wird Ihre Pipeline gestartet, mit einer Ausführung für die geöffnete Pull-Anforderung und einer Ausführung für die geschlossene Pull-Anforderung.

So erstellen Sie eine Pull-Anforderung und starten Ihre Pipeline

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



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



Tutorial: Variablen auf Pipeline-Ebene verwenden

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

Themen

- Voraussetzungen
- 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 CodeCommit Repository.
- Fügen Sie dem Repository eine .txt-Datei hinzu.

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

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

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

So erstellen Sie mit dem Assistenten eine Pipeline

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

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

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

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

7. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie unter Artifact store (Artefaktspeicher) die Option Default location (Standardstandort) aus, um den

Standard-Artefakt-Speicherort für die Pipeline in der entsprechenden Region zu verwenden, beispielsweise mit dem Amazon S3-Artefakt-Bucket als Standard.

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

Wählen Sie Next (Weiter).

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

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

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

   Wählen Sie Weiter.

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

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

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

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

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

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

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

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

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

```
version: 0.2
#env:
  #variables:
     # key: "value"
     # key: "value"
  #parameter-store:
     # key: "value"
     # key: "value"
  #git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
    #commands:
      # - command
      # - command
  #pre_build:
    #commands:
      # - command
      # - command
  build:
    commands:
      - echo $CUSTOM_VAR1
  #post_build:
    #commands:
```

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

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

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

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

j. Wählen Sie Weiter.

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

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

Schritt 2: Änderungen veröffentlichen und Protokolle anzeigen

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

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

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

Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket)

Am einfachsten ist das Erstellen einer Pipeline mithilfe des Assistenten Create Pipeline (Pipeline erstellen) in der AWS CodePipeline-Konsole möglich.

In diesem Tutorial erstellen Sie eine Pipeline mit zwei Phasen, die einen versionierten S3-Bucket und CodeDeploy verwendet, um eine Beispielanwendung zu veröffentlichen.

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, können Sie Ihre Quelldatei (en) in eine einzige ZIP-Datei komprimieren und die ZIP-Datei in Ihren Quell-Bucket hochladen. Sie können auch eine einzelne Datei ungezippt hochladen, aber nachgelagerte Aktionen, die eine ZIP-Datei erwarten, schlagen dann fehl.

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

> **Important**
>
> Viele der Aktionen, die Sie in diesem Verfahren zu Ihrer Pipeline hinzufügen, beinhalten AWS Ressourcen, die Sie erstellen müssen, bevor Sie die Pipeline erstellen. AWSRessourcen 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. AWSRessourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Hinzufügen einer regionsübergreifenden Aktion in CodePipeline.

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

Themen

- Schritt 1: Erstellen eines S3-Buckets für Ihre Anwendung
- Schritt 2: Amazon EC2 EC2-Windows-Instances erstellen und den CodeDeploy Agenten installieren
- 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 Phasen in CodePipeline
- Schritt 7: Bereinigen von Ressourcen

Schritt 1: Erstellen eines S3-Buckets für Ihre Anwendung

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

So erstellen Sie einen S3-Bucket

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

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

Wählen Sie unter Region die Region aus, in der Sie Ihre Pipeline erstellen möchten, z. B. USA West (Oregon), und wählen Sie dann Create Bucket aus.

4. Nachdem der Bucket erstellt wurde, wird ein Erfolgsbanner angezeigt. Wählen Sie Go to bucket details (Zu Bucket-Details wechseln) aus.
5. Wählen Sie auf der Registerkarte Properties (Eigenschaften) die Option Versioning aus. Wählen Sie Enable versioning (Versioning aktivieren) und dann Save (Speichern)aus.

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

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

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

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

      - `appspec.yml`— Die Anwendungsspezifikationsdatei (AppSpecDatei) ist eine Datei im YAML-Format, die CodeDeploy zur Verwaltung einer Bereitstellung verwendet wird. Weitere Informationen zu der AppSpec Datei finden Sie unter CodeDeploy AppSpec Dateireferenz im AWS CodeDeploy Benutzerhandbuch.

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

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

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

   a. Klicken Sie auf Hochladen.

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

   c. Klicken Sie auf Hochladen.

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

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

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

So erstellen Sie eine Instance-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/).
2. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen AWS-Servicedie Option aus. Wählen Sie unter Choose a use case (Anwendungsfall auswählen) die Option EC2 und dann Next: Permissions (Weiter: Berechtigungen) aus.
5. Suchen Sie nach der genannten Richtlinie und wählen Sie sie aus **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Suchen Sie nach der genannten Richtlinie und wählen Sie sie aus **`AmazonSSMManagedInstanceCore`**. Wählen Sie Next: Tags (Weiter: Tags) aus.
7. Klicken Sie auf Weiter: Prüfen. Geben Sie einen Namen für die Rolle ein (z. B. **`EC2InstanceRole`**).

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

   Wählen Sie Rolle erstellen aus.

Starten von Instances

1. Öffnen Sie die Amazon-EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie in der Seitennavigation Instances und dann oben auf der Seite Launch Instances aus.

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

   Sie können auch Proceed without a key pair wählen.

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

7. Gehen Sie unter Netzwerkeinstellungen wie folgt vor.

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

   - Klicken Sie neben Assign a security group (Sicherheitsgruppe zuweisen) auf Create a new security group (Neue Sicherheitsgruppe zuweisen).
   - Wählen Sie in der Zeile für SSH unter Quelltyp die Option Meine IP aus.
   - Wählen Sie Sicherheitsgruppe hinzufügen, wählen Sie HTTP und dann unter Quelltyp die Option Meine IP aus.
8. Erweitern Sie Advanced Details (Erweiterte Details). Wählen Sie im IAM-Instanzprofil die IAM-Rolle aus, die Sie im vorherigen Verfahren erstellt haben (z. B.**`EC2InstanceRole`**).
9. Geben Sie unter Zusammenfassung unter Anzahl der Instanzen den Wert.. 2
10. Wählen Sie Launch Instance (Instance starten) aus.
11. Wählen Sie View all Instances (Alle Instances anzeigen) aus, um die Bestätigungsseite zu schließen und zur Konsole zurückzukehren.

12. Sie können den Status des Starts auf der Seite Instances anzeigen. Wenn Sie eine Instance starten, lautet ihr anfänglicher Status `pending`. Nachdem die Instance gestartet ist, lautet ihr Status `running`. Sie erhält dann einen öffentlichen DNS-Namen. (Wenn die Spalte Public DNS nicht angezeigt wird, wählen Sie das Show/Hide-Symbol aus und wählen Sie dann Public DNS aus.)
13. Es kann ein paar Minuten dauern, bis die Instance zur Verbindung bereitsteht. Stellen Sie sicher, dass Ihre Instance ihre Statusprüfungen bestanden hat. Sie können diese Informationen in der Spalte Statusprüfungen abrufen.

Schritt 3: Erstellen Sie eine Anwendung in CodeDeploy

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

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

Um eine CodeDeploy Servicerolle zu erstellen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/).
2. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Vertrauenswürdige Entität auswählen die Option AWS-Service. Wählen Sie unter Use case (Anwendungsfall) CodeDeploy aus. Wählen Sie CodeDeployaus den aufgelisteten Optionen aus. Wählen Sie Weiter. Die `AWSCodeDeployRole`-verwaltete Richtlinie ist der Rolle bereits zugewiesen.
5. Wählen Sie Weiter.
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 Applications (Anwendungen) nicht angezeigt wird, wählen Sie im AWS CodeDeploy -Menü Applications (Anwendungen) aus.
3. Wählen Sie Erstellen Sie Anwendung aus.
4. Geben Sie unter Application name (Anwendungsname) `MyDemoApplication` ein.
5. Wählen Sie unter Compute Platform (Plattform für die Datenverarbeitung) die Option EC2/On-premises (EC2/Lokal) aus.
6. Wählen Sie Erstellen Sie Anwendung aus.

Um eine Bereitstellungsgruppe zu erstellen in CodeDeploy

1. Wählen Sie auf der Seite mit Ihrer Anwendung Create deployment group (Bereitstellungsgruppe erstellen).
2. Geben Sie unter Deployment group name (Name der Bereitstellungsgruppe) **`MyDemoDeploymentGroup`** ein.
3. Wählen Sie unter Servicerolle die Servicerolle aus, die Sie zuvor erstellt haben. Sie müssen eine Servicerolle verwenden, die mindestens den unter Eine Servicerolle erstellen für CodeDeploy beschriebenen Vertrauensstellungen und Berechtigungen vertraut AWS CodeDeploy . Informationen zum Erhalt des Servicerollen-ARN finden Sie unter Abrufen des Servicerollen-ARN (Konsole).
4. Wählen Sie unter Deployment type (Bereitstellungstyp) die Option In-place (Direkt).
5. Wählen Sie unter Environment configuration (Umgebungskonfiguration) die Option Amazon EC2-Instances aus. Wählen Sie im Feld Schlüssel die Option Name und geben Sie im Feld Wert die Eingabe ein**`MyCodePipelineDemo`**.

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

6. Wählen Sie unter Agentenkonfiguration mit AWS Systems Manager die Option Jetzt und planen Sie Updates. Dadurch wird der Agent auf der Instanz installiert. Die Windows-Instanz ist bereits mit dem SSM-Agenten konfiguriert und wird nun mit dem CodeDeploy Agenten aktualisiert.
7. Wählen Sie unter Bereitstellungseinstellungen die Option`CodeDeployDefault.OneAtaTime`.

8. Stellen Sie sicher, dass unter Load Balancer das Feld Load Balancing aktivieren nicht ausgewählt ist. Sie müssen keinen Load Balancer einrichten oder eine Zielgruppe für dieses Beispiel auswählen. Nachdem Sie das Kontrollkästchen deaktiviert haben, werden die Load Balancer-Optionen nicht angezeigt.
9. Behalten Sie im Abschnitt Advanced (Fortgeschritten) die Standardwerte bei.
10. Wählen Si Create deployment group (Bereitstellungsgruppe erstellen).

Schritt 4: Erstellen Sie Ihre erste Pipeline in CodePipeline

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

Um einen CodePipeline automatisierten Release-Prozess zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) aus, oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyFirstPipeline`** ein.

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

4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:

   - Wählen Sie Neue Servicerolle aus, CodePipeline um die Erstellung einer neuen Servicerolle in IAM zu ermöglichen.

- Wählen Sie Existing service role (Vorhandene Servicerolle), um eine Servicerolle zu verwenden, die in IAM bereits erstellt wurde. Wählen Sie unter Name der Rolle in der Liste Ihre Servicerolle aus.

6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie in Step 2: Add source stage (Schritt 2: Quellstufe hinzufügen) unter Source provider (Quellanbieter) die Option Amazon S3. Geben Sie unter Bucket den Namen des S3-Buckets ein, den Sie in Schritt 1: Erstellen eines S3-Buckets für Ihre Anwendung erstellt haben. Geben Sie im S3-Objektschlüssel den Objektschlüssel mit oder ohne Dateipfad ein, und denken Sie daran, die Dateierweiterung einzuschließen. Geben Sie beispielsweise für `SampleApp_Windows.zip` den Namen der Beispieldatei ein, wie in diesem Beispiel gezeigt:

   ```
   SampleApp_Windows.zip
   ```

   Klicken Sie auf Nächster Schritt.

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

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

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

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

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

- Eine Quellstufe mit dem Namen Source (Quelle), die Änderungen in der Versioning-Beispielanwendung erkennt, die im S3-Bucket gespeichert ist, und diese Änderungen in die Pipeline zieht.
- Eine 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 CodePipelineDemoApplication in CodeDeploy. Anschließend fügen Sie eine Phase hinzu, die eine Aktion enthält, die diese Bereitstellungsgruppe verwendet. Um eine weitere Phase hinzuzufügen, verwenden Sie die CodePipeline Konsole oder die, AWS CLI um die Struktur der Pipeline in einer JSON-Datei abzurufen und manuell zu bearbeiten, und führen dann den update-pipeline Befehl aus, um die Pipeline mit Ihren Änderungen zu aktualisieren.

Themen

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

Erstellen Sie eine zweite Bereitstellungsgruppe in CodeDeploy

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

Um eine zweite Bereitstellungsgruppe in zu erstellen CodeDeploy

1. Öffnen Sie die CodeDeploy Konsole unter https://console.aws.amazon.com/codedeploy.

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

Hinzufügen der Bereitstellungsgruppe als weitere Phase in Ihrer Pipeline

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

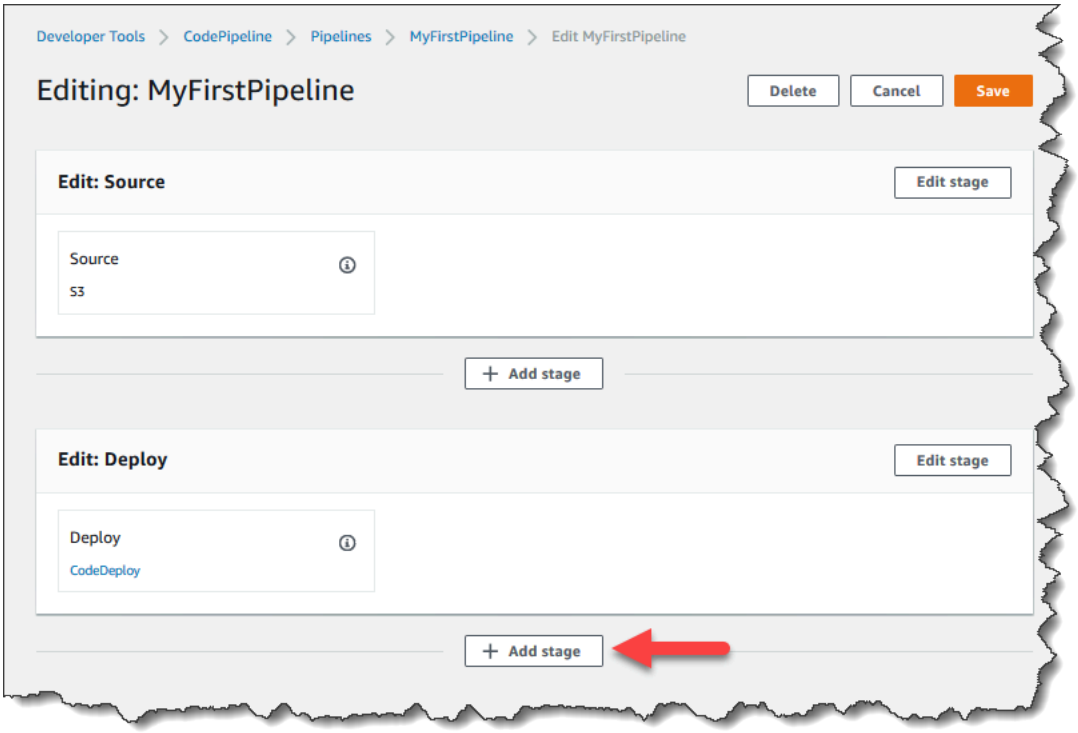
Themen

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

Erstellen einer dritten Phase (Konsole)

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

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie 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 unter Aktionsanbieter unter Bereitstellen die Option aus CodeDeploy.

8. Wählen Sie in dem CodeDeploy Abschnitt unter Anwendungsname eine Option `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 aus.
9. Klicken Sie auf der Seite Edit (Bearbeiten) auf Save (Speichern). Klicken Sie unter Save pipeline changes (Pipeline-Änderungen speichern) auf Save (Speichern).
10. Obwohl die neue Stufe der Pipeline hinzugefügt wurde, wird der Status No executions yet (Zurzeit keine Ausführungen) angezeigt. Dies liegt daran, dass noch keine Änderungen eine weitere Ausführung der Pipeline ausgelöst haben. Sie müssen die letzte Revision manuell erneut ausführen, um die Ausführung der Pipeline nach der Bearbeitung zu testen. Wählen Sie auf der Seite mit den Pipeline-Details die Option Versionsänderung aus und klicken Sie dann auf Freigabe, wenn Sie dazu aufgefordert werden. Dadurch wird die letzte Revision gestartet, die in jedem in einer Quellaktion der Pipeline angegebenen Quellspeicherort vorhanden ist.

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

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

    Dieser Befehl gibt ein `pipelineExecutionId`-Objekt zurück.
11. Kehren Sie zur CodePipeline Konsole zurück und wählen Sie in der Liste der Pipelines, ob Sie die Ansichtsseite öffnen MyFirstPipelinemöchten.

    Die Pipeline zeigt drei Phasen und den Status des ausgeführten Artefakts in den drei Phasen an. Es kann bis zu fünf Minuten dauern, bis die Pipeline alle Phasen durchlaufen hat. Für die ersten zwei Stufen wird wie zuvor die erfolgreiche Bereitstellung angezeigt. Die Stufe Production (Produktion) zeigt jedoch an, dass die Aktion Deploy-Second-Deployment fehlgeschlagen ist.
12. Klicken Sie in der Deploy-Second-Deployment--Aktion auf Details. Sie werden auf die Seite für die CodeDeploy-Bereitstellung weitergeleitet. In diesem Fall wird der Fehler durch die Bereitstellung der ersten Instance-Gruppe in allen EC2-Instances verursacht. Somit verbleiben keine Instances für die zweite Bereitstellungsgruppe.

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

Erstellen einer dritten Phase (CLI)

Auch wenn die Verwendung der AWS CLI für das Hinzufügen einer Phase zu Ihrer Pipeline komplexer als die Verwendung der Konsole ist, bietet dies mehr Transparenz in Bezug auf die Struktur der Pipeline.

So erstellen Sie eine dritte Phase für Ihre Pipeline

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

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

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

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

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

   ```
       ...
           ],
           "artifactStore": {
               "type": "S3"
               "location": "codepipeline-us-east-2-250656481468",
           },
           "name": "MyFirstPipeline",
   ```

```
        "version": 4
    },
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline",
        "updated": 1501626591.112,
        "created": 1501626591.112
    }
}
```

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

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

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

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

```
,
{
    "name": "Production",
     "actions": [
        {
         "inputArtifacts": [
             {
              "name": "MyApp"
             }
```

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

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

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

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

   Speichern Sie die Datei.

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

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

   Dieser Befehl gibt die gesamte Struktur der aktualisierten Pipeline zurück.

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

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

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

   Dieser Befehl gibt ein `pipelineExecutionId`-Objekt zurück.

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

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

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

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

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

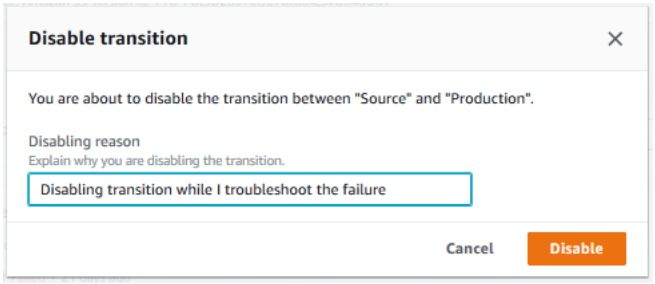
Sie können den Übergang zwischen Phasen in einer Pipeline aktivieren oder deaktivieren. Durch die Deaktivierung des Übergangs zwischen Phasen können Sie die Übergänge zwischen Phasen manuell steuern. Beispielsweise möchten Sie die ersten beiden Phasen einer Pipeline ausführen, den

Übergang zur dritten Phase jedoch deaktivieren, bis Sie für die Bereitstellung für die Produktion bereit sind oder während Sie ein Problem oder einen Ausfall für die betreffende Phase beheben.

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

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

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



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

> **Note**
>
> Wenn Sie möchten, dass diese dritte Phase erfolgreich ist, bearbeiten Sie die CodePipelineProductionFleet Bereitstellungsgruppe, bevor Sie den Übergang aktivieren, und geben Sie einen anderen Satz von EC2-Instances an, auf denen die Anwendung

bereitgestellt wird. Weitere Informationen dazu finden Sie unter Ändern der Einstellungen von Bereitstellungsgruppen. Wenn Sie mehrere EC2-Instances erstellen, können für Sie zusätzliche Kosten entstehen.

Schritt 7: Bereinigen von Ressourcen

Sie können einige der Ressourcen, die Sie in diesem Tutorial erstellt haben, für Tutorial: Erstellen einer vierstufigen Pipeline verwenden. Sie können beispielsweise die CodeDeploy Anwendung und die Bereitstellung wiederverwenden. Sie können eine Build-Aktion mit einem Anbieter wie CodeBuild, einem vollständig verwalteten Build-Service in der Cloud, konfigurieren. Sie können eine Build-Aktion auch so konfigurieren, dass ein Anbieter mit einem Build-Server oder -System verwendet wird, z. B. Jenkins.

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

So bereinigen Sie die in diesem Tutorial verwendeten Ressourcen

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

Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)

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

Pipeline stellt Ihre Änderungen an einer Amazon EC2 EC2-Instance bereit, die CodeDeploy als Bereitstellungsservice verwendet wird.

Die Pipeline hat zwei Phasen:

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

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

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

Schritt 1: Erstelle ein CodeCommit Repository

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

Um ein CodeCommit Repository zu erstellen

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

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

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

Schritt 2: Fügen Sie Ihrem CodeCommit Repository Beispielcode hinzu

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

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

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

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

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

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

   b. Laden Sie die komprimierte ZIP-Datei herunter.

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

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

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

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

   a. So verwenden Sie die CodeCommit Konsole, 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. 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 nun dem `main` Branch in deinem CodeCommit `MyDemoRepo` Repository hinzugefügt und können nun in eine Pipeline aufgenommen werden.

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

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

So erstellen Sie eine Instance-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/).
2. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen AWS-Servicedie Option aus. Wählen Sie unter 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 genannten **`AmazonEC2RoleforAWSCodeDeploy`**Richtlinie und wählen Sie sie aus.
6. Suchen Sie nach der genannten Richtlinie und wählen Sie sie aus **`AmazonSSMManagedInstanceCore`**. Wählen Sie Next: Tags (Weiter: Tags) aus.
7. Klicken Sie auf Weiter: Prüfen. Geben Sie einen Namen für die Rolle ein (z. B. **`EC2InstanceRole`**).

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

   Wählen Sie Rolle erstellen aus.

So starten Sie eine Instance

1. Öffnen Sie die Amazon-EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie in der Seitennavigation Instances und dann oben auf der Seite Launch Instances aus.
3. Geben Sie unter Name **`MyCodePipelineDemo`** ein. Dadurch wird der Instance ein Tag-Schlüssel von **`Name`** und ein Tag-Wert von **`MyCodePipelineDemo`** zugewiesen. Später erstellen Sie eine CodeDeploy Anwendung, die die Beispielanwendung auf dieser Instanz bereitstellt. CodeDeploywählt die bereitzustellenden Instanzen auf der Grundlage der Tags aus.
4. Suchen Sie unter Anwendungs- und Betriebssystem-Images (Amazon Machine Image) die Amazon Linux AMI-Option mit dem AWS Logo und stellen Sie sicher, dass sie ausgewählt ist.

(Dieses AMI wird als Amazon Linux 2 AMI (HVM) bezeichnet und trägt die Bezeichnung „Für das kostenlose Kontingent in Frage“.)

5. Wählen Sie unter Instance-Typ den `t2.micro` Typ aus, für den das kostenlose Kontingent in Frage kommt, als Hardwarekonfiguration für Ihre Instance.
6. Wählen Sie unter key pair (Anmeldung) ein Schlüsselpaar aus oder erstellen Sie eines.

   Sie können auch Proceed without a key pair wählen.

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

7. Gehen Sie unter Netzwerkeinstellungen wie folgt vor.

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

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

8. Erweitern Sie Advanced Details (Erweiterte Details). Wählen Sie im IAM-Instanzprofil die IAM-Rolle aus, die Sie im vorherigen Verfahren erstellt haben (z. B.**`EC2InstanceRole`**).
9. Geben Sie unter Zusammenfassung unter Anzahl der Instanzen den Wert.. 1
10. Wählen Sie Launch Instance (Instance starten) aus.
11. Sie können den Status des Starts auf der Seite Instances anzeigen. Wenn Sie eine Instance starten, lautet ihr anfänglicher Status `pending`. Nachdem die Instance gestartet ist, lautet ihr Status `running`. Sie erhält dann einen öffentlichen DNS-Namen. (Wenn die Spalte Public DNS nicht angezeigt wird, wählen Sie das Show/Hide-Symbol aus und wählen Sie dann Public DNS aus.)

Schritt 4: Erstellen einer Anwendung in CodeDeploy

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

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

Um eine CodeDeploy Servicerolle zu erstellen

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

Um eine Anwendung zu erstellen in CodeDeploy

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

Um eine Bereitstellungsgruppe zu erstellen in CodeDeploy

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

1. Wählen Sie auf der Seite mit Ihrer Anwendung Create deployment group (Bereitstellungsgruppe erstellen).
2. Geben Sie unter Deployment group name (Name der Bereitstellungsgruppe) **`MyDemoDeploymentGroup`** ein.
3. Wählen Sie unter Servicerolle den ARN der Servicerolle aus, die Sie zuvor erstellt haben (z. B.**`arn:aws:iam::`*`account_ID`*`:role/CodeDeployRole`**).
4. Wählen Sie unter Deployment type (Bereitstellungstyp) die Option In-place (Direkt).
5. Wählen Sie unter Environment configuration (Umgebungskonfiguration) die Option Amazon EC2-Instances aus. Geben Sie im Feld Schlüssel den Wert ein**`Name`**. Geben Sie im Feld Wert den Namen ein, mit dem Sie die Instanz markiert haben (z. B.**`MyCodePipelineDemo`**).
6. Wählen Sie unter Agentenkonfiguration mit AWS Systems Manager die Option Jetzt und planen Sie Updates. Dadurch wird der Agent auf der Instanz installiert. Die Linux-Instanz ist bereits mit dem SSM-Agenten konfiguriert und wird nun mit dem CodeDeploy Agenten aktualisiert.
7. Wählen Sie in Deployment configuration (Bereitstellungskonfiguration) die Option `CodeDeployDefault.OneAtaTime` aus.
8. Stellen Sie sicher, dass unter Load Balancer die Option Load Balancing aktivieren nicht ausgewählt ist. Sie müssen keinen Load Balancer einrichten oder eine Zielgruppe für dieses Beispiel auswählen.
9. Wählen Si Create deployment group (Bereitstellungsgruppe erstellen).

Schritt 5: Erstellen Ihrer ersten Pipeline in CodePipeline

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

Um eine CodePipeline Pipeline zu erstellen

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

   Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie Create pipeline (Pipeline erstellen) aus.
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyFirstPipeline`** ein.

4. Wählen Sie unter Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie unter Step 2: Add source stage (Schritt 2: Quell-Phase hinzufügen) unter Source provider (Quell-Anbieter) die Option CodeCommit. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, in Schritt 1: Erstelle ein CodeCommit Repository dem Sie es erstellt haben. Wählen Sie unter Branch name (Name der Verzweigung) die Option `main` und dann Next step (Nächster Schritt) aus.

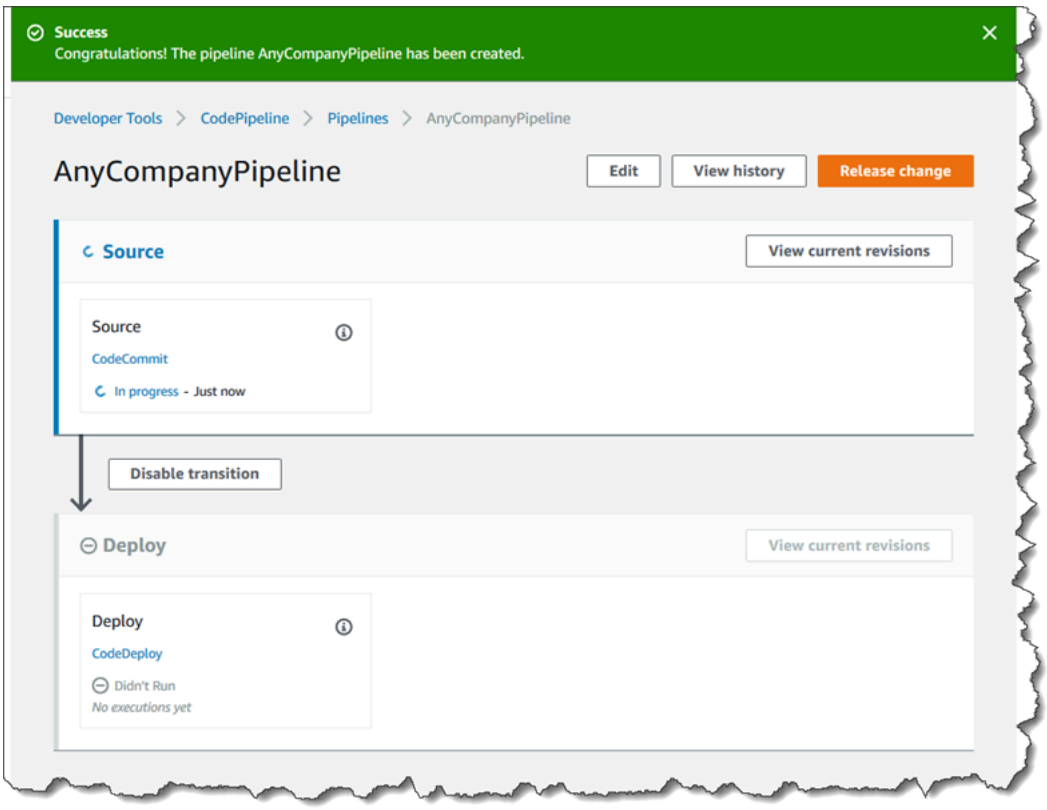
   Nachdem Sie den Repository-Namen und den Branch ausgewählt haben, wird in einer Meldung die Amazon CloudWatch Events-Regel angezeigt, die für diese Pipeline erstellt werden soll.

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

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

   > **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.
9. Wählen Sie in Step 4: Add source stage (Schritt 4: Quell-Stufe hinzufügen) unter Deploy provider (Anbieter bereitstellen) die Option CodeDeploy aus. Wählen Sie unter Application name (Anwendungsname) die Option **`MyDemoApplication`** aus. Wählen Sie unter Deployment group (Bereitstellungsgruppe) die Option **`MyDemoDeploymentGroup`** , und dann Next step (Nächster Schritt) aus.

10. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.
11. Die Pipeline wird ausgeführt, nachdem sie erstellt wurde. Es lädt den Code aus Ihrem CodeCommit Repository herunter und erstellt eine CodeDeploy Bereitstellung für Ihre EC2-Instance. Sie können sich Fortschritts-, Erfolgs- und Fehlschlagsmeldungen anzeigen lassen, während das CodePipeline Beispiel die Webseite für die Amazon EC2 EC2-Instance in der CodeDeploy Bereitstellung bereitstellt.



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

Im nächsten Schritt überprüfen Sie die Ergebnisse.

So verifizieren Sie, ob Ihre Pipeline erfolgreich ausgeführt wurde

1. Zeigen Sie den anfänglichen Fortschritt der Pipeline an. Der Status jeder Phase ändert sich von Zurzeit keine Ausführungen zu Laufend. Danach ändert er sich entweder zu Erfolgreich oder Fehlgeschlagen. Die Pipeline sollte die erste Ausführung innerhalb weniger Minuten abschließen.
2. Nachdem Successed für den Pipeline-Status angezeigt wird, wählen Sie im Statusbereich für die Bereitstellungsphase die Option CodeDeploy. Dadurch wird die CodeDeploy Konsole geöffnet.

Wenn Succeeded (Erfolgreich) nicht angezeigt wird, finden Sie weitere Informationen unter Fehlerbehebung 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 Repository, indem Sie nacheinander die folgenden Befehle ausführen:

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

```
git push
```

So verifizieren Sie, ob Ihre Pipeline erfolgreich ausgeführt wurde

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

   Die aktualisierte Webseite wird angezeigt.

Schritt 7: Bereinigen von Ressourcen

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

So bereinigen Sie die in diesem Tutorial verwendeten Ressourcen

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

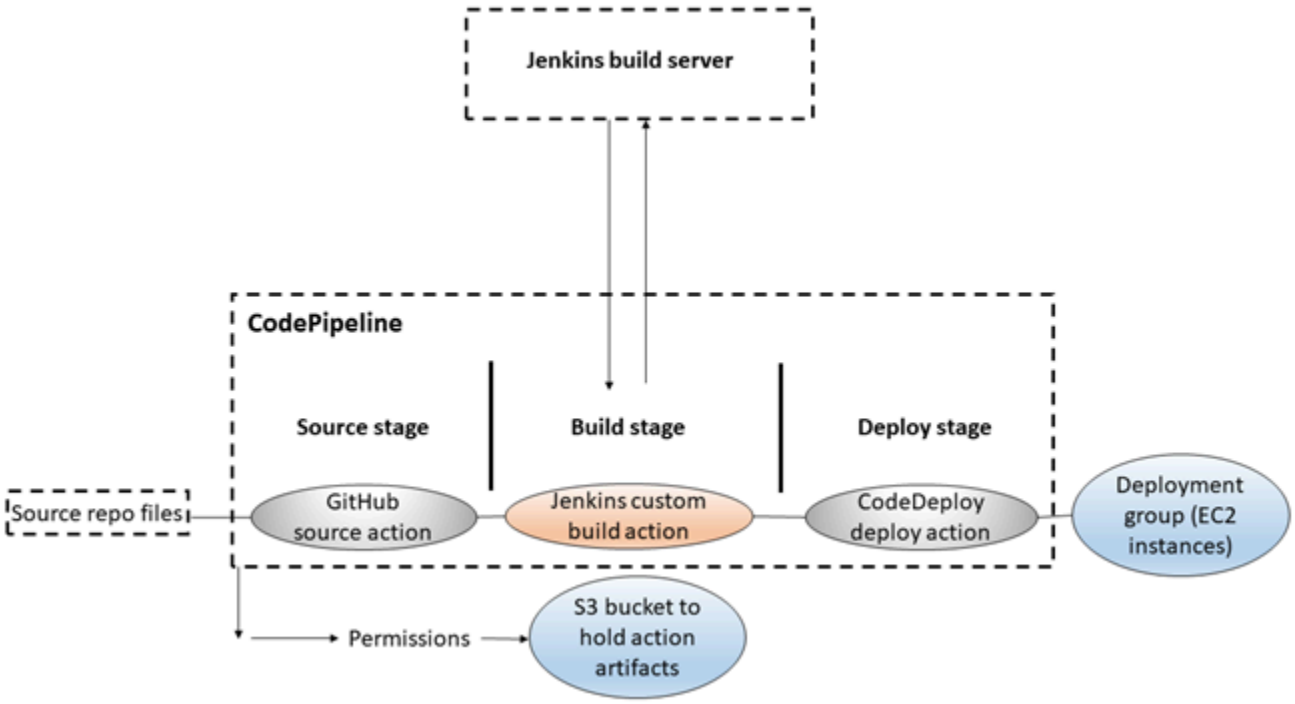
Schritt 8: Weitere Informationen

Erfahre mehr darüber, wie das CodePipeline funktioniert:

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

Tutorial: Erstellen einer vierstufigen Pipeline

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



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

Vor dem Erstellen dieser Pipeline müssen Sie die erforderlichen Ressourcen konfigurieren. Wenn Sie beispielsweise ein GitHub Repository für Ihren Quellcode verwenden möchten, müssen Sie das Repository erstellen, bevor Sie es einer Pipeline hinzufügen können. Im Zuge der Konfiguration führt Sie dieses Tutorial durch die Einrichtung von Jenkins bei einer EC2-Instance für Demonstrationszwecke.

> **Important**
>
> Viele der Aktionen, die Sie Ihrer Pipeline in diesem Verfahren hinzufügen, beinhalten AWS Ressourcen, die Sie erstellen müssen, bevor Sie die Pipeline erstellen. AWSRessourcen 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. AWSRessourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Hinzufügen einer regionsübergreifenden Aktion in CodePipeline.

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

Themen

- Schritt 1: Erfüllen der Voraussetzungen
- Schritt 2: Erstellen einer 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 einer beliebigen Jenkins-Instanz installieren, mit der Sie es verwenden möchten. CodePipeline Sie sollten auch einen dedizierten IAM-Benutzer oder eine spezielle IAM-Rolle konfigurieren, die

für Berechtigungen zwischen Ihrem Jenkins-Projekt und verwendet werden soll. CodePipeline Der einfachste Weg, Jenkins zu integrieren, CodePipeline besteht darin, Jenkins auf einer EC2-Instance zu installieren, die eine IAM-Instanzrolle verwendet, die Sie für die Jenkins-Integration erstellen. Damit Links in der Pipeline für Jenkins-Aktionen erfolgreich eine Verbindung erstellen, müssen Sie Proxy- und Firewall-Einstellungen auf dem Server oder der EC2-Instance entsprechend konfigurieren, um eingehende Verbindungen beim Port zu erlauben, der von Ihrem Jenkins-Projekt verwendet wird. Sie müssen Jenkins für die Authentifizierung von Benutzern und die Durchsetzung der Zugriffskontrolle konfiguriert haben, um Verbindungen auf diesen Ports (z. B. 443 und 8443, wenn Sie Jenkins entsprechend gesichert haben, ausschließlich HTTPS-Verbindungen zu verwenden, oder 80 und 8080, wenn Sie HTTP-Verbindungen zulassen) zu erlauben. Weitere Informationen finden Sie unter Sichern von Jenkins.

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

Themen

- Kopieren oder klonen Sie das Beispiel in ein Repository GitHub
- Erstellen Sie eine IAM-Rolle, die für die Jenkins-Integration verwendet werden soll
- 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 pushen

1. Laden Sie den Beispielcode aus dem GitHub Repository herunter oder klonen Sie die Repositorys auf Ihren lokalen Computer. Es gibt zwei Beispielpakete:
   - Wenn Sie Ihr Beispiel auf Amazon Linux-, RHEL- oder Ubuntu-Server-Instances bereitstellen möchten, wählen Sie codepipeline-jenkins-aws-codedeploy_linux.zip.
   - Wenn Sie Ihr Beispiel auf Windows Server-Instances bereitstellen möchten, wählen Sie CodePipeline-Jenkins - .zip. AWSCodeDeploy_Windows
2. Klicken Sie im Repository auf Fork, um das Beispiel-Repository in Ihrem Github-Konto zu klonen. Weitere Informationen finden Sie in der GitHub-Dokumentation.

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

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

1. Melden Sie sich bei der AWS Management Console an 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 zur Erteilung von Berechtigungen für Anwendungen, die auf Amazon EC2 EC2-Instances ausgeführt werden, und Erstellen einer Rolle zum Delegieren von Berechtigungen an eine. AWS-Service

Installieren und konfigurieren Sie Jenkins und das Plugin für Jenkins CodePipeline

Um Jenkins und das Plugin für Jenkins zu installieren CodePipeline

1. Erstellen Sie eine EC2-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 EC2-Instance im Amazon EC2 EC2-Benutzerhandbuch.

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

2. Installieren Sie Jenkins in der EC2-Instance. Weitere Informationen finden Sie in der Jenkins-Dokumentation unter Installing Jenkins und Starting and accessing Jenkins sowie unter details of integration with Jenkins in Produkt- und Serviceintegrationen mit CodePipeline.
3. Starten Sie Jenkins. Wählen Sie auf der Startseite Manage Jenkins aus.
4. Wählen Sie auf der Seite Manage Jenkins Manage Plugins aus.
5. Wählen Sie die Registerkarte Available (Verfügbar) aus. Geben Sie im Suchfeld Filter „**AWS CodePipeline**“ ein. Wählen Sie CodePipeline Plugin für Jenkins aus der Liste aus und klicken Sie auf Jetzt herunterladen und nach dem Neustart installieren.

6. Klicken Sie auf der Seite Installing Plugins/Upgrades auf Restart Jenkins when installation is complete and no jobs are running.
7. Wählen Sie Back to Dashboard aus.
8. Klicken Sie auf der Hauptseite auf New Item.
9. Geben Sie im Feld Elementname einen Namen für das Jenkins-Projekt ein (z. B.). *MyDemoProject* Wählen Sie Freestyle project und OK aus.

   > **Note**
   >
   > Stellen Sie sicher, dass der Name 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 die AWS CLI mit dem Profil für den IAM-Benutzer konfiguriert haben, den Sie für die Integration zwischen CodePipeline und Jenkins erstellt haben, lassen Sie alle anderen Felder leer.
11. Wählen Sie Erweitert und geben Sie unter Anbieter einen Namen für den Anbieter der Aktion ein, wie er angezeigt werden soll (z. B.). CodePipeline *MyJenkinsProviderName* Stellen Sie sicher, dass der Name eindeutig und leicht zu merken ist. Sie werden ihn verwenden, wenn Sie zu einem späteren Zeitpunkt in diesem Tutorial eine Build- und -Test-Aktion zu Ihrer Pipeline hinzufügen.

    > **Note**
    >
    > Dieser Aktionsname muss 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 anschließend „AWS CodePipeline Publisher“. Wählen Sie Add aus. Lassen Sie den Speicherort in Build Output Locations leer. Dies ist die Standardkonfiguration. Sie erstellt am Ende des Build-Prozesses eine komprimierte Datei.
15. Wählen Sie Save, um Ihr Jenkins-Projekt zu speichern.

Schritt 2: Erstellen einer Pipeline in CodePipeline

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

Um einen CodePipeline automatisierten Release-Prozess zu erstellen

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

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

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

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

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

Schritt 3: Hinzufügen einer weiteren Phase zur Pipeline

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

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

Hinzufügen einer Testphase zur Pipeline

Themen

- Abfragen der IP-Adresse einer Instance
- 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 das AWS CLI mit dem Profil für den IAM-Benutzer konfiguriert haben, den Sie für die Integration zwischen CodePipeline und Jenkins erstellt haben, lassen Sie alle anderen Felder leer.

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

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

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

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

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

```
* * * * *
```

Dieser wird CodePipeline jede Minute abgefragt.

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

   ```
   TEST_IP_ADDRESS=192.168.0.4 rake test
   ```

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

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

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

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

Erstellen einer vierten Phase

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

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie 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 *`MyJenkinsTest-Action`* ein. Wählen Sie unter Testanbieter den Anbieternamen aus, den Sie in Jenkins angegeben haben (z. B.). *`MyJenkinsProviderName`* Geben Sie im Feld Projektname den Namen des Projekts ein, das Sie in Jenkins erstellt haben (z. B.). *`MyTestProject`* Wählen Sie unter Eingabeartefakte das Artefakt aus dem Jenkins-Build aus, dessen Standardname lautet ***`BuildArtifact`***, und wählen Sie dann Fertig aus.

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

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

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

Schritt 4: Bereinigen von Ressourcen

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

nicht weiter verwenden möchten CodePipeline, löschen Sie die Pipeline, dann die CodeDeploy Anwendung und die zugehörigen Amazon EC2 EC2-Instances und schließlich den Amazon S3-Bucket, der zum Speichern von Artefakten verwendet wurde. Sie sollten auch überlegen, ob Sie andere Ressourcen, wie das GitHub Repository, löschen sollten, wenn Sie nicht beabsichtigen, sie weiter zu verwenden.

So bereinigen Sie die in diesem Tutorial verwendeten Ressourcen

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

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

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

Tutorial: Einrichten einer CloudWatch Events-Regel, um E-Mail-Benachrichtigungen zu Änderungen am Pipelinestatus zu empfangen

Nachdem Sie eine Pipeline eingerichtet habenAWS CodePipelineerstellen, können Sie eine CloudWatch Events-Regel zum Senden von Benachrichtigungen bei Änderungen am Ausführungsstatus Ihrer Pipelines oder an den Stufen oder Aktionen in Ihren Pipelines

erstellen. Weitere Informationen zur Verwendung von CloudWatch Events zum Einrichten von Benachrichtigungen zu Änderungen am Pipelinestatus finden Sie unterÜberwachen von CodePipeline Ereignissenaus.

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 eine Eingabetransformationsmethode beim Erstellen der CloudWatch Events-Regel verwendet. Diese wandelt die Details des Nachrichtenschemas um, um die Nachricht in Form von lesbarem Text bereitzustellen.

> **Note**
>
> Stellen Sie beim Erstellen der Ressourcen für dieses Lernprogramm, z. B. die Amazon SNS SNS-Benachrichtigung und die CloudWatch Events-Regel, sicher, dass die Ressourcen in derselben erstellt werdenAWSRegion als Ihre Pipeline.

Themen

- Schritt 1: Einrichten einer E-Mail-Benachrichtigung mit Amazon SNS
- Schritt 2: Erstellen Sie eine Regel und fügen Sie das SNS-Thema als Ziel hinzu
- Schritt 3: Bereinigen von Ressourcen

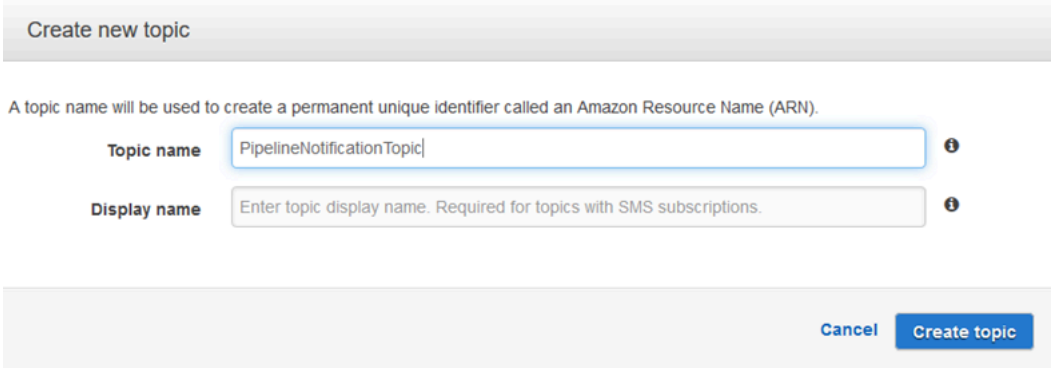
Schritt 1: Einrichten einer E-Mail-Benachrichtigung mit Amazon SNS

Amazon SNS koordiniert die Verwendung von Themen, um Nachrichten an Abonnenten (Endpunkte oder Clients) zu senden. Erstellen Sie mithilfe von Amazon SNS ein Benachrichtigungsthema und abonnieren Sie dieses Thema anschließend über Ihre E-Mail-Adresse. Das Amazon SNS SNS-Thema wird als Ziel zu Ihrer CloudWatch Events-Regel hinzugefügt. Weitere Informationen finden Sie im Amazon Simple Notification Service-Entwicklerhandbuch.

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

1. Öffnen Sie die Amazon SNS SNS-Konsole unterhttps://console.aws.amazon.com/snsaus.
2. Wählen Sie Create topic (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 Create topic (Thema erstellen) aus.

   Weitere Informationen finden Sie unterErstellen eines ThemasimEntwicklerhandbuch von Amazon SNSaus.

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

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

   Weitere Informationen finden Sie unterAbonnieren eines ThemasimEntwicklerhandbuch von Amazon SNSaus.

Schritt 2: Erstellen Sie eine Regel und fügen Sie das SNS-Thema als Ziel hinzu

Erstellen Sie eine CloudWatch Events-Benachrichtigungsregel mit CodePipeline als 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 Create rule (Regel erstellen). 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 Configure details.
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 Create rule (Regel erstellen).
13. Vergewissern Sie sich, dass CodePipeline jetzt Build-Benachrichtigungen sendet. 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 wählen Sie dannAktionen,Bearbeitenaus. Bearbeiten Sie die Regel, wählen Sie Configure details und anschließend Update rule aus.

    Um eine Regel zum Senden von Build-Benachrichtigungen nicht länger zu verwenden, wählen Sie in der CloudWatch -Konsole die Regel und anschließendAktionen,Deaktivieren vonaus.

    Um eine Regel zu löschen, wählen Sie in der CloudWatch -Konsole die Regel aus und wählen Sie dannAktionen,Löschenaus.

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 Löschen der Amazon CloudWatch Events Events-Regel finden Sie unterBereinigen (Abmelden von einem Amazon SNS SNS-Thema)und Referenz`DeleteRule`imAPI-Referenz für Amazon CloudWatch Eventsaus.

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

Sie können verwenden AWS CodePipeline , um einen kontinuierlichen Integrationsablauf zu 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 den Eingang eines neuen GitHub Commit und verwendet dann CodeBuild, um die App zu erstellen, und Device Farm, um sie zu testen.

> **Important**
>
> Viele der Aktionen, die Sie Ihrer Pipeline in diesem Verfahren hinzufügen, umfassen AWS Ressourcen, die Sie erstellen müssen, bevor Sie die pipeline. AWS resources für Ihre Quellaktionen erstellen, müssen immer in derselben AWS Region erstellt werden, in der Sie Ihre Pipeline erstellen. Wenn Sie beispielsweise Ihre Pipeline 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 Hinzufügen einer regionsübergreifenden Aktion in CodePipeline.

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

> **Note**

Bevor Sie beginnen

1. Melden Sie sich bei der - AWS Device Farm Konsole an und wählen Sie Neues Projekt erstellen aus.
2. Wählen Sie Ihr Projekt. Kopieren Sie im Browser die URL Ihres neuen Projekts. Die URL enthält die Projekt-ID.

3. Kopieren Sie diese Produkt-ID und bewahren Sie sie auf. Sie 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
   ```

Konfigurieren Sie CodePipeline für die Verwendung Ihrer Device Farm-Tests

1. Fügen Sie eine Datei mit dem Namen `buildspec.yml` im Stammverzeichnis Ihres App-Codes hinzu und übertragen Sie sie in Ihr Repository. CodeBuild verwendet diese Datei, um Befehle und Zugriffsartefakte auszuführen, die zum Erstellen 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 Durchführung Ihrer Testsuite verwendet werden.

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

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

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

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

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

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

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

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

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

g. 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 unter GitHub Verbindungen.

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

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

j. Behalten Sie die verbleibenden Standardwerte für die Quellaktion bei. Wählen Sie Weiter aus.

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

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

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

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

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

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

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

f. Wählen Sie für Servicerolle Ihre vorhandene 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 aus CodePipeline. Dies kehrt zur CodePipelineKonsole zurück und erstellt ein CodeBuild Projekt, das die `buildspec.yml` in Ihrem Repository für die Konfiguration verwendet. Das Build-Projekt verwendet eine Servicerolle zum Verwalten von AWS-Service Berechtigungen. Dieser Vorgang kann einige Minuten dauern.

i. Wählen Sie Weiter aus.

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

6. Wählen Sie unter Step 5: Review (Schritt 5: Überprüfen) die Option Create pipeline (Pipeline erstellen) aus. Sie sollten ein Diagramm sehen, das Ihre Pipeline-Quell- und -Build-Stufen anzeigt.
7. Fügen Sie Ihrer Pipeline eine Device Farm-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 Stufe, indem Sie den Mauszeiger über das Informationssymbol im Pipeline-Diagramm bewegen. Wenn Ihre Pipeline Ihre App direkt aus der Quellphase testet, wählen Sie SourceArtifact. Wenn die Pipeline eine Build-Phase enthält, wählen Sie BuildArtifact.
   g. Geben Sie ProjectIdin Ihre Device Farm-Projekt-ID ein. Führen Sie die Schritte zu Beginn dieses Tutorials aus, um Ihre Projekt-ID abzurufen.
   h. Geben DevicePoolArnSie in den ARN für den Gerätepool ein. 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
      ```
   i. Geben Sie AppTypeunter 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 TestTypeunter Ihren Testtyp ein und geben Sie dann unter 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_NODE
   - APPIUM_RUBY
   - APPIUM_PYTHRUNG
   - APPIUM_WEB_JAVA_JUNIT
   - APPIUM_WEB_JAVA_TESTNG
   - APPIUM_WEB_NODE
   - APPIUM_WEB_RUBY
   - APPIUM_WEB_PYTHRUNG
   - BUILTIN_FUZ
   - INSTRUMENTATION
   - XCTEST
   - XCTEST_UI

   > **Note**
   >
   > Benutzerdefinierte Umgebungsknoten 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 mit AWS CodePipeline leicht einen kontinuierlichen Integrationsfluss konfigurieren, in dem Ihre App immer dann getestet wird, wenn sich der Quell-Bucket ändert. In diesem Tutorial wird gezeigt, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre erstellte iOS-App aus einem S3-Bucket zu testen. Die Pipeline erkennt das Eintreffen einer gespeicherten Änderung über Amazon CloudWatch Events und verwendet dann Device Farm, um die erstellte Anwendung zu testen.

> **Important**
>
> Viele der Aktionen, die Sie in diesem Verfahren zu Ihrer Pipeline hinzufügen, beinhalten AWS Ressourcen, die Sie erstellen müssen, bevor Sie die Pipeline erstellen. AWSRessourcen 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. AWSRessourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Hinzufügen einer regionsübergreifenden Aktion in CodePipeline.

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

> **Note**

Bevor Sie beginnen

1. Melden Sie sich bei der AWS Device Farm-Konsole an und klicken Sie auf Create a new project (Ein neues Projekt erstellen).

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 eines S3-Buckets für Ihre Anwendung folgen, um einen S3-Bucket zu erstellen.
2. Wählen Sie in der Amazon S3 S3-Konsole für Ihren Bucket Upload und folgen Sie den Anweisungen, um Ihre ZIP-Datei hochzuladen.

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

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

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

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

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

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

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

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

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

h. Wählen Sie Weiter.

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

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

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

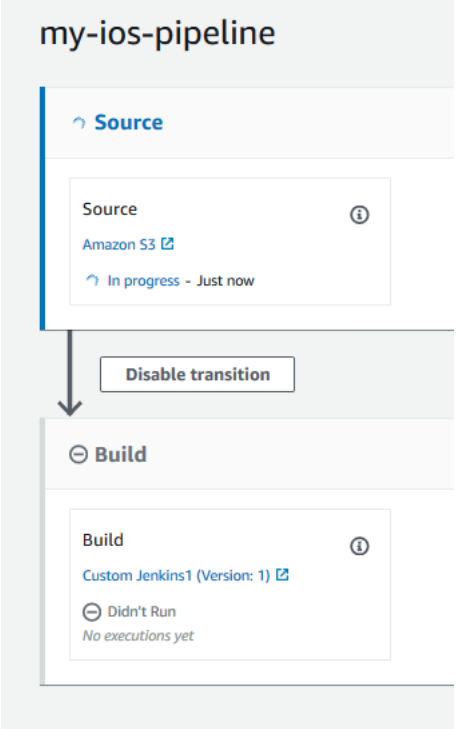
c. Geben Sie unter Server URL Text ein. Der Text ist ein Platzhalter. Sie wird nicht verwendet.

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

e. Wählen Sie Weiter.

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

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



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

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

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

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

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

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

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

g. Wählen Sie unter Aktionsanbieter die Option AWSDevice 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 Ausgangsartefakts für jede Phase, indem Sie mit der Maus auf das Informationssymbol im Pipeline-Diagramm zeigen. Wenn Ihre Pipeline Ihre App direkt von der Quellphase aus testet, wählen Sie SourceArtifact. Wenn die Pipeline eine Build-Phase enthält, wählen Sie BuildArtifact.

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

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

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

k. Geben AppTypeSie iOS ein.

   Im Folgenden finden Sie eine Liste gültiger Werte für AppType:

   - iOS
   - Android
   - Web

l. Geben Sie unter App den Pfad des kompilierten Anwendungspakets ein. Der Pfad ist relativ zum Stamm des Eingabeartefakts für die Testphase. In der Regel sieht dieser Pfad in etwa wie `ios-test.ipa` aus.

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

   Wenn Sie einen der integrierten Device Farm Farm-Tests verwenden, geben Sie den in Ihrem Device Farm Farm-Projekt konfigurierten Testtyp ein, z. B. BUILTIN_FUZZ. Geben Sie unter eine Zeit in FuzzEventCountMillisekunden ein, z. B. 6000. Geben Sie FuzzEventThrottleunter eine Zeit in Millisekunden ein, z. B. 50.

   Wenn Sie keinen der integrierten Device Farm Farm-Tests verwenden, geben Sie Ihren Testtyp und dann im Feld Test den Pfad der Testdefinitionsdatei ein. Der Pfad ist relativ zum Stamm des Eingabeartefakts für Ihren Test.

   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 aus.

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 in Service Catalog bereitgestellt wird

Service Catalog ermöglicht es Ihnen, Produkte auf der Grundlage von AWS CloudFormation Vorlagen zu erstellen und bereitzustellen. In diesem Tutorial erfahren Sie, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre Produktvorlage in Service Catalog bereitzustellen und die Änderungen zu übertragen, die Sie in Ihrem Quell-Repository vorgenommen haben (bereits in GitHub, CodeCommit, oder Amazon S3 erstellt).

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, müssen Sie alle Quelldateien, die als eine einzige ZIP-Datei verpackt sind, in Ihren Bucket hochladen. Andernfalls schlägt die Quellaktion fehl.

Zuerst erstellen Sie ein Produkt im Service Catalog und dann eine Pipeline inAWS CodePipeline. Dieses Tutorial bietet zwei Optionen für die Einrichtung der Bereitstellungskonfiguration:

- Erstellen Sie ein Produkt im Service Catalog und laden Sie eine Vorlagendatei in Ihr Quell-Repository hoch. Geben Sie die Produktversion und die Bereitstellungskonfiguration in der CodePipeline Konsole an (ohne separate Konfigurationsdatei). Siehe Option 1: Bereitstellung in Service Catalog ohne Konfigurationsdatei.

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

- Erstellen Sie ein Produkt im Service Catalog und laden Sie eine Vorlagendatei in Ihr Quell-Repository hoch. Stellen Sie die Produktversion und Bereitstellungskonfiguration in einer separaten Konfigurationsdatei bereit. Siehe Option 2: Bereitstellung in Service Catalog mithilfe einer Konfigurationsdatei.

Option 1: Bereitstellung in Service Catalog ohne Konfigurationsdatei

In diesem Beispiel laden Sie die AWS CloudFormation Beispielvorlagendatei für einen S3-Bucket hoch und erstellen dann Ihr Produkt im Service Catalog. Als Nächstes erstellen Sie Ihre Pipeline und geben die Bereitstellungskonfiguration in der CodePipeline Konsole an.

Schritt 1: Hochladen der Beispielvorlagendatei in das Quellrepository

1. Öffnen Sie einen Texteditor. Erstellen Sie eine Beispielvorlage, indem Sie den folgenden Code in die Datei einfügen. Speichern Sie die Datei als `S3_template.json`.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "CloudFormation Sample Template S3_Bucket: Sample template showing
 how to create a privately accessible S3 bucket. **WARNING** This template creates
 an S3 bucket. You will be billed for the resources used if you create a stack from
 this template.",
  "Resources": {
    "S3Bucket": {
      "Type": "AWS::S3::Bucket",
      "Properties": {}
    }
  },
  "Outputs": {
    "BucketName": {
      "Value": {
        "Ref": "S3Bucket"
      },
      "Description": "Name of Amazon S3 bucket to hold website content"
    }
  }
```

```
}
```

Diese Vorlage ermöglicht AWS CloudFormation die Erstellung eines S3-Buckets, der von Service Catalog verwendet werden kann.

2. Laden Sie die Datei `S3_template.json` in Ihr AWS CodeCommit-Repository hoch.

Schritt 2: Erstellen Sie ein Produkt im Service Catalog

1. Melden Sie sich als IT-Administrator bei der Service Catalog-Konsole an, wechseln Sie zur Produktseite und wählen Sie dann Neues Produkt hochladen aus.
2. Führen Sie auf der Seite Upload new product (Neues Produkt hochladen) die folgenden Schritte aus:
   a. Geben Sie unter Product name (Produktname) den gewünschten Namen für Ihr neues Produkt ein.
   b. Geben Sie unter Description (Beschreibung) die Beschreibung des Produktkatalogs an. Diese Beschreibung wird in der Produktliste angezeigt, um den Benutzern die Auswahl des richtigen Produkts zu erleichtern.
   c. Geben Sie unter Provided by (Bereitgestellt von) den Namen Ihrer IT-Abteilung oder des Administrators ein.
   d. Wählen Sie Weiter.
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.
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 Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   b. Wählen Sie Getting started (Erste Schritte). Wählen Sie Create pipeline (Pipeline erstellen) und geben Sie dann einen Namen für Ihre Pipeline ein.

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

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

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

2. Führen Sie die folgenden Schritte aus, um eine Quellphase hinzuzufügen:

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

   b. Geben Sie unter Repository name (Repository-Name) und Branch name (Name der Verzweigung) das Repository und die Verzweigung ein, die für Ihre Quellaktion verwendet werden sollen.

   c. Wählen Sie Weiter.

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

4. Führen Sie unter Add deploy stage (Bereitstellungsphase hinzufügen) die folgenden Schritte aus:

   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option AWS Service Catalog aus.

   b. Wählen Sie für die Bereitstellungskonfiguration Enter deployment configuration (Bereitstellungskonfiguration eingeben).

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

   d. Geben Sie unter Template file path (Pfad zur Vorlagendatei) den relativen Pfad ein, unter dem die Vorlagendatei gespeichert ist.

   e. Wählen Sie unter Produkttyp die AWS CloudFormationVorlage aus.

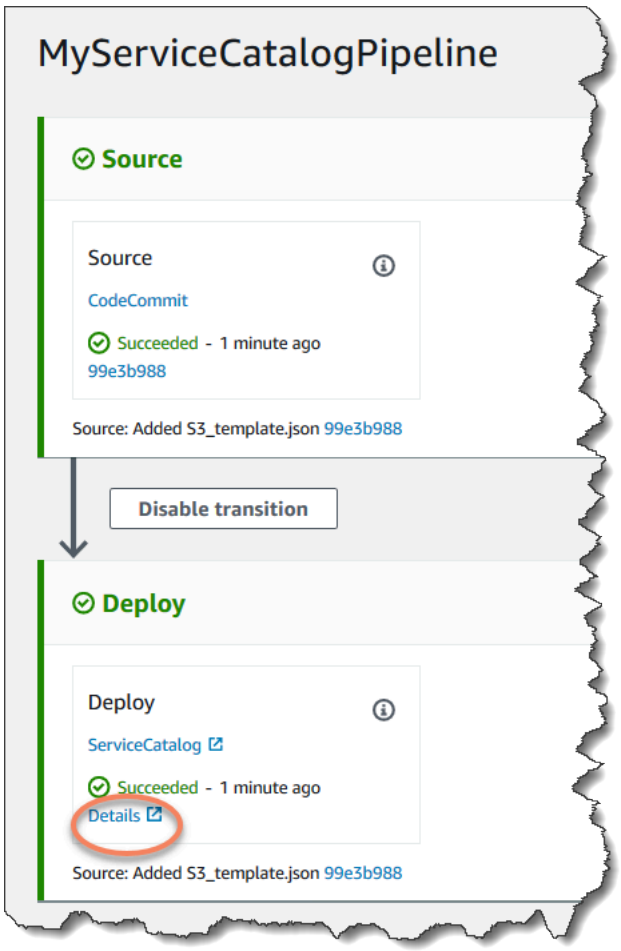
   f. Geben Sie im Feld Produktversionsname den Namen der Produktversion ein, die Sie im Service Catalog angegeben haben. Wenn die Vorlagenänderung in einer neuen Produktversion bereitgestellt werden soll, geben Sie einen Namen für die Produktversion ein, der noch nicht für eine vorherige Produktversion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

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

   h. Wählen Sie Weiter.

5. Prüfen Sie Ihre Pipeline-Einstellungen unter Review (Überprüfen) und wählen Sie dann Create (Erstellen).

6. Wenn Ihre Pipeline in der Bereitstellungsphase erfolgreich ausgeführt wird, wählen Sie Details. Dadurch wird Ihr Produkt im Service Catalog geöffnet.



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

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

1. Sehen Sie sich Ihre Pipeline in der CodePipeline Konsole an und wählen Sie in Ihrer Quellphase Details aus. Ihr AWS CodeCommit-Quellrepository wird in der Konsole geöffnet. Wählen Sie Edit (Bearbeiten) und nehmen Sie eine Änderung an der Datei vor (z. B. an der Beschreibung).

```
"Description": "Name of Amazon S3 bucket to hold and version website content"
```

2. Führen Sie einen Commit für die Änderung durch und übertragen Sie diese. Ihre Pipeline startet, nach der Übertragung der Änderung. Wenn die Ausführung der Pipeline abgeschlossen ist, wählen Sie in der Bereitstellungsphase Details aus, um Ihr Produkt im Service Catalog zu öffnen.

3. Wählen Sie den neuen Namen Ihrer Version in den Produktinformationen, um die Produktvorlage zu öffnen. Zeigen Sie die Änderung an der bereitgestellten Vorlage an.

Option 2: Bereitstellung in Service Catalog mithilfe einer Konfigurationsdatei

In diesem Beispiel laden Sie die AWS CloudFormation Beispielvorlagendatei für einen S3-Bucket hoch und erstellen dann Ihr Produkt im Service Catalog. Sie können auch eine separate Konfigurationsdatei hochladen, in der Ihre Bereitstellungskonfiguration angegeben ist. Als Nächstes erstellen Sie Ihre Pipeline und geben den Speicherort Ihrer Konfigurationsdatei an.

Schritt 1: Hochladen der Beispielvorlagendatei in das Quellrepository

1. Öffnen Sie einen Texteditor. Erstellen Sie eine Beispielvorlage, indem Sie den folgenden Code in die Datei einfügen. Speichern Sie die Datei als `S3_template.json`.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "CloudFormation Sample Template S3_Bucket: Sample template showing
 how to create a privately accessible S3 bucket. **WARNING** This template creates
 an S3 bucket. You will be billed for the resources used if you create a stack from
 this template.",
  "Resources": {
    "S3Bucket": {
      "Type": "AWS::S3::Bucket",
      "Properties": {}
    }
  },
  "Outputs": {
    "BucketName": {
      "Value": {
        "Ref": "S3Bucket"
      },
      "Description": "Name of Amazon S3 bucket to hold website content"
    }
  }
}
```

   Diese Vorlage ermöglicht AWS CloudFormation die Erstellung eines S3-Buckets, der von Service Catalog verwendet werden kann.

2. Laden Sie die Datei `S3_template.json` in Ihr AWS CodeCommit-Repository hoch.

Schritt 2: Erstellen einer Konfigurationsdatei für die Produktbereitstellung

1. Öffnen Sie einen Texteditor. Erstellen Sie die Konfigurationsdatei für Ihr Produkt. Die Konfigurationsdatei wird verwendet, um Ihre Service Catalog-Bereitstellungsparameter/Einstellungen zu definieren. Sie verwenden diese Datei für die Erstellung Ihrer Pipeline.

   In diesem Beispiel wird für `ProductVersionName` "devops S3 v2" und für `ProductVersionDescription` die Option `MyProductVersionDescription` verwendet. Wenn die Vorlagenänderung in einer neuen Produktversion bereitgestellt werden soll, geben Sie einfach einen Namen für die Produktversion ein, der noch nicht für eine vorherige Produktversion im selben Produkt verwendet wurde.

   Speichern Sie die Datei als `sample_config.json`.

   ```
   {
       "SchemaVersion": "1.0",
       "ProductVersionName": "devops S3 v2",
       "ProductVersionDescription": "MyProductVersionDescription",
       "ProductType": "CLOUD_FORMATION_TEMPLATE",
       "Properties": {
           "TemplateFilePath": "/S3_template.json"
       }
   }
   ```

   Diese Datei erstellt bei jeder Ausführung der Pipeline die Informationen zur Produktversion.

2. Laden Sie die Datei `sample_config.json` in Ihr AWS CodeCommit-Repository hoch. Stellen Sie sicher, dass Sie diese Datei in Ihr Quellrepository hochladen.

Schritt 3: Erstellen Sie ein Produkt im Service Catalog

1. Melden Sie sich als IT-Administrator bei der Service Catalog-Konsole an, wechseln Sie zur Produktseite und wählen Sie dann Neues Produkt hochladen aus.
2. Führen Sie auf der Seite Upload new product (Neues Produkt hochladen) die folgenden Schritte aus:

   a. Geben Sie unter Product name (Produktname) den gewünschten Namen für Ihr neues Produkt ein.

b. Geben Sie unter Description (Beschreibung) die Beschreibung des Produktkatalogs an. Diese Beschreibung wird in der Produktliste angezeigt, um den Benutzern die Auswahl des richtigen Produkts zu erleichtern.

c. Geben Sie unter Provided by (Bereitgestellt von) den Namen Ihrer IT-Abteilung oder des Administrators ein.

d. Wählen Sie Weiter.

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.

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 Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   b. Wählen Sie Getting started (Erste Schritte). Wählen Sie Create pipeline (Pipeline erstellen) und geben Sie dann einen Namen für Ihre Pipeline ein.

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

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

2. Führen Sie die folgenden Schritte aus, um eine Quellphase hinzuzufügen:

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

   b. Geben Sie unter Repository name (Repository-Name) und Branch name (Name der Verzweigung) das Repository und die Verzweigung ein, die für Ihre Quellaktion verwendet werden sollen.

   c. Wählen Sie Weiter.

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

4. Führen Sie unter Add deploy stage (Bereitstellungsphase hinzufügen) die folgenden Schritte aus:

   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option AWS Service Catalog aus.

   b. Wählen Sie Use configuration file (Konfigurationsdatei verwenden).

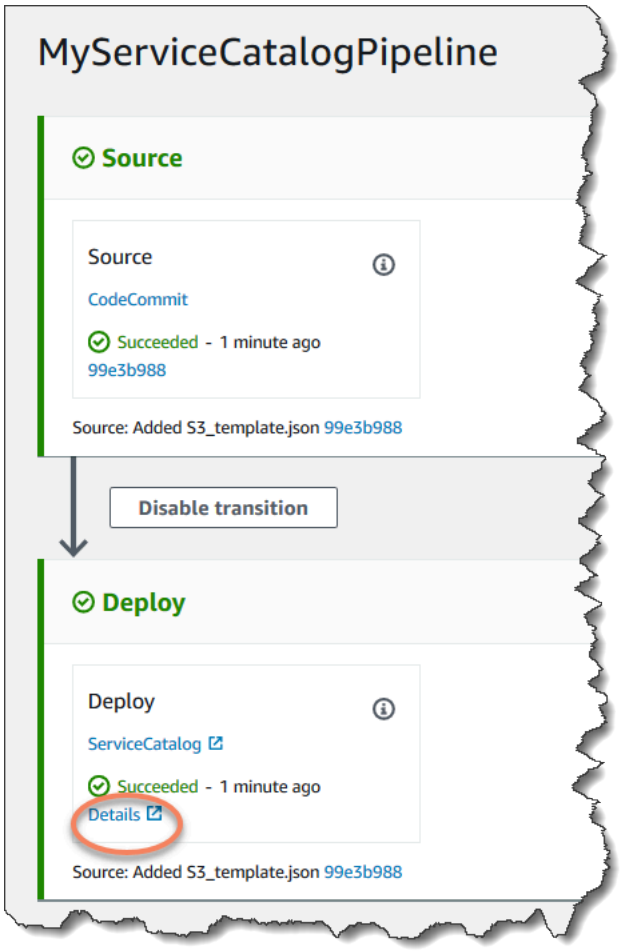
   c. Fügen Sie unter Produkt-ID die Produkt-ID ein, die Sie aus der Service Catalog-Konsole kopiert haben.

d. Geben Sie unter Configuration file path (Pfad zur Konfigurationsdatei) den Dateipfad der Konfigurationsdatei in Ihrem Repository ein.

e. Wählen Sie Weiter.

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-Quellrepository 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 einer Pipeline mit AWS CloudFormation

Die Beispiele bieten Mustervorlagen, mit denen Sie mit AWS CloudFormation eine Pipeline erstellen können, die Ihre Anwendung bei jeder Änderung des Quellcodes auf Ihren Instances bereitstellt. Die Beispielvorlage erstellt eine Pipeline, die in AWS CodePipeline angezeigt werden kann. Die Pipeline erkennt das Eintreffen einer gespeicherten Änderung über Amazon CloudWatch Events.

Themen

- Beispiel 1: Erstellen einer AWS CodeCommit-Pipeline mit AWS CloudFormation
- Beispiel 2: Erstellen einer Amazon-S3-Pipeline mitAWS CloudFormation

Beispiel 1: Erstellen einer AWS CodeCommit-Pipeline mit AWS CloudFormation

In dieser Anleitung wird beschrieben, wie Sie die -Konsole verwenden, um eine Infrastruktur zu erstellen, die eineAWS CloudFormation -Konsole enthält, die eine -Konsole enthält, 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 enthält, z. B. Ihre Amazon CloudWatch Events-Regel. Nachdem Sie Ihren Ressourcen-Stack in AWS CloudFormation erstellt haben, können Sie Ihre

Pipeline in der AWS CodePipeline-Konsole anzeigen. Die Pipeline ist eine zweistufige Pipeline mit einer CodeCommit Quellphase und einer CodeDeploy Bereitstellungsphase.

Voraussetzungen:

Sie müssen die folgenden Ressourcen zur Verwendung mit der AWS CloudFormation-Beispielvorlage erstellt haben:

- Sie müssen ein Quellrepository erstellt haben. Sie können das in Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)erstellte AWS CodeCommit-Repository verwenden.
- Sie müssen eine CodeDeploy Anwendung und eine Bereitstellungsgruppe erstellt haben. Sie können die in Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)erstellten CodeDeploy-Ressourcen verwenden.
- Wählen Sie einen dieser Links, um die AWS CloudFormation-Beispielvorlagendatei zum Erstellen einer Pipeline herunterzuladen: YAML | JSON

  Entpacken Sie die Datei und speichern Sie sie auf Ihrem lokalen Computer.
- Laden Sie die Beispielanwendungsdatei SampleApp_Linux.zip herunter.

Erstellen Ihrer Pipeline in AWS CloudFormation

1. Entpacken Sie die Dateien aus SampleApp_Linux.zip und laden Sie die Dateien in IhrAWS CodeCommit Repository hoch. Sie müssen die entpackten Dateien in das Stammverzeichnis Ihres Repositorys hochladen. Sie können den Anweisungen unter Schritt 2: Fügen Sie Ihrem CodeCommit Repository Beispielcode hinzu folgen, um die Dateien in Ihr Repository zu übertragen.
2. Öffnen Sie die AWS CloudFormation-Konsole und wählen Sie Create Stack (Stack erstellen) aus. 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 Next (Weiter).
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 in ApplicationNameden Namen Ihrer CodeDeploy Anwendung ein.

b. Geben Sie unter den Namen Ihrer CodeDeploy Bereitstellungsgruppe ein. BetaFleet

c. Geben Sie in BranchNameden Repository-Zweig ein, den Sie verwenden möchten.

d. Geben Sie in RepositoryNameden Namen Ihres CodeCommit Quell-Repositorys ein.

5. Wählen Sie Next (Weiter). Übernehmen Sie die Standardeinstellungen auf der folgenden Seite und wählen Sie Next (Weiter) aus.

6. Wählen Sie unter Funktionen die Option Ich bestätige, dassAWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden, und wählen Sie dann Stapel erstellen aus.

7. Nachdem die Erstellung Ihres Stacks abgeschlossen wurde, zeigen Sie die Ereignisliste an, um zu überprüfen, ob Fehler aufgetreten sind.

   Fehlersuche

   Der IAM-Benutzer, der die Pipeline in AWS CloudFormation erstellt, benötigt möglicherweise zusätzliche Berechtigungen, um Ressourcen für die Pipeline erstellen zu können. Die folgenden Berechtigungen sind in der Richtlinie erforderlich, um die erforderlichen Amazon CloudWatch Events-Ressourcen für die CodeCommit Pipeline erstellenAWS 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 anAWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.

> **Note**
>
> Um die erstellte Pipeline anzuzeigen, suchen Sie die Spalte Logische ID auf der Registerkarte Ressourcen für Ihren Stack inAWS CloudFormation. Notieren Sie sich den Namen in der Spalte Physikalische ID für die Pipeline. In CodePipeline können Sie die Pipeline mit derselben physikalischen 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 einer Amazon-S3-Pipeline mitAWS CloudFormation

Diese exemplarische Vorgehensweise zeigt Ihnen, wie Sie dieAWS CloudFormation Konsole verwenden, um eine Infrastruktur zu erstellen, die eine Pipeline enthält, die mit einem Amazon S3 S3-Quell-Bucket verbunden ist. In diesem Tutorial verwenden Sie die bereitgestellte Beispielvorlagendatei, um Ihren Ressourcen-Stack zu erstellen, der Ihren Quell-Bucket, Ihren Artifact-Store, Ihre Pipeline und Ihre Ressourcen zur Änderungserkennung enthält, wie z. B. Ihre Amazon CloudWatch Events-Regel und Ihren Amazon CloudTrail Events-Trail. Nachdem Sie Ihren Ressourcen-Stack in AWS CloudFormation erstellt haben, können Sie Ihre Pipeline in der AWS CodePipeline-Konsole anzeigen. Die Pipeline ist eine zweistufige Pipeline mit einer Amazon S3 S3-Quellphase und einer CodeDeploy Bereitstellungsphase.

Voraussetzungen:

Zur Verwendung mit der AWS CloudFormation-Beispielvorlage müssen Sie über die folgenden Ressourcen verfügen:

- Sie müssen die Amazon EC2 EC2-Instances erstellt haben, in denen Sie den CodeDeploy Agenten auf den Instances installiert haben. Sie müssen eine CodeDeploy Anwendung und eine Bereitstellungsgruppe erstellt haben. Verwenden Sie Amazon EC2 und die CodeDeploy Ressourcen, die Sie in erstellt habenTutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository).
- Wählen Sie die folgenden Links, um dieAWS CloudFormation Beispielvorlagendateien für die Erstellung einer Pipeline mit einer Amazon S3 S3-Quelle herunterzuladen:
  - Laden Sie die Beispielvorlage für Ihre Pipeline herunter: YAML | JSON

- Laden Sie die Beispielvorlage für Ihren CloudTrail Bucket und Trail herunter: YAML | JSON
- Entpacken Sie die Dateien und speichern Sie sie auf Ihrem lokalen Computer.

- Laden Sie die Beispielanwendung von SampleApp_Linux.zip herunter.

  Speichern Sie die ZIP-Datei auf Ihrem lokalen Computer. Laden Sie die ZIP-Datei hoch, nachdem der Stack erstellt wurde.

Erstellen Ihrer Pipeline in AWS CloudFormation

1. Öffnen Sie die AWS CloudFormation-Konsole und wählen Sie Create Stack (Stack erstellen) aus. Wählen Sie Mit neuen Ressourcen (Standard).
2. Wählen Sie unter Vorlage auswählen die Option Vorlage hochladen aus. Wählen Sie Datei auswählen und wählen Sie dann die Vorlagendatei von Ihrem lokalen Computer aus. Wählen Sie Next (Weiter).
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 in ApplicationNameden Namen Ihrer CodeDeploy Anwendung ein. Sie können den `DemoApplication`-Standardnamen ersetzen.

   b. Geben Sie unter den Namen Ihrer CodeDeploy Bereitstellungsgruppe ein. BetaFleet Sie können den `DemoFleet`-Standardnamen ersetzen.

   c. Herein SourceObjectKey, treten 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 Next (Weiter). Übernehmen Sie die Standardeinstellungen auf der folgenden Seite und wählen Sie Next (Weiter) aus.
5. Wählen Sie unter Funktionen die Option Ich bestätige, dassAWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden, und wählen Sie dann Stapel erstellen aus.
6. Nachdem die Erstellung Ihres Stacks abgeschlossen wurde, zeigen Sie die Ereignisliste an, um zu überprüfen, ob Fehler aufgetreten sind.

   Fehlersuche

   Der IAM-Benutzer, der die Pipeline erstellt, benötigtAWS CloudFormation möglicherweise zusätzliche Berechtigungen, um Ressourcen für die Pipeline zu erstellen. Die folgenden

Berechtigungen sind in der Richtlinie erforderlich, um die erforderlichen Amazon CloudWatch Events-Ressourcen für die Amazon S3 S3-Pipeline erstellenAWS CloudFormation zu können:

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

7. Zeigen Sie in AWS CloudFormation auf der Registerkarte Resources (Ressourcen) für Ihren Stack die Ressourcen an, die für Ihren Stack erstellt wurden.

> **Note**
>
> Um die erstellte Pipeline anzuzeigen, suchen Sie die Spalte Logische ID auf der Registerkarte Ressourcen für Ihren Stack inAWS CloudFormation. Notieren Sie sich den Namen in der Spalte Physikalische ID für die Pipeline. In CodePipeline können Sie die Pipeline mit derselben physikalischen 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 AWS CloudFormation neu erstellt wird. Öffnen Sie die Amazon S3 S3-Konsole und suchen Sie Ihren`sourcebucket` Bucket. Wählen Sie Upload (Hochladen) aus, und folgen Sie den Anweisungen, um Ihre `SampleApp_Linux.zip`-ZIP-Datei hochzuladen.

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, müssen Sie alle Quelldateien, die als einzelne ZIP-Datei verpackt sind, in Ihren Bucket hochladen. Andernfalls schlägt die Quellaktion fehl.

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

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.

9. Führen Sie die Schritte im folgenden Verfahren aus, um Ihre AWS CloudTrail-Ressourcen zu erstellen.

Erstellen Ihrer AWS CloudTrail-Ressourcen in AWS CloudFormation

1. Öffnen Sie die AWS CloudFormation-Konsole und wählen Sie Create Stack (Stack erstellen) aus.
2. Wählen Sie unter Choose a template (Auswahl einer Vorlage) die Option Upload a template to Amazon S3 (Vorlage auf Amazon S3 hochladen). Klicken Sie auf Browse (Durchsuchen) und wählen Sie die Vorlagendatei für die AWS CloudTrail-Ressourcen auf Ihrem lokalen Computer aus. Wählen Sie Next (Weiter).
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:

   - Akzeptieren Sie in SourceObjectKeydie Standardeinstellung für die Zip-Datei der Beispielanwendung.

4. Wählen Sie Next (Weiter). Übernehmen Sie die Standardeinstellungen auf der folgenden Seite und wählen Sie Next (Weiter) aus.
5. Wählen Sie unter Funktionen die Option Ich bestätige, dassAWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden, und wählen Sie dann Erstellen aus.
6. Nachdem die Erstellung Ihres Stacks abgeschlossen wurde, zeigen Sie die Ereignisliste an, um zu überprüfen, ob Fehler aufgetreten sind.

Die folgenden Berechtigungen sind in der Richtlinie erforderlich, um die erforderlichen CloudTrail Ressourcen für die Amazon S3 S3-Pipeline erstellenAWS 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 anAWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
8. Führen Sie in Ihrem Quell-Bucket einen Commit für eine Änderung durch und übertragen Sie diese. Ihre Ressourcen für die Änderungserkennung übernehmen die Änderung und Ihre Pipeline startet.

Tutorial: Erstellen einer Pipeline, die Variablen aus AWS CloudFormation-Bereitstellungsaktionen verwendet

In diesem Tutorial erstellen Sie mithilfe der AWS CodePipeline-Konsole eine Pipeline mit einer Bereitstellungsaktion. 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 von der AWS CloudFormation-Aktion in CodePipeline generiert werden.

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.

Themen

- Voraussetzungen: Erstellen Sie eine AWS CloudFormation Servicerolle und ein CodeCommit Repository
- Schritt 1: Herunterladen, Bearbeiten und Hochladen der AWS CloudFormation-Beispielvorlage
- Schritt 2: Erstellen der Pipeline
- Schritt 3: Hinzufügen einer AWS 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 Stack zu löschen

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

Sie müssen bereits über Folgendes verfügen:

- Ein CodeCommit Repository. Sie können das in Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)erstellte AWS CodeCommit-Repository verwenden.
- In diesem Beispiel wird ein Amazon DocumentDB DocumentDB-Stack aus einer Vorlage erstellt. Sie müssen AWS Identity and Access Management (IAM) verwenden, um eine AWS CloudFormation Servicerolle mit den folgenden Berechtigungen für Amazon DocumentDB zu erstellen.

```
"rds:DescribeDBClusters",
"rds:CreateDBCluster",
"rds:DeleteDBCluster",
"rds:CreateDBInstance"
```

Schritt 1: Herunterladen, Bearbeiten und Hochladen der AWS CloudFormation-Beispielvorlage

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

1. Navigieren Sie zur Beispielvorlagenseite für Ihre Region. Die Seite für us-west-2 befindet sich beispielsweise unter https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/sample-templates-services-us-west-2.html. Laden Sie unter Amazon DocumentDB die Vorlage für einen Amazon DocumentDB-Cluster herunter. Der Dateiname lautet `documentdb_full_stack.yaml`.
2. Entpacken Sie die `documentdb_full_stack.yaml`-Datei, und öffnen Sie sie in einem Texteditor. Nehmen Sie die folgenden Änderungen vor.

   a. Fügen Sie in diesem Beispiel den folgenden `Purpose:`-Parameter zu Ihrem `Parameters`-Abschnitt in der Vorlage hinzu.

   ```
   Purpose:
     Type: String
     Default: testing
     AllowedValues:
       - testing
       - production
     Description: The purpose of this instance.
   ```

   b. Fügen Sie in diesem Beispiel die folgende `StackName`-Ausgabe zu Ihrem `Outputs:`-Abschnitt in der Vorlage hinzu.

   ```
   StackName:
     Value: !Ref AWS::StackName
   ```

3. Laden Sie die Vorlagendatei in Ihr AWS CodeCommit-Repository hoch. Sie müssen die entpackte und bearbeitete Vorlagendatei in das Stammverzeichnis Ihres Repositorys hochladen.

   So verwenden Sie die CodeCommit Konsole, um Ihre Dateien hochzuladen:

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

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

   c. Wählen Sie Choose file (Datei auswählen) und navigieren Sie dann zu Ihrer Datei. Übernehmen Sie die Änderung, indem Sie Ihren Benutzernamen und Ihre E-Mail-Adresse eingeben. Wählen Sie Commit changes (Änderungen übernehmen) aus.

   Ihre Datei sollte auf der Stammebene in Ihrem Repository wie folgt aussehen:

```
documentdb_full_stack.yaml
```

Schritt 2: Erstellen der Pipeline

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

- Eine Quellstufe mit einer CodeCommit Aktion, bei der das Quellartefakt Ihre Vorlagendatei ist.
- Eine Bereitstellungsstufe mit einer AWS CloudFormation-Bereitstellungsaktion.

Jeder Aktion in der Quell- und Bereitstellungsstufe, die vom Assistenten erstellt wurde, wird jeweils ein Variablen-Namespace, `SourceVariables`, und `DeployVariables` zugewiesen. Da den Aktionen ein Namespace zugewiesen ist, stehen die in diesem Beispiel konfigurierten Variablen für nachgelagerte Aktionen zur Verfügung. Weitere Informationen finden Sie unter Variablen.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyCFNDeployPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:
   - Wählen Sie Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
   - Wählen Sie Existing service role (Vorhandene Servicerolle). Wählen Sie unter Name der Rolle in der Liste Ihre Servicerolle aus.
6. Unter Artifact store (Artefaktspeicher):

a. Wählen Sie Standardstandort, um den Standard-Artefaktspeicher, z. B. den Amazon S3 S3-Artefakt-Bucket, der als Standard festgelegt wurde, für Ihre Pipeline in der Region zu verwenden, die Sie für Ihre Pipeline ausgewählt haben.

b. Wählen Sie Benutzerdefinierter Standort, wenn Sie bereits einen Artefaktspeicher, z. B. einen Amazon S3 S3-Artefakt-Bucket, in derselben Region wie Ihre Pipeline haben.

> **Note**
>
> Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket. Wenn Sie eine Pipeline erstellen oder bearbeiten, benötigen Sie einen Artefakt-Bucket in der Pipeline-Region und einen Artefakt-Bucket pro AWS Region, in der Sie eine Aktion ausführen.
> Weitere Informationen finden Sie unter Eingabe- und Ausgabe-Artefakte und CodePipeline Referenz zur Pipeline-Struktur.

Wählen Sie Weiter.

7. Unter Step 2: Add source stage (Schritt 2: Quell-Stufe hinzufügen):

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

   b. Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, in dem Sie es erstellt haben. Schritt 1: 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 soll.

   Wählen Sie Weiter.

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

   Wählen Sie Weiter.

9. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

   a. Wählen Sie unter Action name (Aktionsname) die Option Deploy (Bereitstellen). Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option CloudFormation aus.

   b. Wählen Sie unter Action mode (Aktionsmodus) Create or update a stack (Einen Stack erstellen oder aktualisieren) aus.

   c. Geben Sie unter Stack name (Stack-Name) einen Namen für den Stapel ein. Dies ist der Name des Stacks, den die Vorlage erstellt.

   d. Geben Sie unter Output file name (Ausgabedateiname) einen Namen für die Ausgabedatei ein, z. B. **`outputs`**. Dies ist der Name der Datei, die durch die Aktion erstellt wird, nachdem der Stack erstellt wurde.

   e. Erweitern Sie Advanced (Erweitert). Geben Sie unter Parameter overrides (Parameterüberschreibungen) Ihre Vorlagenüberschreibungen als Schlüssel-Wert-Paare ein. Diese Vorlage erfordert beispielsweise die folgenden Überschreibungen.

      ```
      {
      "DBClusterName": "MyDBCluster",
      "DBInstanceName": "MyDBInstance",
      "MasterUser": "UserName",
      "MasterPassword": "Password",
      "DBInstanceClass": "db.r4.large",
      "Purpose": "testing"}
      ```

      Wenn Sie keine Überschreibungen eingeben, erstellt die Vorlage einen Stack mit Standardwerten.

   f. Wählen Sie Weiter.

   g. Wählen Sie Create pipeline (Pipeline erstellen) aus. Lassen Sie die Ausführung Ihrer Pipeline zu. Ihre zweistufige Pipeline ist damit vollständig und bereit für die zusätzlichen Stufen , die hinzugefügt werden sollen.

Schritt 3: Hinzufügen einer AWS CloudFormation-Bereitstellungsaktion zum Erstellen des Änderungssatzes

Erstellen Sie eine Folgeaktion in der Pipeline, die es AWS CloudFormation ermöglicht, 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 unter Action name (Aktionsname) Change_Set ein. Wählen Sie unter Aktionsanbieter die Option aus AWS CloudFormation.

   b. Wählen Sie unter Eingabeartefakt die Option aus SourceArtifact.

   c. Wählen Sie unter Action mode (Aktionsmodus) die Option Create or replace a change set (Einen Änderungssatz erstellen oder ersetzen) aus.

   d. Geben Sie unter Stack name (Stack-Name) die Variablensyntax ein, wie hier gezeigt. Dies ist der Name des Stacks, für den der Änderungssatz erstellt wird, wobei der Standard-Namespace `DeployVariables` der Aktion zugewiesen wird.

   ```
   #{DeployVariables.StackName}
   ```

   e. Geben Sie unter Change set name (Name des Änderungssatzes) den Namen des Änderungssatzes ein.

   ```
   my-changeset
   ```

   f. Ändern Sie unter Parameter overrides (Parameterüberschreibungen) den `Purpose`-Parameter von `testing` zu `production`.

   ```
   {
   "DBClusterName": "MyDBCluster",
   "DBInstanceName": "MyDBInstance",
   "MasterUser": "UserName",
   "MasterPassword": "Password",
   "DBInstanceClass": "db.r4.large",
   "Purpose": "production"}
   ```

g. Wählen Sie Done (Fertig), um die Aktion zu speichern.

Schritt 4: Hinzufügen einer manuellen Genehmigungsaktion

Erstellen Sie eine manuelle Genehmigungsaktion in Ihrer Pipeline.

1. Wählen Sie, ob Sie die Pipeline bearbeiten möchten, oder fahren Sie mit der Anzeige der Pipeline im Modus Edit (Bearbeiten) fort.
2. Wählen Sie, ob Sie die Bereitstellungsphase bearbeiten möchten.
3. Fügen Sie nach der Bereitstellungsaktion, die den Änderungssatz erstellt, eine manuelle Genehmigungsaktion hinzu. Mit dieser Aktion können Sie den in AWS CloudFormation erstellten Ressourcenänderungssatz überprüfen, bevor die Pipeline den Änderungssatz ausführt.

Schritt 5: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Änderungssatz auszuführen

Erstellen Sie eine Folgeaktion in Ihrer Pipeline, die es AWS 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 unter Action name (Aktionsname) Execute_Change_Set ein. Wählen Sie unter Aktionsanbieter die Option aus AWS CloudFormation.

   b. Wählen Sie unter Eingabeartefakt die Option aus SourceArtifact.

   c. Wählen Sie unter Action mode (Aktionsmodus) die Option Execute a change set (Einen Änderungssatz ausführen).

d. Geben Sie unter Stack name (Stack-Name) die Variablensyntax ein, wie hier gezeigt. Dies ist der Name des Stacks, für den der Änderungssatz erstellt wird.

```
#{DeployVariables.StackName}
```

e. Geben Sie unter Change set name (Name des Änderungssatzes) den Namen des Änderungssatzes ein, den Sie in der vorherigen Aktion erstellt haben.

```
my-changeset
```

f. Wählen Sie Done (Fertig), um die Aktion zu speichern.

g. Fahren Sie mit der Pipeline-Ausführung fort.

Schritt 6: Fügen Sie eine CloudFormation Bereitstellungsaktion hinzu, um den Stack zu löschen

Erstellen Sie eine abschließende Aktion in Ihrer Pipeline, die es AWS CloudFormation ermöglicht, den Stack-Namen 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 im Feld Aktionsname die Option DeleteStack. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option CloudFormation aus.

   b. Wählen Sie unter Action mode (Aktionsmodus) die Option Delete a Stack (Stack löschen) aus.

   c. Geben Sie unter Stack name (Stack-Name) die Variablensyntax ein, wie hier gezeigt. Dies ist der Name des Stacks, den die Aktion löschen wird.

   d. Wählen Sie Done (Fertig), um die Aktion zu speichern.

   e. Wählen Sie Save (Speichern), um die Pipeline zu speichern.

Die Pipeline wird ausgeführt, wenn sie gespeichert wird.

Tutorial: Amazon-ECS-Standardbereitstellung mit CodePipeline

Dieses Tutorial hilft Ihnen beim Erstellen einer vollständigen CD-Pipeline (end-to-end Continuous Deployment) mit Amazon ECS mit CodePipeline.

> **Note**
>
> Dieses Tutorial bezieht sich auf die Amazon-ECS-Standardbereitstellungsaktion für CodePipeline. Ein Tutorial, das die Blau CodeDeploy /Grün-Bereitstellungsaktion von Amazon ECS in verwendet CodePipeline, finden Sie unter Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy .

Voraussetzungen

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

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

- Ein Quellcodeverwaltungs-Repository (in diesem Tutorial wird verwendet CodeCommit) mit Ihrer Dockerfile- und Anwendungsquelle. Weitere Informationen finden Sie unter Erstellen eines CodeCommit Repositorys im AWS CodeCommit -Benutzerhandbuch.
- Ein Docker-Image-Repository (dieses Tutorial verwendet Amazon ECR), das ein Image enthält, das Sie aus Ihrer Dockerfile- und Anwendungsquelle erstellt haben. Weitere Informationen finden Sie unter Erstellen eines Repositorys und Verschieben 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-Entwicklerhandbuch.

> **Important**
>
> Die Amazon-ECS-Standardbereitstellungsaktion für CodePipeline erstellt eine eigene Revision der Aufgabendefinition basierend auf der Revision, die vom Amazon-ECS-Service verwendet wird. Wenn Sie neue Revisionen für die Aufgabendefinition erstellen, ohne den Amazon-ECS-Service zu aktualisieren, ignoriert die Bereitstellungsaktion diese Revisionen.

Im Folgenden finden Sie ein Beispiel für eine Aufgabendefinition, die für dieses Tutorial verwendet wird. Der Wert, den Sie für `name` und verwenden`family`, wird im nächsten Schritt für Ihre Build-Spezifikationsdatei verwendet.

```
{
  "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 Erstellen eines Clusters und Erstellen eines Services im Amazon Elastic Container Service-Entwicklerhandbuch.

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

Schritt 1: Hinzufügen einer Build-Spezifikationsdatei zu Ihrem Quell-Repository

In diesem Tutorial wird verwendet CodeBuild , um 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:

- Phase vor dem Build:
  - Melden Sie sich bei Amazon ECR an.
  - Stellen Sie die Repository-URI auf Ihr ECR-Image ein und fügen Sie ein Image-Tag mit den ersten sieben Zeichen der Git-Commit-ID der Quelle hinzu.
- Build-Phase:
  - Erstellen Sie das Docker-Image und taggen Sie das Image sowohl als `latest` als auch mit der Git-Commit-ID.
- Phase nach dem Build:
  - Führen Sie die Push-Übertragung des Image zu Ihrem ECR-Repository mit beiden Tags durch.
  - Schreiben Sie eine Datei mit dem Namen `imagedefinitions.json` in das Build-Stammverzeichnis, die den Containernamen Ihres Amazon-ECS-Services sowie das Image und das Tag enthält. Die Bereitstellungsphase Ihrer CD-Pipeline erstellt anhand dieser Informationen eine neue Revision Ihrer Service-Aufgabendefinition und aktualisiert dann den Service für die

Nutzung der neuen Aufgabendefinition. Die Datei `imagedefinitions.json` wird für den ECS-Auftragsworker benötigt.

Fügen Sie diesen Beispieltext ein, um Ihre `buildspec.yml` Datei zu erstellen, und ersetzen Sie die Werte für Ihr Bild und Ihre Aufgabendefinition. In diesem Text wird die Beispiel-Konto-ID 111122223333 verwendet.

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
      - REPOSITORY_URI=012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"hello-world","imageUri":"%s"}]' $REPOSITORY_URI:$IMAGE_TAG >
 imagedefinitions.json
artifacts:
    files: imagedefinitions.json
```

Die Build-Spezifikation wurde für die Beispielaufgabendefinition geschrieben, die in bereitgestellt wurde Voraussetzungenund vom Amazon-ECS-Service für dieses Tutorial verwendet wird. Der `REPOSITORY_URI`-Wert entspricht dem `image`-Repository (ohne Image-Tag), und der *`hello-`*

world-Wert am Ende der Datei entspricht dem Container-Namen in der Aufgabendefinition des Service.

So fügen Sie eine **buildspec.yml**-Datei zu Ihrem Quell-Repository hinzu

1. Öffnen Sie einen Text-Editor und kopieren Sie die Build-Spezifikation in eine neue Datei.
2. Ersetzen Sie den `REPOSITORY_URI` Wert (*012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world*) durch Ihren 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-Phasen 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 zum ersten Mal verwenden CodePipeline, wird anstelle von Willkommen eine einleitende Seite angezeigt. Wählen Sie Get Started Now.

3. Geben Sie auf der Seite Schritt 1: Name für Pipeline-Name den Namen für Ihre Pipeline ein. Bei diesem Tutorial lautet der Pipeline-Name hello-world.
4. Wählen Sie im Pipeline-Typ V1 für die Zwecke dieses Tutorials aus. V1 Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in Bezug auf Merkmale und Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen. Wählen Sie Weiter.
5. Wählen Sie auf der Seite Schritt 2: Quellstufe hinzufügen für Quellanbieter aus 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.
6. Wählen Sie auf der Seite Schritt 3: Build-Stufe hinzufügen für Build-Anbieter AWS CodeBuildund dann Projekt erstellen aus.
   a. Wählen Sie für Project name einen eindeutigen Namen für Ihr Build-Projekt aus. Bei diesem Tutorial lautet der Projektname hello-world.
   b. Wählen Sie für Environment image (Umgebungs-Abbild) die Option Managed image (Verwaltetes Abbild) aus.
   c. Wählen Sie für Operating system (Betriebssystem) die Option Amazon Linux 2 aus.
   d. Wählen Sie unter Runtime (Laufzeit) die Option Standard aus.
   e. Wählen Sie für Image aus**`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 die Auswahl von CloudWatch Protokollen. Möglicherweise müssen Sie Advanced erweitern.
   i. Wählen Sie Weiter zu aus CodePipeline.
   j. Wählen Sie Weiter aus.

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

7. Wählen Sie auf der Seite Schritt 4: Bereitstellungsstufe hinzufügen für Bereitstellungsanbieter die Option Amazon ECS aus.

   a. Wählen Sie für Clustername den Amazon-ECS-Cluster aus, in dem Ihr Service ausgeführt wird. Für dieses Tutorial lautet der Cluster default.

   b. Wählen Sie für Service name (Service-Name) den zu aktualisierenden Service und dann Next (Weiter) aus. Bei diesem Tutorial lautet der Servicename hello-world.

8. Überprüfen Sie auf der Seite Step 5: Review (Schritt 5: Überprüfen) die Pipeline-Konfiguration und wählen Sie Create pipeline (Pipeline erstellen), um die Pipeline zu erstellen.

> **Note**
>
> Nachdem die Pipeline erstellt wurde, versucht sie, die verschiedenen Pipeline-Phasen zu durchlaufen. Die vom Assistenten erstellte CodeBuild Standardrolle verfügt jedoch nicht über Berechtigungen zum Ausführen aller in der `buildspec.yml` Datei enthaltenen Befehle, sodass die Build-Phase fehlschlägt. Der nächste Abschnitt fügt die Berechtigungen für die Build-Phase hinzu.

Schritt 3: Hinzufügen von Amazon-ECR-Berechtigungen zur CodeBuild Rolle

Der CodePipeline Assistent hat eine IAM-Rolle für das CodeBuild Build-Projekt namens codebuild*build-project-name*-service-role erstellt. In diesem Tutorial lautet der Name codebuild-hello-world-service-role . Da die -`buildspec.yml`Datei Aufrufe an Amazon-ECR-API-Operationen vornimmt, muss die Rolle über eine Richtlinie verfügen, die Berechtigungen für diese Amazon-ECR-Aufrufe gewährt. Das folgende Verfahren hilft Ihnen, die richtigen Berechtigungen an die Rolle anzufügen.

So fügen Sie der CodeBuild Rolle Amazon-ECR-Berechtigungen hinzu

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

Schritt 4: Testen Ihrer Pipeline

Ihre Pipeline sollte über alles verfügen, um eine end-to-end native AWS kontinuierliche Bereitstellung auszuführen. Testen Sie nun ihre Funktionalität, indem Sie eine Code-Änderung mithilfe von Push zu Ihrem Quell-Repository übertragen.

So testen Sie Ihre Pipeline

1. Nehmen Sie eine Code-Änderung an Ihrem konfigurierten Quell-Repository vor und führen Sie den Commit-Vorgang und die Push-Übertragung der Änderung durch.
2. Öffnen Sie die - CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
3. Wählen Sie Ihre Pipeline in der Liste aus.
4. Sehen Sie sich an, wie die Pipeline ihre Phasen durchläuft. Ihre Pipeline sollte abgeschlossen sein und Ihr Amazon-ECS-Service führt das Docker-Image aus, das aus Ihrer Codeänderung erstellt wurde.

Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy

In diesem Tutorial konfigurieren Sie eine Pipeline in AWS CodePipeline, die Container-Anwendungen mithilfe einer Blau/Grün-Bereitstellung bereitstellt, die Docker-Images unterstützt. In einer Blau/Grün-Bereitstellung können Sie die neue Version Ihrer Anwendung zusammen mit der alten Version

starten und die neue Version testen, bevor Sie Datenverkehr umleiten. Sie können auch den Bereitstellungsprozess überwachen und bei Problemen schnell einen Rollback durchführen.

> **Note**
>
> Dieses Tutorial bezieht sich auf die Aktion Amazon ECS to CodeDeploy Blue/Green Deployment für CodePipeline. Ein Tutorial, das die Amazon ECS-Standardbereitstellungsaktion in verwendet CodePipeline, finden Sie unterTutorial: Amazon-ECS-Standardbereitstellung mit CodePipeline.

Die abgeschlossene Pipeline erkennt Änderungen an Ihrem Image, das in einem Image-Repository wie Amazon ECR gespeichert ist und verwendet wird, CodeDeploy um den Datenverkehr an einen Amazon ECS-Cluster und einen Load Balancer weiterzuleiten und bereitzustellen. CodeDeploy verwendet einen Listener, um den Datenverkehr an den Port des aktualisierten Containers umzuleiten, der in der Datei angegeben ist. AppSpec Informationen darüber, wie der Load Balancer, der Produktions-Listener, die Zielgruppen und Ihre Amazon ECS-Anwendung in einer blauen/grünen Bereitstellung verwendet werden, finden Sie unter Tutorial: Bereitstellen eines Amazon ECS-Service.

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

Ihre Continuous Delivery-Pipeline erstellt und stellt automatisch Container-Images bereit, wenn der Quellcode geändert oder ein neues Basis-Image in Amazon ECR hochgeladen wird.

In diesem Ablauf werden die folgenden Artefakte verwendet:

- Eine Docker-Image-Datei, die den Container-Namen und die Repository-URI Ihres Amazon ECR-Image-Repositorys angibt.
- Eine Amazon ECS-Aufgabendefinition, die Ihren Docker-Image-Namen, Container-Namen, Amazon ECS-Servicenamen und Load Balancer-Konfiguration auflistet.
- Eine CodeDeploy AppSpec Datei, die den Namen der Amazon ECS-Aufgabendefinitionsdatei, den Namen des Containers der aktualisierten Anwendung und den Container-Port angibt, an den der CodeDeploy Produktionsdatenverkehr umgeleitet wird. Sie kann auch optionale Netzwerkkonfigurations- und Lambda-Funktionen angeben, die Sie für Lebenszyklusereignis-Hooks während der Bereitstellung ausführen.

> **Note**
>
> Wenn Sie eine Änderung an Ihrem Amazon ECR-Image-Repository festschreiben, erstellt die Pipeline-Quellaktion eine `imageDetail.json` Datei für diesen Commit. Informationen zur `imageDetail.json`-Datei finden Sie unter imageDetail.json-Datei für Amazon ECS-Blau/Grün-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 Aufgabendefinitions- und 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 Ihrer CodeDeploy-Anwendung und Bereitstellungsgruppe (ECS-Datenverarbeitungsplattform)
- 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 in Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)erstellte AWS CodeCommit-Repository verwenden.
- Starten Sie eine Amazon EC2 EC2-Linux-Instance und installieren Sie Docker, um ein Image zu erstellen, wie in diesem Tutorial gezeigt. Wenn Sie bereits ein Image haben, das Sie verwenden möchten, können Sie diesen Schritt überspringen.

Schritt 1: Image erstellen und in ein Amazon ECR-Repository übertragen

In diesem Abschnitt verwenden Sie Docker, um ein Image zu erstellen, und verwenden dann das, AWS CLI um ein Amazon ECR-Repository zu erstellen und das Image in das Repository zu übertragen.

> **Note**
>
> Wenn Sie bereits ein Image haben, das Sie verwenden möchten, können Sie diesen Schritt überspringen.

So erstellen Sie ein Image

1. Melden Sie sich bei Ihrer Linux-Instance an, in der Sie Docker installiert haben.

   Rufen Sie ein Image für `nginx` ab. Dieser Befehl stellt das `nginx:latest` Bild bereit:

   ```
   docker pull nginx
   ```

2. Führen Sie docker images. Das Image sollte in der Liste angezeigt werden.

   ```
   docker images
   ```

Um ein Amazon ECR-Repository zu erstellen und Ihr Image zu pushen

1. Erstellen Sie ein Amazon ECR-Repository, um Ihr -Image zu speichern. Notieren Sie sich den `repositoryUri` in der Ausgabe.

   ```
   aws ecr create-repository --repository-name nginx
   ```

   Ausgabe:

   ```
   {
       "repository": {
           "registryId": "aws_account_id",
           "repositoryName": "nginx",
           "repositoryArn": "arn:aws:ecr:us-east-1:aws_account_id:repository/nginx",
           "createdAt": 1505337806.0,
   ```

```
        "repositoryUri": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx"
    }
}
```

2. Versehen Sie das Image mit dem `repositoryUri`-Wert aus dem vorherigen Schritt als Tag.

```
docker tag nginx:latest aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
```

3. Führen Sie den aws ecr get-login-password Befehl aus, wie in diesem Beispiel für die `us-west-2` Region und die Konto-ID 111122223333 gezeigt.

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com/nginx
```

4. Pushen Sie das Bild mit dem `repositoryUri` aus dem vorherigen Schritt an Amazon ECR.

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

Schritt 2: Erstellen Sie Aufgabendefinitions- und AppSpec Quelldateien und übertragen Sie sie in ein Repository CodeCommit

In diesem Abschnitt erstellen Sie eine JSON-Datei mit Aufgabendefinitionen und registrieren sie bei Amazon ECS. Anschließend erstellen Sie eine AppSpec Datei für Ihren Git-Client CodeDeploy und verwenden diesen, um die Dateien in Ihr CodeCommit Repository zu übertragen.

So erstellen Sie eine Aufgabendefinition für Ihr Image

1. Erstellen Sie eine Datei mit dem Namen `taskdef.json` und dem folgenden Inhalt. Geben Sie für `image` Ihren Image-Namen, z. B. nginx, ein. Dieser Wert wird aktualisiert, wenn Ihre Pipeline ausgeführt wird.

> **Note**
>
> Stellen Sie sicher, dass die in der Aufgabendefinition angegebene Ausführungsrolle den `AmazonECSTaskExecutionRolePolicy` enthält. Weitere Informationen finden Sie unter Amazon ECS Task Execution IAM Role im Amazon ECS Developer Guide.

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "name": "sample-website",
            "image": "nginx",
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
    "family": "ecs-demo"
}
```

2. Registrieren Sie Ihre Aufgabendefinition mit der Datei `taskdef.json`.

```
aws ecs register-task-definition --cli-input-json file://taskdef.json
```

3. Nachdem die Aufgabendefinition registriert ist, bearbeiten Sie Ihre Datei, um den Image-Namen zu entfernen und den <IMAGE1_NAME>-Platzhaltertext in das Image-Feld einzufügen.

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "name": "sample-website",
            "image": "<IMAGE1_NAME>",
            "essential": true,
            "portMappings": [
                {
```

```
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
    "family": "ecs-demo"
}
```

Um eine Datei zu erstellen AppSpec

- Die AppSpec Datei wird für CodeDeploy Bereitstellungen verwendet. Die Datei, die optionale Felder enthält, verwendet das folgende Format:

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "task-definition-ARN"
        LoadBalancerInfo:
          ContainerName: "container-name"
          ContainerPort: container-port-number
# Optional properties
        PlatformVersion: "LATEST"
        NetworkConfiguration:
            AwsvpcConfiguration:
              Subnets: ["subnet-name-1", "subnet-name-2"]
              SecurityGroups: ["security-group"]
              AssignPublicIp: "ENABLED"
Hooks:
- BeforeInstall: "BeforeInstallHookFunctionName"
- AfterInstall: "AfterInstallHookFunctionName"
- AfterAllowTestTraffic: "AfterAllowTestTrafficHookFunctionName"
- BeforeAllowTraffic: "BeforeAllowTrafficHookFunctionName"
```

```
- AfterAllowTraffic: "AfterAllowTrafficHookFunctionName"
```

Weitere Informationen zu der AppSpec Datei, einschließlich Beispielen, finden Sie unter CodeDeploy AppSpec Dateireferenz.

Erstellen Sie eine Datei mit dem Namen `appspec.yaml` und dem folgenden Inhalt. Ändern Sie für `TaskDefinition` den <TASK_DEFINITION>-Platzhaltertext nicht. Dieser Wert wird aktualisiert, wenn Ihre Pipeline ausgeführt wird.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: <TASK_DEFINITION>
        LoadBalancerInfo:
          ContainerName: "sample-website"
          ContainerPort: 80
```

Um Dateien in Ihr CodeCommit Repository zu übertragen

1. Push oder lade die Dateien in dein CodeCommit Repository hoch. Diese Dateien sind das Quellartefakt, das vom Assistenten 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. Führen Sie den folgenden Befehl aus, um die Dateien von Ihrem lokalen Repository in Ihr CodeCommit Repository zu übertragen:

```
git push
```

b. So verwenden Sie die CodeCommit Konsole, um Ihre Dateien hochzuladen:

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

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

iii. Wählen Sie Choose file (Datei auswählen) aus und navigieren Sie anschließend zu Ihrer Datei. Übernehmen Sie die Änderung, indem Sie Ihren Benutzernamen und Ihre E-Mail-Adresse eingeben. Wählen Sie Commit changes (Änderungen übernehmen) aus.

iv. Wiederholen Sie diesen Schritt für jede Datei, die Sie hochladen möchten.

Schritt 3: Erstellen Ihres Application Load Balancer und der Zielgruppen

In diesem Abschnitt erstellen Sie einen Amazon EC2 Application Load Balancer. Sie verwenden die Subnetznamen und Zielgruppenwerte, die Sie mit Ihrem Load Balancer erstellen, später, wenn Sie Ihren Amazon ECS-Service erstellen. Sie können einen Application Load Balancer oder einen Network Load Balancer erstellen. Der Load Balancer muss eine VPC mit zwei öffentlichen Subnetzen in verschiedenen Availability Zones verwenden. In diesen Schritten bestätigen Sie Ihre Standard-VPC, erstellen einen Load Balancer und legen dann zwei Zielgruppen für Ihren Load Balancer fest. Weitere Informationen finden Sie unter Zielgruppen für Ihre Network Load Balancer.

So überprüfen Sie Ihre Standard-VPC und öffentliche Subnetze

1. Melden Sie sich an der AWS Management Console an 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.

So erstellen Sie einen Amazon EC2 Application Load Balancer

1. Melden Sie sich bei der AWS Management Console an 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 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 (zu einem Amazon ECS-Cluster statt zu EC2-Instances). Um Ihren Amazon ECS-Service zu erstellen, müssen Sie die Subnetznamen, Sicherheitsgruppen und Zielgruppenwerte verwenden, die Sie mit Ihrem Load Balancer erstellt haben, um Ihren Service zu erstellen.

> **Note**
>
> Wenn Sie diese Schritte verwenden, um Ihren Amazon ECS-Cluster zu erstellen, verwenden Sie die Cluster-Vorlage nur für Netzwerke, die AWS Fargate-Container bereitstellt. AWS Fargate ist eine Technologie, die Ihre Container-Instance-Infrastruktur für Sie verwaltet. Sie müssen Amazon EC2-Instances für Ihren Amazon ECS-Cluster nicht auswählen oder manuell erstellen.

Erstellen eines Amazon ECS-Clusters

1. Öffnen Sie die klassische Amazon-ECS-Konsole unter https://console.aws.amazon.com/ecs/.
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Wählen Sie Cluster erstellen.
4. Wählen Sie die Clustervorlage Nur Networking aus, die AWS Fargate verwendet, und klicken Sie dann auf Nächster Schritt.
5. Geben Sie auf der Seite Configure cluster (Cluster konfigurieren) einen Namen für den Cluster ein. Sie können ein optionales Tag für Ihre Ressource hinzufügen. Wählen Sie Erstellen aus.

So erstellen Sie einen Amazon ECS-Service

Verwenden Sie denAWS CLI, um Ihren Service in Amazon ECS zu erstellen.

1. Erstellen Sie eine JSON-Datei und geben Sie ihr den Namen `create-service.json`. Fügen Sie Folgendes in die JSON-Datei ein.

   Wenn Sie für dieses `taskDefinition` Feld eine Aufgabendefinition in Amazon ECS registrieren, geben Sie ihr eine Familie. Dies ähnelt einem Namen für mehrere Versionen der Aufgabendefinition und wird mit einer Versionsnummer versehen. Verwenden Sie in diesem Beispiel für Ihre Datei „`ecs-demo:1`“ als Familie und Versionsnummer. Verwenden Sie die Subnetz-Namen, die Sicherheitsgruppe und den Zielgruppenwert, die bzw. den Sie mit Ihrem Load Balancer in Schritt 3: Erstellen Ihres Application Load Balancer und der Zielgruppen erstellt haben.

   > **Note**
   >
   > Sie müssen Ihren Zielgruppen-ARN in dieser Datei angeben. Öffnen Sie die Amazon EC2 EC2-Konsole und wählen Sie im Navigationsbereich unter LOAD BALANCING die Option Target Groups aus. Wählen Sie Ihre erste Zielgruppe aus. Kopieren Sie Ihren ARN aus der Registerkarte Description (Beschreibung).

```
{
    "taskDefinition": "family:revision-number",
    "cluster": "my-cluster",
    "loadBalancers": [
```

```
        {
            "targetGroupArn": "target-group-arn",
            "containerName": "sample-website",
            "containerPort": 80
        }
    ],
    "desiredCount": 1,
    "launchType": "FARGATE",
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "CODE_DEPLOY"
    },
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                "subnet-1",
                "subnet-2"
            ],
            "securityGroups": [
                "security-group"
            ],
            "assignPublicIp": "ENABLED"
        }
    }
}
```

2. Führen Sie den Befehl create-service aus und geben Sie dabei die JSON-Datei an:

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

In diesem Beispiel wird ein Service mit dem Namen `my-service` erstellt.

> **Note**
>
> Dieser Beispielbefehl erstellt einen Service mit dem Namen my-Service. Wenn Sie bereits über einen Service mit diesem Namen verfügen, gibt der Befehl einen Fehler zurück.

```
aws ecs create-service --service-name my-service --cli-input-json file://create-
service.json
```

Die Ausgabe gibt die Beschreibungsfelder für Ihren Service zurück.

3. Führen Sie den Befehl describe-services aus, um zu überprüfen, ob Ihr Service erstellt wurde.

```
aws ecs describe-services --cluster cluster-name --services service-name
```

Schritt 5: Erstellen Ihrer CodeDeploy-Anwendung und Bereitstellungsgruppe (ECS-Datenverarbeitungsplattform)

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, die richtigen Zielgruppen, Listener und das richtige Verhalten bei der Umleitung des Datenverkehrs zu verweisen.

Um eine Anwendung zu erstellen CodeDeploy

1. Öffnen Sie die CodeDeploy Konsole und wählen Sie Anwendung erstellen.
2. Geben Sie unter Application name (Anwendungsname) den Namen ein, den Sie verwenden möchten.
3. Wählen Sie unter Compute Platform (Datenverarbeitungsplattform) die Option Amazon ECS aus.
4. Wählen Sie Erstellen Sie Anwendung aus.

Um eine CodeDeploy Bereitstellungsgruppe zu erstellen

1. Wählen Sie auf Ihrer Anwendungsseite über die Registerkarte Deployment groups (Bereitstellungsgruppen) die Option Create deployment group (Bereitstellungsgruppe erstellen) aus.
2. Geben Sie im Feld Deployment group name (Name der Bereitstellungsgruppe) einen Namen ein, der die Bereitstellungsgruppe beschreibt.
3. Wählen Sie unter Servicerolle eine Servicerolle aus, die CodeDeploy Zugriff auf Amazon ECS gewährt. Gehen Sie folgendermaßen vor, um eine neue Servicerolle zu erstellen:

a. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/).
b. Wählen Sie im Dashboard der Konsole die Option Rollen aus.
c. Wählen Sie Rolle erstellen aus.
d. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen AWS-Servicedie Option aus. Wählen Sie unter Anwendungsfall auswählen die Option aus CodeDeploy. Wählen Sie unter Wählen Sie Ihren Anwendungsfall die Option CodeDeploy - ECS aus. Wählen Sie Weiter: Berechtigungen aus. Die `AWSCodeDeployRoleForECS`-verwaltete Richtlinie ist der Rolle bereits zugewiesen.
e. Wählen Sie Next: Tags (Weiter: Tags) und Next: Review (Weiter: Prüfen) aus.
f. Geben Sie einen Namen für die Rolle ein (beispielsweise **`CodeDeployECSRole`**) und wählen Sie Next (Weiter) aus.

4. Wählen Sie in der Umgebungskonfiguration Ihren Amazon ECS-Clusternamen und Servicenamen aus.
5. Wählen Sie unter Load Balancers den Namen des Load Balancers aus, der den Datenverkehr an Ihren Amazon ECS-Service weiterleitet.
6. Wählen Sie unter Production listener port (Produktions-Listener-Port) den Port und das Protokoll für den Listener aus, der den Produktionsdatenverkehr an Ihren Amazon ECS Service weiterleitet. Wählen Sie unter Test listener port (Test-Listener-Port) den Port und das Protokoll für den Test-Listener aus.
7. Wählen Sie unter Target group 1 name (Name der Zielgruppe 1) und Target group 2 name (Name der Zielgruppe 2) die Zielgruppen für die Weiterleitung des Datenverkehrs während der Bereitstellung aus. Stellen Sie sicher, dass es sich hierbei um die Zielgruppen handelt, die Sie für Ihren Load Balancer erstellt haben.
8. Wählen Sie Traffic sofort umleiten, um zu bestimmen, wie lange nach einer erfolgreichen Bereitstellung der Verkehr zu Ihrer aktualisierten Amazon ECS-Aufgabe umgeleitet werden soll.
9. Wählen Si Create deployment group (Bereitstellungsgruppe erstellen).

Schritt 6: Erstellen Ihrer Pipeline

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

- Eine CodeCommit Aktion, bei der die Quellartefakte die Aufgabendefinition und die AppSpec Datei sind.
- Eine Quellstufe mit einer Amazon ECR-Quellaktion, bei der das Quellartefakt die Bilddatei ist.

- Eine Bereitstellungsphase mit einer Amazon ECS-Bereitstellungsaktion, in der die Bereitstellung mit einer CodeDeploy Anwendung und einer Bereitstellungsgruppe ausgeführt wird.

So erstellen Sie eine zweistufige Pipeline mit dem Assistenten

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) aus, oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyImagePipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie in Service role (Servicerolle) die Option New service role (Neue Servicerolle) aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie unter Step 2: Add source stage (Schritt 2: Quell-Phase hinzufügen) unter Source provider (Quell-Anbieter) die Option AWS CodeCommit. Wählen Sie 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.
8. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken. Wählen Sie Weiter.
9. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Amazon ECS (Blue/Green) (Amazon ECS (Blau/Grün)) aus. Geben Sie unter Application name (Anwendungsname) den Anwendungsnamen ein oder wählen Sie ihn aus der Liste aus, z. B. `codedeployapp`. Geben Sie unter Deployment group (Bereitstellungsgruppe) den Namen einer Bereitstellungsgruppe ein oder wählen Sie ihn in der Liste aus, z. B. `codedeploydeplgroup`

> **Note**
>
> Der Name "Deploy" (Bereitstellen) ist der Name, der der Stufe in Step 4: Deploy (Schritt 4: bereitstellen) standardmäßig gegeben wird, so wie der ersten Stufe der Pipeline der Namen "Source" (Quelle) gegeben wird.

b. Wählen Sie unter Amazon ECS-Aufgabendefinition die Option SourceArtifact. Geben Sie im Feld **`taskdef.json`** ein.

c. Wählen Sie unter AWS CodeDeploy AppSpec Datei die Option aus 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.

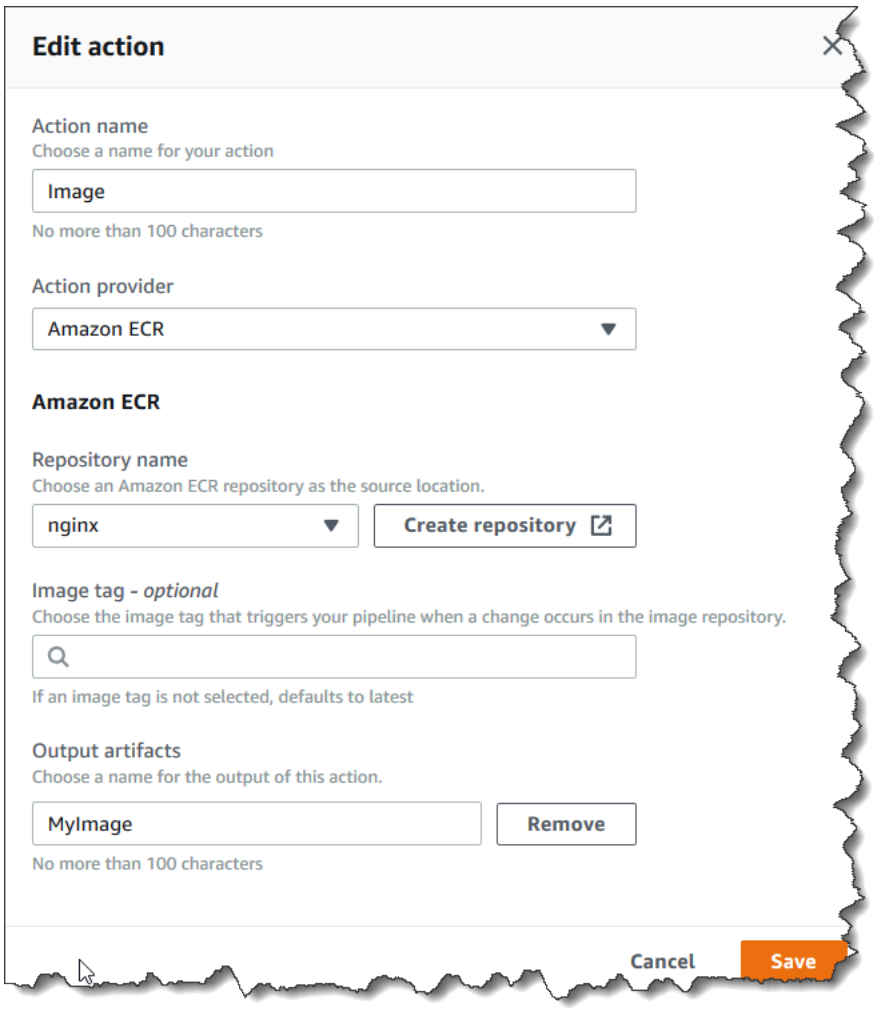
d. Wählen Sie Weiter.

10. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.

So fügen Sie Ihrer Pipeline eine Amazon ECR-Quellaktion hinzu

Sehen Sie sich Ihre Pipeline an und fügen Sie Ihrer Pipeline eine Amazon ECR-Quellaktion hinzu.

1. Wählen Sie Ihre Pipeline aus. Wählen Sie links oben Edit (Bearbeiten) aus.
2. Wählen Sie in der Quellstufe die Option Edit stage (Stufe bearbeiten) aus.
3. Fügen Sie eine parallel Aktion hinzu, indem Sie + Aktion hinzufügen neben Ihrer CodeCommit Quellaktion auswählen.
4. Geben Sie im Feld Action name (Name der Aktion) einen Namen ein (zum Beispiel **`Image`**).
5. Wählen Sie unter Action provider (Aktionsanbieter) die Option Amazon ECR aus.



6. Wählen Sie unter Repository-Name den Namen Ihres Amazon ECR-Repositorys aus.
7. Geben Sie im Feld Image tag (Image-Tag) den Image-Namen und die Version an, falls sie nicht LATEST lautet.
8. Wählen Sie im Feld Output artifacts (Ausgabeartefakte) den Standardwert für die Ausgabeartefakte aus (z. B. `MyImage`), der den Image-Namen und Repository-URI-Informationen enthält, die in der nächsten Stufe verwendet werden sollen.
9. Wählen Sie im Aktionsbildschirm Save (Speichern) aus. Wählen Sie im Phasenbilschirm Done (Fertig) aus. Wählen Sie in der Pipeline Save (Speichern) aus. In einer Meldung wird die Amazon CloudWatch Events-Regel angezeigt, die für die Amazon ECR-Quellaktion erstellt werden soll.

So senden Sie Ihre Quellartefakte an die Bereitstellungsaktion

1. Wählen Sie in Ihrer Bereitstellungsphase Bearbeiten und wählen Sie das Symbol, um die Amazon ECS-Aktion (Blau/Grün) zu bearbeiten.

2. Scrollen Sie nach unten bis zum Ende des Bereichs. Wählen Sie unter Input artifacts (Eingabeartefakte) die Option Add (Hinzufügen) aus. Fügen Sie das Quellartefakt aus Ihrem neuen Amazon ECR-Repository hinzu (z. B.`MyImage`).
3. Wählen Sie unter Aufgabendefinition die Option aus SourceArtifact, und geben Sie dann verifizieren **`taskdef.json`** ein.
4. Wählen Sie unter AWS CodeDeploy AppSpec Datei die Option aus SourceArtifact, und bestätigen Sie, **`appspec.yaml`** dass verifizieren eingegeben wurde.
5. Wählen Sie unter Aufgabendefinitionsbild dynamisch aktualisieren unter Eingabeartefakt mit Bild-URI den Platzhaltertext aus MyImage, der in der Datei verwendet wird, und geben Sie dann den Platzhaltertext ein, der in der `taskdef.json` Datei verwendet wird:. **`IMAGE1_NAME`** Wählen Sie Speichern aus.
6. Wählen Sie im Bereich AWS CodePipeline die Option Save pipeline change (Pipeline-Änderung speichern) und dann Save change (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 anzuzeigen CodeDeploy und den Fortschritt der Verkehrsverlagerung zu verfolgen.

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

gewartet, bevor der ursprüngliche Tasksatz beendet wird. Sie können diese Zeit nutzen, um die Aufgabe zurückzusetzen oder zu beenden. Ihre Bereitstellung wird andernfalls abgeschlossen, wenn der Aufgabensatz beendet wird.

Schritt 7: Vornehmen einer Änderung an Ihrer Pipeline und Überprüfen der Bereitstellung

Nehmen Sie eine Änderung an Ihrem Image vor und übertragen Sie die Änderung dann in Ihr Amazon ECR-Repository. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Überprüfen Sie, ob die Änderung Ihrer Image-Quelle bereitgestellt wird.

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

In diesem Tutorial konfigurieren Sie eine Pipeline, über die Ihre Alexa-Qualifikation kontinuierlich bereitgestellt wird, indem das Alexa Skills Kit auf Ihrer Bereitstellungsstufe als Bereitstellungsanbieter verwendet wird. Die fertige Pipeline erkennt Änderungen an Ihrer Qualifikation, wenn Sie in Ihrem Quell-Repository eine Änderung an den Quelldateien vornehmen. Die Pipeline nutzt dann das Alexa Skills Kit für Bereitstellungen für die Entwicklungsstufe der Alexa-Qualifikation.

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

Informationen zum Erstellen Ihres benutzerdefinierten Skills als Lambda-Funktion finden Sie unter Hosten eines benutzerdefinierten Skills als AWS Lambda-Funktion. Sie können auch eine Pipeline erstellen, die Lambda-Quelldateien und ein CodeBuild Projekt verwendet, um Änderungen an Lambda für Ihre Fähigkeiten bereitzustellen. Wenn Sie beispielsweise eine neue Qualifikation und eine zugehörige Lambda-Funktion erstellen möchten, können Sie ein AWS CodeStar-Projekt erstellen. Informationen hierzu finden Sie unter Erstellen eines Alexa-Skill-Projekts in AWS CodeStar. Für diese Option enthält die Pipeline eine dritte Build-Stufe mit einer CodeBuild-Aktion und einer Aktion in der Bereitstellungsstufe für AWS CloudFormation.

Voraussetzungen

Sie müssen bereits über Folgendes verfügen:

- Ein CodeCommit Repository. Sie können das in Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)erstellte AWS CodeCommit-Repository verwenden.
- 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.
- Achten Sie darauf, dass Sie die URLs unter Allowed Return URLs (Zugelassene Rückgabe-URLs) gemäß der Anleitung eingeben. Mit den URLs kann der ASK-CLI-Befehl Anforderungen von Aktualisierungs-Token umleiten.

Schritt 2: Erstellen Sie Alexa-Skill-Quelldateien und übertragen Sie sie in Ihr CodeCommit Repository

In diesem Abschnitt erstellen Sie die Quelldateien für Ihre Alexa-Qualifikation und übertragen sie per Push an das Repository, das von der Pipeline für Ihre Quellstufe verwendet wird. Für die Qualifikation, die Sie in der Amazon-Entwicklerkonsole erstellt haben, erstellen und übertragen Sie Folgendes:

- Die Datei `skill.json`.
- Einen `interactionModel/custom`-Ordner.

> **Note**
>
> Diese Verzeichnisstruktur entspricht den Anforderungen des Alexa Skills Kit-Skill-Pakets, wie in Skill-Paketformat beschrieben. Wenn Ihre Verzeichnisstruktur nicht das richtige Skill-Paketformat verwendet, werden Änderungen nicht erfolgreich in der Alexa Skills Kit-Konsole bereitgestellt.

So erstellen Sie Quelldateien für Ihre Qualifikation

1. Rufen Sie Ihre Qualifikations-ID aus der Alexa Skills Kit-Entwicklerkonsole ab. Verwenden Sie diesen Befehl:

   ```
   ask api list-skills
   ```

   Suchen Sie anhand des Namens nach Ihrer Qualifikation und kopieren Sie die zugehörige ID anschließend in das Feld `skillId`.

2. Generieren Sie die Datei `skill.json`, die Ihre Qualifikationsdetails enthält. Verwenden Sie diesen Befehl:

   ```
   ask api get-skill -s skill-ID > skill.json
   ```

3. (Optional) Erstellen Sie einen `interactionModel/custom`-Ordner.

   Verwenden Sie diesen Befehl, um die Interaktionsmodelldatei im Ordner zu generieren. Als Gebietsschema wird in diesem Tutorial „en-US“ im Dateinamen verwendet.

   ```
   ask api get-model --skill-id skill-ID --locale locale >
       ./interactionModel/custom/locale.json
   ```

Um Dateien in Ihr CodeCommit Repository zu übertragen

1. Push oder lade die Dateien in dein CodeCommit Repository hoch. Diese Dateien stellen das Quellartefakt dar, das vom Assistenten Create Pipeline (Pipeline erstellen) für Ihre Bereitstellungsaktion in AWS CodePipeline erstellt wurde. Ihre Dateien sollten in Ihrem lokalen Verzeichnis wie folgt aussehen:

```
skill.json
/interactionModel
  /custom
    |en-US.json
```

2. Wählen Sie die Methode aus, mit der Sie Ihre Dateien hochladen möchten:

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

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

      ```
      git add -A
      ```

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

      ```
      git commit -m "Added Alexa skill files"
      ```

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

      ```
      git push
      ```

   b. So verwenden Sie die CodeCommit Konsole, um Ihre Dateien hochzuladen:

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

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

      iii. Wählen Sie Choose file (Datei auswählen) aus und navigieren Sie anschließend zu Ihrer Datei. Übernehmen Sie die Änderung, indem Sie Ihren Benutzernamen und Ihre E-Mail-Adresse eingeben. Wählen Sie Commit changes (Änderungen übernehmen) aus.

      iv. Wiederholen Sie diesen Schritt für jede Datei, die Sie hochladen möchten.

Schritt 3: Verwenden von ASK-CLI-Befehlen zum Erstellen eines Aktualisierungs-Tokens

CodePipeline verwendet ein Aktualisierungstoken, das auf der Client-ID und dem geheimen Schlüssel in Ihrem Amazon-Entwicklerkonto basiert, um Aktionen zu autorisieren, die es in Ihrem Namen ausführt. In diesem Abschnitt verwenden Sie die ASK-CLI zum Erstellen des Tokens. Sie nutzen diese Anmeldeinformationen bei der Verwendung des Assistenten Create Pipeline (Pipeline erstellen).

So erstellen Sie ein Aktualisierungs-Token mit den Anmeldeinformationen Ihres Amazon-Entwicklerkontos

1. Verwenden Sie den folgenden Befehl:

   ```
   ask util generate-lwa-tokens
   ```

2. Geben Sie bei entsprechender Aufforderung Ihre Client-ID und den Schlüssel wie in diesem Beispiel ein:

   ```
   ? Please type in the client ID:
   amzn1.application-client.example112233445566
   ? Please type in the client secret:
   example112233445566
   ```

3. Die Browserseite für die Anmeldung wird angezeigt. Melden Sie sich mit den Anmeldeinformationen für Ihr Amazon-Entwicklerkonto an.
4. Wechseln Sie zurück zum Fenster mit der Befehlszeile. Das Zugriffstoken und das Aktualisierungstoken werden in der Ausgabe generiert. Kopieren Sie das zurückgegebene Aktualisierungs-Token, das in der Ausgabe enthalten ist.

Schritt 4: Erstellen Ihrer Pipeline

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

- Eine Quellphase mit einer CodeCommit Aktion, bei der es sich bei den Quellartefakten um die Alexa-Skill-Dateien handelt, die Ihren Skill unterstützen.
- Eine Bereitstellungsstufe mit einer Alexa Skills Kit-Bereitstellungsaktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie die AWS-Region, in der Sie das Projekt und seine Ressourcen erstellen möchten. Die Alexa Skill-Laufzeit ist nur in den folgenden Regionen verfügbar:

   - Asien-Pazifik (Tokio)
   - Europa (Irland)
   - USA Ost (Nord-Virginia)
   - USA West (Oregon)
3. Wählen sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) aus, oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
4. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyAlexaPipeline`** ein.
5. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
6. Wählen Sie in Service role (Servicerolle) die Option New service role (Neue Servicerolle) aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
7. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
8. Wählen Sie unter Step 2: Add source stage (Schritt 2: Quell-Phase hinzufügen) unter Source provider (Quell-Anbieter) die Option AWS CodeCommit. Wählen Sie 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.

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

Wählen Sie Weiter.

10. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

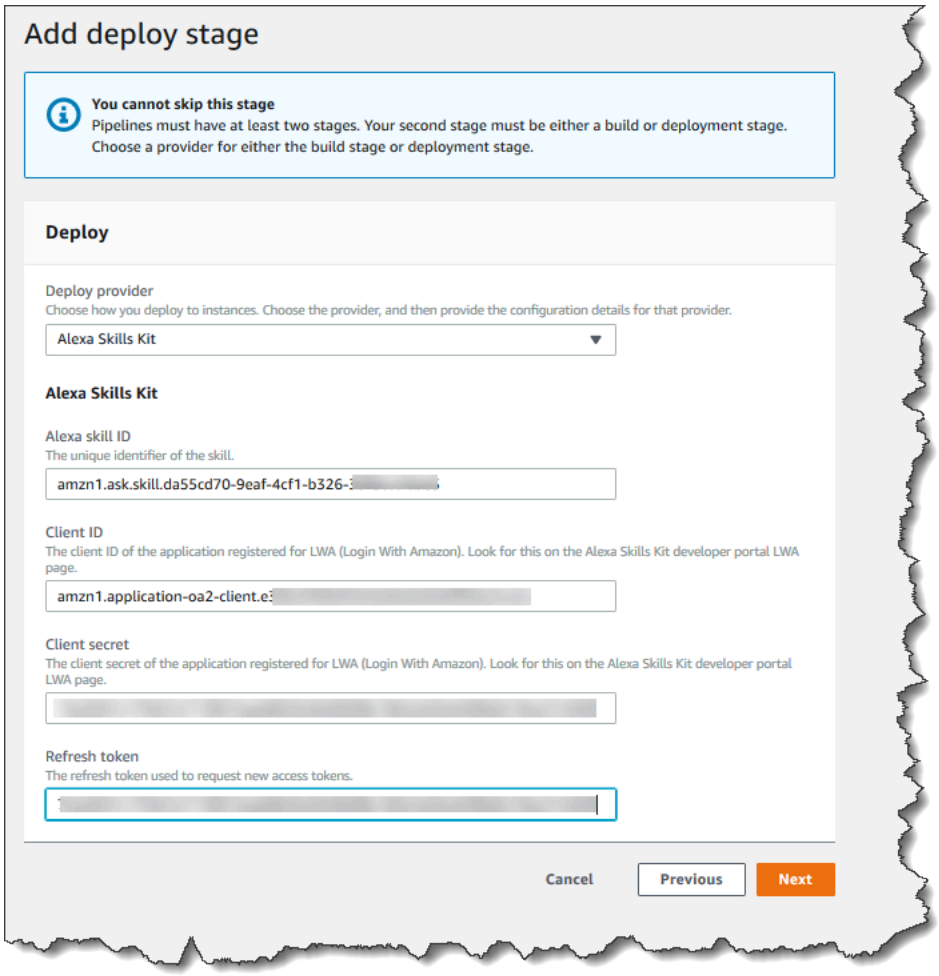
    a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Alexa Skills Kit.

    b. Geben Sie unter Alexa Skill-ID die Qualifikations-ID ein, die Ihrer Qualifikation in der Alexa Skills Kit-Entwicklerkonsole zugewiesen ist.

    c. Geben Sie unter Client-ID die ID der von Ihnen registrierten Anwendung ein.

    d. Geben Sie unter Clientschlüssel den Schlüssel ein, den Sie während der Registrierung gewählt haben.

    e. Geben Sie unter Aktualisierungs-Token das Token ein, das Sie in Schritt 3 generiert haben.



    f. Wählen Sie Weiter.

11. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.

Schritt 5: Vornehmen einer Änderung an einer beliebigen Quelldatei und Überprüfen der Bereitstellung

Nehmen Sie eine Änderung an Ihrer Qualifikation vor und übertragen Sie die Änderung per Push in Ihr Repository. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Vergewissern Sie sich, dass Ihre Qualifikation in der Alexa Skills Kit-Entwicklerkonsole aktualisiert wurde.

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

In diesem Tutorial konfigurieren Sie eine Pipeline, die kontinuierlich Dateien mithilfe von Amazon S3 als Bereitstellungsaktionsanbieter in Ihrer Bereitstellungsphase bereitstellt. Die fertige Pipeline erkennt Änderungen, wenn Sie in Ihrem Quell-Repository eine Änderung an den Quelldateien vornehmen. Die Pipeline verwendet dann Amazon S3, um die Dateien in Ihrem Bucket bereitzustellen. Jedes Mal, wenn Sie Ihre Website-Dateien an Ihrem Quellspeicherort ändern oder hinzufügen, erstellt die Bereitstellung die Website mit Ihren neuesten Dateien.

> **Note**
>
> Selbst wenn Sie Dateien aus dem Quell-Repository löschen, löscht die S3-Bereitstellungsaktion keine S3-Objekte, die gelöschten Dateien entsprechen.

In diesem Tutorial haben Sie zwei Optionen:

- Erstellen einer Pipeline, mit der eine statische Website für Ihren öffentlichen S3-Bucket bereitgestellt wird. In diesem Beispiel wird eine Pipeline mit einer AWS CodeCommit Quellaktion und einer Amazon S3 S3-Bereitstellungsaktion erstellt. Siehe Option 1: Statische Website-Dateien auf Amazon S3 bereitstellen.
- Erstellen Sie eine Pipeline, die TypeScript Beispielcode kompiliert JavaScript und das CodeBuild Ausgabeartefakt zur Archivierung in Ihrem S3-Bucket bereitstellt. In diesem Beispiel wird eine Pipeline mit einer Amazon S3 S3-Quellaktion, einer CodeBuild Build-Aktion und einer Amazon S3 S3-Bereitstellungsaktion erstellt. Siehe Option 2: Bereitstellen von erstellten Archivdateien aus einem S3-Quell-Bucket in Amazon S3.

> **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. AWSRessourcen 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. AWSRessourcen für regionsübergreifende Aktionen müssen sich in derselben AWS Region befinden, in der Sie die Aktion ausführen möchten. Weitere Informationen finden Sie unter Hinzufügen einer regionsübergreifenden Aktion in CodePipeline.

Option 1: Statische Website-Dateien auf Amazon S3 bereitstellen

In diesem Beispiel laden Sie die Vorlagendatei mit dem Beispiel für eine statische Website herunter, laden die Dateien in Ihr AWS CodeCommit-Repository hoch, erstellen Ihren Bucket und führen die Konfiguration für das Hosten durch. 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 in Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)erstellte AWS CodeCommit-Repository verwenden.
- 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: Push Quelldateien in dein 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 Repository in dein CodeCommit Repository zu übertragen:

```
git push
```

b. So verwenden Sie die CodeCommit Konsole, um Ihre Dateien hochzuladen:

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

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

iii. Wählen Sie Choose file (Datei auswählen) und navigieren Sie dann zu Ihrer Datei. Übernehmen Sie die Änderung, indem Sie Ihren Benutzernamen und Ihre E-Mail-Adresse eingeben. Wählen Sie Commit changes (Änderungen übernehmen) aus.

iv. Wiederholen Sie diesen Schritt für jede Datei, die Sie hochladen möchten.

Schritt 2: Erstellen der Pipeline

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

- Eine Quellstufe mit einer CodeCommit Aktion, bei der die Quellartefakte die Dateien für Ihre Website sind.
- Eine Bereitstellungsphase mit einer Amazon S3 S3-Bereitstellungsaktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyS3DeployPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.

5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie unter Step 2: Add source stage (Schritt 2: Quell-Phase hinzufügen) unter Source provider (Quell-Anbieter) die Option AWS CodeCommit. Wählen Sie 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.
8. Wählen Sie unter Step 3: Add build stage (Schritt 3: Build-Stufe hinzufügen) die Option Skip build stage (Build-Stufe überspringen) und akzeptieren Sie die Warnmeldung, indem Sie erneut auf Skip (Überspringen) klicken.

   Wählen Sie Weiter.
9. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

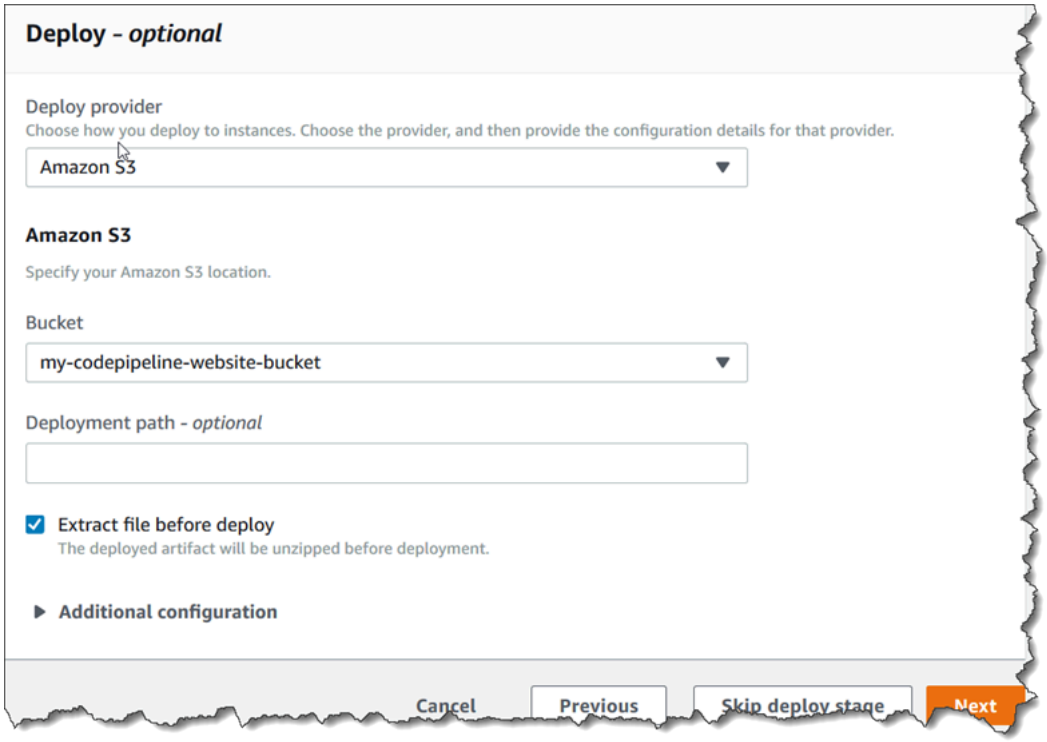
   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Amazon S3.

   b. Geben Sie unter Bucket den Namen Ihres öffentlichen Buckets ein.

   c. Wählen Sie Extract file before deploy (Datei vor Bereitstellung extrahieren).

   > **Note**
   >
   > Die Bereitstellung schlägt fehl, wenn Sie nicht Datei vor der Bereitstellung extrahieren auswählen. Dies liegt daran, dass die AWS CodeCommit-Aktion in Ihrer Pipeline Quellartefakte komprimiert und Ihre Datei eine ZIP-Datei ist.

   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.

10. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.

11. Nachdem Ihre Pipeline erfolgreich ausgeführt wurde, öffnen Sie die Amazon S3 S3-Konsole und überprüfen Sie, ob Ihre Dateien in Ihrem öffentlichen Bucket wie folgt angezeigt werden:

```
index.html
main.css
graphic.jpg
```

12. Greifen Sie auf Ihren Endpunkt zu, um die Website zu testen. Ihr Endpunkt hat das folgende Format: `http://`*`bucket-name`*`.s3-website-`*`region`*`.amazonaws.com/`.

    Beispielendpunkt: `http://my-bucket.s3-website-us-west-2.amazonaws.com/`.

Die Beispiel-Webseite wird angezeigt.

Schritt 3: Vornehmen einer Änderung an einer beliebigen Quelldatei und Überprüfen der Bereitstellung

Nehmen Sie eine Änderung an Ihren Quelldateien vor und übertragen Sie die Änderung per Push in Ihr Repository. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Überprüfen Sie, ob Ihre Website aktualisiert wurde.

Option 2: Bereitstellen von erstellten Archivdateien aus einem S3-Quell-Bucket in Amazon S3

Bei dieser Option kompilieren die Build-Befehle in Ihrer Build-Phase TypeScript Code in JavaScript Code und stellen die Ausgabe in Ihrem S3-Ziel-Bucket in einem separaten Ordner mit Zeitstempel bereit. Zunächst erstellen Sie TypeScript Code und eine buildspec.yml-Datei. Nachdem Sie die Quelldateien in einer ZIP-Datei kombiniert haben, laden Sie die Quell-ZIP-Datei in Ihren S3-Quell-Bucket hoch und verwenden einen CodeBuild Stagingbereich, um eine ZIP-Datei für eine erstellte Anwendung in Ihrem S3-Ziel-Bucket bereitzustellen. Der kompilierte Code wird in Ihrem Ziel-Bucket als Archiv aufbewahrt.

Voraussetzungen

Sie müssen bereits über Folgendes verfügen:

- Ein S3-Quell-Bucket. Sie können den Bucket verwenden, den Sie unter Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) erstellt haben.
- Ein S3-Ziel-Bucket. Siehe Hosten einer statischen Website auf Amazon S3. Stellen Sie sicher, dass Sie Ihren Bucket genauso erstellen AWS-Region wie die Pipeline, die Sie erstellen möchten.

> **Note**
>
> In diesem Beispiel wird die Bereitstellung von Dateien in einem privaten Bucket veranschaulicht. Aktivieren Sie für Ihren Ziel-Bucket nicht das Website-Hosting und fügen Sie keine Richtlinien an, die dazu führen, dass der Bucket öffentlich verfügbar ist.

Schritt 1: Erstellen und Hochladen von Quelldateien in Ihren S3-Quell-Bucket

In diesem Abschnitt erstellen Sie Ihre Quelldateien und laden sie in den Bucket hoch, der von der Pipeline für Ihre Quellstufe verwendet wird. Dieser Abschnitt enthält eine Anleitung zum Erstellen der folgenden Quelldateien:

- Eine `buildspec.yml` Datei, die für CodeBuild Build-Projekte verwendet wird.
- Die Datei `index.ts`.

So erstellen Sie die Datei „buildspec.yml“

- Erstellen Sie eine Datei mit dem Namen `buildspec.yml` und dem folgenden Inhalt. Diese Build-Befehle installieren TypeScript und verwenden den TypeScript Compiler, um den Code in `index.ts` Code umzuschreiben. JavaScript

```
version: 0.2

phases:
  install:
    commands:
      - npm install -g typescript
  build:
    commands:
      - tsc index.ts
artifacts:
  files:
    - index.js
```

So erstellen Sie die Datei „index.ts“

- Erstellen Sie eine Datei mit dem Namen `index.ts` und dem folgenden Inhalt.

```
interface Greeting {
    message: string;
}

class HelloGreeting implements Greeting {
    message = "Hello!";
}
```

```
function greet(greeting: Greeting) {
    console.log(greeting.message);
}

let greeting = new HelloGreeting();

greet(greeting);
```

So laden Sie Dateien in Ihren S3-Quell-Bucket hoch

1. Ihre Dateien sollten in Ihrem lokalen Verzeichnis wie folgt aussehen:

   ```
   buildspec.yml
   index.ts
   ```

   Zippen Sie die Dateien und geben Sie der Datei den Namen `source.zip`.
2. Wählen Sie in der Amazon S3 S3-Konsole für Ihren Quell-Bucket Upload aus. Wählen Sie Dateien hinzufügen und suchen Sie nach der von Ihnen erstellten ZIP-Datei.
3. Klicken Sie auf Hochladen. Diese Dateien 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 Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyS3DeployPipeline`** ein.
4. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
5. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
6. Wählen Sie in Step 2: Add source stage (Schritt 2: Quellstufe hinzufügen) unter Source provider (Quellanbieter) die Option Amazon S3. Wählen Sie unter Bucket den Namen Ihres Quell-Buckets aus. Geben Sie unter S3 object key (S3-Objektschlüssel) den Namen Ihrer ZIP-Quelldatei ein. Stellen Sie sicher, dass Sie die Dateierweiterung .zip angeben.

   Wählen Sie Weiter.
7. Unter Schritt 3: Build-Stufe hinzufügen:

   a. Wählen Sie unter Build provider (Build-Anbieter) die Option CodeBuild aus.
   b. Wählen Sie Create build project (Build-Projekt erstellen) aus. Auf der Seite Projekt erstellen:
   c. Geben Sie unter Project name (Projektname) einen Namen für dieses Build-Projekt ein.
   d. Wählen Sie für Environment (Umgebung) die Option Managed image (Verwaltetes Image) aus. Wählen Sie für Operating system (Betriebssystem) die Option Ubuntu aus.
   e. Wählen Sie unter Runtime (Laufzeit) die Option Standard aus. Wählen Sie für Runtime version (Laufzeitversion) aws/codebuild/standard:1.0 aus.
   f. Wählen Sie unter Image version (Abbildversion) die Option Always use the latest image for this runtime version (Immer aktuelles Abbild für diese Laufzeitversion verwenden).
   g. Wählen Sie unter Servicerolle Ihre CodeBuild Servicerolle aus, oder erstellen Sie eine.
   h. Wählen Sie unter Build specifications (Build-Spezifikationen) die Option Use a buildspec file (Eine buildspec-Datei verwenden).
   i. Wählen Sie Weiter zu CodePipeline. Eine Meldung mit dem Hinweis, ob die Erstellung des Projekts erfolgreich war, wird angezeigt.
   j. Wählen Sie Weiter.
8. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

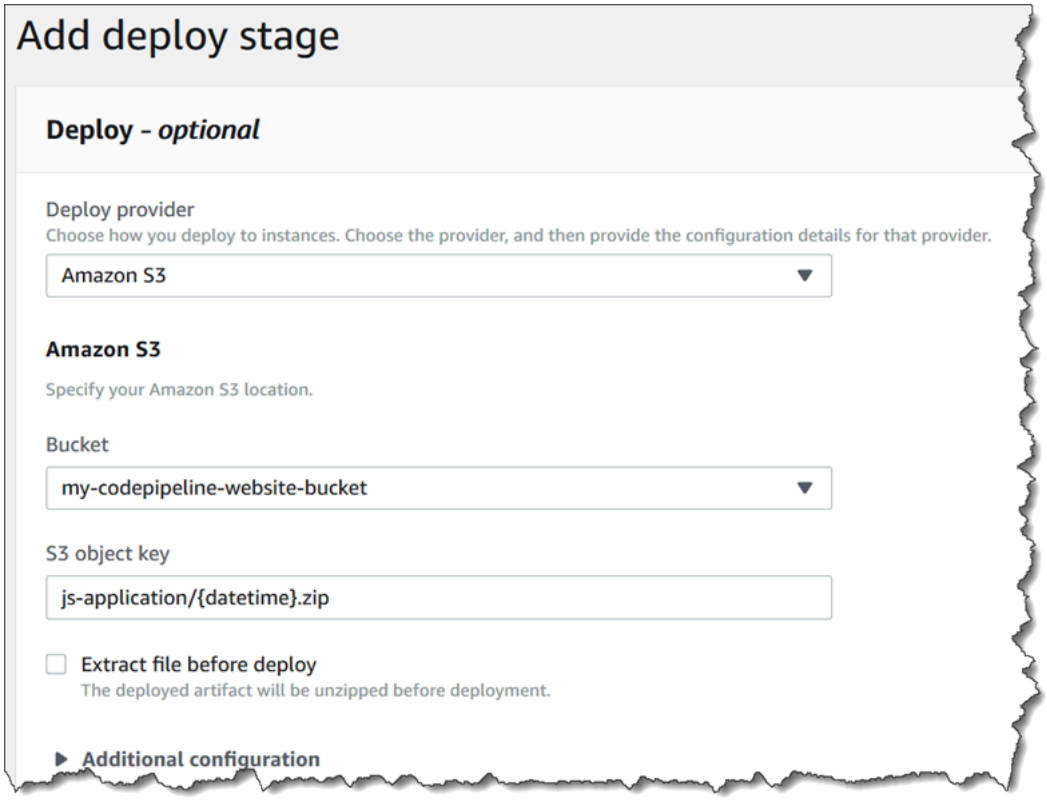
   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option Amazon S3.

b. Geben Sie unter Bucket den Namen Ihres S3-Ziel-Buckets ein.

c. Stellen Sie sicher, dass Extract file before deploy (Datei vor Bereitstellung extrahieren) deaktiviert ist.

   Wenn die Option Extract file before deploy (Datei vor Bereitstellung extrahieren) deaktiviert ist, wird die Option S3 object key (S3-Objektschlüssel) angezeigt. Geben Sie den Namen des gewünschten Pfads ein: `js-application/{datetime}.zip`

   Dadurch wird ein `js-application` Ordner in Amazon S3 erstellt, in den die Dateien extrahiert werden. In diesem Ordner wird bei Ausführung Ihrer Pipeline mit der Variablen `{datetime}` ein Zeitstempel für jede Ausgabedatei erstellt.



d. (Optional) Unter Canned ACL (Vordefinierte ACL) können Sie einen Satz vordefinierter Berechtigungen, als vordefinierte ACL bezeichnet, auf die hochgeladenen Artefakte anwenden.

e. (Optional) Geben Sie unter Cache control (Cache-Kontrolle) die Caching-Parameter ein. Sie können hiermit das Caching-Verhalten für Anforderungen/Antworten festlegen. Informationen zu gültigen Werten finden Sie im `Cache-Control`-Header-Feld für HTTP-Operationen.

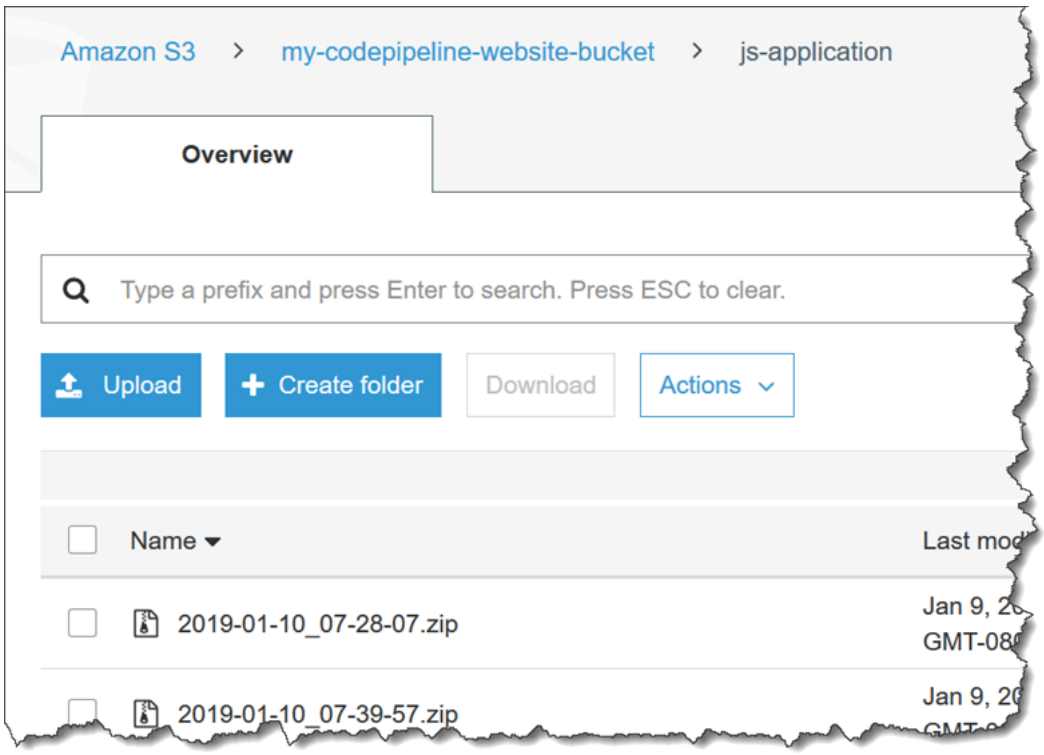
f. Wählen Sie Weiter.

9. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.
10. Nachdem Ihre Pipeline erfolgreich ausgeführt wurde, sehen Sie sich Ihren Bucket in der Amazon S3 S3-Konsole an. Vergewissern Sie sich, dass Ihre bereitgestellte ZIP-Datei in Ihrem Ziel-Bucket unter dem Ordner `js-application` angezeigt wird. Die in der JavaScript ZIP-Datei enthaltene Datei sollte sein`index.js`. Die Datei `index.js` enthält die folgende Ausgabe:

```
var HelloGreeting = /** @class */ (function () {
    function HelloGreeting() {
        this.message = "Hello!";
    }
    return HelloGreeting;
}());
function greet(greeting) {
    console.log(greeting.message);
}
var greeting = new HelloGreeting();
greet(greeting);
```

Schritt 3: Vornehmen einer Änderung an einer beliebigen Quelldatei und Überprüfen der Bereitstellung

Nehmen Sie eine Änderung an Ihren Quelldateien vor und laden Sie sie in Ihren Quell-Bucket hoch. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Zeigen Sie Ihren Ziel-Bucket an und vergewissern Sie sich, dass die bereitgestellten Ausgabedateien im Ordner `js-application` enthalten sind (wie hier dargestellt):



Tutorial: Erstellen einer Pipeline, die Ihre Serverless-Anwendung in veröffentlicht AWS Serverless Application Repository

Sie können verwenden AWS CodePipeline , um Ihre AWS SAM Serverless-Anwendung kontinuierlich an die zu übermitteln AWS Serverless Application Repository.

Dieses Tutorial zeigt, wie Sie eine Pipeline erstellen und konfigurieren, um Ihre Serverless-Anwendung zu erstellen, die in gehostet wird, GitHub und sie AWS Serverless Application Repository automatisch in der zu veröffentlichen. Die Pipeline verwendet GitHub als Quellanbieter und CodeBuild als Build-Anbieter. Um Ihre Serverless-Anwendung in der zu veröffentlichen AWS Serverless Application Repository, stellen Sie eine Anwendung bereit (aus der AWS Serverless Application Repository) und ordnen die von dieser Anwendung erstellte Lambda-Funktion als Invoke Action Provider in Ihrer Pipeline zu. Anschließend können Sie kontinuierlich Anwendungsaktualisierungen an die bereitstellen AWS Serverless Application Repository, ohne Code schreiben zu müssen.

> **Important**
>
> Viele der Aktionen, die Sie Ihrer Pipeline in diesem Verfahren hinzufügen, umfassen AWS Ressourcen, die Sie erstellen müssen, bevor Sie die pipeline. AWS resources für Ihre

Quellaktionen erstellen, müssen immer in derselben AWS Region erstellt werden, in der Sie Ihre Pipeline erstellen. Wenn Sie beispielsweise Ihre Pipeline 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 Hinzufügen einer regionsübergreifenden Aktion 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 verfügen über eine Serverless-Anwendung GitHub , die in gehostet wird und die Sie AWS Serverless Application Repository mithilfe der AWS SAM CLI in der veröffentlicht haben. Informationen zum AWS Serverless Application Repository Veröffentlichen einer Beispielanwendung in der finden Sie unter Schnellstart: Veröffentlichen von Anwendungen im AWS Serverless Application Repository -Entwicklerhandbuch. Informationen zum Veröffentlichen Ihrer eigenen Anwendung in der 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 Serverless-Anwendung hinzu. Ersetzen Sie *`template.yml`* durch die AWS SAM Vorlage Ihrer Anwendung und *`bucketname`* durch den S3-Bucket, in dem Ihre verpackte Anwendung gespeichert ist.

```
version: 0.2
phases:
  install:
    runtime-versions:
        python: 3.8
  build:
    commands:
```

```
      - sam package --template-file template.yml --s3-bucket bucketname --output-
template-file packaged-template.yml
artifacts:
  files:
    - packaged-template.yml
```

Schritt 2: Erstellen und Konfigurieren Ihrer Pipeline

Gehen Sie wie folgt vor, um Ihre Pipeline in der zu erstellen AWS-Region , in der Sie Ihre Serverless-Anwendung veröffentlichen möchten.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die - CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wechseln Sie bei Bedarf zu der , AWS-Region in der Sie Ihre Serverless-Anwendung veröffentlichen möchten.
3. Wählen Sie Create pipeline (Pipeline erstellen) aus. Geben Sie auf der Seite Choose pipeline settings (Auswahl der Pipelineeinstellungen) unter Pipeline name (Pipelinename) den Namen für Ihre Pipeline ein.
4. Wählen Sie im Pipeline-Typ V1 für die Zwecke dieses Tutorials aus. V1 Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in Bezug auf Merkmale und Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, damit eine Servicerolle in IAM CodePipeline erstellen kann.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie auf der Seite Quellstufe hinzufügen unter Quellanbieter die Option ausGitHub.
8. 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 unter GitHub Verbindungen.
9. Wählen Sie unter Repository Ihr GitHub Quell-Repository aus.
10. Wählen Sie unter Verzweigung Ihre GitHub Verzweigung aus.
11. Behalten Sie die verbleibenden Standardwerte für die Quellaktion bei. Wählen Sie Weiter aus.
12. Fügen Sie auf der Seite Add build stage (Build-Phase hinzufügen) eine Build-Phase hinzu:

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

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

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

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

e. Wählen Sie für Runtime (Laufzeit) und Runtime version (Laufzeitversion) die erforderliche Laufzeit und die Version für Ihre serverlose Anwendung.

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

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

h. Wählen Sie Weiter zu aus 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 zum Verwalten von AWS-Service Berechtigungen. Dieser Vorgang kann einige Minuten dauern.

i. Wählen Sie Weiter aus.

13. Wählen Sie auf der Seite Add deploy stage (Bereitstellungsphase hinzufügen) die Option Skip deploy stage (Bereitstellungsphase übergehen) und akzeptieren Sie dann die Warnmeldung, indem Sie erneut Skip (Übergehen) auswählen. Wählen Sie Weiter aus.

14. Wählen Sie Create pipeline (Pipeline erstellen) aus. Sie sollten ein Diagramm sehen, das Ihre Pipeline-Quell- und -Build-Stufen anzeigt.

15. Erteilen Sie der CodeBuild Servicerolle die Berechtigung für den Zugriff auf den S3-Bucket, 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 in Umgebung die CodeBuild Servicerolle aus, um die IAM-Konsole zu öffnen.

    d. Erweitern Sie die Auswahl für `CodeBuildBasePolicy` und wählen Sie Edit policy (Richtlinie bearbeiten).

    e. Wählen Sie JSON.

    f. Fügen Sie eine neue Richtlinienanweisung mit den folgenden Inhalten hinzu. Die Anweisung ermöglicht es CodeBuild , Objekte in den S3-Bucket zu legen, in dem Ihre verpackte Anwendung gespeichert ist. Ersetzen Sie *`bucketname (Bucket-Name)`* durch den Namen Ihres S3-Buckets.

```
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::bucketname/*"
            ],
            "Action": [
                "s3:PutObject"
            ]
        }
```

g. Wählen Sie Richtlinie prüfen.

h. Wählen Sie Änderungen speichern aus.

Schritt 3: Bereitstellen der Veröffentlichungsanwendung

Gehen Sie wie folgt vor, um die Anwendung bereitzustellen, die die Lambda-Funktion enthält, die die Veröffentlichung in der durchführt AWS Serverless Application Repository. Diese Anwendung ist aws-serverless-codepipeline-serverlessrepo-publish.

> **Note**
>
> Sie müssen die Anwendung in derselben AWS-Region wie Ihre 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 AWS CloudFormation Stack anzeigen, um die - AWS CloudFormation Konsole zu öffnen.
5. Erweitern Sie den Abschnitt Resources (Ressourcen). Sie sehen ServerlessRepoPublish, das vom Typ istAWS::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 ausBuildArtifact.
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 vor GitHub. Ändern Sie beispielsweise die Beschreibung der Anwendung im `Metadata` Abschnitt Ihrer AWS SAM Vorlagendatei. Übernehmen Sie die Änderung und übertragen Sie sie an Ihren GitHub Zweig. 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: Verwenden von Variablen mit Lambda-Aufrufaktionen

Eine Lambda-Aufrufaktion kann Variablen aus einer anderen Aktion als Teil ihrer Eingabe verwenden und neue Variablen zusammen mit ihrer Ausgabe zurückgeben. Informationen zu Variablen für Aktionen in CodePipeline finden Sie unter Variablen.

Am Ende dieses Tutorials werden Sie Folgendes haben::

- Eine Lambda-Aufrufaktion, die:

  - -Konsument`CommitId`-Variable aus einer CodeCommit-Quellaktion
  - 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 CodeCommit-Quelle inTutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)aus.
- Bearbeiten Sie Ihre vorhandene Pipeline so, dass die CodeCommit-Quellaktion einen Namespace aufweist. Weisen Sie der Aktion `SourceVariables` den Namespace zu.

Schritt 1: Erstellen einer Lambda-Funktion

Führen Sie die folgenden Schritte aus, um eine Lambda-Funktion und eine Lambda-Ausführungsrolle zu erstellen. Fügen Sie der Pipeline die Lambda-Aktion hinzu, nachdem Sie die Lambda-Funktion erstellt haben.

So erstellen Sie eine Lambda-Funktions- und Ausführungsrolle

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Lambda-Konsole an https://console.aws.amazon.com/lambda.
2. Wählen Sie Create function (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 Create function (Funktion erstellen).
6. Um eine Variable aus einer anderen Aktion zu verwenden, muss sie an die `UserParameters` in der Lambda-Aufrufaktionskonfiguration übergeben werden. Sie werden die Aktion in unserer Pipeline später in diesem Tutorial konfigurieren, Sie fügen jedoch den Code hinzu, vorausgesetzt, dass die Variable übergeben wird.

   ```
   const commitId =
   event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;
   ```

   Um neue Variablen zu erzeugen, setzen Sie eine Eigenschaft mit der Bezeichnung `outputVariables` auf die Eingabe zu `putJobSuccessResult`. Beachten Sie, dass Sie keine Variablen als Teil eines `putJobFailureResult` erzeugen können.

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

   Lassen Sie in Ihrer neuen Funktion Edit code inline (Code inline bearbeiten) ausgewählt, und fügen Sie den folgenden Beispielcode unter `index.js` ein.

   ```
   var AWS = require('aws-sdk');

   exports.handler = function(event, context) {
       var codepipeline = new AWS.CodePipeline();

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

       // Retrieve the value of UserParameters from the Lambda action configuration in
    CodePipeline,
       // in this case it is the Commit ID of the latest change of the pipeline.
   ```

```
    var params =
event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;

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

    // Notify CodePipeline of a successful job
    var putJobSuccess = function(message) {
        var params = {
            jobId: jobId,
            outputVariables: {
                testRunId: Math.floor(Math.random() * 1000).toString(),
                dateTime: Date(Date.now()).toString(),
                region: lambdaRegion
            }
        };
        codepipeline.putJobSuccessResult(params, function(err, data) {
            if(err) {
                context.fail(err);
            } else {
                context.succeed(message);
            }
        });
    };

    // Notify CodePipeline of a failed job
    var putJobFailure = function(message) {
        var params = {
            jobId: jobId,
            failureDetails: {
                message: JSON.stringify(message),
                type: 'JobFailed',
                externalExecutionId: context.invokeid
            }
        };
        codepipeline.putJobFailureResult(params, function(err, data) {
            context.fail(message);
        });
    };

    var sendResult = function() {
        try {
            console.log("Testing commit - " + params);
```

```
            // Your tests here

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

    sendResult();
};
```

7. Wählen Sie Save (Speichern) aus.
8. Kopieren Sie den Amazon-Ressourcenname (ARN) oben im Bildschirm.
9. Öffnen Sie als letzten SchrittAWS Identity and Access Management(IAM) -Konsole unterhttps://console.aws.amazon.com/iam/aus. Ändern Sie die Lambda-Ausführungsrolle, um die folgende Richtlinie hinzuzufügen: AWSCodePipelineCustomActionAccessaus. Die Schritte zum Erstellen einer Lambda-Ausführungsrolle oder zum Ändern der Rollenrichtlinie finden Sie unter Schritt 2: Erstellen einer 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.

So fügen Sie der Pipeline eine Lambda-Aktion und eine manuelle Genehmigungsaktion hinzu

1. Öffnen Sie die CodePipeline-Konsole unter https://console.aws.amazon.com/codepipeline.

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt. Wählen Sie die Pipeline aus, in der Sie die Aktion hinzufügen möchten.
2. Fügen Sie Ihrer Pipeline die Lambda-Testaktion hinzu.

   a. Um die Pipeline zu bearbeiten, wählen Sie Edit (Bearbeiten). Fügen Sie nach der Quellaktion einen Schritt in der vorhandenen Pipeline hinzu. Geben Sie einen Namen für die Phase ein, z. B. **`Test`**.

b. Wählen Sie in der neuen Phase das Symbol zum Hinzufügen einer Aktion aus. Geben Sie unter Action name (Aktionsname) den Namen der Aufrufaktion ein, z. B. **`Test_Commit`**.

c. In :Aktionsanbieter, wählenAWS Lambdaaus.

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

e. In :FunktionsnameWählen Sie den Namen der Lambda-Funktion, die Sie erstellt haben.

f. In :BenutzerparameterGeben Sie die Variablensyntax für die CodeCommit-Commit-ID ein. Dadurch wird die Ausgabevariable erstellt, die jedes Mal überprüft und genehmigt wird, wenn die Pipeline ausgeführt wird.

```
#{SourceVariables.CommitId}
```

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

h. Wählen Sie Done (Erledigt) aus.

3. Fügen Sie der Pipeline die manuelle Genehmigungsaktion hinzu.

a. Wenn sich Ihre Pipeline noch im Bearbeitungsmodus befindet, fügen Sie nach der Aufrufaktion eine Phase hinzu. Geben Sie einen Namen für die Phase ein, z. B. **`Approval`**.

b. Wählen Sie in der neuen Phase das Symbol zum Hinzufügen einer Aktion aus. Geben Sie unter Action name (Aktionsname) den Namen der Genehmigungsaktion ein, z. B. **`Change_Approval`**.

c. Wählen Sie unter Action provider (Aktionsanbieter) die Option Manual approval (Manuelle Genehmigung).

d. Erstellen Sie unter URL for review (URL zur Überprüfung) die URL, indem Sie die Variablensyntax für die `region`-Variable und die `CommitId`-Variable hinzufügen. Stellen Sie sicher, dass Sie die Namespaces verwenden, die denjenigen Aktionen zugewiesen sind, die die Ausgabevariablen bereitstellen.

In diesem Beispiel hat die URL mit der Variablensyntax für eine CodeCommit-Aktion den Standard-Namespace`SourceVariables`aus. 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 einerAWS Step FunctionsAufrufen einer Aktion in einer Pipeline

Sie können zum Erstellen und Konfigurieren von Zustandsautomaten AWS Step Functions verwenden. In diesem Tutorial erfahren Sie, wie Sie einer Pipeline eine Aufrufaktion hinzufügen, mit der die Ausführung von Zustandsautomaten über Ihre Pipeline aktiviert wird.

In diesem Tutorial führen Sie folgende Aufgaben aus:

- Sie erstellen eine Standardzustandsautomaten in AWS Step Functions.
- Sie geben die Eingabedaten für den Zustandsautomaten direkt im JSON-Format ein. Sie können die Eingabedatei für Zustandsautomaten auch in einen Amazon Simple Storage Service (Amazon S3) -Bucket hochladen.
- Sie aktualisieren Ihre Pipeline, indem Sie die Zustandsautomaten-Aktion hinzufügen.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong) und Europa (Mailand) nicht verfügbar. Weitere Informationen zu anderen verfügbaren Aktionen finden Sie unterProdukt- und Serviceintegrationen mit CodePipelineaus.

Themen

- Voraussetzung: Erstellen oder Auswählen einer einfachen Pipeline
- Schritt 1: Erstellen des Beispielzustandsautomaten
- Schritt 2: Aufrufen Step Functions -Aufrufaktion zu Ihrer Pipeline hinzufügen

Voraussetzung: Erstellen oder Auswählen einer einfachen Pipeline

In diesem Tutorial fügen Sie einer vorhandenen Pipeline eine Aufrufaktion hinzu. Sie können die Pipeline verwenden, die Sie in Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) oder Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository) 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 dieAWS Identity and Access Management(IAM) -Konsole suchen Sie die Rolle und fügen Sie dann die Berechtigungen zur Richtlinie der Rolle hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle .

Schritt 1: Erstellen des Beispielzustandsautomaten

Erstellen Sie in der Step Functions -Konsole einen Zustandsautomaten mit der`HelloWorld`Beispielvorlage. Detaillierte Anweisungen finden Sie unterErstellen eines ZustandsautomatenimAWS Step FunctionsEntwicklerhandbuchaus.

Schritt 2: Aufrufen Step Functions -Aufrufaktion zu Ihrer Pipeline hinzufügen

Fügen Sie Ihrer Pipeline wie folgt eine -Aufrufaktion hinzu:

1. Melden Sie sich beim anAWS Management Consoleund öffnen Sie die CodePipeline -Konsole unterhttp://console.aws.amazon.com/codesuite/codepipeline/homeaus.

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines 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 Delete (Löschen). 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. In :Aktionsanbieter, wählenAWSSchrittfunktionenaus. 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 Ausführung der Zustandsautomaten unterscheidet sich von dem Term, der in CodePipeline verwendet wird, um Eingastartefakte für Aktionen zu beschreiben.

Geben Sie in diesem Beispiel den folgenden JSON-Code ein:

```
{"IsHelloWorldExample": true}
```

15. Wählen Sie Done (Erledigt) aus.
16. Wählen Sie die Phase aus, die Sie bearbeiten, und wählen Sie dann Done (Fertig). Wählen Sie im AWS CodePipeline-Fenster Save (Speichern) und dann in der Warnmeldung Save (Speichern).
17. Um Ihre Änderungen zu übertragen und eine Pipelineausführung zu starten, wählen Sie Release change (Änderung freigeben) und dann Release (Freigeben).
18. Wählen Sie in der abgeschlossenen PipelineAWSSchrittfunktionenin Ihrer Aufrufen Aktion. Zeigen Sie in der AWS Step Functions-Konsole die Ausführungs-ID des Zustandsautomaten 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: Erstellen einer Pipeline mit AWS AppConfig als Bereitstellungsanbieter

In diesem Tutorial konfigurieren Sie eine Pipeline, die kontinuierlich Konfigurationsdateien bereitstellt und in Ihrer Bereitstellungsphase AWS AppConfig als Anbieter für Bereitstellungsaktionen verwendet wird.

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 eines S3-Buckets für Ihre Anwendung folgen, um einen S3-Bucket zu erstellen.

Schritt 1: Erstellen Sie Ihre Ressourcen AWS AppConfig

In diesem Abschnitt erstellen Sie die folgenden Ressourcen:

- Eine Anwendung AWS AppConfig ist eine logische Codeeinheit, die Ihren Kunden Funktionen zur Verfügung stellt.
- Eine Umgebung in AWS AppConfig ist eine logische Bereitstellungsgruppe von AppConfig Zielen, z. B. Anwendungen in einer Beta- oder Produktionsumgebung.
- Ein Konfigurationsprofil ist eine Sammlung von Einstellungen, die das Verhalten Ihrer Anwendung beeinflussen. Das Konfigurationsprofil ermöglicht AWS AppConfig den Zugriff auf Ihre Konfiguration an ihrem gespeicherten Speicherort.
- (Optional) Eine Bereitstellungsstrategie AWS AppConfig definiert das Verhalten einer Konfigurationsbereitstellung, z. B. wie viel Prozent der Clients die neu bereitgestellte Konfiguration zu einem bestimmten Zeitpunkt während einer Bereitstellung erhalten sollen.

Um eine Anwendung, eine Umgebung, ein Konfigurationsprofil und eine Bereitstellungsstrategie zu erstellen

1. Melden Sie sich an der AWS Management Console an.
2. Verwenden Sie die Schritte in den folgenden Themen, um Ihre Ressourcen in zu erstellen AWS AppConfig.

   - Erstellen Sie eine Anwendung.

- Erstellen Sie eine Umgebung.
- Erstellen Sie ein AWS CodePipeline Konfigurationsprofil.
- (Optional) Wählen Sie eine vordefinierte Bereitstellungsstrategie oder erstellen Sie Ihre eigene.

Schritt 2: Laden Sie Dateien in Ihren S3-Quell-Bucket hoch

Erstellen Sie in diesem Abschnitt Ihre Konfigurationsdatei (en). Dann komprimieren Sie Ihre Quelldateien und übertragen Sie sie in den Bucket, den die Pipeline für Ihre Quellphase verwendet.

Um Konfigurationsdateien zu erstellen

1. Erstellen Sie eine `configuration.json` Datei für jede Konfiguration in jeder Region. Fügen Sie den folgenden Inhalt hinzu:

   ```
   Hello World!
   ```

2. Gehen Sie wie folgt vor, um Ihre Konfigurationsdateien zu komprimieren und hochzuladen.

Um Quelldateien zu komprimieren und hochzuladen

1. Erstellen Sie eine .zip-Datei mit Ihren Dateien und geben Sie der ZIP-Datei einen Namen. `configuration-files.zip` Beispielsweise kann Ihre ZIP-Datei die folgende Struktur haben:

   ```
   .
   ### appconfig-configurations
       ### MyConfigurations
           ### us-east-1
           #   ### configuration.json
           ### us-west-2
               ### configuration.json
   ```

2. Wählen Sie in der Amazon S3 S3-Konsole für Ihren Bucket Upload und folgen Sie den Anweisungen, um Ihre ZIP-Datei hochzuladen.

Schritt 3: Erstellen Ihrer Pipeline

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

- Eine Quellstufe mit einer Amazon S3 S3-Aktion, bei der die Quellartefakte die Dateien für Ihre Konfiguration sind.
- Eine Bereitstellungsphase mit einer AppConfig Bereitstellungsaktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyAppConfigPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.
6. Belassen Sie die Einstellungen unter Erweiterte Einstellungen bei den Standardeinstellungen, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie in Step 2: Add source stage (Schritt 2: Quellstufe hinzufügen) unter Source provider (Quellanbieter) die Option Amazon S3. Wählen Sie in Bucket den Namen Ihres S3-Quell-Buckets aus.

   Geben Sie im Feld S3-Objektschlüssel den Namen Ihrer .zip-Datei ein:`configuration-files.zip`.

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

   Wählen Sie Weiter.
9. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

   a. Wählen Sie unter Deploy provider (Bereitstellungsanbieter) die Option AWS AppConfig aus.

b. Wählen Sie unter Anwendung den Namen der Anwendung aus, in AWS AppConfig der Sie sie erstellt haben. Das Feld zeigt die ID für Ihre Anwendung.

c. Wählen Sie unter Umgebung den Namen der Umgebung aus, in der Sie erstellt haben AWS AppConfig. Das Feld zeigt die ID für Ihre Umgebung.

d. Wählen Sie unter Konfigurationsprofil den Namen des Konfigurationsprofils aus, in dem Sie es erstellt haben AWS AppConfig. Das Feld zeigt die ID für Ihr Konfigurationsprofil.

e. Wählen Sie unter Bereitstellungsstrategie den Namen Ihrer Bereitstellungsstrategie aus. Dabei kann es sich entweder um eine Bereitstellungsstrategie handeln, die Sie in erstellt haben, AppConfig oder um eine, die Sie unter vordefinierten Bereitstellungsstrategien ausgewählt haben AppConfig. Das Feld zeigt die ID für Ihre Bereitstellungsstrategie.

f. Geben Sie im Feld Konfigurationspfad für das Eingabeartefakt den Dateipfad ein. Stellen Sie sicher, dass Ihr Konfigurationspfad für das Eingabeartefakt mit der Verzeichnisstruktur in Ihrer S3-Bucket-ZIP-Datei übereinstimmt. Geben Sie für dieses Beispiel den folgenden Dateipfad ein:. `appconfig-configurations/MyConfigurations/us-west-2/configuration.json`

g. Wählen Sie Weiter.

10. Prüfen Sie in Step 5: Review die Informationen und wählen Sie dann Create pipeline aus.

Schritt 4: Nehmen Sie eine Änderung an einer beliebigen Quelldatei vor und überprüfen Sie die Bereitstellung

Nehmen Sie eine Änderung an Ihren Quelldateien vor und laden Sie die Änderung in Ihren Bucket hoch. Damit wird die Ausführung Ihrer Pipeline ausgelöst. Überprüfen Sie anhand der Version, ob Ihre Konfiguration verfügbar ist.

Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden

Sie können die Option „Vollständiges Klonen“ für Ihre GitHub Quellaktion unter auswählen CodePipeline. Verwenden Sie diese Option, um CodeBuild Befehle für Git-Metadaten in Ihrer Pipeline-Build-Aktion auszuführen.

In diesem Tutorial erstellen Sie eine Pipeline, die eine Verbindung zu Ihrem GitHub Repository herstellt, die vollständige Klonoption für Quelldaten verwendet und einen CodeBuild Build ausführen, der Ihr Repository klont und Git-Befehle für das Repository ausführt.

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

Themen

- Voraussetzungen
- 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 verwendenAWS Management Console, um eine Verbindung herzustellen, werden Sie aufgefordert, sich mit Ihren GitHub Anmeldeinformationen anzumelden.

Schritt 1: Erstellen Sie eine README-Datei

Nachdem Sie Ihr GitHub Repository erstellt haben, gehen Sie wie folgt vor, um eine README-Datei hinzuzufügen.

1. Loggen Sie sich in Ihr GitHub Repository ein und wählen Sie Ihr Repository aus.
2. Um eine neue Datei zu erstellen, wähle Datei hinzufügen > Neue Datei erstellen. Benennen Sie die Datei`README.md`. Datei und fügen Sie den folgenden Text hinzu.

```
This is a GitHub repository!
```

3. Wählen Sie Commit changes (Änderungen übernehmen) aus.

   Die Datei `README.md` muss sich im Stammverzeichnis Ihres Repositorys befinden.

Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt

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

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

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich unter https://console.aws.amazon.com/codepipeline/ bei der CodePipeline Konsole an.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyGitHubPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Service role (Servicerolle) die Option New service role (Neue Servicerolle).

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

6. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie unter Artifact store (Artefaktspeicher) die Option Default location (Standardstandort) aus, um den

Standard-Artefakt-Speicherort für die Pipeline in der entsprechenden Region zu verwenden, beispielsweise mit dem Amazon S3-Artefakt-Bucket als Standard.

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

Wählen Sie Next (Weiter).

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

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

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

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

   d. Wählen Sie unter Branch-Name den Repository-Branch aus, den Sie verwenden möchten.

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

   f. Wählen Sie unter Ausgabeartefaktformat die Option Vollständiger Klon aus, um die Git-Clone-Option für das Quell-Repository zu aktivieren. Nur Aktionen, die von bereitgestellt werden, CodeBuild können die Git-Clone-Option verwenden. Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um Verbindungen zu verwendenIn diesem Tutorial erfahren Sie, wie Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, sodass Sie diese Option verwenden können.

   Wählen Sie Weiter.

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

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

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

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

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

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

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

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

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

> **Note**
>
> Stellen Sie im `env` Abschnitt der Build-Spezifikation sicher, dass der Credential-Helper für Git-Befehle aktiviert ist, wie in diesem Beispiel gezeigt.

```
version: 0.2

env:
  git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
      # name: version
    #commands:
      # - command
```

```
      # - command
  pre_build:
    commands:
      - ls -lt
      - cat README.md
  build:
    commands:
      - git log | head -100
      - git status
      - ls
      - git archive --format=zip HEAD > application.zip
  #post_build:
    #commands:
      # - command
      # - command
artifacts:
  files:
    - application.zip
    # - location
  #name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

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

i. Wählen Sie Weiter.

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

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

Schritt 3: Aktualisieren Sie die CodeBuild Servicerollenrichtlinie, um Verbindungen zu verwenden

Der erste Pipeline-Lauf schlägt fehl, da die CodeBuild Servicerolle mit Berechtigungen zur Verwendung von Verbindungen aktualisiert werden muss. Fügen Sie die `codestar-connections:UseConnection` IAM-Berechtigung zu Ihrer Servicerollenrichtlinie hinzu. Anweisungen zum Aktualisieren der Richtlinie in der IAM-Konsole finden Sie unter. Hinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder 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 „Vollständiges Klonen“ für Ihre CodeCommit Quellaktion unter auswählen CodePipeline. Verwenden Sie diese Option, um den Zugriff CodeBuild auf Git-Metadaten in Ihrer Pipeline-Build-Aktion zu ermöglichen.

In diesem Tutorial erstellen Sie eine Pipeline, die auf Ihr CodeCommit Repository zugreift, die Option Full Clone für Quelldaten verwendet und einen CodeBuild Build ausführt, der Ihr Repository klont und Git-Befehle für das Repository ausführt.

> **Note**
>
> CodeBuild Aktionen sind die einzigen Downstream-Aktionen, die die Verwendung von Git-Metadaten unterstützen, die mit der Git-Clone-Option verfügbar sind. Außerdem kann deine Pipeline kontoübergreifende Aktionen enthalten, aber die CodeCommit Aktion und die CodeBuild Aktion müssen sich im selben Konto befinden, damit die Option „Vollständiges Klonen“ erfolgreich ist.

Themen

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

Um eine README-Datei hinzuzufügen

1. Loggen Sie sich in Ihr Repository ein und wählen Sie Ihr Repository aus.
2. Um eine neue Datei zu erstellen, wähle Datei hinzufügen > Datei erstellen. Benennen Sie die Datei`README.md`. Datei und fügen Sie den folgenden Text hinzu.

   ```
   This is a CodeCommit repository!
   ```

3. Wählen Sie Commit changes (Änderungen übernehmen) aus.

Die Datei `README.md` muss sich im Stammverzeichnis Ihres Repositorys befinden.

Schritt 2: Erstellen Sie Ihre Pipeline und erstellen Sie ein Projekt

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

- Eine Quellphase mit einer CodeCommit Quellaktion.
- Eine Build-Phase mit einer AWS CodeBuild-Build-Aktion.

So erstellen Sie mit dem Assistenten eine Pipeline

1. Melden Sie sich unter https://console.aws.amazon.com/codepipeline/ bei der CodePipeline Konsole an.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyCodeCommitPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:
   - Wählen Sie Existing service role (Vorhandene Servicerolle).
   - Wählen Sie Ihre bestehende CodePipeline Servicerolle. Diese Rolle muss über die `codecommit:GetRepository` IAM-Berechtigung gemäß Ihrer Servicerollenrichtlinie verfügen. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline Servicerolle.
6. Lassen Sie die Standardwerte bei Erweiterte Einstellungen unverändert. Wählen Sie Weiter.
7. Gehen Sie auf der Seite Schritt 2: Quellstufe hinzufügen wie folgt vor:
   a. Wählen Sie unter Source provider (Quell-Anbieter) die Option CodeCommit.
   b. Wählen Sie unter Repository-Name den Namen Ihres Repositorys aus.
   c. Wählen Sie unter Branchname Ihren Branchennamen aus.
   d. Stellen Sie sicher, dass die Option Starten der Pipeline bei Änderung des Quellcodes ausgewählt ist.

e. Wählen Sie unter Ausgabeartefaktformat die Option Vollständiger Klon aus, um die Git-Clone-Option für das Quell-Repository zu aktivieren. Nur Aktionen, die von bereitgestellt werden, CodeBuild können die Git-Clone-Option verwenden.

Wählen Sie Weiter.

8. Gehen Sie in der Phase Build hinzufügen wie folgt vor:

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

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

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

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

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

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

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

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

```
version: 0.2

env:
  git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
```

```
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
      # name: version
    #commands:
      # - command
      # - command
  pre_build:
    commands:
      - ls -lt
      - cat README.md
  build:
    commands:
      - git log | head -100
      - git status
      - ls
      - git describe --all
  #post_build:
    #commands:
      # - command
      # - command
#artifacts:
  #files:
    # - location
  #name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

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

i. Wählen Sie Weiter.

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

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

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

Der erste Pipeline-Lauf schlägt fehl, da Sie die CodeBuild Servicerolle mit den Berechtigungen zum Abrufen aus Ihrem Repository aktualisieren müssen.

Fügen Sie die `codecommit:GitPull` IAM-Berechtigung zu Ihrer Servicerollenrichtlinie hinzu. Anweisungen zum Aktualisieren der Richtlinie in der IAM-Konsole finden Sie unter. Hinzufügen von CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen

Schritt 4: Repository-Befehle in der Build-Ausgabe anzeigen

Um die Build-Ausgabe anzuzeigen

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

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

   Die Befehle geben den `README.md` Dateiinhalt aus, listen die Dateien im Verzeichnis auf, klonen das Repository, zeigen das Protokoll an und werden ausgeführt`git describe --all`.

Tutorial: Eine Pipeline mit AWS CloudFormation StackSets Bereitstellungsaktionen erstellen

In diesem Tutorial verwenden Sie die AWS CodePipeline Konsole, um eine Pipeline mit Bereitstellungsaktionen zum Erstellen eines Stack-Sets und zum Erstellen von Stack-Instances zu erstellen. Wenn die Pipeline ausgeführt wird, erstellt die Vorlage ein Stack-Set und erstellt und aktualisiert auch die Instanzen, in denen das Stack-Set bereitgestellt wird.

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

Um Stacksets am effektivsten einsetzen zu können CodePipeline, sollten Sie ein klares Verständnis der dahinterstehenden Konzepte AWS CloudFormation StackSets und ihrer Funktionsweise haben. Weitere Informationen zu den StackSets Konzepten finden Sie im AWS CloudFormationBenutzerhandbuch.

> **Note**
>
> Die `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und sind in den Regionen Asien-Pazifik (Hongkong), Europa (Zürich), Europa (Mailand), Afrika (Kapstadt) und Naher Osten (Bahrain) nicht verfügbar. Weitere verfügbare Aktionen finden Sie unterProdukt- und Serviceintegrationen mit CodePipeline.

Themen

- Voraussetzungen
- Schritt 1: Laden Sie die AWS CloudFormation Beispielvorlage und die Parameterdatei hoch
- Schritt 2: Erstellen der Pipeline
- Schritt 3: Erste Bereitstellung anzeigen
- Schritt 4: Fügen Sie eine CloudFormationStackInstances Aktion hinzu
- Schritt 5: Sehen Sie sich die Stackset-Ressourcen für Ihre Bereitstellung an
- Schritt 6: Nehmen Sie ein Update für Ihr Stack-Set vor

Voraussetzungen

Für Stack-Set-Operationen verwenden Sie zwei verschiedene Konten: ein Administratorkonto und ein Zielkonto. Sie erstellen Stack-Sets im Administratorkonto. Sie erstellen einzelne Stacks, die zu einem Stack-Set im Zielkonto gehören.

Um eine Administratorrolle mit Ihrem Administratorkonto zu erstellen

- Folgen Sie den Anweisungen unter Grundberechtigungen für Stack-Set-Operationen einrichten. Ihre Rolle muss benannt werden **`AWSCloudFormationStackSetAdministrationRole`**.

Um eine Servicerolle im Zielkonto zu erstellen

- Erstellen Sie eine Servicerolle im Zielkonto, die dem Administratorkonto vertraut. Folgen Sie den Anweisungen unter Grundberechtigungen für Stack-Set-Operationen einrichten. Ihre Rolle muss benannt werden **`AWSCloudFormationStackSetExecutionRole`**.

Schritt 1: Laden Sie die AWS CloudFormation Beispielvorlage und die Parameterdatei hoch

Erstellen Sie einen Quell-Bucket für Ihre Stackset-Vorlagen- und Parameterdateien. Laden Sie die AWS CloudFormation Beispielvorlagendatei herunter, richten Sie eine Parameterdatei ein und komprimieren Sie dann die Dateien, bevor Sie sie in Ihren S3-Quell-Bucket hochladen.

> **Note**
>
> Stellen Sie sicher, dass Sie die Quelldateien komprimieren, bevor Sie sie in Ihren S3-Quell-Bucket hochladen, auch wenn die einzige Quelldatei die Vorlage ist.

Um einen S3-Quell-Bucket zu erstellen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Create Bucket (Bucket erstellen).
3. Geben Sie in Bucketname einen Namen für Ihren Bucket ein.

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

Um die AWS CloudFormation Vorlagendatei zu erstellen

1. Laden Sie die folgende Beispielvorlagendatei zum Generieren der CloudTrail Konfiguration für Stack-Sets herunter:https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml.
2. Speichern Sie die Datei als `template.yml`.

Um die Datei parameters.txt zu erstellen

1. Erstellen Sie eine Datei mit den Parametern für Ihre Bereitstellung. Parameter sind Werte, die Sie zur Laufzeit in Ihrem Stack aktualisieren möchten. Die folgende Beispieldatei aktualisiert die Vorlagenparameter für Ihr Stack-Set, um die Protokollierung von Validierungen und globalen Ereignissen zu ermöglichen.

```
[
  {
    "ParameterKey": "EnableLogFileValidation",
    "ParameterValue": "true"
  },
  {
    "ParameterKey": "IncludeGlobalEvents",
    "ParameterValue": "true"
  }
]
```

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

Um die Datei accounts.txt zu erstellen

1. Erstellen Sie eine Datei mit den Konten, für die Sie Instanzen erstellen möchten, wie in der folgenden Beispieldatei gezeigt.

```
[
    "111111222222","333333444444"
]
```

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

Um Quelldateien zu erstellen und hochzuladen

1. Kombinieren Sie die Dateien zu einer einzigen ZIP-Datei. Ihre Dateien sollten in Ihrer ZIP-Datei so aussehen.

   ```
   template.yml
   parameters.txt
   accounts.txt
   ```

2. Laden Sie die ZIP-Datei in Ihren S3-Bucket hoch. Diese Datei ist das Quellartefakt, das vom Assistenten „Pipeline erstellen“ für Ihre Bereitstellungsaktion in CodePipeline erstellt wurde.

Schritt 2: Erstellen der Pipeline

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

- Eine Quellstufe mit einer S3-Quellaktion, bei der das Quellartefakt Ihre Vorlagendatei und alle unterstützenden Quelldateien ist.
- Eine Bereitstellungsphase mit einer AWS CloudFormation Stack-Set-Bereitstellungsaktion, die das Stack-Set erstellt.
- Eine Bereitstellungsphase mit einer Bereitstellungsaktion für AWS CloudFormation Stack-Instances, die die Stacks und Instances innerhalb der Zielkonten erstellt.

Um eine Pipeline mit einer CloudFormationStackSet Aktion zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie auf der Seite Welcome (Willkommen) die Option Getting started (Erste Schritte) oder auf der Seite Pipelines die Option Create pipeline (Pipeline erstellen).
3. Geben Sie unter Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipeline-Einstellungen) unter Pipeline name (Pipeline-Name) **`MyStackSetsPipeline`** ein.
4. Wählen Sie im Feld Pipeline-Typ die Option V1 für die Zwecke dieses Tutorials aus. Sie können auch V2 wählen. Beachten Sie jedoch, dass sich die Pipeline-Typen in ihren Eigenschaften und im Preis unterscheiden. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, um CodePipeline die Erstellung einer Servicerolle in IAM zu ermöglichen.

6. Behalten Sie im Artifact Store die Standardeinstellungen bei.

   > **Note**
   >
   > Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket. Wenn Sie eine Pipeline erstellen oder bearbeiten, benötigen Sie einen Artefakt-Bucket in der Pipeline-Region und einen Artefakt-Bucket pro AWS Region, in der Sie eine Aktion ausführen.
   > Weitere Informationen finden Sie unter Eingabe- und Ausgabe-Artefakte und CodePipeline Referenz zur Pipeline-Struktur.

   Wählen Sie Weiter.

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

8. Geben Sie im Feld Bucket den S3-Quell-Bucket ein, den Sie für dieses Tutorial erstellt haben, z. B. `BucketName` Geben Sie im Feld S3-Objektschlüssel den Dateipfad und den Dateinamen für Ihre ZIP-Datei ein, z. `MyFiles.zip` B.

9. Wählen Sie Weiter.

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

    Wählen Sie Weiter.

11. Unter Step 4: Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen):

    a. Wählen Sie unter Deploy Provider die Option AWS CloudFormationStack Set aus.

    b. Geben Sie im Feld Name des Stack-Sets einen Namen für das Stack-Set ein. Dies ist der Name des Stack-Sets, das die Vorlage erstellt.

       > **Note**
       >
       > Notieren Sie sich den Namen Ihres Stack-Sets. Sie werden ihn verwenden, wenn Sie Ihrer Pipeline die zweite StackSets Bereitstellungsaktion hinzufügen.

c. Geben Sie im Feld Vorlagenpfad den Namen des Artefakts und den Dateipfad ein, in den Sie Ihre Vorlagendatei hochgeladen haben. Geben Sie beispielsweise Folgendes ein und verwenden Sie dabei den Standardnamen des Quellartefakts. `SourceArtifact`

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

d. Geben Sie unter Bereitstellungsziele den Artefaktnamen und den Dateipfad ein, in den Sie Ihre Kontodatei hochgeladen haben. Geben Sie beispielsweise Folgendes ein und verwenden Sie dabei den Standardnamen des Quellartefakts. `SourceArtifact`

```
SourceArtifact::accounts.txt
```

e. Geben Sie im Feld Bereitstellungsziel AWS-Regionen eine Region für die Bereitstellung Ihrer ersten Stack-Instance ein, z. B. `us-east-1`

f. Erweitern Sie die Bereitstellungsoptionen. Geben Sie unter Parameter den Artefaktnamen und den Dateipfad ein, in den Sie Ihre Parameterdatei hochgeladen haben. Geben Sie beispielsweise Folgendes ein und verwenden Sie dabei den Standardnamen des Quellartefakts. `SourceArtifact`

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

Um die Parameter als Literaleingabe und nicht als Dateipfad einzugeben, geben Sie Folgendes ein:

```
ParameterKey=EnableLogFileValidation,ParameterValue=true
ParameterKey=IncludeGlobalEvents,ParameterValue=true
```

g. Wählen Sie unter Capabilities die Optionen CAPABILITY_IAM und CAPABILITY_NAMED_IAM aus.

h. Wählen Sie im Berechtigungsmodell die Option SELF_MANAGED aus.

i. Geben Sie im Feld Prozentsatz der Fehlertoleranz den Wert ein. `20`

j. Geben `25` Sie im Feld Maximaler Prozentsatz gleichzeitig den Wert ein.

k. Wählen Sie Weiter.

l. Wählen Sie Create pipeline (Pipeline erstellen) aus. Ihre Pipeline wird angezeigt.

m. Lassen Sie die Ausführung Ihrer Pipeline zu.

Schritt 3: Erste Bereitstellung anzeigen

Sehen Sie sich die Ressourcen und den Status Ihrer ersten Bereitstellung an. Nachdem Sie überprüft haben, ob das Deployment Ihr Stack-Set erfolgreich erstellt hat, können Sie die zweite Aktion zu Ihrer Bereitstellungsphase hinzufügen.

Um die Ressourcen anzusehen

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
3. Wählen Sie die AWS CloudFormation Aktion für die CloudFormationStackSetAktion in Ihrer Pipeline aus. Die Vorlage, die Ressourcen und Ereignisse für Ihr Stack-Set werden in der AWS CloudFormation Konsole angezeigt.
4. Wählen Sie im linken Navigationsbereich StackSets. Wählen Sie in der Liste das neue Stack-Set aus.
5. Wählen Sie den Tab Stack-Instances. Stellen Sie sicher, dass für jedes von Ihnen angegebene Konto eine Stack-Instance in der Region us-east-1 erstellt wurde. Stellen Sie sicher, dass der Status für jede Stack-Instance lautet. `CURRENT`

Schritt 4: Fügen Sie eine CloudFormationStackInstances Aktion hinzu

Erstellen Sie eine nächste Aktion in Ihrer Pipeline, mit der Sie AWS CloudFormation StackSets die verbleibenden Stack-Instanzen erstellen können.

Um eine nächste Aktion in Ihrer Pipeline zu erstellen

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.

   Wählen Sie unter Pipelines Ihre Pipeline und dann View (Anzeigen) aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
2. Wählen Sie diese Option, um die Pipeline zu bearbeiten. Die Pipeline wird im Bearbeitungsmodus angezeigt.
3. Wählen Sie in der Bereitstellungsphase die Option Bearbeiten aus.
4. Wählen Sie unter der Aktion AWS CloudFormationStack Set Deploy die Option Aktionsgruppe hinzufügen aus.

5. Fügen Sie auf der Seite Aktion bearbeiten die Aktionsdetails hinzu:

   a. Geben Sie im Feld Aktionsname einen Namen für die Aktion ein.

   b. Wählen Sie unter Aktionsanbieter die Option AWS CloudFormationStack Instances aus.

   c. Wählen Sie unter Eingabeartefakte die Option aus SourceArtifact.

   d. Geben Sie im Feld Name des Stack-Sets den Namen für das Stack-Set ein. Dies ist der Name des Stack-Sets, das Sie in der ersten Aktion angegeben haben.

   e. Geben Sie unter Bereitstellungsziele den Namen des Artefakts und den Dateipfad ein, in den Sie Ihre Kontodatei hochgeladen haben. Geben Sie beispielsweise Folgendes ein und verwenden Sie dabei den Standardnamen des Quellartefakts. `SourceArtifact`

   ```
   SourceArtifact::accounts.txt
   ```

   f. Geben Sie im Feld Bereitstellungsziel AWS-Regionen die Regionen für die Bereitstellung Ihrer verbleibenden Stack-Instances ein, z. B. `us-east-2` und `eu-central-1` wie folgt:

   ```
   us-east2, eu-central-1
   ```

   g. Geben Sie im Feld Prozentsatz der Fehlertoleranz den Wert ein`20`.

   h. Geben `25` Sie im Feld Maximaler Prozentsatz gleichzeitig den Wert ein.

   i. Wählen Sie Speichern aus.

   j. . Geben Sie eine Änderung manuell frei. Ihre aktualisierte Pipeline wird in der Bereitstellungsphase mit zwei Aktionen angezeigt.

Schritt 5: Sehen Sie sich die Stackset-Ressourcen für Ihre Bereitstellung an

Sie können die Ressourcen und den Status für Ihre Stack-Set-Bereitstellung einsehen.

Um die Ressourcen anzusehen

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie unter Pipelines Ihre Pipeline und dann Ansicht aus. Das Diagramm zeigt die Quell- und Bereitstellungsphase Ihrer Pipeline an.
3. Wählen Sie die AWS CloudFormation Aktion für die **`AWS CloudFormation Stack Instances`**Aktion in Ihrer Pipeline aus. Die Vorlage, die Ressourcen und Ereignisse für Ihr Stack-Set werden in der AWS CloudFormation Konsole angezeigt.

4. Wählen Sie im linken Navigationsbereich StackSets. Wählen Sie in der Liste Ihr Stack-Set aus.
5. Wählen Sie den Tab Stack-Instances. Stellen Sie sicher, dass alle verbleibenden Stack-Instances für jedes von Ihnen angegebene Konto in den erwarteten Regionen erstellt oder aktualisiert wurden. Stellen Sie sicher, dass der Status für jede Stack-Instance lautet`CURRENT`.

Schritt 6: Nehmen Sie ein Update für Ihr Stack-Set vor

Nehmen Sie ein Update für Ihr Stack-Set vor und stellen Sie das Update für Instances bereit. In diesem Beispiel ändern Sie auch die Bereitstellungsziele, die Sie für das Update festlegen möchten. Die Instanzen, die nicht Teil des Updates sind, erhalten einen veralteten Status.

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie unter Pipelines Ihre Pipeline aus und klicken Sie dann auf Bearbeiten. Wählen Sie in der Bereitstellungsphase die Option Bearbeiten aus.
3. Wählen Sie, ob Sie die Aktion AWS CloudFormationStack Set in Ihrer Pipeline bearbeiten möchten. Überschreiben Sie unter Beschreibung die bestehende Beschreibung mit einer neuen Beschreibung für das Stack-Set.
4. Wählen Sie, ob Sie die Aktion AWS CloudFormationStack Instances in Ihrer Pipeline bearbeiten möchten. Löschen Sie im Feld Bereitstellungsziel AWS-Regionen den `us-east-2` Wert, der bei der Erstellung der Aktion eingegeben wurde.
5. Speichern Sie die Änderungen. Wählen Sie Änderung veröffentlichen, um Ihre Pipeline auszuführen.
6. Öffnen Sie Ihre Aktion inAWS CloudFormation. Wählen Sie die Registerkarte „StackSet Informationen“. 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`

CodePipeline Bewährte Methoden und Anwendungsfälle

In den folgenden Abschnitten werden bewährte Methoden für beschrieben CodePipeline.

Themen

- Anwendungsfälle für CodePipeline

Anwendungsfälle für CodePipeline

Sie können Pipelines erstellen, die sich in andere integrieren lassen AWS-Services. Dabei kann es sich um handeln AWS-Services, z. B. Amazon S3, oder um Produkte von Drittanbietern, wie z. B. GitHub. Dieser Abschnitt enthält Beispiele für die Verwendung von CodePipeline zur Automatisierung Ihrer Codeversionen mithilfe verschiedener Produktintegrationen. Eine vollständige Liste der Integrationen mit , die nach Aktionstyp CodePipeline organisiert sind, finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

Themen

- Verwenden von CodePipeline mit Amazon S3 AWS CodeCommit, und AWS CodeDeploy
- Verwenden von CodePipeline mit Aktionsanbietern von Drittanbietern (GitHub und Jenkins)
- Verwenden von CodePipeline mit AWS CodeStar zum Erstellen einer Pipeline in einem Codeprojekt
- Verwenden von CodePipeline zum Kompilieren, Erstellen und Testen von Code mit CodeBuild
- Verwenden von CodePipeline mit Amazon ECS für die kontinuierliche Bereitstellung von containerbasierten Anwendungen in der Cloud
- Verwenden von CodePipeline mit Elastic Beanstalk für die kontinuierliche Bereitstellung von Webanwendungen in der Cloud
- Verwenden von CodePipeline mit AWS Lambda für die kontinuierliche Bereitstellung von Lambda-basierten und Serverless-Anwendungen
- Verwenden von CodePipeline mit - AWS CloudFormation Vorlagen für die kontinuierliche Bereitstellung in der Cloud

Verwenden von CodePipeline mit Amazon S3 AWS CodeCommit, und AWS CodeDeploy

Wenn Sie eine Pipeline erstellen, CodePipeline ist in AWS Produkte und Services integriert, die in jeder Phase Ihrer Pipeline als Aktionsanbieter fungieren. 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 in Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) helfen Ihnen, den Assistenten zum Erstellen einer Pipeline mit zwei Standardphasen zu verwenden: „Quelle“ und „Staging“, wobei Ihr Amazon S3-Repository der Quellanbieter ist. In diesem Tutorial wird eine Pipeline erstellt, die verwendet, AWS CodeDeploy um eine Beispielanwendung aus einem Amazon S3-Bucket auf Amazon EC2-Instances bereitzustellen, auf denen Amazon Linux ausgeführt wird.
- Die Schritte in Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository) helfen Ihnen dabei, den Assistenten zum Erstellen einer Pipeline mit einer „Quellphase“ zu verwenden, die Ihr AWS CodeCommit Repository als Quellanbieter verwendet. In diesem Tutorial wird eine Pipeline erstellt, die verwendet, AWS CodeDeploy um eine Beispielanwendung aus einem - AWS CodeCommit Repository auf einer Amazon EC2-Instance bereitzustellen, auf der Amazon Linux ausgeführt wird.

Verwenden von CodePipeline mit Aktionsanbietern von Drittanbietern (GitHub und Jenkins)

Sie können Pipelines erstellen, die in Produkte von Drittanbietern wie GitHub und Jenkins integriert werden können. Die Schritte in Tutorial: Erstellen einer vierstufigen Pipeline zeigen, wie Sie eine Pipeline erstellen, die:

- Ruft Quellcode aus einem GitHub Repository ab,
- Jenkins verwendet, um den Quellcode zu erstellen und zu testen,
- Verwendet AWS CodeDeploy , um den erstellten und getesteten Quellcode auf Amazon EC2-Instances bereitzustellen, auf denen Amazon Linux oder Microsoft Windows Server ausgeführt wird.

Verwenden von CodePipeline mit AWS CodeStar zum Erstellen einer Pipeline in einem Codeprojekt

AWS CodeStar ist ein cloudbasierter Service, der eine einheitliche Benutzeroberfläche für die Verwaltung von Softwareentwicklungsprojekten auf AWS. AWS CodeStar works mit bereitstellt, CodePipeline um AWS Ressourcen zu einer Projektentwicklungstoolchain zu kombinieren. Sie können Ihr AWS CodeStar Dashboard verwenden, um die Pipeline, Repositorys, den Quellcode, Build-Spezifikationsdateien, die Bereitstellungsmethode und die Hosting-Instances oder Serverless-Instances, die für ein vollständiges Codeprojekt erforderlich sind, automatisch zu erstellen.

Um Ihr AWS CodeStar Projekt zu erstellen, wählen Sie Ihre Codierungssprache und den Typ der Anwendung aus, die Sie bereitstellen möchten. Sie können die folgenden Projekttypen erstellen: eine Webanwendung, einen Web-Service oder eine Alexa Skill.

Sie können Ihre bevorzugte IDE jederzeit in Ihr AWS CodeStar Dashboard integrieren. Sie können außerdem Teammitglieder hinzufügen und entfernen sowie Berechtigungen für Teammitglieder in Ihrem Projekt verwalten. Ein Tutorial, das Ihnen zeigt, wie Sie verwenden, AWS CodeStar um eine Beispiel-Pipeline für eine Serverless-Anwendung zu erstellen, finden Sie unter Tutorial: Erstellen und Verwalten eines Serverless-Projekts in AWS CodeStar.

Verwenden von CodePipeline zum Kompilieren, Erstellen und Testen von Code 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 mit , CodeBuild um laufende Revisionen über die Pipeline für die kontinuierliche Bereitstellung von Software-Builds zu automatisieren, wenn sich der Quellcode ändert. Weitere Informationen finden Sie unter Verwenden von CodePipeline mit , CodeBuild um Code zu testen und Builds auszuführen.

Verwenden von CodePipeline mit Amazon ECS für die kontinuierliche Bereitstellung von containerbasierten 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 CodePipeline mit Amazon ECS, um laufende Revisionen über die Pipeline für die kontinuierliche Bereitstellung von containerbasierten Anwendungen zu automatisieren, wenn sich das Quell-Image-Repository ändert. Weitere Informationen finden Sie im Tutorial: Kontinuierliche Bereitstellung mit CodePipeline.

Verwenden von CodePipeline mit Elastic Beanstalk für die kontinuierliche Bereitstellung von Webanwendungen in der Cloud

Elastic Beanstalk ist ein Datenverarbeitungsservice, mit dem Sie Webanwendungen und Services auf Webservern bereitstellen können. Verwenden Sie CodePipeline mit Elastic Beanstalk für die kontinuierliche Bereitstellung von Webanwendungen in Ihrer Anwendungsumgebung. Sie können auch verwenden AWS CodeStar , um eine Pipeline mit einer Elastic Beanstalk-Bereitstellungsaktion zu erstellen.

Verwenden von CodePipeline mit AWS Lambda für die kontinuierliche Bereitstellung von Lambda-basierten und Serverless-Anwendungen

Sie können AWS Lambda mit verwenden, um eine CodePipeline - AWS Lambda Funktion aufzurufen, wie unter Bereitstellen von Serverless-Anwendungen beschrieben. Sie können auch AWS Lambda und verwenden AWS CodeStar , um eine Pipeline für die Bereitstellung von Serverless-Anwendungen zu erstellen.

Verwenden von CodePipeline mit - AWS CloudFormation Vorlagen für die kontinuierliche Bereitstellung in der Cloud

Sie können AWS CloudFormation mit CodePipeline für kontinuierliche Bereitstellung und Automatisierung verwenden. Weitere Informationen finden Sie unter Kontinuierliche Bereitstellung mit CodePipeline. AWS CloudFormation wird auch verwendet, um die Vorlagen für Pipelines zu erstellen, die in erstellt wurden AWS CodeStar.

Markieren von Ressourcen

Ein Tag ist eine benutzerdefinierte Attributbezeichnung, die Sie oder AWS einer AWS-Ressource 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.

Tags helfen Ihnen, Ihre AWS-Ressourcen zu identifizieren und zu organisieren. Viele AWS-Services unterstützen das Markieren mit Tags (kurz: Tagging). So können Ressourcen aus verschiedenen Services das gleiche Tag zuweisen, um anzugeben, dass die Ressourcen zusammengehören. Sie können beispielsweise das gleiche Tag einer Pipeline zuweisen, das Sie einem Amazon S3 S3-Quellbucket 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 CodePipeline
- Markieren einer benutzerdefinierten Aktion in CodePipeline

Sie können dieAWS CLI CodePipeline APIs oderAWS 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 Ressourcen mit Tags können Sie Tags in IAM-Richtlinien verwenden, um zu steuern, 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 CodePipeline-Ressourcen.

Verwenden Sie CodePipeline mit Amazon Virtual Private Cloud

AWS CodePipelineunterstützt jetzt jetzt Amazon-Virtual-Private-Cloud-Endpunkte (Amazon VPC), die von bereitgestellt werden AWS PrivateLink. Das bedeutet, dass Sie eine direkte Verbindung CodePipeline über einen privaten Endpunkt in Ihrer VPC herstellen können, sodass der gesamte Datenverkehr innerhalb Ihrer VPC und imAWS Netzwerk bleibt.

Amazon VPC ist einAWS-Service, den Sie verwenden können, umAWS -Ressourcen in einem von Ihnen definierten virtuellen Netzwerk auszuführen. Mit einer VPC haben Sie die Kontrolle über Ihre Netzwerkeinstellungen, z. B.:

- IP-Adressbereich
- Subnetze
- Routing-Tabellen
- Netzwerk-Gateways

Schnittstellen-VPC-Endpunkte werden über eineAWS -Technologie bereitgestelltAWS PrivateLink, die eine private KommunikationAWS-Services unter Verwendung einer elastic network interface mit privaten IP-Adressen ermöglicht. Zum Herstellen einer Verbindung Ihrer VPC mit definieren Sie eine Schnittstellen-VPC-Endpunkt für CodePipeline. CodePipeline Dieser Endpunkttyp ermöglicht es Ihnen, Ihre VPC mit zu verbindenAWS-Services. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität zu, CodePipeline ohne dass ein Internet-Gateway, eine Network-Address-Translation-Instance (Network Address Translation) oder eine VPN-Verbindung erforderlich ist. Weitere Informationen zur Einrichtung einer VPC finden Sie im VPC-Benutzerhandbuch.

Verfügbarkeit

CodePipeline Zurzeit unterstützt VPC-Endpunkte in den folgenden LändernAWS-Regionen:

- US East (Ohio)
- USA Ost (Nord-Virginia)
- USA West (Nordkalifornien)
- USA West (Oregon)

- Kanada (Zentral)
- Europe (Frankfurt)
- Europa (Irland)
- Europe (London)
- Europa (Mailand (Mailand (Mailand (Mailand (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, damit Sie sie verwenden können.

Erstellen eines VPC-Endpunkts für CodePipeline

Sie können die Amazon VPC-Konsole zum Erstellen von com.amazonaws verwenden. ***Region*** .codepipeline VPC-Endpunkt. In der Konsole ist *Region* die Regions-ID für eineAWS-Region unterstützte von 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**
>
> AndereAWS-Services, die VPC-Unterstützung bieten und sich integrieren lassen CodePipeline, z. B. CodeCommit, unterstützen möglicherweise nicht die Verwendung von Amazon VPC-Endpunkten für diese Integration. Beispielsweise kann der Datenverkehr

zwischen CodePipeline und CodeCommit nicht auf den VPC-Subnetzbereich beschränkt werden.

Fehlerbehebung für Ihre VPC-Einrichtung

Verwenden Sie bei der Fehlerbehebung von VPC-Problemen die in den Fehlermeldungen zur Internetverbindung angegebenen Informationen, um Probleme zu identifizieren, zu diagnostizieren und zu beheben.

1. Stellen Sie sicher, dass Ihr Internet-Gateway Ihrer VPC angefügt ist..
2. Stellen Sie sicher, dass die Routing-Tabelle für Ihr öffentliches Subnetz auf das Internet-Gateway verweist..
3. Stellen Sie sicher, dass Ihre Netzwerk-ACLs den Datenverkehr zulassen.
4. Stellen Sie sicher, dass Ihre Sicherheitsgruppen den Datenverkehr zulassen.
5. Stellen Sie sicher, dass die Routing-Tabelle für private Subnetze auf das Virtual Private Gateway zeigt.
6. Stellen Sie sicher, dass die von verwendete Servicerolle CodePipeline über die entsprechenden Berechtigungen verfügt. Wenn Sie beispielsweise CodePipeline nicht über die Amazon EC2 EC2-Berechtigungen verfügen, die für die Arbeit mit einer Amazon VPC erforderlich sind, erhalten Sie möglicherweise die Fehlermeldung „Unerwarteter EC2-Fehler:“ UnauthorizedOperation.

Arbeiten mit Pipelines in CodePipeline

Um ein automatisiertes Freigabeverfahren in AWS CodePipeline zu definieren, erstellen Sie eine Pipeline, d. h. einen Workflow, der beschreibt, wie Softwareänderungen das Freigabeverfahren durchlaufen. Eine Pipeline besteht aus mehreren Phasen und Aktionen, die Sie konfigurieren.

> **Note**
>
> Wenn Sie die Stufen Build, Deploy, Test oder Invake hinzufügen, können Sie zusätzlich zu den Standardoptionen von benutzerdefinierte Aktionen auswählen CodePipeline, die Sie bereits für die Verwendung mit Ihren Pipelines erstellt haben. Benutzerdefinierte Aktionen können für das Ausführen von Aufgaben wie beispielsweise des intern entwickelten Erstellungsprozesses oder einer Testsuite eingesetzt werden. Die enthaltenen Versionsbezeichner sollen Ihnen dabei helfen, zwischen den verschiedenen Versionen einer benutzerdefinierten Aktion in den Anbieterlisten zu unterscheiden. Weitere Informationen finden Sie unter Erstellen und Hinzufügen einer benutzerdefinierten Aktion in CodePipeline.

Vor dem Erstellen einer Pipeline müssen Sie die Schritte unter Erste Schritte mit CodePipeline ausführen.

Weitere Informationen zu Pipelines finden Sie unter CodePipeline -Konzepte, CodePipeline -Tutorials oder unter CodePipeline Referenz zur Pipeline-Struktur, wenn Sie eine Pipeline mit der AWS CLI erstellen möchten. Eine Liste der Pipelines finden Sie unter Pipelines und Details anzeigen in CodePipeline.

Themen

- Starten einer Pipeline in CodePipeline
- Stoppen Sie eine Pipeline-Ausführung in CodePipeline
- Erstellen einer Pipeline in CodePipeline
- Bearbeiten einer Pipeline in CodePipeline
- Pipelines und Details anzeigen in CodePipeline
- Löschen einer Pipeline in CodePipeline
- Erstellen einer Pipeline in CodePipeline, die Ressourcen aus einem anderen AWS-Konto verwendet
- Migrieren Sie Abfrage-Pipelines, um die ereignisbasierte Änderungserkennung zu nutzen

- CodePipeline-Servicerolle erstellen
- Markieren CodePipeline
- Erstellen einer Benachrichtigungsregel

Starten einer Pipeline in CodePipeline

Jede Pipeline-Ausführung kann basierend auf einem anderen Auslöser gestartet werden. Jede Pipeline-Ausführung kann je nachdem, wie die Pipeline gestartet wird, eine andere Art von Auslöser haben. Der Auslösertyp für jede Ausführung wird im Ausführungsverlauf für eine Pipeline angezeigt. Auslösertypen können wie folgt vom Quellaktionsanbieter abhängen:

> **Note**
>
> Sie können nicht mehr als einen Auslöser pro Quellaktion angeben.

- Pipeline-Erstellung: Wenn eine Pipeline erstellt wird, wird eine Pipeline-Ausführung automatisch gestartet. Dies ist der `CreatePipeline` Auslösertyp im Ausführungsverlauf .
- Änderungen an überarbeiteten Objekten: Diese Kategorie stellt den `PutActionRevision` Auslösertyp im Ausführungsverlauf dar.
- Änderungserkennung auf Verzweigung und Commit für einen Code-Push: Diese Kategorie stellt den `CloudWatchEvent` Auslösertyp im Ausführungsverlauf dar. Wenn eine Änderung an einem Quell-Commit und einer Verzweigung im Quell-Repository erkannt wird, wird Ihre Pipeline gestartet. Dieser Auslösertyp verwendet die automatische Änderungserkennung. Die Quellaktionsanbieter, die diesen Auslösertyp verwenden, sind S3 und CodeCommit. Dieser Typ wird auch für einen Zeitplan verwendet, der Ihre Pipeline startet. Siehe Starten einer Pipeline nach einem Zeitplan.
- Abfrage für Quelländerungen : Diese Kategorie stellt den `PollForSourceChanges` Auslösertyp im Ausführungsverlauf dar. Wenn eine Änderung an einem Quell-Commit erkannt wird und diese durch Abfragen im Quell-Repository verzweigt wird, wird Ihre Pipeline gestartet. Dieser Auslösertyp wird nicht empfohlen und sollte zur Verwendung der automatisierten Änderungserkennung migriert werden. Die Quellaktionsanbieter, die diesen Auslösertyp verwenden, sind S3 und CodeCommit.
- Webhook-Ereignisse für Drittanbieterquellen: Diese Kategorie stellt den `Webhook` Auslösertyp im Ausführungsverlauf dar. Wenn eine Änderung durch ein Webhook-Ereignis erkannt wird, beginnt Ihre Pipeline. Dieser Auslösertyp verwendet die automatische Änderungserkennung. Die

Quellaktionsanbieter, die diesen Auslösertyp verwenden, sind Verbindungen, die für Code-Push konfiguriert sind (Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltet).

- WebhookV2-Ereignisse für Drittanbieterquellen: Diese Kategorie stellt den `WebhookV2` Auslösertyp im Ausführungsverlauf dar. Dieser Typ gilt für Ausführungen, die basierend auf Auslösern ausgelöst werden, die in der Pipeline-Definition definiert sind. Wenn eine Version mit einem angegebenen Git-Tag erkannt wird, beginnt Ihre Pipeline. Mit Git-Tags können Sie einen Commit mit einem Namen oder einer anderen Kennung versehen, damit andere Repository-Benutzer dessen Bedeutung verstehen. Sie können Git-Tags auch dazu verwenden, um einen bestimmten Commit im Repository-Verlauf zu identifizieren. Dieser Auslösertyp deaktiviert die automatische Änderungserkennung. Die Quellaktionsanbieter, die diesen Auslösertyp verwenden, sind Verbindungen, die für Git-Tags konfiguriert sind (Bitbucket Cloud, GitHub, GitHub Enterprise Server und GitLab.com).
- Manuelles Starten einer Pipeline: Diese Kategorie stellt den `StartPipelineExecution` Auslösertyp im Ausführungsverlauf dar. Sie können die -Konsole oder die verwenden AWS CLI , um eine Pipeline manuell zu starten. Weitere Informationen finden Sie unter Manuelles Starten einer Pipeline.

Wenn Sie Ihrer Pipeline eine Quellaktion hinzufügen, die Auslösertypen für die automatische Änderungserkennung verwendet, funktionieren die Aktionen mit zusätzlichen Ressourcen. Das Erstellen jeder Quellaktion wird aufgrund dieser zusätzlichen Ressourcen zur Änderungserkennung in separaten Abschnitten detailliert beschrieben. Einzelheiten zu den einzelnen Quellanbietern und den für die automatische Änderungserkennung erforderlichen Methoden finden Sie unter Quellaktionen und Methoden zur Erkennung von Änderungen.

Themen

- Quellaktionen und Methoden zur Erkennung von Änderungen
- Manuelles Starten einer Pipeline
- Starten einer Pipeline nach einem Zeitplan
- Starten einer Pipeline mit einer Quellrevisionsüberschreibung

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 S3-Quellaktionen CodeCommit und erfordern entweder eine konfigurierte Änderungserkennungsressource (eine EventBridge Regel) oder verwenden die Option , um das Repository nach Quelländerungen abzufragen. Für Pipelines mit einer Bitbucket GitHub- oder GitHub Enterprise Server-Quellaktion müssen Sie keinen Webhook einrichten oder standardmäßig abfragen. Die Verbindungsaktion verwaltet die Änderungserkennung für Sie.

Quelle	Verwendet zusätzliche Ressourcen?	Schritte
Amazon S3	Diese Quellaktion verwendet zusätzliche Ressourcen. Wenn Sie die CLI oder verwenden, CloudFormation um diese Aktion zu erstellen, erstellen und verwalten Sie auch diese Ressourcen.	Siehe Erstellen einer Pipeline in CodePipeline und Amazon S3-Quellaktionen und EventBridge mit AWS CloudTrail
Bitbucket Cloud	Diese Quellaktion verwendet eine Verbindungsressource.	Siehe Bitbucket-Cloud-Verbindungen
AWS CodeCommit	Amazon EventBridge (empfohlen). Dies ist die Standardeinstellung für eine Pipeline mit einer CodeCommit -Quelle, die in der Konsole erstellt oder bearbeitet wurde.	Siehe Erstellen einer Pipeline in CodePipeline und CodeCommit - Quellaktionen und EventBridge
Amazon ECR	Amazon EventBridge. Dies wird vom Assistenten für Pipelines mit einer Amazon-ECR-Quelle erstellt, die in der Konsole erstellt oder bearbeitet wurde.	Siehe Erstellen einer Pipeline in CodePipeline und Amazon-ECR-Quellaktionen und - EventBridge Ressourcen.
GitHub oder GitHub Enterprise Cloud	Diese Quellaktion verwendet eine Verbindungsressource.	Siehe GitHub Verbindungen

Quelle	Verwendet zusätzliche Ressourcen?	Schritte
GitHub Enterprise Server	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 selbstverwaltete

Wenn Sie eine Pipeline verwenden, die Abfragen verwendet, können Sie sie aktualisieren, um die empfohlene Erkennungsmethode zu verwenden. Weitere Informationen finden Sie unter Aktivieren von Pipelines für die empfohlenen Methode zur Änderungserkennung.

Informationen zum Deaktivieren der Änderungserkennung für eine Quellaktion, die Verbindungen verwendet, finden Sie unter CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen.

Manuelles Starten einer Pipeline

Standardmäßig startet eine Pipeline automatisch bei ihrer Erstellung und bei jeder Änderung eines Quellrepositorys. Möglicherweise möchten Sie jedoch die letzte Revision über die Pipeline ein zweites Mal ausführen. Sie können die CodePipeline Konsole oder den AWS CLI start-pipeline-execution Befehl und verwenden, um die neueste Revision manuell über Ihre Pipeline erneut 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 Management Console und öffnen Sie die - CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie starten möchten.
3. Wählen Sie auf der Seite mit den Pipeline-Details die Option Release change aus. Wenn die Pipeline so konfiguriert ist, dass Parameter (Pipeline-Variablen) übergeben werden, wird bei Auswahl von Release change das Fenster Release change geöffnet. Geben Sie in Pipeline-Variablen im Feld oder den Feldern für die Variablen auf Pipeline-Ebene den Wert oder die Werte ein, die Sie für diese Pipeline-Ausführung übergeben möchten. Weitere Informationen finden Sie unter Variablen.

   Dadurch wird die letzte Revision gestartet, die in jedem in einer Quellaktion der Pipeline angegebenen Quellspeicherort vorhanden ist.

Manuelles Starten einer Pipeline (CLI)

So starten Sie eine Pipeline und führen die letzte Version eines Artefakts manuell über eine Pipeline aus

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) und verwenden Sie die , um den start-pipeline-execution Befehl auszuführen, wobei Sie den Namen der Pipeline angeben, die Sie starten möchten. AWS CLI 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 sind, verwenden Sie den start-pipeline-execution Befehl mit dem optionalen Argument --variables , um die Pipeline zu starten, und fügen Sie die Variablen hinzu, die in der Ausführung verwendet werden. Um beispielsweise eine Variable `var1` mit dem Wert hinzuzufügen1, verwenden Sie den folgenden Befehl:

   ```
   aws codepipeline start-pipeline-execution --name MyFirstPipeline --variables
    name=var1,value=1
   ```

2. Zeigen Sie das zurückgegebene Objekt an, um den Erfolg zu überprüfen. Dieser Befehl gibt eine Ausführungs-ID zurück, die in etwa wie folgt aussieht:

```
{
    "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
}
```

> **Note**
>
> Nachdem Sie die Pipeline gestartet haben, können Sie ihren Fortschritt in der CodePipeline Konsole oder durch Ausführen des get-pipeline-state Befehls überwachen. Weitere Informationen finden Sie unter Pipelines anzeigen (Konsole) und Anzeigen von der Pipeline-Details und des Verlaufs (CLI).

Starten einer Pipeline nach einem Zeitplan

Sie können eine Regel in einrichten EventBridge , um eine Pipeline nach einem Zeitplan zu starten.

Erstellen einer - EventBridge Regel, die den Start Ihrer Pipeline plant (Konsole)

So erstellen Sie eine - EventBridge Regel mit einem Zeitplan als Ereignisquelle

1. Öffnen Sie die Amazon- EventBridge 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 Planen 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 ausCodePipeline.
6. Geben Sie den Pipeline-ARN für die Pipeline-Ausführung für diesen Zeitplan ein.

> **Note**
>
> Sie finden den Pipeline-ARN unter Einstellungen in der -Konsole. Siehe Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen.

7. Wählen Sie eine der folgenden Optionen aus, um eine IAM-Servicerolle zu erstellen oder anzugeben, die EventBridge Berechtigungen zum Aufrufen des Ziels erteilt, das Ihrer EventBridge Regel zugeordnet ist (in diesem Fall ist das Ziel CodePipeline).

   - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen, um eine Servicerolle zu erstellen, die Berechtigungen zum Starten Ihrer Pipeline-Ausführungen erteilt EventBridge.
   - Wählen Sie Vorhandene Rolle verwenden, um eine Servicerolle einzugeben, die EventBridge Berechtigungen zum Starten Ihrer Pipeline-Ausführungen erteilt.
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 einer - EventBridge Regel, die den Start Ihrer Pipeline plant (CLI)

Um die AWS CLI zum Erstellen einer Regel 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, mit denen Sie erstellen, die Ihrem AWS Konto CodePipeline zugeordnet sind, eindeutig sein.
- Den Zeitplanausdruck für die Regel.

So erstellen Sie eine - EventBridge Regel mit einem Zeitplan als Ereignisquelle

1. Rufen Sie den Befehl put-rule auf und beziehen Sie die Parameter `--name` und `--schedule-expression` ein.

   Beispiele:

   Der folgende Beispielbefehl verwendet --schedule-expression, um eine Regel namens zu erstellen`MyRule2`, die EventBridge nach einem Zeitplan filtert.

   ```
   aws events put-rule --schedule-expression 'cron(15 10 ? * 6L 2002-2005)' --name
    MyRule2
   ```

2. Erteilen Sie Berechtigungen für EventBridge , um zum Aufrufen der Regel CodePipeline zu verwenden. Weitere Informationen finden Sie unter Verwenden von ressourcenbasierten 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`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

   c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie in diesem Beispiel für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
```

```
        }
    ]
}
```

d. Führen Sie den folgenden Befehl aus, um der erstellten `Role-for-MyRule`-Rolle die neue `CodePipeline-Permissions-Policy-for-EB`-Berechtigungsrichtlinie anzufügen.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforCWE.json
```

Starten einer Pipeline mit einer Quellrevisionsüberschreibung

Sie können Überschreibungen 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 beim Starten Ihrer Pipeline als Überschreibung hinzufügen.

Es gibt drei Arten von Quellrevisionen für revisionType :

- COMMIT_ID
- IMAGE_DIGEST
- S3_OBJECT_VERSION_ID

> **Note**
>
> Für die Typen COMMIT_ID und IMAGE_DIGEST von Quellrevisionen gilt die Quellrevisions-ID für alle Inhalte im Repository in allen Zweigen.

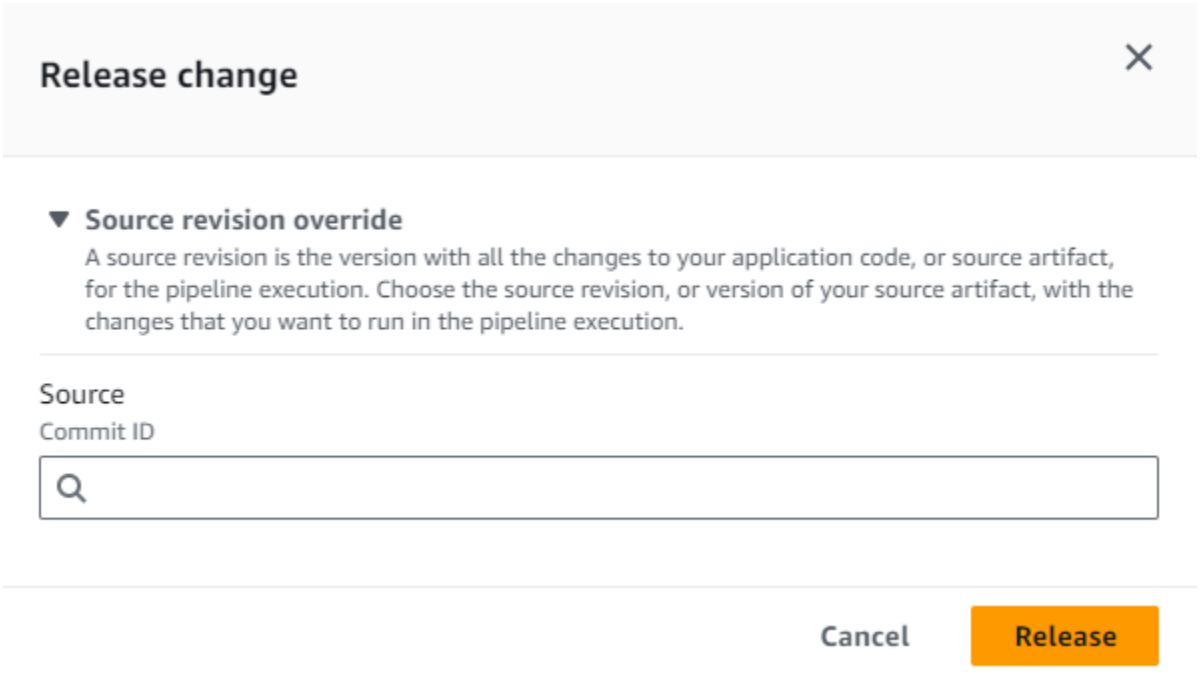
Themen

- Starten einer Pipeline mit einer Quellrevisionsüberschreibung (Konsole)
- Starten einer Pipeline mit einer Quellrevisionsüberschreibung (CLI)

Starten einer 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 Management Console und öffnen Sie die - CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie starten möchten.
3. Wählen Sie auf der Seite mit den Pipeline-Details die Option Release change aus. Wenn Sie Release change auswählen, wird das Fenster Release change geöffnet. Wählen Sie für Quellrevisionsüberschreibung den Pfeil aus, um das Feld zu erweitern. Geben Sie unter Quelle die ID der Quellrevision ein. Wenn Ihre Pipeline beispielsweise über eine CodeCommit Quelle verfügt, wählen Sie die Commit-ID aus dem Feld aus, das Sie verwenden möchten.



Starten einer Pipeline mit einer Quellrevisionsüberschreibung (CLI)

So starten Sie eine Pipeline manuell und führen die angegebene Quellrevisions-ID für ein Artefakt über eine Pipeline aus

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) und verwenden Sie die , um den start-pipeline-execution Befehl auszuführen, wobei Sie den Namen der Pipeline angeben, die Sie starten möchten. AWS CLI Sie verwenden auch das --source-revisions -Argument, um die Quellrevisions-ID anzugeben. Die Quellrevision besteht

aus dem actionName , revisionType und revisionValue . Gültige Werte für revisionType sind `COMMIT_ID | IMAGE_DIGEST | S3_OBJECT_VERSION_ID`.

Um die angegebene Änderung über eine Pipeline mit dem Namen auszuführencodecommit-pipeline, erzeugt der folgende Befehl im folgenden Beispiel einen Quellaktionsnamen von Source`COMMIT_ID`, einen Revisionstyp von und eine Commit-ID von `78a25c18755ccac3f2a9eec099dEXAMPLE`.

```
aws codepipeline start-pipeline-execution --name codecommit-pipeline --source-
revisions
 actionName=Source,revisionType=COMMIT_ID,revisionValue=78a25c18755ccac3f2a9eec099dEXAMPLE
 --region us-west-1
```

2. Zeigen Sie das zurückgegebene Objekt an, um den Erfolg zu überprüfen. Dieser Befehl gibt eine Ausführungs-ID zurück, die in etwa wie folgt aussieht:

```
{
    "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
}
```

> **Note**
>
> Nachdem Sie die Pipeline gestartet haben, können Sie ihren Fortschritt in der CodePipeline Konsole oder durch Ausführen des get-pipeline-state Befehls überwachen. Weitere Informationen finden Sie unter Pipelines anzeigen (Konsole) und Anzeigen von der Pipeline-Details und des Verlaufs (CLI).

Stoppen Sie eine Pipeline-Ausführung in CodePipeline

Wenn eine Pipeline-Ausführung mit dem Durchlaufen einer Pipeline beginnt, tritt sie in eine Phase nach der anderen ein und sperrt die entsprechende Phase, während alle Aktionsausführungen der Phase ausgeführt werden. Diese laufenden Aktionen müssen so verarbeitet werden, dass beim Anhalten der Pipeline-Ausführung die Aktionen entweder abgeschlossen oder abgebrochen werden können.

Es gibt zwei Möglichkeiten, eine Pipeline-Ausführung anzuhalten:

- Anhalten und warten: AWS CodePipeline wartet mit dem Anhalten der Ausführung, bis alle laufenden Aktionen abgeschlossen sind (die Aktionen haben also den Status `Succeeded` oder `Failed`). 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.
- Anhalten und beenden: AWS CodePipeline hält die Ausführung an, 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 einer eingehenden Ausführung (CLI)

Pipeline-Ausführung beenden (Konsole)

Sie können die Konsole verwenden, um die Ausführung einer Pipeline zu beenden. Wählen Sie eine Ausführung und dann die Methode zum Stoppen der Pipeline-Ausführung.

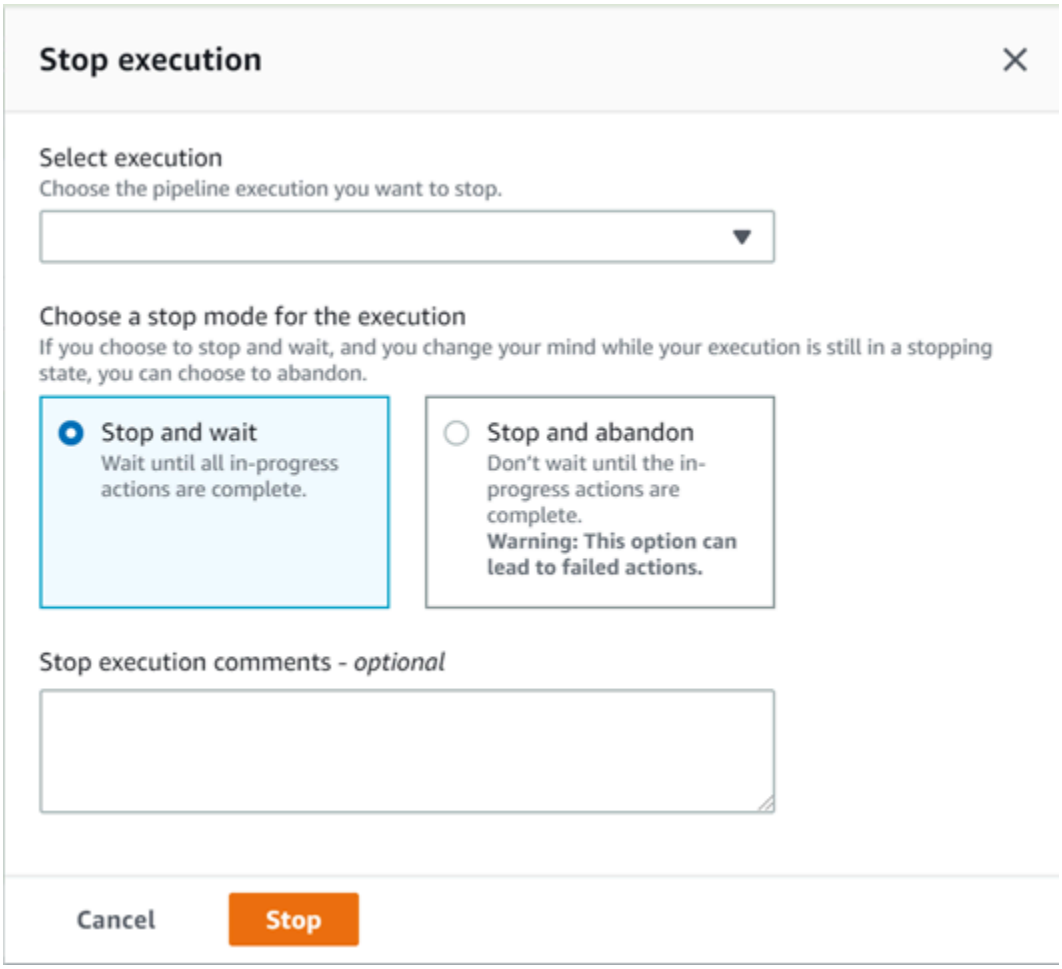
> **Note**
>
> Sie können auch eine Pipeline-Ausführung beenden, bei der es sich um eine eingehende Ausführung handelt. Weitere Informationen zum Stoppen einer eingehenden Ausführung finden Sie unter. Stoppen Sie eine eingehende Ausführung (Konsole)

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

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

   - Wählen Sie in Name den Namen der Pipeline mit der Ausführung aus, die Sie beenden möchten. Wählen Sie auf der Seite mit den Pipeline-Details Stop execution (Ausführung anhalten) aus.
   - Wählen Sie View history (Verlauf anzeigen). Wählen Sie auf der Verlaufsseite Stop execution (Ausführung stoppen) aus.
3. Wählen Sie auf der Seite Stop execution (Ausführung stoppen) unter Select execution (Ausführung auswählen) die Ausführung aus, die Sie beenden möchten.

   > **Note**
   >
   > Die Ausführung wird nur dann angezeigt, wenn sie noch läuft. Bereits abgeschlossene Ausführungen werden nicht angezeigt.



4. Wählen Sie unter Select an action to apply to execution (Aktion für Anwendung auf Ausführung auswählen) eine der folgenden Optionen aus:

   - Um sicherzustellen, dass die Ausführung erst dann beendet wird, wenn alle laufenden Aktionen abgeschlossen sind, wählen Sie Anhalten und warten aus.

     > **Note**
     >
     > Sie können "Anhalten und warten" nicht auswählen, wenn sich die Ausführung bereits im Status Stopping (Wird angehalten) befindet. Sie können jedoch "Anhalten und beenden" auswählen.

   - Um zu beenden, ohne auf den Abschluss laufender Aktionen zu warten, wählen Sie Stop and abandon (Anhalten und beenden) aus.

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

5. (Optional) Geben Sie Kommentare ein. Diese Kommentare werden zusammen mit dem Ausführungsstatus auf der Verlaufsseite für die Ausführung angezeigt.
6. Wählen Sie Stop (Anhalten) aus.

> **Important**
>
> Diese Aktion kann nicht mehr rückgängig gemacht werden.

7. Zeigen Sie den Ausführungsstatus in der Pipeline-Visualisierung wie folgt an:

   - Wenn Sie sich für "Anhalten und Warten" entschieden haben, wird die ausgewählte Ausführung fortgesetzt, bis die laufenden Aktionen abgeschlossen sind.
     - Die Erfolgs-Bannermeldung wird oben in der Konsole angezeigt.
     - In der aktuellen Phase werden die laufenden Aktionen in einem `InProgress`-Status fortgesetzt. Während die Aktionen in Bearbeitung sind, befindet sich die Pipeline-Ausführung in einem `Stopping`-Status.

       Nachdem die Aktionen abgeschlossen sind (d. h. die Aktion schlägt fehl oder ist erfolgreich), wechselt die Pipeline-Ausführung in einen `Stopped`-Status und die Aktion in einen `Failed`- oder `Succeeded`-Status. Sie können den Aktionsstatus auch auf der Seite mit den Ausführungsdetails anzeigen. Sie können den Ausführungsstatus auf der Seite mit dem Ausführungsverlauf oder auf der Seite mit den Ausführungsdetails anzeigen.
     - Die Pipeline-Ausführung wechselt kurz in den Status `Stopping` und dann in den Status `Stopped`. Sie können den Ausführungsstatus auf der Seite mit dem Ausführungsverlauf oder auf der Seite mit den Ausführungsdetails anzeigen.
   - Wenn Sie sich für "Anhalten und beenden" entscheiden, wartet die Ausführung nicht auf den Abschluss der laufenden Aktionen.
     - Die Erfolgs-Bannermeldung wird oben in der Konsole angezeigt.
     - In der aktuellen Phase wechseln die laufenden Aktionen in den Status `Abandoned`. Sie können den Aktionsstatus auch auf der Seite mit den Ausführungsdetails anzeigen.

- Die Pipeline-Ausführung wechselt kurz in den Status `Stopping` und dann in den Status `Stopped`. Sie können den Ausführungsstatus auf der Seite mit dem Ausführungsverlauf oder auf der Seite mit den Ausführungsdetails anzeigen.

Sie können den Pipeline-Ausführungsstatus im Ausführungsverlauf und in der detaillierten Verlaufsansicht anzeigen.

Stoppen Sie eine eingehende Ausführung (Konsole)

Sie können die Konsole verwenden, um eine eingehende Ausführung zu beenden. Eine eingehende Ausführung ist eine Pipeline-Ausführung, die darauf wartet, in eine Phase überzugehen, in der der Übergang deaktiviert wurde. Wenn der Übergang aktiviert ist, 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 in einer deaktivierten Phase des Übergangs keine ausstehenden Ausführungen.

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

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.

2. Wählen Sie den Namen der Pipeline, für die Sie die eingehende Ausführung beenden möchten, und führen Sie einen der folgenden Schritte aus:

   - Wählen Sie in der Pipeline-Ansicht die Ausführungs-ID für den eingehenden Datenverkehr aus und entscheiden Sie dann, ob die Ausführung gestoppt werden soll.
   - Wählen Sie die Pipeline aus und klicken Sie auf Verlauf anzeigen. Wählen Sie in der Ausführungshistorie die Ausführungs-ID für eingehende Nachrichten aus und wählen Sie dann, ob die Ausführung gestoppt werden soll.

3. Folgen Sie im Modal „Ausführung beenden“ den Schritten im obigen Abschnitt, um die Ausführungs-ID auszuwählen und die Stoppmethode anzugeben.

   Verwenden Sie den get-pipeline-state Befehl, um den Status der eingehenden Ausführung anzuzeigen.

Pipeline-Ausführung anhalten (CLI)

Um eine Pipeline mit der AWS CLI manuell zu beenden, verwenden Sie den Befehl stop-pipeline-execution mit den folgenden Parametern:

- Ausführungs-ID (erforderlich)
- Kommentare (optional)
- Pipeline-Name (erforderlich)
- Beenden-Flag (optional, der Standardwert ist Falsch)

Befehlsformat:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --pipeline-
execution-id Execution_ID [--abandon | --no-abandon] [--reason STOP_EXECUTION_REASON]
```

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix).
2. Um eine Pipeline-Ausführung zu beenden, wählen Sie eine der folgenden Optionen aus:

   - Um sicherzustellen, dass die Ausführung nicht beendet wird, bis alle laufenden Aktionen abgeschlossen sind, wählen Sie "Anhalten und warten" aus. Sie können dies tun, indem Sie den Parameter no-abandon einschließen. Wenn Sie den Parameter nicht angeben, wird standardmäßig "Anhalten und warten" verwendet. Verwenden Sie die AWS CLI, um den Befehl stop-pipeline-execution auszuführen, wobei Sie den Namen der Pipeline und die Ausführungs-ID angeben. 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 die AWS CLI, um den Befehl stop-pipeline-execution auszuführen, wobei Sie den Namen der Pipeline und die Ausführungs-ID angeben.

  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 einer eingehenden Ausführung (CLI)

Sie können die CLI verwenden, um eine eingehende Ausführung zu stoppen. Eine eingehende Ausführung ist eine Pipeline-Ausführung, die darauf wartet, in eine Phase überzugehen, in der der Übergang deaktiviert wurde. Wenn der Übergang aktiviert ist, 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 in einer deaktivierten Phase des Übergangs keine ausstehenden Ausführungen.

Verwenden Sie AWS CLI den stop-pipeline-execution Befehl mit den folgenden Parametern, um eine eingehende Ausführung manuell zu beenden:

- Ausführungs-ID für eingehende Nachrichten (erforderlich)
- Kommentare (optional)
- Pipeline-Name (erforderlich)
- Beenden-Flag (optional, der Standardwert ist Falsch)

Befehlsformat:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --
pipeline-execution-id Inbound_Execution_ID [--abandon | --no-abandon] [--
reason STOP_EXECUTION_REASON]
```

Gehen Sie wie im obigen Verfahren beschrieben vor, um den Befehl einzugeben und die Stoppmethode anzugeben.

Verwenden Sie den get-pipeline-state Befehl, um den Status der eingehenden Ausführung anzuzeigen.

Erstellen einer Pipeline in CodePipeline

Sie können die - AWS CodePipeline Konsole oder die verwenden AWS CLI , um eine Pipeline zu erstellen. Pipelines müssen mindestens zwei Stufen enthalten. Die erste Stufe einer Pipeline muss eine Quellstufe sein. Die Pipeline muss mindestens eine weitere Stufe besitzen, bei der es sich um eine Build- oder Bereitstellungsstufe handelt.

Sie können Aktionen zu Ihrer Pipeline hinzufügen, die sich in einer anderen - AWS Region befinden als Ihre Pipeline. Eine regionsübergreifende Aktion ist eine, bei der ein der Anbieter für eine Aktion AWS-Service ist und sich der Aktionstyp oder Anbietertyp in einer - AWS Region befindet, die sich von Ihrer Pipeline unterscheidet. Weitere Informationen finden Sie unter Hinzufügen einer regionsübergreifenden Aktion 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 beschriebenReferenz zu Abbild-Definitionsdateien.

CodePipeline verwendet Methoden zur Änderungserkennung, um Ihre Pipeline zu starten, wenn eine Quellcodeänderung per Push übertragen wird. Diese Erkennungsmethoden basieren auf dem Quelltyp:

- CodePipeline verwendet Amazon CloudWatch Events, um Änderungen in Ihrem CodeCommit Quell-Repository und Ihrer Verzweigung 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 zum Erstellen der Pipeline verwenden, müssen Sie die zusätzlichen Ressourcen selbst erstellen. Weitere Informationen finden Sie unter CodeCommit -Quellaktionen und EventBridge.

Themen

- Erstellen einer Pipeline (Konsole)
- Erstellen einer Pipeline (CLI)
- Amazon-ECR-Quellaktionen und - EventBridge Ressourcen
- Amazon S3-Quellaktionen und EventBridge mit AWS CloudTrail
- Bitbucket-Cloud-Verbindungen
- CodeCommit -Quellaktionen und EventBridge
- GitHub Verbindungen
- GitHub Enterprise Server-Verbindungen
- GitLab.com-Verbindungen
- Verbindungen für GitLab selbstverwaltete

Erstellen einer Pipeline (Konsole)

Um eine Pipeline in der Konsole zu erstellen, müssen Sie den Standort der Quelldatei sowie Informationen zu den Providern angeben, die Sie für Ihre Aktionen verwenden werden.

Wenn Sie die Konsole zum Erstellen einer Pipeline verwenden, müssen Sie eine Source-Stufe und eine der folgenden Stufen einschließen:

- Eine Build-Stufe.
- Eine Bereitstellungsstufe.

Wenn Sie den Pipeline-Assistenten verwenden, CodePipeline erstellt die Namen von Phasen (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 Management Console und öffnen Sie die - CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Klicken Sie auf der Seite Welcome auf Create pipeline.

   Wenn Sie zum ersten Mal verwenden CodePipeline, wählen Sie Erste Schritte aus.
3. Geben Sie auf der Seite Step 1: Choose pipeline settings (Schritt 1: Pipeline-Einstellungen auswählen) unter Pipeline name (Pipeline-Name) den Namen für Ihre Pipeline ein.

   In einem einzigen AWS Konto muss jede Pipeline, die Sie in einer - AWS Region erstellen, einen eindeutigen Namen haben. Namen können für Pipelines in anderen Regionen wiederverwendet werden.

   > **Note**
   >
   > Der Name einer erstellten Pipeline kann nicht nachträglich geändert werden. Weitere Informationen zu Einschränkungen finden Sie unter Kontingente in AWS CodePipeline.
4. Wählen Sie unter Pipeline-Typ eine der folgenden Optionen aus. Pipeline-Typen unterscheiden sich in Eigenschaften und Preis. Weitere Informationen finden Sie unter Pipeline-Typen.

   - Pipelines vom Typ V1 verfügen über eine JSON-Struktur, die Standardparameter auf Pipeline-, Stufen- und Aktionsebene enthält.
   - Pipelines vom Typ V2 haben dieselbe Struktur wie ein V1-Typ sowie zusätzliche Parameterunterstützung, z. B. Auslöser für Git-Tags und Variablen auf Pipeline-Ebene.
5. Führen Sie unter Service role (Service-Rolle) einen der folgenden Schritte aus:

- Wählen Sie Neue Servicerolle aus, damit eine neue Servicerolle in IAM CodePipeline erstellen kann.
- 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 zusätzliche zu unterstützen AWS-Services. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.

Weitere Informationen über die Service-Rolle und ihre Richtlinienanweisung finden Sie unter Verwalten der CodePipeline Servicerolle.

6. (Optional) Wählen Sie unter Variablen die Option Variable hinzufügen aus, um Variablen auf Pipeline-Ebene hinzuzufügen.

   Weitere Informationen zu Variablen auf Pipeline-Ebene finden Sie unter Variablen. 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.

   > **Note**
   >
   > Obwohl es optional ist, Variablen auf Pipeline-Ebene hinzuzufügen, schlägt die Pipeline-Ausführung für eine Pipeline fehl, die mit Variablen auf Pipeline-Ebene angegeben ist, in der keine Werte angegeben werden.

7. (Optional) Erweitern Sie den Abschnitt Advanced Settings (Erweiterte Einstellungen).
8. Führen Sie unter Artifact store (Artefakt speichern) einen der folgenden Schritte aus:

   a. Wählen Sie Standardspeicherort aus, um den Standardartefaktspeicher, z. B. den als Standard bezeichneten S3-Artefakt-Bucket, für Ihre Pipeline in der zu verwenden, die AWS-Region Sie für Ihre Pipeline ausgewählt haben.

b. Wählen Sie Custom location (Benutzerdefinierter Speicherort), wenn Sie in derselben Region wie Ihre Pipeline bereits über einen Artefakt-Speicher, wie z. B. einen S3-Artefakt-Bucket, verfügen. Wählen Sie unter Bucket den Bucket-Namen aus.

> **Note**
>
> Dabei handelt es sich nicht um den Quell-Bucket für Ihren Quellcode, sondern um den Artefaktspeicher für Ihre Pipeline. Für jede Pipeline benötigen Sie einen separaten Artefaktspeicher, z. B. einen S3 Bucket. Wenn Sie eine Pipeline erstellen oder bearbeiten, benötigen Sie einen Artefakt-Bucket in der Pipeline-Region und einen Artefakt-Bucket pro AWS Region, in der Sie eine Aktion ausführen.
> Weitere Informationen finden Sie unter Eingabe- und Ausgabe-Artefakte und CodePipeline Referenz zur Pipeline-Struktur.

9. Führen Sie unter Encryption key (Verschlüsselungsschlüssel) einen der folgenden Schritte aus:

   a. Um den CodePipeline Standard AWS KMS key zum Verschlüsseln der Daten im Pipeline-Artefaktspeicher (S3-Bucket) zu verwenden, wählen Sie Standard AWS -verwalteter Schlüssel aus.

   b. Um Ihren kundenverwalteten Schlüssel zum Verschlüsseln der Daten im Pipeline-Artefaktspeicher (S3-Bucket) zu verwenden, wählen Sie Kundenverwalteter Schlüssel . Wählen Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN aus.

10. Wählen Sie Weiter aus.

Schritt 2: Erstellen Sie eine Quellstufe

- Wählen Sie auf der Seite Step 2: Add source stage (Schritt 2: Quelle) in der Dropdown-Liste Source provider (Quell-Stufe hinzufügen) den Typ des Repositorys aus, in dem der Quellcode gespeichert ist. Geben Sie die erforderlichen Optionen an und wählen Sie dann Next step (Nächster Schritt).

  - Für Bitbucket Cloud, GitHub (Version 2), GitHub Enterprise Server, GitLab.com oder GitLab selbstverwaltet:

    1. 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 unter GitHub Verbindungen.

2. Wählen Sie das Repository aus, das Sie als Quellspeicherort für Ihre Pipeline verwenden möchten.

   Wählen Sie aus, ob Sie einen Auslöser hinzufügen oder nach Auslösertypen filtern möchten, um Ihre Pipeline zu starten. Weitere Informationen zum Arbeiten mit Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen. Weitere Informationen zum Filtern mit globalen Mustern finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

3. Wählen Sie unter Ausgabeartefaktformat das Format für Ihre Artefakte aus.

   - Um Ausgabeartefakte aus 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 Projektservicerolle aktualisieren, wie in gezeigtFehlerbehebung CodePipeline. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiger Klon verwenden, finden Sie unter Tutorial: 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 oder Dateien in einen einzigen ZIP-Code komprimieren und den ZIP-Code 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 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 kann Amazon CloudWatch Events CodePipeline verwenden und Änderungen für Ihre neue Pipeline AWS CloudTrail erkennen. Wählen Sie Weiter aus.

- AWS CodeCommit:
  - Wählen Sie unter Repository-Name den Namen des CodeCommit Repositorys aus, das Sie als Quellspeicherort 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 Ausgabeartefakte aus der CodeCommit Aktion mit der Standardmethode zu speichern, wählen Sie CodePipeline Standard . 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 die `-codecommit:GitPull`Berechtigung zu Ihrer CodeBuild Servicerolle hinzufügen, wie in gezeigtHinzufügen von CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen. Sie müssen Ihrer CodePipeline Servicerolle auch die `codecommit:GetRepository` Berechtigungen hinzufügen, wie in gezeigtHinzufügen von Berechtigungen zur CodePipeline-Servicerolle. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiger Klon verwenden, finden Sie unter Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.
  - Nachdem Sie den CodeCommit Repository-Namen und die Verzweigung ausgewählt haben, wird in den Optionen zur Änderungserkennung eine Meldung angezeigt, in der die für diese Pipeline zu erstellende Amazon- CloudWatch Events-Regel angezeigt wird. Lassen Sie die Standardwerte in Change detection options (Optionen für die Änderungserkennung) unverändert. Auf diese Weise kann Amazon CloudWatch Events verwenden CodePipeline , um Änderungen an Ihrer neuen 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 den Standard für Ausgabeartefakte aus, z. B. MyApp, der den Image-Namen und die Repository-URI-Informationen enthält, die Sie in der nächsten Phase verwenden möchten.

  Ein Tutorial zum Erstellen einer Pipeline für Amazon ECS mit CodeDeploy Blau/Grün-Bereitstellungen, die eine Amazon-ECR-Quellphase enthält, finden Sie unter Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy .

Wenn Sie eine Amazon-ECR-Quellphase in Ihre Pipeline aufnehmen, generiert die Quellaktion eine `-imageDetail.json`Datei als Ausgabeartefakt, wenn Sie eine Änderung festschreiben. Informationen zur `imageDetail.json`-Datei finden Sie unter imageDetail.json-Datei für Amazon ECS-Blau/Grün-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-Umgebungen und unterstützten Plattformen. Weitere Informationen zum Bereitstellen von Revisionen mit finden Sie CodeDeployunter Hochladen Ihrer Anwendungsrevision und Vorbereiten einer Revision.

Schritt 3: Erstellen Sie eine Build-Stufe

Dieser Schritt ist optional, wenn Sie die Erstellung einer Bereitstellungsstufe planen.

- Führen Sie auf der Seite Step 3:Add build stage (Schritt 3: Build-Stufe hinzufügen) einen der folgenden Schritte aus und klicken Sie dann auf Next (Weiter):
  - Wählen Sie Skip build stage (Build-Stufe überspringen), wenn Sie die Erstellung einer Bereitstellungsstufe planen.
  - Wählen Sie in Build provider (Build-Anbieter) einen benutzerdefinierten Aktionsanbieter von Build-Services aus und geben Sie die Konfigurationsdetails für diesen Anbieter ein. Ein

Beispiel zum Hinzufügen von Jenkins als Build-Anbieter finden Sie unter Tutorial: Erstellen einer vierstufigen Pipeline.

- Wählen Sie unter Build provider (Build-Anbieter) AWS CodeBuild aus.

  Wählen Sie in Region die AWS Region aus, in der sich die Ressource befindet. 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 ist standardmäßig dieselbe AWS Region wie Ihre Pipeline.

  Wählen Sie unter Project name (Projektname) Ihr Build-Projekt aus. Wenn Sie bereits ein Build-Projekt in erstellt haben CodeBuild, wählen Sie es aus. Oder Sie können 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.

  Um in Umgebungsvariablen Umgebungsvariablen zu Ihrer Build-Aktion hinzuzufügen CodeBuild, wählen Sie 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 auswählen, 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 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 Build-Typ eine der folgenden Optionen aus:

- Um jeden Build in einer einzigen Build-Aktionsausführung auszuführen, wählen Sie Single build .
- Um mehrere Builds in derselben Build-Aktionsausführung auszuführen, wählen Sie Batch-Build aus.

(Optional) Wenn Sie Batch-Builds ausführen möchten, können Sie Alle Artefakte aus Batch an einem einzigen Ort kombinieren auswählen, um alle Build-Artefakte in einem einzigen Ausgabeartefakt zu platzieren.

Schritt 4: Erstellen Sie eine Bereitstellungsstufe

Dieser Schritt ist optional, falls Sie bereits eine Build-Stufe erstellt haben.

- Führen Sie auf der Seite Step 4:Add deploy stage (Schritt 4: Bereitstellungsstufe hinzufügen) einen der folgenden Schritte aus und klicken Sie dann auf Next (Weiter):
  - Wählen Sie Skip deploy stage (Bereitstellungsstufe überspringen), wenn Sie im vorherigen Schritt eine Build-Stufe erstellt haben.

    > **Note**
    >
    > Diese Option wird nicht angezeigt, falls Sie die Build-Stufe bereits übersprungen haben.

  - Wählen Sie unter Deploy provider (Bereitstellungsanbieter) eine benutzerdefinierte Aktion aus, die Sie für einen Bereitstellungsanbieter erstellt haben.

    Wählen Sie in 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 zeigt nur für Aktionen an, bei denen der Aktionsanbieter ein ist AWS-Service. Das Feld Region ist standardmäßig dieselbe AWS Region wie Ihre Pipeline.
  - Unter Deploy provider (Bereitstellungsanbieter) finden Sie folgende Felder für Standardanbieter:
    - CodeDeploy

      Geben Sie unter 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 unter Anwendungsname den Namen einer vorhandenen Elastic 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 auch eine Anwendung, Umgebung oder beides in der Elastic Beanstalk-Konsole 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 einer Ebene in AWS OpsWorks finden Sie unter Hinzufügen von Apps im AWS OpsWorks -Benutzerhandbuch.

  Ein end-to-end Beispiel für die Verwendung einer einfachen Pipeline in CodePipeline als Quelle für Code, den Sie auf AWS OpsWorks Ebenen ausführen, finden Sie unter Verwenden von CodePipeline mit AWS OpsWorks Stacks.

- AWS CloudFormation

  Führen Sie eine der folgenden Aktionen aus:

  - Wählen Sie im Aktionsmodus Erstellen oder Aktualisieren eines Stacks aus, geben Sie einen Stack-Namen und einen Vorlagendateinamen ein und wählen Sie dann den Namen einer Rolle aus, die übernehmen AWS CloudFormation soll. Geben Sie optional den Namen einer Konfigurationsdatei ein und wählen Sie eine IAM-Funktionsoption aus.
  - Wählen Sie im Aktionsmodus Erstellen oder Ersetzen eines Änderungssatzes aus, geben Sie einen Stack-Namen und einen Änderungssatznamen ein und wählen Sie dann den Namen einer Rolle aus, die übernehmen AWS CloudFormation soll. Geben Sie optional den Namen einer Konfigurationsdatei ein und wählen Sie eine IAM-Funktionsoption aus.

Informationen zur Integration von AWS CloudFormation Funktionen in eine Pipeline in CodePipelinefinden Sie unter Kontinuierliche Bereitstellung mit CodePipeline im AWS CloudFormation -Benutzerhandbuch.

- Amazon ECS

  Geben Sie unter 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**
  >
  > Die Amazon-ECS-Bereitstellungsaktion erfordert eine `-imagedefinitions.json`Datei als Eingabe für die Bereitstellungsaktion. Der Standardname der Datei ist imagedefinitions.json. Wenn Sie einen anderen Dateinamen verwenden, müssen Sie diesen während der Erstellung der Pipeline-Bereitstellungsphase angeben. Weitere Informationen finden Sie unter imagedefinitions.json-Datei für Amazon ECS-SStandardbereitstellungsaktionen.

  Wählen Sie Weiter.

  > **Note**
  >
  > Stellen Sie sicher, dass Ihr Amazon-ECS-Cluster mit zwei oder mehr Instances konfiguriert ist. Amazon-ECS-Cluster müssen mindestens zwei Instances enthalten, damit eine als primäre Instance und eine 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) (Amazon ECS (Blau/Grün))

Geben Sie die CodeDeploy Anwendung und Bereitstellungsgruppe, die Amazon-ECS-Aufgabendefinition und die AppSpec Dateiinformationen ein und wählen Sie dann Weiter aus.

> **Note**
>
> Die Aktion Amazon ECS (Blue/Green) (Amazon ECS (Blau/Grün)) erfordert eine imageDetail.json-Datei als Eingabeartefakt für die Bereitstellungsaktion. Da die Amazon-ECR-Quellaktion diese Datei erstellt, müssen Pipelines mit einer Amazon-ECR-Quellaktion keine `-imageDetail.json`Datei bereitstellen. Weitere Informationen finden Sie unter imageDetail.json-Datei für Amazon ECS-Blau/Grün-Bereitstellungsaktionen.

Ein Tutorial zum Erstellen einer Pipeline für Blau/Grün-Bereitstellungen in einem Amazon-ECS-Cluster mit CodeDeployfinden Sie unter Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen 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 zum Bereitstellen von Produktänderungen im Service Catalog mit Ihrer Pipeline finden Sie unter Tutorial: Erstellen Sie eine Pipeline, die in Service Catalog bereitgestellt wird.

- Alexa Skills Kit

  Geben Sie unter Alexa Skill-ID die Skill-ID für Ihre Alexa-Qualifikation ein. Geben Sie unter Client-ID und Clientschlüssel die Anmeldeinformationen ein, die über ein Login with Amazon-Sicherheitsprofil (LWA) generiert wurden. Geben Sie unter Aktualisierungs-Token das Aktualisierungs-Token ein, das Sie mit dem ASK-CLI-Befehl zum Abrufen eines Aktualisierungs-Tokens generiert haben. Wählen Sie Weiter aus.

Ein Tutorial zur Bereitstellung von Alexa-Qualifikationen mit Ihrer Pipeline und zur Generierung der LWA-Anmeldeinformationen finden Sie unter Tutorial: Erstellen einer Pipeline zum Bereitstellen eines Amazon Alexa-Skills.

- Amazon S3

  Geben Sie unter Bucket den Namen des gewünschten S3-Buckets ein. Wählen Sie Extract file before deploy (Datei vor Bereitstellung extrahieren), falls das Eingabeartefakt für Ihre Bereitstellungsstufe eine ZIP-Datei ist. Wenn Extract file before deploy (Datei vor Bereitstellung extrahieren) ausgewählt ist, können Sie optional einen Wert für Deployment path (Bereitstellungspfad) als Speicherort eingeben, an dem Ihre ZIP-Datei entzippt wird. Wenn die Option nicht aktiviert ist, müssen Sie einen Wert unter S3 object key (S3-Objektschlüssel) eingeben.

  > **Note**
  >
  > Die meisten Ausgabeartefakte von Quell- und Build-Stufen werden gezippt. Alle Pipeline-Quellanbieter außer Amazon S3 komprimieren Ihre Quelldateien, bevor sie als Eingabeartefakt für die nächste Aktion bereitgestellt werden.

  (Optional) Geben Sie unter Vordefinierte ACL die vordefinierte 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-Bereitstellungsaktionsanbieter finden Sie unter Tutorial: Erstellen Sie eine Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet.

Schritt 5: Überprüfen der Pipeline

- Prüfen Sie auf der Seite Step 5: Review die Konfiguration Ihrer Pipeline und wählen Sie dann Create pipeline zur Erstellung der Pipeline oder Previous zum Bearbeiten aus. Um den Assistenten ohne Erstellen einer Pipeline zu beenden, wählen Sie Cancel aus.

Nachdem Sie die Pipeline erstellt haben, können Sie sie in der Konsole anzeigen. Die Pipeline wird nach dem Erstellen ausgeführt. Weitere Informationen finden Sie unter Pipelines und Details anzeigen in CodePipeline. Weitere Informationen über die Durchführung von Änderungen an Ihrer Pipeline finden Sie unter Bearbeiten einer Pipeline in CodePipeline.

Erstellen einer Pipeline (CLI)

Um eine Pipeline mit der AWS CLI zu erstellen, erstellen Sie eine JSON-Datei zur Definition der Pipeline-Struktur und führen dann den create-pipeline Befehl mit dem `--cli-input-json` Parameter aus.

> **Important**
>
> Sie können die 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 zu .

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 den Servicerollen CodePipeline -ARN in der Pipeline-JSON-Datei, wenn Sie den create-pipeline Befehl ausführen. Weitere Informationen zum Erstellen einer Servicerolle finden Sie unter CodePipeline-Servicerolle erstellen. Im Gegensatz zur Konsole AWS CLI hat das Ausführen des create-pipeline Befehls in der keine Option, 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 erstellt das Ausführen des create-pipeline Befehls in AWS CLI der keinen S3-Bucket zum Speichern von Artefakten. 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 in einem Terminal (Linux, macOS oder Unix) oder einer Eingabeaufforderung (Windows) eine neue Textdatei in einem lokalen Verzeichnis.
2. (Optional) Sie können eine oder mehrere Variablen auf Pipeline-Ebene hinzufügen. Sie können diesen Wert in der Konfiguration von CodePipeline Aktionen referenzieren. Sie können die Variablennamen und -werte hinzufügen, wenn Sie die Pipeline erstellen, und Sie können auch Werte zuweisen, wenn Sie die Pipeline in der Konsole starten.

> **Note**
>
> Obwohl es optional ist, Variablen auf Pipeline-Ebene hinzuzufügen, schlägt die Pipeline-Ausführung für eine Pipeline fehl, die mit Variablen auf Pipeline-Ebene angegeben ist, in der keine Werte angegeben werden.

Eine Variable auf Pipeline-Ebene 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 für jede Ausführung angezeigt.

Sie stellen Variablen auf Pipeline-Ebene mithilfe des Variablenattributs 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 zu Ihrem Pipeline-JSON oder zum Beispiel-JSON im folgenden Schritt hinzu. Weitere Informationen zu Variablen, einschließlich Namespace-Informationen, finden Sie unter Variablen.

3. Öffnen Sie die Datei in einem Texteditor und bearbeiten Sie die Werte entsprechend der Struktur, die Sie erstellen möchten. Sie müssen mindestens den Namen der Pipeline ändern. Sie sollten auch überlegen, ob Sie Folgendes ändern möchten:

   - Den S3-Bucket, in dem Artefakte für diese Pipeline gespeichert sind.
   - Den Speicherort für Ihren Code.
   - Den Bereitstellungsanbieter.
   - Art der Bereitstellung des Codes.
   - Die Tags für Ihre Pipeline.

Die folgende Beispielstruktur für eine Pipeline mit zwei Phasen zeigt die Werte an, deren Änderung Sie für Ihre Pipeline in Betracht ziehen sollten. Ihre Pipeline enthält wahrscheinlich mehr als zwei Phasen:

```
{
    "pipeline": {
        "roleArn": "arn:aws:iam::80398EXAMPLE::role/AWS-CodePipeline-Service",
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "inputArtifacts": [],
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "S3"
                        },
                        "outputArtifacts": [
                            {
                                "name": "MyApp"
                            }
                        ],
                        "configuration": {
                            "S3Bucket": "awscodepipeline-demobucket-example-date",
                            "S3ObjectKey": "ExampleCodePipelineSampleBundle.zip",
                            "PollForSourceChanges": "false"
                        },
                        "runOrder": 1
                    }
                ]
            },
            {
                "name": "Staging",
                "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "MyApp"
```

```
                        }
                    ],
                    "name": "Deploy-CodeDeploy-Application",
                    "actionTypeId": {
                        "category": "Deploy",
                        "owner": "AWS",
                        "version": "1",
                        "provider": "CodeDeploy"
                    },
                    "outputArtifacts": [],
                    "configuration": {
                        "ApplicationName": "CodePipelineDemoApplication",
                        "DeploymentGroupName": "CodePipelineDemoFleet"
                    },
                    "runOrder": 1
                }
            ]
        }
    ],
    "artifactStore": {
        "type": "S3",
        "location": "codepipeline-us-east-2-250656481468"
    },
    "name": "MyFirstPipeline",
    "version": 1,
    "variables": [
        {
            "name": "Timeout",
            "defaultValue": "1000",
            "description": "description"
        }
      ]
    },
    "triggers": [
        {
            "providerType": "CodeStarSourceConnection",
            "gitConfiguration": {
                "sourceActionName": "Source",
                "push": [
                    {
                        "tags": {
                            "includes": [
                                "v1"
                            ],
```

```
                                "excludes": [
                                    "v2"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline",
        "updated": 1501626591.112,
        "created": 1501626591.112
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
    }]
}
```

In diesem Beispiel wird der Pipeline Tagging hinzugefügt, indem der Pipeline der Tag-Schlüssel `Project` und der Wert `ProjectA` hinzugefügt werden. Weitere Informationen zum Markieren von Ressourcen in CodePipelinefinden Sie unter Markieren von Ressourcen.

Stellen Sie sicher, dass der `PollForSourceChanges`-Parameter in Ihrer JSON-Datei wie folgt festgelegt ist:

```
                "PollForSourceChanges": "false",
```

CodePipeline verwendet Amazon CloudWatch Events, um Änderungen in Ihrem CodeCommit Quell-Repository und Ihrer Verzweigung oder Ihrem S3-Quell-Bucket zu erkennen. Der nächste Schritt enthält Anweisungen zum manuellen Erstellen dieser Ressourcen für Ihre Pipeline. Wenn das Flag auf `false` gesetzt wird, werden regelmäßige Prüfungen deaktiviert, die nicht erforderlich sind, wenn die empfohlenen Methoden zur Änderungserkennung verwendet werden.

4. Um eine Build-, Test- oder Bereitstellungsaktion in einer anderen Region als Ihre Pipeline zu erstellen, müssen Sie Ihrer Pipeline-Struktur Folgendes hinzufügen. Anweisungen finden Sie unter Hinzufügen einer regionsübergreifenden Aktion 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 über eine -Aktion verfügen.

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`* namens mit einer JSON-Datei namens pipeline.json zu erstellen, die den Namen „*`MySecondPipeline`*“ als Wert für `name` im JSON 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 Liste der Pipelines aus, oder verwenden Sie den get-pipeline-state Befehl . 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 einer - EventBridge Regel für eine CodeCommit Quelle (CLI).
   - Für eine Pipeline mit einer Amazon S3-Quelle müssen Sie die CloudWatch Ereignisregel und den AWS CloudTrail Trail manuell erstellen, wie unter beschriebenAmazon S3-Quellaktionen und EventBridge mit AWS CloudTrail.

Amazon-ECR-Quellaktionen und - EventBridge Ressourcen

Um eine Amazon-ECR-Quellaktion in hinzuzufügen CodePipeline, können Sie entweder:

- Wählen Sie auf der Seite Pipeline CodePipeline erstellen (Erstellen einer Pipeline (Konsole)) oder Aktion bearbeiten die Option Amazon-ECR-Anbieter aus. Die Konsole erstellt eine - EventBridge Regel, die Ihre Pipeline startet, wenn sich die Quelle ändert.
- Verwenden Sie die CLI, um die Aktionskonfiguration für die `ECR` Aktion hinzuzufügen und zusätzliche Ressourcen wie folgt zu erstellen:
  - Verwenden Sie die `ECR` Beispielaktionskonfiguration in Amazon ECR, um Ihre Aktion zu erstellen, wie in gezeigtErstellen einer Pipeline (CLI).
  - Die Methode zur Erkennung von Änderungen startet standardmäßig die Pipeline, indem die Quelle abgefragt wird. Sie sollten regelmäßige Prüfungen deaktivieren und die Änderungserkennungsregel manuell erstellen. Verwenden Sie eine der folgenden Methoden: Erstellen einer - EventBridge Regel für eine Amazon-ECR-Quelle (Konsole)Erstellen einer - EventBridge Regel für eine Amazon-ECR-Quelle (CLI), oder Erstellen einer - EventBridge Regel für eine Amazon-ECR-Quelle (AWS CloudFormation Vorlage) .

Themen

- Erstellen einer - EventBridge Regel für eine Amazon-ECR-Quelle (Konsole)
- Erstellen einer - EventBridge Regel für eine Amazon-ECR-Quelle (CLI)
- Erstellen einer - EventBridge Regel für eine Amazon-ECR-Quelle (AWS CloudFormation Vorlage)

Erstellen einer - EventBridge Regel für eine Amazon-ECR-Quelle (Konsole)

So erstellen Sie eine - EventBridge Regel für die Verwendung in - CodePipeline Operationen (Amazon-ECR-Quelle)

1. Öffnen Sie die Amazon- EventBridge Konsole unter https://console.aws.amazon.com/events/.
2. Wählen Sie im Navigationsbereich die Option Events.
3. Wählen Sie Regel erstellen und dann unter Ereignisquelle unter Servicename 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 und EventBridge oder Amazon-Elastic-Container-Registry-Ereignisse.

5. Wählen Sie Speichern.

   Zeigen Sie die Regel im Bereich Event Pattern Preview an.

6. Wählen Sie unter Ziele die Option ausCodePipeline.
7. Geben Sie den Pipeline-ARN für die Pipeline ein, die von 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 oder geben Sie eine IAM-Servicerolle an, die EventBridge Berechtigungen zum Aufrufen des Ziels erteilt, das Ihrer EventBridge Regel zugeordnet ist (in diesem Fall ist das Ziel CodePipeline).
   - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen, um eine Servicerolle zu erstellen, die EventBridge Ihren Pipeline-Ausführungen Berechtigungen erteilt.
   - Wählen Sie Vorhandene Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihrem EventBridge Berechtigungen zum Starten Ihrer Pipeline-Ausführungen erteilt.
9. Überprüfen Sie die eingerichteten Regeln, um sicherzustellen, dass sie Ihren Anforderungen entsprechen.
10. Wählen Sie Details konfigurieren.
11. Geben Sie auf der Seite Configure rule details (Regeldetails konfigurieren) einen Namen und eine Beschreibung für die Regel ein und wählen Sie dann State (Status), um die Regel zu aktivieren.
12. Wenn Sie mit der Regel zufrieden sind, klicken Sie auf Create rule (Regel erstellen).

Erstellen einer - 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, mit denen Sie erstellen, die Ihrem AWS Konto CodePipeline zugeordnet sind, eindeutig sein.
- Das Ereignismuster für die Quelle und von der Regel verwendete Detailfelder. Weitere Informationen finden Sie unter Amazon EventBridge und Ereignismuster.

So erstellen Sie eine - EventBridge Regel mit Amazon ECR als Ereignisquelle und CodePipeline als Ziel

1. Fügen Sie Berechtigungen für hinzu EventBridge , um zum Aufrufen der Regel CodePipeline zu verwenden. Weitere Informationen finden Sie unter Verwenden von ressourcenbasierten 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`.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

   b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

   ```
   aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
    file://trustpolicyforEB.json
   ```

   c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie in diesem Beispiel für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "codepipeline:StartPipelineExecution"
   ```

```
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Verwenden Sie den folgenden Befehl, um die Berechtigungsrichtlinie `CodePipeline-Permissions-Policy-for-EB` der Rolle `Role-for-MyRule` anzufügen.

Warum nehme ich diese Änderung vor? Wenn Sie diese Richtlinie der Rolle hinzufügen, 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 angibt, wie ein Image-Push durchgeführt werden muss, und ein Ziel, das die Pipeline benennt, die vom 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 und EventBridge oder Amazon-Elastic-Container-Registry-Ereignisse.

3. Um CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und schließen 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
```

Erstellen einer - EventBridge Regel für eine Amazon-ECR-Quelle (AWS CloudFormation Vorlage)

Um zum Erstellen einer Regel AWS CloudFormation zu verwenden, verwenden Sie den Vorlagenausschnitt, wie hier gezeigt.

So aktualisieren Sie Ihre Pipeline- AWS CloudFormation Vorlage und erstellen eine EventBridge Regel

1. Verwenden Sie in der Vorlage unter die Ressource `Resources`, `AWS::IAM::Role` AWS CloudFormation um die IAM-Rolle zu konfigurieren, die es Ihrem Ereignis 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? Sie müssen eine Rolle erstellen, die von angenommen 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 Ressource `Resources`, `AWS::Events::Rule` AWS CloudFormation um eine EventBridge Regel für die Amazon-ECR-Quelle hinzuzufügen. Dieses Ereignismuster erstellt ein Ereignis, das Commits in Ihrem Repository überwacht. Wenn eine Repository-Statusänderung EventBridge erkennt, wird die Regel `StartPipelineExecution` für Ihre Ziel-Pipeline aufgerufen.

   Warum nehme ich diese Änderung vor? Sie müssen ein Ereignis mit einer Regel erstellen, die angibt, wie ein Image-Push durchgeführt werden muss, und ein Ziel, das die Pipeline benennt, die vom 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 und EventBridge oder Amazon-Elastic-Container-Registry-Ereignisse.

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation -Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.

6. Wählen Sie Execute (Ausführen).

Amazon S3-Quellaktionen und EventBridge mit AWS CloudTrail

Um eine Amazon S3-Quellaktion in hinzuzufügen CodePipeline, wählen Sie eine der folgenden Optionen:

- Wählen Sie auf der Seite Pipeline CodePipeline erstellen (Erstellen einer Pipeline (Konsole)) oder Aktion bearbeiten die Option S3-Anbieter aus. Die Konsole erstellt eine - EventBridge Regel und einen CloudTrail Trail, der Ihre Pipeline startet, wenn sich die Quelle ändert.
- Verwenden Sie die AWS CLI , um die Aktionskonfiguration für die `S3` Aktion hinzuzufügen und zusätzliche Ressourcen wie folgt zu erstellen:
  - Verwenden Sie die `S3` Beispielaktionskonfiguration in Quellaktion für Amazon S3, um Ihre Aktion zu erstellen, wie in gezeigtErstellen einer Pipeline (CLI).
  - Die Methode zur Erkennung von Änderungen startet standardmäßig die Pipeline, indem die Quelle abgefragt wird. Sie sollten regelmäßige Prüfungen deaktivieren und die Änderungserkennungsregel und den Trail manuell erstellen. Verwenden Sie eine der folgenden Methoden: Erstellen einer - EventBridge Regel für eine Amazon S3-Quelle (Konsole)Erstellen einer - EventBridge Regel für eine Amazon S3-Quelle (CLI), oder Erstellen einer - EventBridge Regel für eine Amazon S3-Quelle (AWS 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 seine Protokolldateien speichern AWS CloudTrail kann. 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 einen Amazon S3-Bucket im Rahmen der Erstellung oder Aktualisierung eines Trails in der Konsole erstellen,

AWS CloudTrail fügt die erforderlichen Berechtigungen einem Bucket für Sie hinzu. Weitere Informationen finden Sie unter Amazon S3-Bucket-Richtlinie für CloudTrail.

Erstellen einer - EventBridge Regel für eine Amazon S3-Quelle (Konsole)

Bevor Sie eine Regel in einrichten EventBridge, müssen Sie einen - AWS CloudTrail Trail erstellen. Weitere Informationen finden Sie unter Erstellen eines Trails in der Konsole.

> **Important**
>
> Wenn Sie die Konsole verwenden, um Ihre Pipeline zu erstellen oder zu bearbeiten, werden Ihre EventBridge Regel und Ihr AWS CloudTrail Trail für Sie erstellt.

Sie erstellen einen Trail wie folgt:

1. Öffnen Sie die - AWS CloudTrail Konsole.
2. Wählen Sie im Navigationsbereich Trails aus.
3. Wählen Sie Create Trail (Trail erstellen) aus. Geben Sie unter Trail name (Trail-Name) einen Namen für Ihren Trail ein.
4. Erstellen oder geben Sie unter Storage location (Speicherort) den Bucket an, der zum Speichern der Protokolldateien verwendet werden soll. Standardmäßig werden Amazon-S3-Buckets und -Objekte als privat eingestuft. Nur der 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-Protokoll-Bucket und Ordner einen Amazon-S3-Bucket und das Objektpräfix (Ordnername) an, um Datenereignisse für alle Objekte im Ordner zu protokollieren. Amazon S3 Sie können jedem Trail 250 Amazon S3-Objekte hinzufügen. Füllen Sie die erforderlichen Verschlüsselungsschlüsselinformationen aus und wählen Sie Weiter aus.
6. Wählen Sie für Ereignistyp die Option Verwaltungsereignisse aus.
7. Wählen Sie für Verwaltungsereignisse die Option Schreiben aus. Der Trail zeichnet die API-Aktivitäten auf Objektebene von Amazon S3 (z. B. `GetObject` und `PutObject`) im angegebenen Bucket und Präfix auf.
8. Wählen Sie Write (Schreiben) aus.
9. Wenn Sie mit dem Trail zufrieden sind, wählen Sie Trail erstellen aus.

So erstellen Sie eine - EventBridge Regel, die auf Ihre Pipeline mit einer Amazon S3-Quelle abzielt

1. Öffnen Sie die Amazon- EventBridge 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 aus. Wählen Sie Regel erstellen aus.
3. Geben Sie unter 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 die Option AWS Ereignisse oder EventBridge Partnerereignisse aus.
6. Wählen Sie unter Beispielereignistyp die Option AWS Ereignisse aus.
7. Geben Sie unter Beispielereignisse S3 als das Schlüsselwort ein, nach dem gefiltert werden soll. Wählen Sie AWS API-Aufruf über aus CloudTrail.
8. Wählen Sie unter Erstellungsmethode die Option Kundenmuster (JSON-Editor) aus.

   Fügen Sie das unten angegebene Ereignismuster ein. Stellen Sie sicher, dass Sie den Bucket-Namen und den S3-Objektschlüssel (oder Schlüsselnamen) hinzufügen, der das Objekt im Bucket eindeutig als identifiziert`requestParameters`. In diesem Beispiel wird eine Regel für einen Bucket namens `my-bucket` und einen Objektschlüssel von erstellt`my-files.zip`. Wenn Sie das Fenster Edit (Bearbeiten) zum Angeben von Ressourcen verwenden, wird Ihre Regel für die Verwendung eines benutzerdefinierten Ereignismusters aktualisiert.

   Im Folgenden finden Sie ein Beispiel-Ereignismuster, das Sie kopieren und einfügen können:

```
{
    "source": [
        "aws.s3"
    ],
    "detail-type": [
        "AWS API Call via CloudTrail"
    ],
    "detail": {
        "eventSource": [
            "s3.amazonaws.com"
        ],
        "eventName": [
            "CopyObject",
            "CompleteMultipartUpload",
```

```
            "PutObject"
        ],
        "requestParameters": {
            "bucketName": [
                "my-bucket"
            ],
            "key": [
                "my-files.zip"
            ]
        }
    }
}
```

9. Wählen Sie Weiter aus.
10. Wählen Sie unter Zieltypen die Option -AWS Service aus.
11. Wählen Sie unter Ziel auswählen die Option ausCodePipeline. Geben Sie unter Pipeline-ARN den Pipeline-ARN für die Pipeline ein, die von 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- 2:80398EXAMPLE:MyFirstPipeline

12. So erstellen oder geben Sie eine IAM-Servicerolle an, die EventBridge Berechtigungen zum Aufrufen des Ziels erteilt, das Ihrer EventBridge Regel zugeordnet ist (in diesem Fall ist das Ziel CodePipeline):

    - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen, um eine Servicerolle zu erstellen, die Ihren Pipeline-Ausführungen Berechtigungen zum Starten erteilt EventBridge.
    - Wählen Sie Vorhandene Rolle verwenden aus, um eine Servicerolle einzugeben, die Ihrem 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).

Erstellen einer - EventBridge Regel für eine Amazon S3-Quelle (CLI)

So erstellen Sie einen - AWS CloudTrail Trail und aktivieren die Protokollierung

Um einen Trail mit der AWS CLI zu erstellen, rufen Sie den create-trail Befehl auf und geben Sie Folgendes an:

- Den Trail-Namen.
- Der Bucket, auf den Sie bereits die Bucket-Richtlinie für AWS CloudTrail angewendet haben.

Weitere Informationen finden Sie unter Erstellen eines Trails mit der AWS Befehlszeilenschnittstelle .

1. Rufen Sie den Befehl create-trail auf und beziehen Sie die Parameter `--name` und `--s3-bucket-name` ein.

   Warum nehme ich diese Änderung vor? Dadurch wird der für Ihren S3-Quell-Bucket erforderliche CloudTrail-Trail erstellt.

   Der folgende Befehl verwendet `--name` und `--s3-bucket-name` zum Erstellen eines Trails mit dem Namen `my-trail` und eines Buckets mit dem Namen `myBucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name myBucket
   ```

2. Rufen Sie den Befehl start-logging auf und beziehen Sie den `--name`-Parameter ein.

   Warum nehme ich diese Änderung vor? Dieser Befehl startet die CloudTrail Protokollierung für Ihren Quell-Bucket und sendet Ereignisse an EventBridge.

   Beispiel:

   Im folgenden Befehl wird `--name` verwendet, um die Protokollierung auf einem Trail mit der Bezeichnung `my-trail` zu starten.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Rufen Sie den Befehl put-event-selectors auf und beziehen Sie die Parameter `--trail-name` und `--event-selectors` ein. Verwenden Sie Ereignisauswahlen, um anzugeben, dass Ihr Trail Datenereignisse für Ihren Quell-Bucket protokollieren und die Ereignisse an die EventBridge Regel senden soll.

Warum nehme ich diese Änderung vor? Dieser Befehl filtert Ereignisse.

Beispiel:

Im folgenden Beispielbefehl werden `--trail-name` und `--event-selectors` verwendet, um die Verwaltung von Datenereignissen für einen Quell-Bucket und einen Präfix namens `myBucket/myFolder` anzugeben.

```
aws cloudtrail put-event-selectors --trail-name my-trail --event-selectors
 '[{ "ReadWriteType": "WriteOnly", "IncludeManagementEvents":false,
 "DataResources": [{ "Type": "AWS::S3::Object", "Values": ["arn:aws:s3:::myBucket/
myFolder/file.zip"] }] }]'
```

So erstellen Sie eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel und wenden die Berechtigungsrichtlinie an

1. Erteilen Sie Berechtigungen für EventBridge , um zum Aufrufen der Regel CodePipeline zu verwenden. Weitere Informationen finden Sie unter Verwenden von ressourcenbasierten Richtlinien für Amazon EventBridge.

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie EventBridge zu erstellen, damit die Servicerolle übernehmen kann. Geben Sie ihr den Namen `trustpolicyforEB.json`.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

   b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

Warum nehme ich diese Änderung vor? Wenn Sie diese Vertrauensrichtlinie zur Rolle hinzufügen, werden Berechtigungen für erstellt EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie hier für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Führen Sie den folgenden Befehl aus, um der erstellten `Role-for-MyRule`-Rolle die neue `CodePipeline-Permissions-Policy-for-EB`-Berechtigungsrichtlinie anzufügen.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

   Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „`MyS3SourceRule`“ erstellt.

```
aws events put-rule --name "MyS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"AWS API Call via CloudTrail\"],\"detail\":
{\"eventSource\":[\"s3.amazonaws.com\"],\"eventName\":[\"CopyObject\",\"PutObject
\",\"CompleteMultipartUpload\"],\"requestParameters\":{\"bucketName\":[\"my-bucket
\"],\"key\":[\"my-key\"]}}}
```

```
  --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Um CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und schließen Sie die `--targets` Parameter `--rule` und ein.

   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
   ```

So bearbeiten Sie den PollForSourceChanges Parameter Ihrer Pipeline

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem Texteditor und bearbeiten Sie die Quellphase, indem Sie den Parameter `PollForSourceChanges` für einen Bucket mit dem Namen `storage-bucket`in `false` ändern wie in diesem Beispiel gezeigt.

Warum nehme ich diese Änderung vor? Durch Festlegen dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

```
"configuration": {
    "S3Bucket": "storage-bucket",
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

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

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

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

   Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

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

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

   Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den start-pipeline-execution-Befehl, um Ihre Pipeline manuell zu starten.

Erstellen einer - EventBridge Regel für eine Amazon S3-Quelle (AWS CloudFormation Vorlage)

Um zum Erstellen einer Regel AWS CloudFormation zu verwenden, aktualisieren Sie Ihre Vorlage wie hier gezeigt.

So erstellen Sie eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel und wenden die Berechtigungsrichtlinie an

1. Verwenden Sie in der Vorlage unter die Ressource `Resources`, `AWS::IAM::Role` AWS CloudFormation um die IAM-Rolle zu konfigurieren, die es Ihrem Ereignis 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? Durch das Hinzufügen von `AWS::IAM::Role` Ressourcen kann Berechtigungen für AWS CloudFormation erstellen EventBridge. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

   ```
   EventRole:
     Type: AWS::IAM::Role
     Properties:
       AssumeRolePolicyDocument:
         Version: 2012-10-17
         Statement:
           -
   ```

```
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
 'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]


...
```

JSON

```
  "EventRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": [
                "events.amazonaws.com"
              ]
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "Path": "/",
      "Policies": [
        {
```

```
          "PolicyName": "eb-pipeline-execution",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Action": "codepipeline:StartPipelineExecution",
                "Resource": {
                  "Fn::Join": [
                    "",
                    [
                      "arn:aws:codepipeline:",
                      {
                        "Ref": "AWS::Region"
                      },
                      ":",
                      {
                        "Ref": "AWS::AccountId"
                      },
                      ":",
                      {
                        "Ref": "AppPipeline"
                      }
                    ]
                  ]

...
```

2. Verwenden Sie die `AWS::Events::Rule` AWS CloudFormation Ressource , um eine - EventBridge Regel hinzuzufügen. Dieses Ereignismuster erstellt ein Ereignis`CopyObject`, das und `PutObject` `CompleteMultipartUpload` auf Ihrem 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 das Hinzufügen der `AWS::Events::Rule` Ressource kann das Ereignis AWS CloudFormation erstellen. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

   ```
     EventRule:
   ```

```
    Type: AWS::Events::Rule
    Properties:
      EventPattern:
        source:
          - aws.s3
        detail-type:
          - 'AWS API Call via CloudTrail'
        detail:
          eventSource:
            - s3.amazonaws.com
          eventName:
            - CopyObject
            - PutObject
            - CompleteMultipartUpload
          requestParameters:
            bucketName:
              - !Ref SourceBucket
            key:
              - !Ref SourceObjectKey
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline


...
```

JSON

```
  "EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
      "EventPattern": {
        "source": [
          "aws.s3"
        ],
        "detail-type": [
          "AWS API Call via CloudTrail"
        ],
```

```
        "detail": {
          "eventSource": [
            "s3.amazonaws.com"
          ],
          "eventName": [
            "CopyObject",
            "PutObject",
            "CompleteMultipartUpload"
          ],
          "requestParameters": {
            "bucketName": [
              {
                "Ref": "SourceBucket"
              }
            ],
            "key": [
              {
                "Ref": "SourceObjectKey"
              }
            ]
          }
        }
      },
      "Targets": [
        {
          "Arn": {
            "Fn::Join": [
              "",
              [
                "arn:aws:codepipeline:",
                {
                  "Ref": "AWS::Region"
                },
                ":",
                {
                  "Ref": "AWS::AccountId"
                },
                ":",
                {
                  "Ref": "AppPipeline"
                }
              ]
            ]
          },
```

```
          "RoleArn": {
            "Fn::GetAtt": [
              "EventRole",
              "Arn"
            ]
          },
          "Id": "codepipeline-AppPipeline"
        }
      ]
    }
  }
},

...
```

3. Fügen Sie diesen Ausschnitt zu Ihrer ersten Vorlage hinzu, um Stack-übergreifende Funktionalität zu ermöglichen:

   YAML

   ```
   Outputs:
     SourceBucketARN:
       Description: "S3 bucket ARN that Cloudtrail will use"
       Value: !GetAtt SourceBucket.Arn
       Export:
         Name: SourceBucketARN
   ```

   JSON

   ```
     "Outputs" : {
       "SourceBucketARN" : {
         "Description" : "S3 bucket ARN that Cloudtrail will use",
         "Value" : { "Fn::GetAtt": ["SourceBucket", "Arn"] },
         "Export" : {
           "Name" : "SourceBucketARN"
         }
       }

   ...
   ```

4. Speichern Sie Ihre aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die - AWS CloudFormation Konsole.

5. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
6. Laden Sie Ihre aktualisierte Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
7. Wählen Sie Execute (Ausführen).

So bearbeiten Sie den PollForSourceChanges Parameter Ihrer Pipeline

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

  Warum nehme ich diese Änderung vor? Durch Ändern von `PollForSourceChanges` in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

  YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
```

```
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
```

JSON

```
{
   "Name": "SourceAction",
   "ActionTypeId": {
     "Category": "Source",
     "Owner": "AWS",
     "Version": 1,
     "Provider": "S3"
   },
   "OutputArtifacts": [
     {
       "Name": "SourceOutput"
     }
   ],
   "Configuration": {
     "S3Bucket": {
       "Ref": "SourceBucket"
     },
     "S3ObjectKey": {
       "Ref": "SourceObjectKey"
     },
     "PollForSourceChanges": false
   },
   "RunOrder": 1
 }
```

So erstellen Sie eine zweite Vorlage für die CloudTrail Ressourcen Ihrer Amazon S3-Pipeline

- Verwenden Sie in einer separaten Vorlage unter die `AWS::CloudTrail::Trail` AWS CloudFormation Ressourcen `AWS::S3::BucketPolicy`, und `AWS::S3::BucketResources`, um eine einfache Bucket-Definition und einen Trail für bereitzustellen CloudTrail.

Warum nehme ich diese Änderung vor? Angesichts des aktuellen Limits von fünf Trails pro Konto muss der CloudTrail Trail separat erstellt und verwaltet werden. (Siehe Limits in AWS CloudTrail.) Sie können jedoch viele Amazon S3-Buckets in einen einzelnen 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
      }
    }
  }
}

...
```

Bitbucket-Cloud-Verbindungen

Mit Verbindungen können Sie Konfigurationen autorisieren und einrichten, die Ihren Drittanbieter mit Ihren - AWS Ressourcen verknüpfen. Um Ihr Drittanbieter-Repository als Quelle für Ihre Pipeline zuzuordnen, verwenden Sie eine -Verbindung.

> **Note**
>
> Diese Funktion ist in den Regionen Asien-Pazifik (Hongkong), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Asien-Pazifik (Osaka), Afrika (Kapstadt), Naher Osten (Bahrain), Naher Osten (VAE), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv) oder AWS GovCloud (USA-West) nicht verfügbar. Informationen zum Verweisen auf andere verfügbare Aktionen finden Sie unter Produkt- und Serviceintegrationen mit CodePipeline. Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie im

Hinweis unter CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen.

Um eine Bitbucket-Cloud-Quellaktion in hinzuzufügen CodePipeline, können Sie entweder:

- Wählen Sie auf der Seite Pipeline erstellen oder Aktion bearbeiten der CodePipeline Konsole die Option Bitbucket-Anbieter aus. Informationen Erstellen einer Verbindung mit Bitbucket Cloud (Konsole) zum Hinzufügen der Aktion finden Sie 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 , Erstellen einer Verbindung mit Bitbucket Cloud (CLI) um eine Verbindungsressource mit der CLI zu erstellen.
  - Verwenden Sie die `CreateSourceConnection` Beispielaktionskonfiguration in CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen, um Ihre Aktion hinzuzufügen, wie in gezeigtErstellen einer Pipeline (CLI).

> **Note**
>
> Sie können eine Verbindung auch über die Entwicklertools-Konsole unter Einstellungen erstellen. Weitere Informationen finden Sie unter Erstellen einer Verbindung.

Bevor Sie beginnen:

- Sie müssen ein Konto beim Anbieter des Drittanbieter-Repositorys erstellt haben, z. B. Bitbucket Cloud.
- Sie müssen 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, das zum Erstellen der Verbindung verwendet wurde.
> Wenn die Anwendung in einem Bitbucket-Cloud-Workspace installiert wird, benötigen Sie die Berechtigungen Workspace verwalten. Andernfalls wird die Option zum Installieren der App nicht angezeigt.

Themen

- Erstellen einer Verbindung mit Bitbucket Cloud (Konsole)
- Erstellen einer Verbindung mit Bitbucket Cloud (CLI)

Erstellen einer Verbindung mit Bitbucket Cloud (Konsole)

Gehen Sie wie folgt vor, um mit der CodePipeline Konsole eine Verbindungsaktion für Ihr Bitbucket-Repository hinzuzufügen.

> **Note**
>
> Sie können Verbindungen mit einem Bitbucket-Cloud-Repository erstellen. Installierte Bitbucket-Anbietertypen wie Bitbucket Server werden nicht unterstützt.

Schritt 1: Erstellen oder Bearbeiten Ihrer Pipeline

So 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 , um eine Pipeline zu erstellen. Folgen Sie den Schritten unter Erstellen einer Pipeline, um den ersten Bildschirm abzuschließen und Weiter zu wählen. Wählen Sie auf der Seite Quelle unter Quellanbieter die Option Bitbucket aus.
   - Wählen Sie , um eine vorhandene Pipeline zu bearbeiten. Wählen Sie Bearbeiten und dann Stufe bearbeiten aus. Wählen Sie , um Ihre Quellaktion hinzuzufügen oder zu bearbeiten.

Geben Sie auf der Seite Aktion bearbeiten unter Aktionsname den Namen für Ihre Aktion ein. Wählen Sie unter Aktionsanbieter die Option Bitbucket aus.

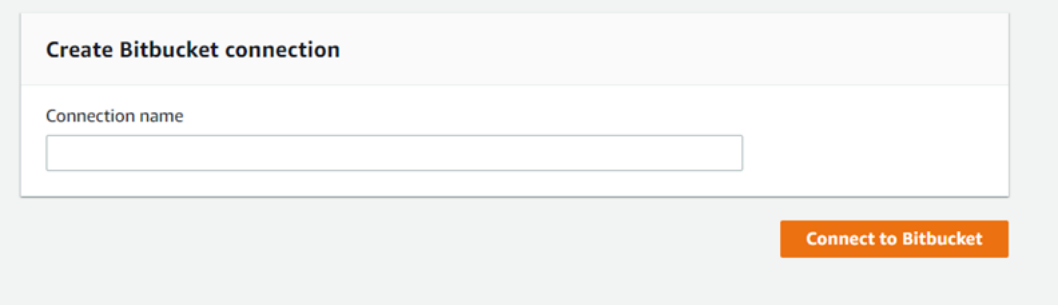
3. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter erstellt haben, wählen Sie unter Verbindung die Option Mit Bitbucket verbinden aus. Fahren Sie mit Schritt 2: Erstellen einer Verbindung zu Bitbucket fort.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter erstellt haben, wählen Sie die Verbindung aus. Fahren Sie mit Schritt 3: Speichern der Quellaktion für Ihre Verbindung fort.

Schritt 2: Erstellen einer Verbindung mit Bitbucket Cloud

So erstellen Sie eine Verbindung zu Bitbucket Cloud

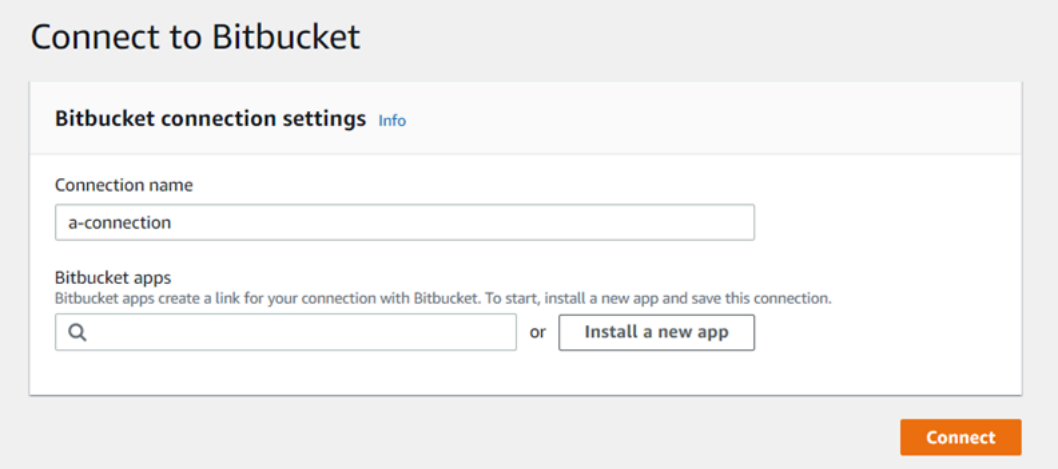
1. Geben Sie auf der Seite Mit Bitbucket verbinden Ihren Verbindungsnamen ein und wählen Sie Mit Bitbucket verbinden aus.



   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.

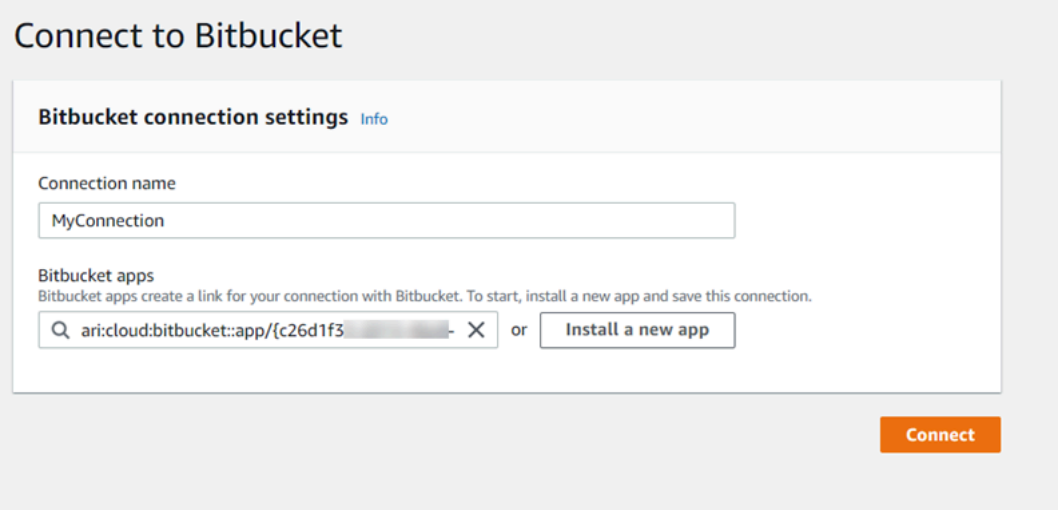
> **Note**
>
> Sie installieren die App nur einmal für jeden Bitbucket-Cloud-Workspace oder -Konto. Wenn Sie die Bitbucket-App bereits installiert haben, wählen Sie sie aus und fahren Sie mit Schritt 4 fort.



3. Wenn die Anmeldeseite für Bitbucket Cloud angezeigt wird, melden Sie sich mit Ihren - Anmeldeinformationen an und fahren Sie dann fort.
4. Auf der App-Installationsseite wird eine Meldung angezeigt, dass die AWS CodeStar App versucht, eine Verbindung zu Ihrem 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: Speichern der Bitbucket-Cloud-Quellaktion

Führen Sie diese Schritte auf dem Assistenten oder der Seite Aktion bearbeiten aus, um Ihre Quellaktion mit Ihren Verbindungsinformationen zu speichern.

So schließen Sie die Quellaktion ab und speichern sie mit Ihrer Verbindung

1. Wählen Sie unter Repository name (Repository-Name) den Namen Ihres Drittanbieter-Repositorys aus.
2. Unter Pipeline-Auslöser können Sie Auslöser hinzufügen, wenn Ihre Aktion eine CodeConnections Aktion ist. Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration und zum optionalen Filtern mit Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen.
3. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

   - Um Ausgabeartefakte aus der Bitbucket-Cloud-Aktion mit der Standardmethode zu speichern, wählen Sie 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 CodeBuild nachgelagerten Aktionen verwendet werden.

     Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservicerolle aktualisieren, wie in gezeigtHinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder GitLab.com.
4. Wählen Sie im Assistenten Weiter oder Speichern auf der Seite Aktion bearbeiten aus.

Erstellen einer Verbindung mit Bitbucket Cloud (CLI)

Sie können die AWS Command Line Interface (AWS CLI) verwenden, um eine Verbindung zu erstellen.

> **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 über die AWS CLI oder erstellte Verbindung AWS CloudFormation befindet sich standardmäßig im `PENDING` Status . Nachdem Sie eine Verbindung mit der CLI oder erstellt haben AWS CloudFormation, verwenden Sie die Konsole, um die Verbindung so zu bearbeiten, dass sie ihren Status hat`AVAILABLE`.

So stellen Sie eine Verbindung her

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie die , AWS CLI um den create-connection Befehl auszuführen, und geben Sie die `--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 am Code-Push in das Verbindungsquellen-Repository. Führen Sie einen der folgenden Schritte aus, um die Pipeline-Auslöserkonfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

   - Um die Pipeline-Auslöserkonfiguration so zu konfigurieren, dass sie nur mit einer manuellen Version beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

     ```
     "DetectChanges": "false",
     ```

   - Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration zum Filtern nach Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen. Im Folgenden werden beispielsweise Git-Tags zur Pipeline-Ebene der Pipeline-JSON-Definition hinzugefügt. In diesem Beispiel `release-v1` sind `release-v0` und die einzuschließenden Git-Tags und `release-v2` ist das auszuschließende Git-Tag.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
     ```

CodeCommit -Quellaktionen und EventBridge

Um eine CodeCommit Quellaktion in hinzuzufügen CodePipeline, können Sie entweder:

- Wählen Sie auf der Seite Pipeline CodePipeline erstellen (Erstellen einer Pipeline (Konsole)) oder Aktion bearbeiten die CodeCommit Anbieteroption aus. Die Konsole erstellt eine EventBridge - Regel, die Ihre Pipeline startet, wenn sich die Quelle ändert.
- Verwenden Sie die AWS CLI , um die Aktionskonfiguration für die `CodeCommit` Aktion hinzuzufügen und zusätzliche Ressourcen wie folgt zu erstellen:
  - Verwenden Sie die `CodeCommit` Beispielaktionskonfiguration in CodeCommit, um Ihre Aktion zu erstellen, wie in gezeigtErstellen einer Pipeline (CLI).
  - Die Methode zur Erkennung von Änderungen startet standardmäßig die Pipeline, indem die Quelle abgefragt wird. Sie sollten regelmäßige Prüfungen deaktivieren und die Änderungserkennungsregel manuell erstellen. Verwenden Sie eine der folgenden Methoden: Erstellen einer - EventBridge Regel für eine CodeCommit Quelle (Konsole)Erstellen einer - EventBridge Regel für eine CodeCommit Quelle (CLI), oder Erstellen einer - EventBridge Regel für eine CodeCommit Quelle (AWS CloudFormation Vorlage) .

Themen

- Erstellen einer - EventBridge Regel für eine CodeCommit Quelle (Konsole)
- Erstellen einer - EventBridge Regel für eine CodeCommit Quelle (CLI)
- Erstellen einer - EventBridge Regel für eine CodeCommit Quelle (AWS CloudFormation Vorlage)

Erstellen einer - 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.

So erstellen Sie eine - EventBridge Regel für die Verwendung in - CodePipeline Operationen

1. Öffnen Sie die Amazon- EventBridge 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 aus. Wählen Sie Regel erstellen aus.
3. Geben Sie unter 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 die Option AWS Ereignisse oder EventBridge Partnerereignisse aus.
6. Wählen Sie unter Beispielereignistyp die Option AWS Ereignisse aus.
7. Geben Sie unter Beispielereignisse CodeCommit als 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 ausCodePipeline.
10. Geben Sie den Pipeline-ARN für die Pipeline ein, die von 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 geben Sie eine IAM-Servicerolle an, die EventBridge Berechtigungen zum Aufrufen des Ziels erteilt, das Ihrer EventBridge Regel zugeordnet ist (in diesem Fall ist das Ziel CodePipeline):
    - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen, um eine Servicerolle zu erstellen, die Ihren Pipeline-Ausführungen Berechtigungen zum Starten erteilt EventBridge.
    - Wählen Sie Vorhandene Rolle verwenden, um eine Servicerolle einzugeben, die Ihren Pipeline-Ausführungen EventBridge Berechtigungen erteilt.
12. Wählen Sie Weiter aus.
13. Wählen Sie auf der Seite Tags die Option Weiter aus.
14. Überprüfen Sie auf der Seite Überprüfen und erstellen die Regelkonfiguration. Wenn Sie mit der Regel zufrieden sind, klicken Sie auf Create rule (Regel erstellen).

Erstellen einer - 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, mit denen Sie erstellen, die Ihrem AWS Konto CodePipeline zugeordnet sind, eindeutig sein.
- Das Ereignismuster für die Quelle und von der Regel verwendete Detailfelder. Weitere Informationen finden Sie unter Amazon EventBridge und Ereignismuster.

So erstellen Sie eine - EventBridge Regel mit CodeCommit als Ereignisquelle und CodePipeline als Ziel

1. Fügen Sie Berechtigungen für hinzu EventBridge , um zum Aufrufen der Regel CodePipeline zu verwenden. Weitere Informationen finden Sie unter Verwenden von ressourcenbasierten Richtlinien für Amazon EventBridge.
    a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie EventBridge zu erstellen, die die Übernahme der Servicerolle ermöglicht. Geben Sie der Vertrauensrichtlinie den Namen `trustpolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie in diesem Beispiel für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Verwenden Sie den folgenden Befehl, um die Berechtigungsrichtlinie `CodePipeline-Permissions-Policy-for-EB` der Rolle `Role-for-MyRule` anzufügen.

Warum nehme ich diese Änderung vor? Wenn Sie diese Richtlinie der Rolle hinzufügen, werden Berechtigungen für erstellt EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

   Warum nehme ich diese Änderung vor? Dieser Befehl aktiviert AWS CloudFormation , um das Ereignis zu erstellen.

   Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „`MyCodeCommitRepoRule`“ erstellt.

   ```
   aws events put-rule --name "MyCodeCommitRepoRule" --event-pattern "{\"source\":
   [\"aws.codecommit\"],\"detail-type\":[\"CodeCommit Repository State Change\"],
   \"resources\":[\"repository-ARN\"],\"detail\":{\"referenceType\":[\"branch\"],
   \"referenceName\":[\"main\"]}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-
   MyRule"
   ```

3. Um CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und schließen 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
   ```

So bearbeiten Sie den PollForSourceChanges Parameter Ihrer Pipeline

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem beliebigen Texteditor und bearbeiten Sie die Quellstufe, indem Sie den Parameter `PollForSourceChanges` in `false` ändern, wie in diesem Beispiel gezeigt.

   Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

   ```
   "configuration": {
       "PollForSourceChanges": "false",
       "BranchName": "main",
       "RepositoryName": "MyTestRepo"
   },
   ```

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

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

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

Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

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

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

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den **`start-pipeline-execution`**-Befehl, um Ihre Pipeline manuell zu starten.

Erstellen einer - EventBridge Regel für eine CodeCommit Quelle (AWS CloudFormation Vorlage)

Um zum Erstellen einer Regel AWS CloudFormation zu verwenden, aktualisieren Sie Ihre Vorlage wie hier gezeigt.

So aktualisieren Sie Ihre Pipeline- AWS CloudFormation Vorlage und erstellen eine EventBridge Regel

1. Verwenden Sie in der Vorlage unter die Ressource `Resources`, `AWS::IAM::Role` AWS CloudFormation um die IAM-Rolle zu konfigurieren, die es Ihrem Ereignis 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? Durch das Hinzufügen der `AWS::IAM::Role` Ressource kann Berechtigungen für AWS CloudFormation erstellen EventBridge. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

```
EventRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        -
          Effect: Allow
          Principal:
            Service:
              - events.amazonaws.com
          Action: sts:AssumeRole
    Path: /
    Policies:
      -
        PolicyName: eb-pipeline-execution
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            -
              Effect: Allow
              Action: codepipeline:StartPipelineExecution
              Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
```

JSON

```
"EventRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "events.amazonaws.com"
            ]
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "eb-pipeline-execution",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": "codepipeline:StartPipelineExecution",
              "Resource": {
                "Fn::Join": [
                  "",
                  [
                    "arn:aws:codepipeline:",
                    {
                      "Ref": "AWS::Region"
                    },
                    ":",
                    {
                      "Ref": "AWS::AccountId"
                    },
                    ":",
                    {
```

```
                    "Ref": "AppPipeline"
                  }
                ]

...
```

2. Verwenden Sie in der Vorlage unter die Ressource `Resources`, `AWS::Events::Rule` AWS CloudFormation um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erstellt ein Ereignis, das Push-Änderungen an Ihrem Repository überwacht. Wenn eine Repository-Statusänderung EventBridge erkennt, wird die Regel `StartPipelineExecution` für Ihre Ziel-Pipeline aufgerufen.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::Events::Rule` Ressource kann das Ereignis AWS CloudFormation erstellen. Diese Ressource wird Ihrem AWS CloudFormation Stack hinzugefügt.

   YAML

```
 EventRule:
   Type: AWS::Events::Rule
   Properties:
     EventPattern:
       source:
         - aws.codecommit
       detail-type:
         - 'CodeCommit Repository State Change'
       resources:
         - !Join [ '', [ 'arn:aws:codecommit:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref RepositoryName ] ]
       detail:
         event:
           - referenceCreated
           - referenceUpdated
         referenceType:
           - branch
         referenceName:
           - main
     Targets:
       -
         Arn:
           !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
```

```
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
```

JSON

```
"EventRule": {
  "Type": "AWS::Events::Rule",
  "Properties": {
    "EventPattern": {
      "source": [
        "aws.codecommit"
      ],
      "detail-type": [
        "CodeCommit Repository State Change"
      ],
      "resources": [
        {
          "Fn::Join": [
            "",
            [
              "arn:aws:codecommit:",
              {
                "Ref": "AWS::Region"
              },
              ":",
              {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "RepositoryName"
              }
            ]
          ]
        }
      ],
      "detail": {
        "event": [
          "referenceCreated",
          "referenceUpdated"
        ],
        "referenceType": [
```

```
          "branch"
        ],
        "referenceName": [
          "main"
        ]
      }
    },
    "Targets": [
      {
        "Arn": {
          "Fn::Join": [
            "",
            [
              "arn:aws:codepipeline:",
              {
                "Ref": "AWS::Region"
              },
              ":",
              {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "AppPipeline"
              }
            ]
          ]
        },
        "RoleArn": {
          "Fn::GetAtt": [
            "EventRole",
            "Arn"
          ]
        },
        "Id": "codepipeline-AppPipeline"
      }
    ]
  }
},
```

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation -Konsole.

4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

So bearbeiten Sie den PollForSourceChanges Parameter Ihrer Pipeline

> **Important**
>
> In vielen Fällen ist der Parameter `PollForSourceChanges` „true“, wenn Sie eine Pipeline erstellen. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

  Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

  YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: CodeCommit
            OutputArtifacts:
              - Name: SourceOutput
```

```
            Configuration:
              BranchName: !Ref BranchName
              RepositoryName: !Ref RepositoryName
              PollForSourceChanges: false
            RunOrder: 1
```

JSON

```
{
  "Name": "Source",
  "Actions": [
    {
      "Name": "SourceAction",
      "ActionTypeId": {
        "Category": "Source",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "CodeCommit"
      },
      "OutputArtifacts": [
        {
          "Name": "SourceOutput"
        }
      ],
      "Configuration": {
        "BranchName": {
          "Ref": "BranchName"
        },
        "RepositoryName": {
          "Ref": "RepositoryName"
        },
        "PollForSourceChanges": false
      },
      "RunOrder": 1
    }
  ]
},
```

GitHub Verbindungen

Sie verwenden Verbindungen, um Konfigurationen zu autorisieren und einzurichten, die Ihren Drittanbieter mit Ihren -AWSRessourcen verknüpfen.

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

Um eine Quellaktion für Ihr - GitHub oder GitHub Enterprise Cloud-Repository in hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Wählen Sie auf der Seite Pipeline erstellen oder Aktion bearbeiten der CodePipeline Konsole die Option GitHub (Version 2)-Anbieter aus. Siehe Erstellen einer Verbindung zu GitHub Enterprise Server (Konsole), um die Aktion hinzuzufügen. Die Konsole hilft Ihnen beim Erstellen einer Verbindungsressource.

  > **Note**
  >
  > Ein Tutorial, das Sie durch das Hinzufügen einer GitHub Verbindung führt und die Option Vollständiger Klon in Ihrer Pipeline verwendet, finden Sie unter Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

- Verwenden Sie die -CLI, um die Aktionskonfiguration für die `CodeStarSourceConnection` Aktion beim `GitHub` Anbieter mit den unter gezeigten CLI-Schritten hinzuzufügenErstellen einer Pipeline (CLI).

> **Note**
>
> Sie können eine Verbindung auch mithilfe der Entwicklertools-Konsole unter Einstellungen erstellen. Weitere Informationen finden Sie unter Erstellen einer Verbindung.

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 zu verwenden. Anweisungen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.

> **Note**
>
> Um die Verbindung zu erstellen, müssen Sie der Eigentümer der GitHub Organisation sein. Bei Repositorys, die keiner Organisation angehören, müssen Sie der Repository-Besitzer sein.

Themen

- Erstellen einer Verbindung zu GitHub (Konsole)
- Erstellen einer Verbindung zu GitHub (CLI)

Erstellen einer Verbindung zu GitHub (Konsole)

Gehen Sie wie folgt vor, um mithilfe der CodePipeline Konsole eine Verbindungsaktion für Ihr GitHub - oder GitHub Enterprise Cloud-Repository hinzuzufügen.

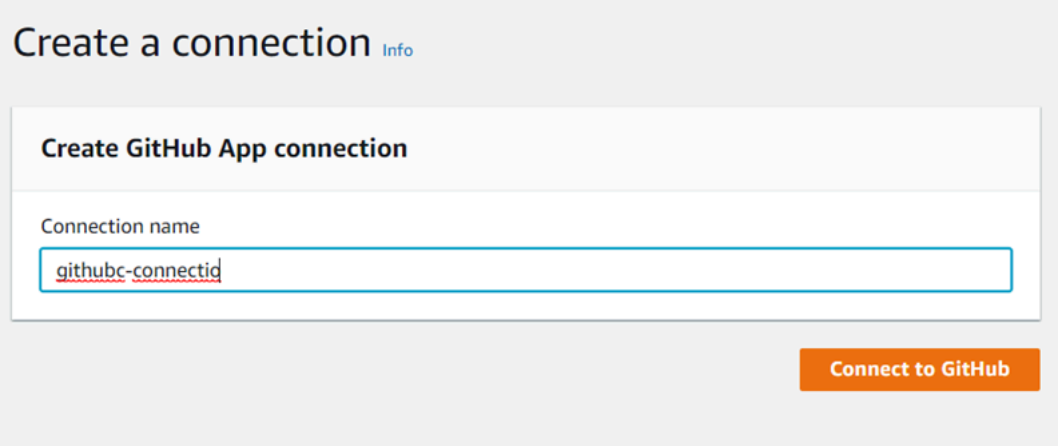
> **Note**
>
> In diesen Schritten können Sie bestimmte Repositorys unter Repository-Zugriff auswählen. Auf alle nicht ausgewählten Repositorys kann nicht zugegriffen werden oder sie sind von nicht sichtbar CodePipeline.

Schritt 1: Erstellen oder Bearbeiten Ihrer Pipeline

1. Melden Sie sich bei der - CodePipeline Konsole an.
2. Wählen Sie eine der folgenden Optionen aus.
   - Wählen Sie , um eine Pipeline zu erstellen. Folgen Sie den Schritten unter Erstellen einer Pipeline, um den ersten Bildschirm abzuschließen und Weiter zu wählen. Wählen Sie auf der Seite Quelle unter Quellanbieter GitHub (Version 2) aus.
   - Wählen Sie , um eine vorhandene Pipeline zu bearbeiten. Wählen Sie Bearbeiten und dann Stufe bearbeiten aus. Wählen Sie , um Ihre Quellaktion hinzuzufügen oder zu bearbeiten. Geben Sie auf der Seite Aktion bearbeiten unter Aktionsname den Namen für Ihre Aktion ein. Wählen Sie unter Aktionsanbieter GitHub (Version 2) aus.
3. Führen Sie eine der folgenden Aktionen aus:
   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter erstellt haben, wählen Sie unter Verbindung die Option Mit verbinden aus GitHub. Fahren Sie mit Schritt 2: Erstellen einer Verbindung zu fort GitHub.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter erstellt haben, wählen Sie die Verbindung aus. Fahren Sie mit Schritt 3: Speichern der Quellaktion für Ihre Verbindung fort.

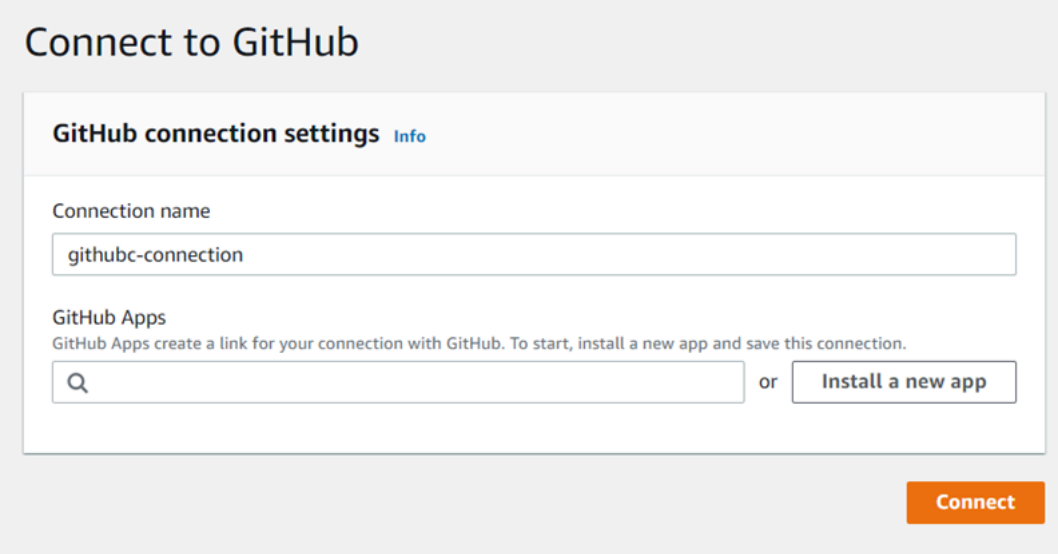
Schritt 2: Erstellen einer Verbindung zu GitHub

Nachdem Sie die Verbindung erstellt haben, wird die Seite Mit verbinden GitHub angezeigt.



So erstellen Sie eine Verbindung zu GitHub

1. Unter GitHub Verbindungseinstellungen wird Ihr Verbindungsname unter Verbindungsname angezeigt. Wählen Sie Mit verbinden aus GitHub. Die Seite für die Zugriffsanforderung wird angezeigt.
2. Wählen Sie AWSKonnektor autorisieren für GitHub. Auf der Verbindungsseite wird das Feld GitHub Apps angezeigt.



3. Wählen Sie unter GitHub Apps eine App-Installation aus oder wählen Sie Neue App installieren, um eine zu erstellen.

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

4. Wählen Sie auf der Seite AWSKonnektor installieren für GitHub 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 Seite AWSKonnektor installieren für GitHub die Standardwerte bei und wählen Sie Installieren aus.
6. Auf der Seite Mit verbinden GitHub wird die Verbindungs-ID für Ihre neue Installation in GitHub Apps angezeigt. Wählen Sie Connect aus.

Schritt 3: Speichern Ihrer GitHub Quellaktion

Führen Sie diese Schritte auf der Seite Aktion bearbeiten aus, um Ihre Quellaktion mit Ihren Verbindungsinformationen zu speichern.

So speichern Sie Ihre GitHub Quellaktion

1. Wählen Sie unter Repository name (Repository-Name) den Namen Ihres Drittanbieter-Repositorys aus.
2. Unter Pipeline-Auslöser können Sie Auslöser hinzufügen, wenn Ihre Aktion eine CodeConnections Aktion ist. Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration und zum optionalen Filtern mit Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen.
3. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

   - Um Ausgabeartefakte aus 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 Projektservicerolle aktualisieren, wie in gezeigtHinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder GitLab.com. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiger Klon verwenden, finden Sie unter Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

4. Wählen Sie im Assistenten Weiter oder Speichern auf der Seite Aktion bearbeiten aus.

Erstellen einer Verbindung zu GitHub (CLI)

Sie können mit der AWS Command Line Interface (AWS CLI, Befehlszeilenschnittstelle) eine Verbindung erstellen.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über die AWS CLI oder AWS CloudFormation erstellt wird, ist standardmäßig im Status `PENDING` (Ausstehend). Nachdem Sie mit der CLI oder mit AWS CloudFormation eine Verbindung erstellt haben, bearbeiten Sie die Verbindung mit der Konsole, um ihren Status auf `AVAILABLE` zu setzen.

So stellen Sie eine Verbindung her

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie die AWS CLI, um den Befehl create-connection auszuführen, und geben Sie die Parameter `--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 am Code-Push in das Verbindungsquellen-Repository. Führen Sie einen der folgenden Schritte aus, um die Pipeline-Auslöserkonfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

   - Um die Pipeline-Auslöserkonfiguration so zu konfigurieren, dass sie nur mit einer manuellen Version beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

     ```
     "DetectChanges": "false",
     ```

   - Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration zum Filtern nach Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen. Im Folgenden wird beispielsweise die Pipeline-Ebene der Pipeline-JSON-Definition erweitert. In diesem Beispiel `release-v1` sind `release-v0` und die einzuschließenden Git-Tags und `release-v2` das auszuschließende Git-Tag.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
     ```

GitHub Enterprise Server-Verbindungen

Mit Verbindungen können Sie Konfigurationen autorisieren und einrichten, die Ihren Drittanbieter mit Ihren -AWSRessourcen verknüpfen. Um Ihr Drittanbieter-Repository als Quelle für Ihre Pipeline zuzuordnen, verwenden Sie eine -Verbindung.

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

Um eine GitHub Enterprise Server-Quellaktion in hinzuzufügen CodePipeline, können Sie eine der folgenden Optionen wählen:

- Wählen Sie auf der Seite Pipeline erstellen oder Aktion bearbeiten der CodePipeline Konsole die Option GitHub Enterprise-Server-Anbieter aus. Siehe Erstellen einer Verbindung zu GitHub Enterprise Server (Konsole), um die Aktion hinzuzufügen. Die Konsole hilft Ihnen beim Erstellen einer Hostressource und einer Verbindungsressource.
- Verwenden Sie die CLI, um die Aktionskonfiguration für die `CreateSourceConnection` Aktion beim `GitHubEnterpriseServer` Anbieter hinzuzufügen und Ihre Ressourcen zu erstellen:
  - Informationen zum Erstellen Ihrer Verbindungsressourcen finden Sie unter , Erstellen eines Hosts und einer Verbindung mit GitHub Enterprise Server (CLI) um eine Hostressource und eine Verbindungsressource mit der CLI zu erstellen.
  - Verwenden Sie die `CreateSourceConnection` Beispielaktionskonfiguration in CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen, um Ihre Aktion hinzuzufügen, wie in gezeigtErstellen einer Pipeline (CLI).

> **Note**
>
> Sie können eine Verbindung auch mithilfe der Entwicklertools-Konsole unter Einstellungen erstellen. Weitere Informationen finden Sie unter Erstellen einer Verbindung.

Bevor Sie beginnen:

- Sie müssen ein Konto bei GitHub Enterprise Server erstellt und die GitHub Enterprise Server-Instance auf Ihrer Infrastruktur installiert haben.

  > **Note**
  >
  > Jede VPC kann jeweils nur einem Host (GitHub Enterprise Server-Instance) zugeordnet werden.

- Sie müssen bereits ein Code-Repository mit GitHub Enterprise Server erstellt haben.

Themen

- Erstellen einer Verbindung zu GitHub Enterprise Server (Konsole)
- Erstellen eines Hosts und einer Verbindung mit GitHub Enterprise Server (CLI)

Erstellen einer Verbindung zu GitHub Enterprise Server (Konsole)

Gehen Sie wie folgt vor, um mit 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, das zum Erstellen der Verbindung verwendet wurde.

Bevor Sie beginnen:

Für eine Host-Verbindung zu GitHub Enterprise Server müssen Sie die Schritte zum Erstellen einer Host-Ressource für Ihre Verbindung abgeschlossen haben. Weitere Informationen finden Sie unter Verwalten von Hosts für Verbindungen.

Schritt 1: Erstellen oder Bearbeiten Ihrer Pipeline

So 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 , um eine Pipeline zu erstellen. Führen Sie die Schritte unter Pipeline erstellen aus, um den ersten Bildschirm abzuschließen und Weiter zu wählen. Wählen Sie auf der Seite Quelle unter Quellanbieter die Option GitHub Enterprise Server aus.
   - Wählen Sie , um eine vorhandene Pipeline zu bearbeiten. Wählen Sie Bearbeiten und dann Stufe bearbeiten aus. Wählen Sie , um Ihre Quellaktion hinzuzufügen oder zu bearbeiten. 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 erstellt haben, wählen Sie unter Verbindung die Option Mit GitHub Enterprise Server verbinden aus. Fahren Sie mit Schritt 2: Erstellen einer Verbindung zu GitHub Enterprise Server fort.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter erstellt haben, wählen Sie die Verbindung aus. Fahren Sie mit Schritt 3: Speichern der Quellaktion für Ihre Verbindung fort.

Erstellen einer Verbindung mit GitHub Enterprise Server

Nachdem Sie die Verbindung erstellt haben, wird die Seite Mit GitHub Enterprise Server verbinden angezeigt.

> **Important**
>
> AWS CodeStar Verbindungen unterstützen 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.

So stellen Sie eine Verbindung zu GitHub Enterprise Server her

1. Geben Sie unter Connection Name (Verbindungsname) den Namen für die Verbindung ein.

2. Geben Sie unter URL den Endpunkt für Ihren Server ein.

> **Note**
>
> Wenn die bereitgestellte URL bereits verwendet wurde, um einen GitHub Enterprise Server für eine Verbindung einzurichten, werden Sie aufgefordert, den Hostressourcen-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 auswählen, 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.

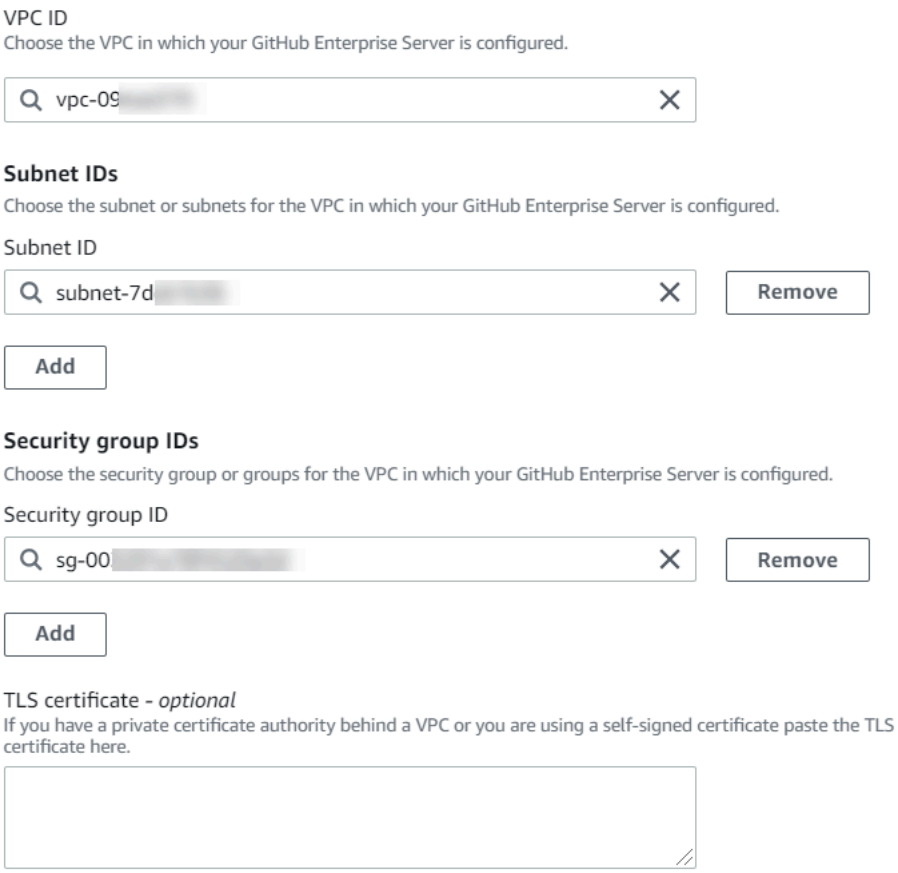
   b. Wählen Sie unter Subnetz-ID (Subnetz-ID) die Option Add (Hinzufügen) aus. Wählen Sie im Feld die Subnetz-ID aus, die Sie für Ihren Host verwenden möchten. Sie können bis zu 10 Subnetze wählen.

      Stellen Sie sicher, dass Sie das Subnetz für die Infrastruktur auswählen, in der Ihre GitHub Enterprise-Server-Instance installiert ist, oder ein Subnetz mit Zugriff auf Ihre installierte GitHub Enterprise-Server-Instance 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-Instance installiert ist, oder eine Sicherheitsgruppe, die über VPN oder Direct Connect Zugriff auf Ihre installierte GitHub Enterprise-Server-Instance hat.

   d. Wenn Sie eine private VPC konfiguriert haben und Ihre GitHub Enterprise Server-Instance so konfiguriert haben, dass eine TLS-Validierung mit einer nicht öffentlichen Zertifizierungsstelle durchgeführt wird, geben Sie unter TLS-Zertifikat Ihre Zertifikat-ID ein. Der TLS-Zertifikatwert sollte der öffentliche Schlüssel des Zertifikats sein.



4. Wählen Sie Mit GitHub Enterprise Server verbinden 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 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 Create App einen Namen für Ihre App aus.
8. Wählen Sie auf der GitHub Autorisierungsseite Autorisieren <app-name> aus.
9. Auf der App-Installationsseite wird eine Meldung angezeigt, dass die AWS CodeStar Connector-App für die 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 der GitHub Enterprise Server-Quellaktion

Führen Sie diese Schritte auf der Seite Assistent oder Aktion bearbeiten aus, um Ihre Quellaktion mit Ihren Verbindungsinformationen zu speichern.

So schließen Sie die Quellaktion ab und speichern sie mit Ihrer Verbindung

1. Wählen Sie unter Repository name (Repository-Name) den Namen Ihres Drittanbieter-Repositorys aus.
2. Unter Pipeline-Auslöser können Sie Auslöser hinzufügen, wenn Ihre Aktion eine CodeConnections Aktion ist. Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration und zum optionalen Filtern mit Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen.
3. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.
   - Um Ausgabeartefakte aus der GitHub Enterprise Server-Aktion mit der Standardmethode zu speichern, wählen Sie CodePipeline Standard . 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 Speichern auf der Seite Aktion bearbeiten aus.

Erstellen eines Hosts und einer Verbindung mit GitHub Enterprise Server (CLI)

Sie können mit der AWS Command Line Interface (AWS CLI, Befehlszeilenschnittstelle) eine Verbindung erstellen.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über die AWS CLI oder AWS CloudFormation erstellt wird, ist standardmäßig im Status `PENDING` (Ausstehend). Nachdem Sie mit der CLI oder mit AWS CloudFormation eine Verbindung erstellt haben, bearbeiten Sie die Verbindung mit der Konsole, um ihren Status auf `AVAILABLE` zu setzen.

Sie können mit der AWS Command Line Interface (AWS CLI, Befehlszeilenschnittstelle) einen Host für installierte Verbindungen 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 Host-Erstellung 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 verschieben. Sobald der Host verfügbar ist, führen Sie die Schritte zum Erstellen einer Verbindung aus.

Verwenden Sie dazu den Befehl create-host.

> **Important**
>
> Ein Host, der über die AWS CLI erstellt wird, ist standardmäßig im Status `Pending` (Ausstehend). 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 hat`Available`.

Einen Host erstellen

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie die AWS CLI, um den Befehl create-host auszuführen, und geben Sie die Parameter `--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).

So erstellen Sie eine Verbindung zu GitHub Enterprise Server

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie die AWS CLI, um den Befehl create-connection auszuführen, und geben Sie die Parameter `--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 am Code-Push in das Verbindungsquellen-Repository. Führen Sie einen der folgenden Schritte aus, um die Pipeline-Auslöserkonfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

   - Um die Pipeline-Auslöserkonfiguration so zu konfigurieren, dass sie nur mit einer manuellen Version beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

     ```
     "DetectChanges": "false",
     ```

   - Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration zum Filtern mit Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen. Im Folgenden wird beispielsweise die Pipeline-Ebene der Pipeline-JSON-Definition erweitert. In diesem Beispiel `release-v1` sind `release-v0` und die einzuschließenden Git-Tags und `release-v2` das auszuschließende Git-Tag.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
     ```

GitLab.com-Verbindungen

Mit Verbindungen können Sie Konfigurationen autorisieren und einrichten, die Ihren Drittanbieter mit Ihren -AWSRessourcen verknüpfen. Um Ihr Drittanbieter-Repository als Quelle für Ihre Pipeline zuzuordnen, verwenden Sie eine -Verbindung.

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

Um eine GitLab.com-Quellaktion in hinzuzufügen CodePipeline, können Sie entweder:

- Wählen Sie auf der Seite Pipeline erstellen oder Aktion bearbeiten der CodePipeline Konsole die Option GitLab Anbieter aus. Siehe Erstellen einer Verbindung mit GitLab.com (Konsole), um die Aktion hinzuzufügen. 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 , Erstellen einer Verbindung mit GitLab.com (CLI) um eine Verbindungsressource mit der CLI zu erstellen.
  - Verwenden Sie die `CreateSourceConnection` Beispielaktionskonfiguration in CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen, um Ihre Aktion hinzuzufügen, wie in gezeigtErstellen einer Pipeline (CLI).

> **Note**
>
> Sie können eine Verbindung auch mithilfe der Entwicklertools-Konsole unter Einstellungen erstellen. Weitere Informationen finden Sie unter Erstellen einer Verbindung.

> **Note**
>
> Indem Sie diese Verbindungsinstallation in GitLab.com autorisieren, erteilen Sie unserem Service Berechtigungen zur Verarbeitung Ihrer Daten, indem Sie auf Ihr Konto zugreifen, und Sie können die Berechtigungen jederzeit widerrufen, indem Sie die Anwendung deinstallieren.

Bevor Sie beginnen:

- Sie müssen bereits ein Konto mit GitLab.com erstellt haben.

  > **Note**
  >
  > Verbindungen bieten nur Zugriff auf Repositorys, die dem Konto gehören, das zum Erstellen und Autorisieren der Verbindung verwendet wurde.

  > **Note**
  >
  > Sie können Verbindungen zu einem Repository erstellen, in dem Sie die Rolle Besitzer in haben GitLab. Anschließend 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 Ihre Pipeline anzugeben, müssen Sie bereits ein Repository auf gitlab.com erstellt haben.

Themen

- Erstellen einer Verbindung mit GitLab.com (Konsole)
- Erstellen einer Verbindung mit GitLab.com (CLI)

Erstellen einer Verbindung mit GitLab.com (Konsole)

Gehen Sie wie folgt vor, um mithilfe der CodePipeline Konsole eine Verbindungsaktion für Ihr Projekt (Repository) in hinzuzufügen GitLab.

So erstellen oder bearbeiten Sie Ihre Pipeline

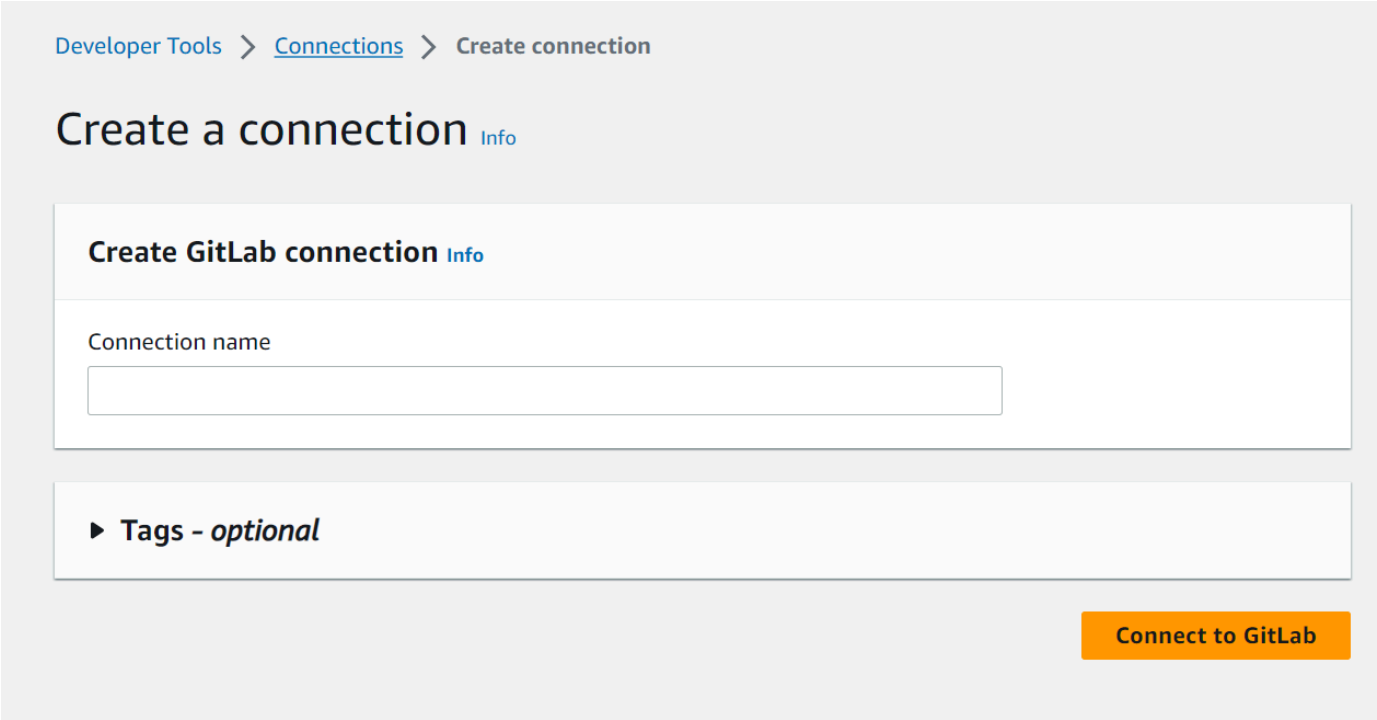
1. Melden Sie sich bei der - CodePipeline Konsole an.
2. Wählen Sie eine der folgenden Optionen aus.

   - Wählen Sie , um eine Pipeline zu erstellen. Folgen Sie den Schritten unter Erstellen einer Pipeline, um den ersten Bildschirm abzuschließen und Weiter zu wählen. Wählen Sie auf der Seite Quelle unter Quellanbieter die Option ausGitLab.
   - Wählen Sie , um eine vorhandene Pipeline zu bearbeiten. Wählen Sie Bearbeiten und dann Stufe bearbeiten aus. Wählen Sie , um Ihre Quellaktion hinzuzufügen oder zu bearbeiten. Geben Sie auf der Seite Aktion bearbeiten unter Aktionsname den Namen für Ihre Aktion ein. Wählen Sie unter Aktionsanbieter die Option ausGitLab.
3. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter erstellt haben, wählen Sie unter Verbindung die Option Mit verbinden aus GitLab. Fahren Sie mit Schritt 4 fort, um die Verbindung zu erstellen.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter erstellt haben, wählen Sie die Verbindung aus. Fahren Sie mit Schritt 9 fort.

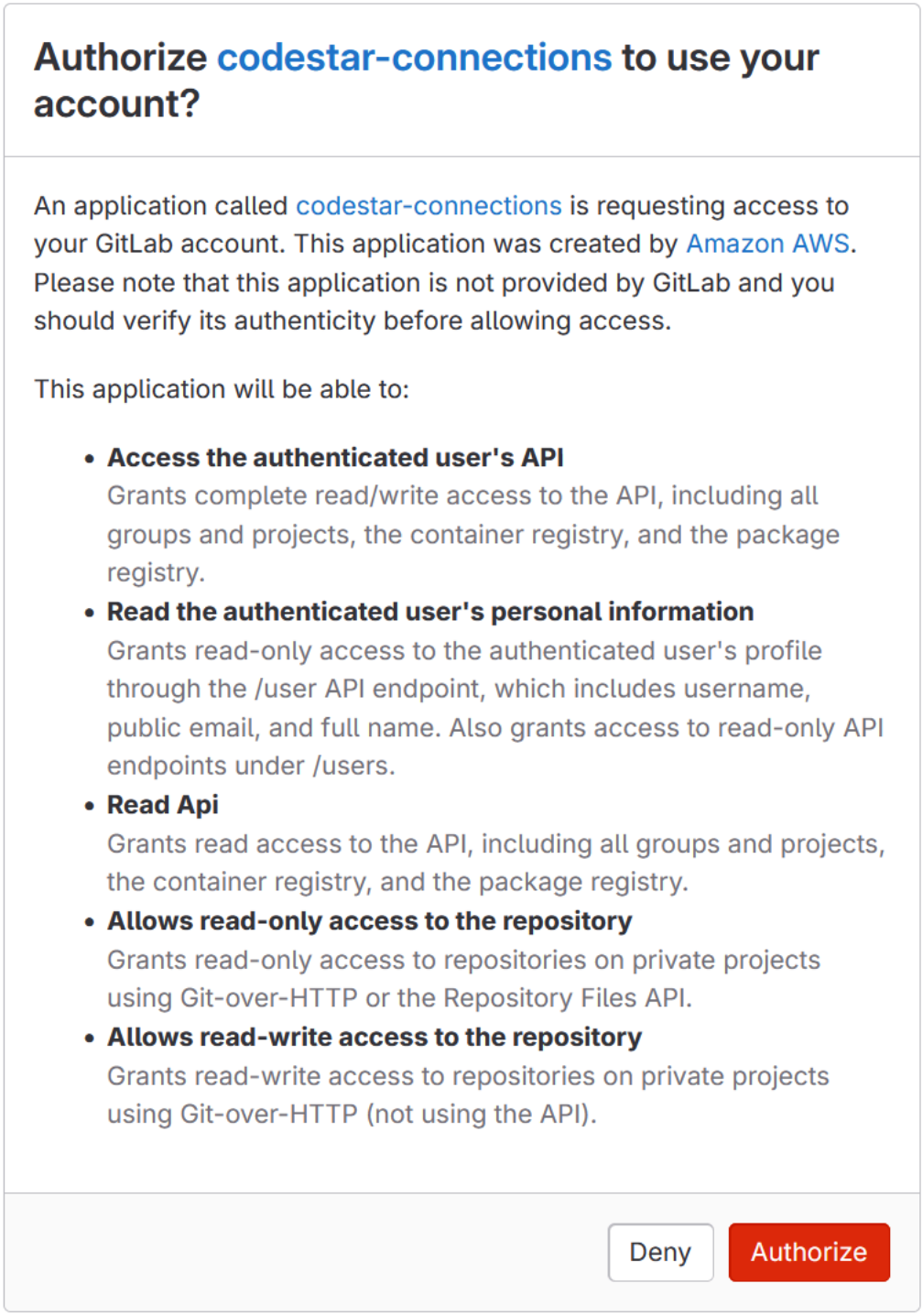
   > **Note**
   >
   > Wenn Sie das Popup-Fenster schließen, bevor eine GitLab.com-Verbindung erstellt wird, müssen Sie die Seite aktualisieren.

4. Um eine Verbindung zu einem GitLab.com-Repository herzustellen, wählen Sie unter Anbieter auswählen die Option ausGitLab. Geben Sie unter Connection name (Verbindungsname) den Namen für die Verbindung ein, die Sie erstellen möchten. Wählen Sie Mit verbinden aus 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, die die Autorisierung für die Verbindung zum Zugriff auf Ihr GitLab.com-Konto anfordert.

   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 Mit verbinden aus GitLab.

   Sie kehren zur CodePipeline Konsole zurück.

> **Note**
>
> Nachdem eine GitLab.com-Verbindung erfolgreich erstellt wurde, wird im Hauptfenster ein Erfolgsbanner angezeigt.
> Wenn Sie sich noch nicht bei GitLab auf dem aktuellen Computer angemeldet haben, müssen Sie das Popup-Fenster manuell schließen.

9. Wählen Sie unter Repository-Name den Namen Ihres Projekts in 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 im `path_with_namespace` Feld unter https://docs.gitlab.com/ee/api/projects.html#get-single-project. Weitere Informationen zum Namespace in GitLabfinden Sie unter https://docs.gitlab.com/ee/user/namespace/.

   > **Note**
   >
   > Für Gruppen in müssen GitLabSie den Projektpfad manuell mit dem Namespace angeben. Geben Sie beispielsweise für ein Repository mit dem Namen `myrepo` in einer Gruppe Folgendes `mygroup`ein: `mygroup/myrepo`. Sie finden den Projektpfad mit dem Namespace in der URL in GitLab.

10. Unter Pipeline-Auslöser können Sie Auslöser hinzufügen, wenn Ihre Aktion eine CodeConnections Aktion ist. Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration und zum optionalen Filtern mit Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen.
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 ausgefüllt wird, haben Sie keinen Besitzerzugriff auf das Repository. Entweder ist der Projektname ungültig oder die verwendete Verbindung hat keinen Zugriff auf das Projekt/Repository.

12. Im Output artifact format (Format des Ausgabeartefakts) müssen Sie das Format für Ihre Artefakte auswählen.

- Um Ausgabeartefakte aus 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 Projektservicerolle aktualisieren, wie in gezeigtHinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder GitLab.com. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiger Klon verwenden, finden Sie unter Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

13. Wählen Sie , um die Quellaktion zu speichern und fortzufahren.

Erstellen einer Verbindung mit GitLab.com (CLI)

Sie können mit der AWS Command Line Interface (AWS CLI, Befehlszeilenschnittstelle) eine Verbindung erstellen.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über die AWS CLI oder AWS CloudFormation erstellt wird, ist standardmäßig im Status `PENDING` (Ausstehend). Nachdem Sie mit der CLI oder mit AWS CloudFormation eine Verbindung erstellt haben, bearbeiten Sie die Verbindung mit der Konsole, um ihren Status auf `AVAILABLE` zu setzen.

So stellen Sie eine Verbindung her

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie die AWS CLI, um den Befehl create-connection auszuführen, und geben Sie die Parameter `--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 am Code-Push in das Verbindungsquellen-Repository. Führen Sie einen der folgenden Schritte aus, um die Pipeline-Auslöserkonfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

   - Um die Pipeline-Auslöserkonfiguration so zu konfigurieren, dass sie nur mit einer manuellen Version beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

     ```
     "DetectChanges": "false",
     ```

   - Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration zum Filtern nach Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen. Im Folgenden wird beispielsweise die Pipeline-Ebene der Pipeline-JSON-Definition erweitert. In diesem Beispiel `release-v1` sind `release-v0` und die einzuschließenden Git-Tags und `release-v2` ist das auszuschließende Git-Tag.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
     ```

```
                            "excludes": [
                                "release-v2"
                            ]
                        }
                    }
                ]
            }
        }
    ]
```

Verbindungen für GitLab selbstverwaltete

Mit Verbindungen können Sie Konfigurationen autorisieren und einrichten, die Ihren Drittanbieter mit Ihren - AWS Ressourcen verknüpfen. Um Ihr Drittanbieter-Repository als Quelle für Ihre Pipeline zuzuordnen, verwenden Sie eine -Verbindung.

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

Um eine GitLab selbstverwaltete Quellaktion in hinzuzufügen CodePipeline, können Sie entweder:

- Wählen Sie auf der Seite Pipeline erstellen oder Aktion bearbeiten der CodePipeline Konsole die GitLab Option Selbstverwalteter Anbieter aus. Siehe Erstellen einer Verbindung zu GitLab selbstverwalteten (Konsole), um die Aktion hinzuzufügen. Die Konsole hilft Ihnen beim Erstellen einer Hostressource und einer Verbindungsressource.
- Verwenden Sie die CLI, um die Aktionskonfiguration für die `CreateSourceConnection` Aktion beim `GitLabSelfManaged` Anbieter hinzuzufügen und Ihre Ressourcen zu erstellen:

- Informationen zum Erstellen Ihrer Verbindungsressourcen finden Sie unter , Erstellen eines Hosts und einer Verbindung zu GitLab selbstverwalteten (CLI) um eine Hostressource und eine Verbindungsressource mit der CLI zu erstellen.
- Verwenden Sie die `CreateSourceConnection` Beispielaktionskonfiguration in CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen, um Ihre Aktion hinzuzufügen, wie in gezeigtErstellen einer Pipeline (CLI).

> **Note**
>
> Sie können eine Verbindung auch mithilfe der Entwicklertools-Konsole unter Einstellungen erstellen. Weitere Informationen finden Sie unter Erstellen einer Verbindung.

Bevor Sie beginnen:

- Sie müssen bereits ein Konto bei erstellt haben GitLab und GitLab Enterprise Edition oder GitLab Community Edition mit einer selbstverwalteten Installation haben. Weitere Informationen erhalten Sie unter https://docs.gitlab.com/ee/subscriptions/self_managed/.

  > **Note**
  >
  > Verbindungen bieten nur Zugriff für das Konto, das zum Erstellen und Autorisieren der Verbindung verwendet wurde.

  > **Note**
  >
  > Sie können Verbindungen zu einem Repository erstellen, in dem Sie die Rolle Besitzer in haben GitLab, und dann kann die Verbindung mit mit Ressourcen wie verwendet werden CodePipeline. Bei Repositorys in Gruppen müssen Sie nicht der Gruppenbesitzer sein.

- Sie müssen bereits ein GitLab persönliches Zugriffstoken (PAT) mit der folgenden eingeschränkten Berechtigung erstellt haben: api. Weitere Informationen erhalten Sie unter https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html. Sie müssen Administrator sein, um den 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 Ihren Host im Voraus einrichten. Sie können einen Host mit oder ohne VPC einrichten. Weitere Informationen zur VPC-Konfiguration und zusätzliche Informationen zum Erstellen eines Hosts finden Sie unter Erstellen eines Hosts.

Themen

- Erstellen einer Verbindung zu GitLab selbstverwalteten (Konsole)
- Erstellen eines Hosts und einer Verbindung zu GitLab selbstverwalteten (CLI)

Erstellen einer Verbindung zu GitLab selbstverwalteten (Konsole)

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, das zum Erstellen der Verbindung verwendet wurde.

Bevor Sie beginnen:

Für eine Host-Verbindung zu GitLab selbstverwalteten müssen Sie die Schritte zum Erstellen einer Host-Ressource für Ihre Verbindung abgeschlossen haben. Weitere Informationen finden Sie unter Verwalten von Hosts für Verbindungen.

Schritt 1: Erstellen oder Bearbeiten Ihrer Pipeline

So 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 , um eine Pipeline zu erstellen. Führen Sie die Schritte unter Erstellen einer Pipeline aus, um den ersten Bildschirm abzuschließen, und wählen Sie Weiter aus. Wählen Sie auf der Seite Quelle unter Quellanbieter die Option GitLab Selbstverwaltet aus.
   - Wählen Sie , um eine vorhandene Pipeline zu bearbeiten. Wählen Sie Bearbeiten und dann Stufe bearbeiten aus. Wählen Sie , um Ihre Quellaktion hinzuzufügen oder zu bearbeiten. 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 erstellt haben, wählen Sie unter Verbindung die Option Mit GitLab selbstverwaltetem verbinden aus. Fahren Sie mit Schritt 2: Erstellen einer Verbindung zu GitLab selbstverwalteten fort.
   - Wenn Sie unter Verbindung bereits eine Verbindung zu Ihrem Anbieter erstellt haben, wählen Sie die Verbindung aus. Fahren Sie mit Schritt 3: Speichern der Quellaktion für Ihre Verbindung fort.

Erstellen einer Verbindung zu GitLab selbstverwalteten

Nachdem Sie die Verbindung erstellt haben, wird die Seite Mit GitLab selbstverwaltetem verbinden angezeigt.

So stellen Sie eine Verbindung zu GitLab selbstverwalteten her

1. Geben Sie unter Connection Name (Verbindungsname) den Namen für die Verbindung ein.
2. Geben Sie unter URL den Endpunkt für Ihren Server ein.

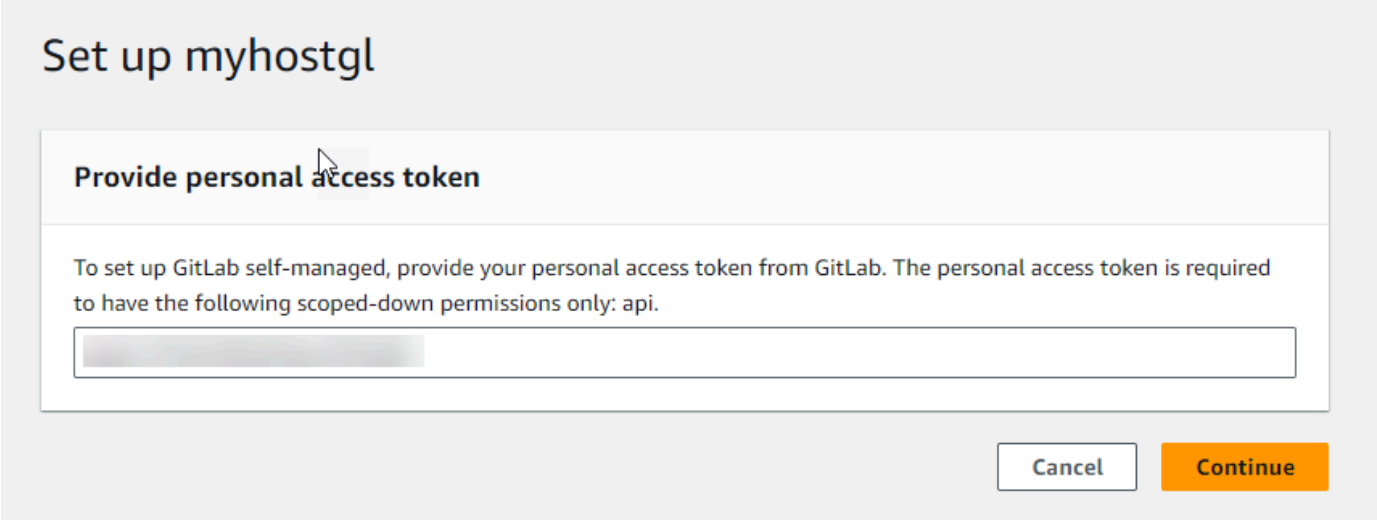
   > **Note**
   >
   > Wenn die bereitgestellte URL bereits verwendet wurde, um einen Host für eine Verbindung einzurichten, werden Sie aufgefordert, den Hostressourcen-ARN auszuwählen, der zuvor für diesen Endpunkt erstellt wurde.

3. Wenn Sie Ihren Server in einer Amazon VPC gestartet haben und eine Verbindung zu Ihrer VPC herstellen möchten, wählen Sie VPC verwenden und füllen Sie die Informationen für die VPC aus.

4. Wählen Sie Mit GitLab selbstverwaltetem verbinden 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 Umleitungsmeldung geöffnet wird, die bestätigt, dass Sie mit dem Anbieter fortfahren möchten, wählen Sie Weiter aus. Geben Sie die Autorisierung für den Anbieter ein.
7. Eine Seite ***`Hostname`*** einrichten wird angezeigt. Geben Sie unter Persönliches Zugriffstoken bereitstellen Ihrem GitLab PAT nur die folgende eingeschränkte Berechtigung: `api`.

   > **Note**
   >
   > Nur ein Administrator kann den PAT erstellen und verwenden.

   Klicken Sie auf Weiter.



8. Die Verbindungsseite zeigt die erstellte Verbindung im Status Available (Verfügbar).

Schritt 3: Speichern Ihrer GitLab selbstverwalteten Quellaktion

Führen Sie diese Schritte auf dem Assistenten oder der Seite Aktion bearbeiten aus, um Ihre Quellaktion mit Ihren Verbindungsinformationen zu speichern.

So schließen Sie die Quellaktion ab und speichern sie mit Ihrer Verbindung

1. Wählen Sie unter Repository name (Repository-Name) den Namen Ihres Drittanbieter-Repositorys aus.
2. Unter Pipeline-Auslöser können Sie Auslöser hinzufügen, wenn Ihre Aktion eine CodeConnections Aktion ist. Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration und zum optionalen Filtern mit Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen.
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 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.
4. Wählen Sie im Assistenten Weiter oder Speichern auf der Seite Aktion bearbeiten aus.

Erstellen eines Hosts und einer Verbindung zu GitLab selbstverwalteten (CLI)

Sie können die AWS Command Line Interface (AWS CLI) verwenden, um eine Verbindung herzustellen.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine über die AWS CLI oder erstellte Verbindung AWS CloudFormation befindet sich standardmäßig im `PENDING` Status . Nachdem Sie eine Verbindung mit der CLI oder erstellt haben AWS CloudFormation, verwenden Sie die Konsole, um die Verbindung so zu bearbeiten, dass sie ihren Status hat`AVAILABLE`.

Sie können die 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 Host-Erstellung 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 verschieben. Sobald der Host verfügbar ist, führen Sie die Schritte zum Erstellen einer Verbindung aus.

Verwenden Sie dazu den Befehl create-host.

> **Important**
>
> Ein über die erstellter Host AWS CLI befindet sich standardmäßig im `Pending` Status . Nachdem Sie einen Host mit der CLI erstellt haben, richten Sie den Host mit der Konsole oder der CLI so ein, dass er seinen Status hat`Available`.

Einen Host erstellen

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie die , AWS CLI um den create-host Befehl auszuführen, und geben Sie die `--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).

So erstellen Sie eine Verbindung zu GitLab selbstverwalteten

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie die , AWS CLI um den create-connection Befehl auszuführen, und geben Sie die `--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 am Code-Push in das Verbindungsquellen-Repository. Führen Sie einen der folgenden Schritte aus, um die Pipeline-Auslöserkonfiguration für die manuelle Veröffentlichung oder für Git-Tags zu konfigurieren:

   - Um die Pipeline-Auslöserkonfiguration so zu konfigurieren, dass sie nur mit einer manuellen Version beginnt, fügen Sie der Konfiguration die folgende Zeile hinzu:

     ```
     "DetectChanges": "false",
     ```

   - Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration zum Filtern nach Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen. Im Folgenden wird beispielsweise die Pipeline-Ebene der Pipeline-JSON-Definition erhöht. In diesem Beispiel `release-v1` sind `release-v0` und die einzuschließenden Git-Tags und `release-v2` das auszuschließende Git-Tag.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
     ```

```
                    {
                        "tags": {
                            "includes": [
                                "release-v0", "release-v1"
                            ],
                            "excludes": [
                                "release-v2"
                            ]
                        }
                    }
                ]
            }
        }
    ]
```

Bearbeiten einer Pipeline in CodePipeline

Eine Pipeline beschreibt das Freigabeverfahren, dem AWS CodePipeline folgen soll, und enthält die Phasen und Aktionen, die ausgeführt 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 Auslöser über die Pipeline-Edit-Seite bearbeiten. Sie können auch Phasen und Aktionen in Ihrer Pipeline 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 Aktionen zu Ihrer Pipeline hinzufügen, die sich in einer anderen -AWSRegion befinden als Ihre Pipeline. Wenn ein der Anbieter für eine Aktion AWS-Service ist und sich dieser Aktionstyp/Anbietertyp in einer anderen AWS Region als Ihre Pipeline befindet, ist dies eine regionsübergreifende Aktion. Weitere Informationen zu regionsübergreifenden Aktionen finden Sie unter Hinzufügen einer regionsübergreifenden Aktion in CodePipeline.

CodePipeline verwendet Methoden zur Änderungserkennung, um Ihre Pipeline zu starten, wenn eine Quellcodeänderung per Push übertragen wird. Diese Erkennungsmethoden basieren auf dem Quelltyp:

- CodePipeline verwendet Amazon CloudWatch Events, um Änderungen in Ihrem CodeCommit Quell-Repository oder Ihrem Amazon 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 die AWS CLI zum Erstellen der Pipeline verwenden, müssen Sie zusätzliche Ressourcen selbst erstellen. Weitere Informationen zum Erstellen oder Aktualisieren einer CodeCommit Pipeline finden Sie unter Erstellen einer - EventBridge Regel für eine CodeCommit Quelle (CLI). Weitere Informationen zur Verwendung der CLI zum Erstellen oder Aktualisieren einer Amazon S3-Pipeline finden Sie unter Erstellen einer - 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 Stufen in einer Pipeline hinzuzufügen, zu bearbeiten oder zu entfernen und Aktionen in einer Stufe hinzuzufügen, zu bearbeiten oder zu entfernen.

Wenn Sie eine Pipeline aktualisieren, schließt alle laufenden Aktionen CodePipeline ordnungsgemäß ab und schlägt dann die Phasen und Pipeline-Ausführungen fehl, in denen die laufenden Aktionen abgeschlossen wurden. Wenn eine Pipeline aktualisiert wird, müssen Sie Ihre Pipeline erneut ausführen. Weitere Informationen zum Ausführen einer Pipeline finden Sie unter Manuelles Starten einer Pipeline.

Bearbeiten einer Pipeline

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

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.

2. Wählen Sie im Feld Name den Namen der Pipeline aus, 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 Bearbeiten auf der Karte Bearbeiten: Pipeline-Eigenschaften aus. Wählen Sie eine der folgenden Optionen und dann Fertig aus.

   - Pipelines vom Typ V1 haben eine JSON-Struktur, die Standardparameter für Pipeline, Phase und Aktionsebene enthält.
   - Pipelines vom Typ V2 haben dieselbe Struktur wie ein V1-Typ sowie zusätzliche Parameterunterstützung, wie Auslöser und Variablen auf Pipeline-Ebene.

   Pipeline-Typen unterscheiden sich in den Merkmalen und im Preis. Weitere Informationen finden Sie unter Pipeline-Typen.
5. Um die Pipeline-Variablen zu bearbeiten, wählen Sie auf der Karte Bearbeiten: Variablen die Option Variablen bearbeiten aus. Fügen Sie Variablen für die Pipeline-Ebene hinzu oder ändern Sie sie und wählen Sie dann Fertig aus.

   Weitere Informationen zu Variablen auf Pipeline-Ebene finden Sie unter Variablen. 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.

   > **Note**
   >
   > Obwohl es optional ist, Variablen auf Pipeline-Ebene hinzuzufügen, schlägt die Pipeline-Ausführung für eine Pipeline fehl, die mit Variablen auf Pipeline-Ebene angegeben ist, in der keine Werte angegeben werden.
6. Um die Pipeline-Auslöser zu bearbeiten, wählen Sie auf der Karte Bearbeiten: Auslöser die Option Auslöser bearbeiten aus. Fügen Sie Auslöser hinzu oder ändern Sie sie und wählen Sie dann Fertig aus.

   Weitere Informationen zum Hinzufügen von Auslösern finden Sie in den Schritten zum Erstellen einer Verbindung zu Bitbucket Cloud, GitHub (Version 2), GitHub Enterprise Server, GitLab.com oder GitLab selbstverwaltet, z. B. GitHub Verbindungen.
7. Führen Sie einen der folgenden Schritte aus, um Phasen und Aktionen auf der Seite Bearbeiten zu bearbeiten:

- 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 eine Verzweigung wird eine Meldung mit der Amazon CloudWatch Events-Regel angezeigt, die für diese Pipeline erstellt werden soll. Wenn Sie die CodeCommit Quelle entfernen, wird eine Meldung mit der zu löschenden Amazon CloudWatch Events-Regel angezeigt.
  - Für einen Amazon S3-Quell-Bucket wird eine Meldung mit der Amazon- CloudWatch Events-Regel und dem AWS CloudTrail Trail angezeigt, die für diese Pipeline erstellt werden sollen. Wenn Sie die Amazon S3-Quelle entfernen, wird eine Meldung mit der Amazon-CloudWatch Events-Regel und dem zu löschenden AWS CloudTrail Trail angezeigt. Wenn der AWS CloudTrail-Pfad von anderen Pipelines verwendet wird, wird der Pfad nicht entfernt und die Datenereignisse werden 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.

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 in 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 zeigt nur für Aktionen an, bei denen der Aktionsanbieter ein istAWS-Service. Das Feld Region ist standardmäßig dieselbe AWS Region wie Ihre Pipeline.

Weitere Informationen zu den Anforderungen für Aktionen in CodePipeline, einschließlich Namen für Eingabe- und Ausgabeartefakte und deren Verwendung, finden Sie unter Anforderungen an die Aktionsstruktur in CodePipeline. Beispiele zum Hinzufügen von Aktionsanbietern mithilfe der Standard-Felder für die einzelnen Anbieter finden Sie unter Erstellen einer Pipeline (Konsole).

Informationen zum Hinzufügen von CodeBuild als Build-Aktion oder Testaktion zu einer Stufe finden Sie unter Verwenden von CodePipeline mit CodeBuild zum Testen von Code und Ausführen von Builds im CodeBuild -Benutzerhandbuch.

> **Note**
>
> Einige Aktionsanbieter, wie z. B. GitHub, erfordern, dass 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 Release change aus, um zu verarbeiten, die sich durch die Pipeline und eine Änderung an der Quelle übergeben, die in der Quellphase der Pipeline angegeben ist. Oder befolgen Sie die Schritte unter Manuelles Starten einer Pipeline zur Verwendung der AWS CLI zur manuellen Veröffentlichung einer Änderung.

Bearbeiten einer Pipeline (AWS CLI)

Sie können den Befehl update-pipeline verwenden, um eine Pipeline zu bearbeiten.

Wenn Sie eine Pipeline aktualisieren, schließt alle laufenden Aktionen CodePipeline ordnungsgemäß ab und schlägt dann die Phasen und Pipeline-Ausführungen fehl, in denen die laufenden Aktionen abgeschlossen wurden. Wenn eine Pipeline aktualisiert wird, müssen Sie Ihre Pipeline erneut ausführen. Weitere Informationen zum Ausführen einer Pipeline finden Sie unter Manuelles Starten einer Pipeline.

> **Important**
>
> Sie können Pipelines, die Partneraktionen enthalten, zwar mithilfe der AWS CLI bearbeiten, dürfen die JSON einer Partneraktion jedoch 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 Eingabeaufforderung (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 Bereitstellungsstufe hinzufügen würden. Diese Stufe wird nach der ersten Bereitstellungsstufe mit dem Namen *`Staging`* ausgeführt.

   > **Note**
   >
   > Dies ist nur ein Teil und nicht die komplette Struktur der Datei. Weitere Informationen finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

```
,
        {
            "name": "Staging",
            "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "MyApp"
                            }
                        ],
                        "name": "Deploy-CodeDeploy-Application",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
```

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

Informationen zum Verwenden der CLI zum Hinzufügen einer Genehmigungsaktion zu einer Pipeline finden Sie unter Hinzufügen einer manuellen Genehmigungsaktion zu einer Pipeline 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 Ihrer Verzweigung oder Ihrem Amazon 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 Hinzufügen einer regionsübergreifenden Aktion 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 einer - EventBridge Regel für eine CodeCommit Quelle (CLI).

- Für eine Amazon S3-Quelle müssen Sie die CloudWatch Ereignisregel und den AWS CloudTrail Trail erstellen, wie unter beschriebenAmazon S3-Quellaktionen und EventBridge mit AWS CloudTrail.

6. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

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

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

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen.

7. Öffnen Sie die - CodePipeline Konsole und wählen Sie die Pipeline aus, die Sie gerade bearbeitet haben.

   Die Pipeline zeigt die Änderungen an. Wenn Sie das nächste Mal eine Änderung am Quellspeicherort durchführen, führt die Pipeline die Revision über die überarbeitete Struktur der Pipeline aus.

8. Um die letzte Revision manuell über die überarbeitete Struktur der Pipeline auszuführen, nutzen Sie den Befehl start-pipeline-execution. Weitere Informationen finden Sie unter Manuelles Starten einer Pipeline.

Weitere Informationen zur Struktur einer Pipeline und zu den erwarteten Werten finden Sie unter CodePipeline Referenz zur Pipeline-Struktur und APIAWS CodePipeline-Referenz .

Pipelines und Details anzeigen in CodePipeline

Sie können die AWS CodePipeline-Konsole oder die AWS CLI verwenden, um Details 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)

Pipelines anzeigen (Konsole)

Sie können den Status, die Übergänge und die Artefaktaktualisierungen für eine Pipeline anzeigen.

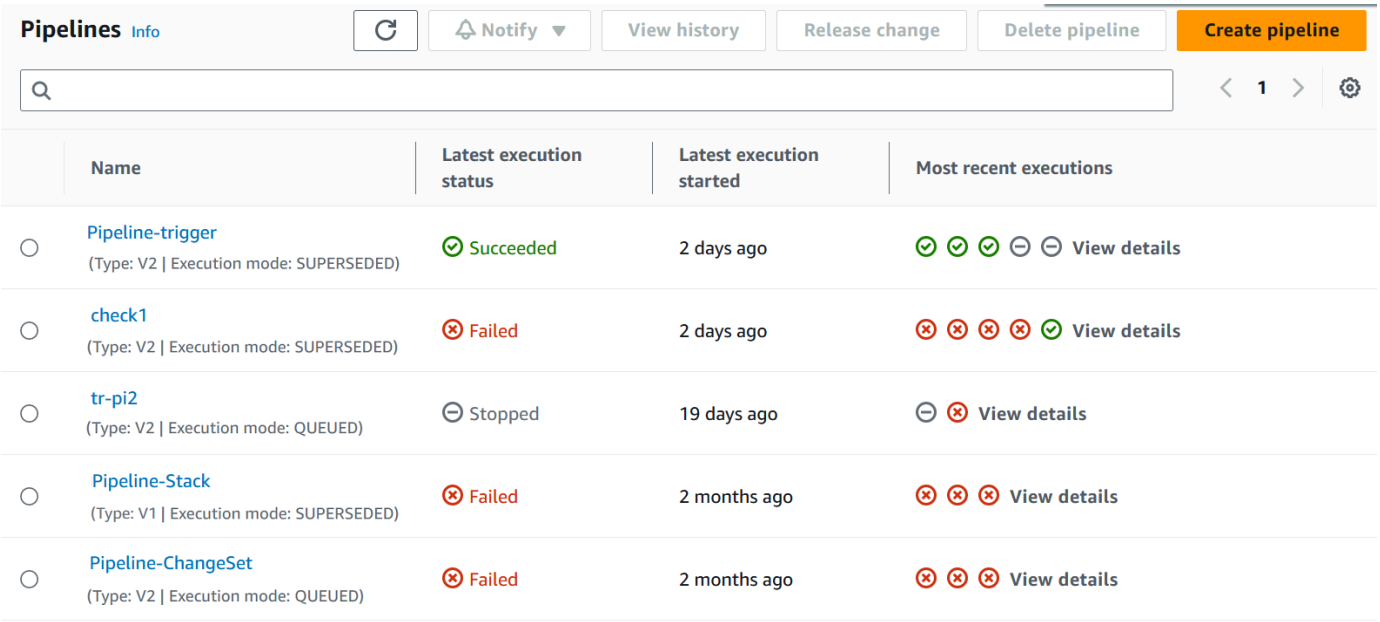
> **Note**
>
> Nach einer Stunde wird die detaillierte Anzeige der Pipeline in Ihrem Browser nicht mehr automatisch aktualisiert. Um aktuelle Informationen anzuzeigen, müssen Sie die Seite aktualisieren.

Um Pipelines anzuzeigen

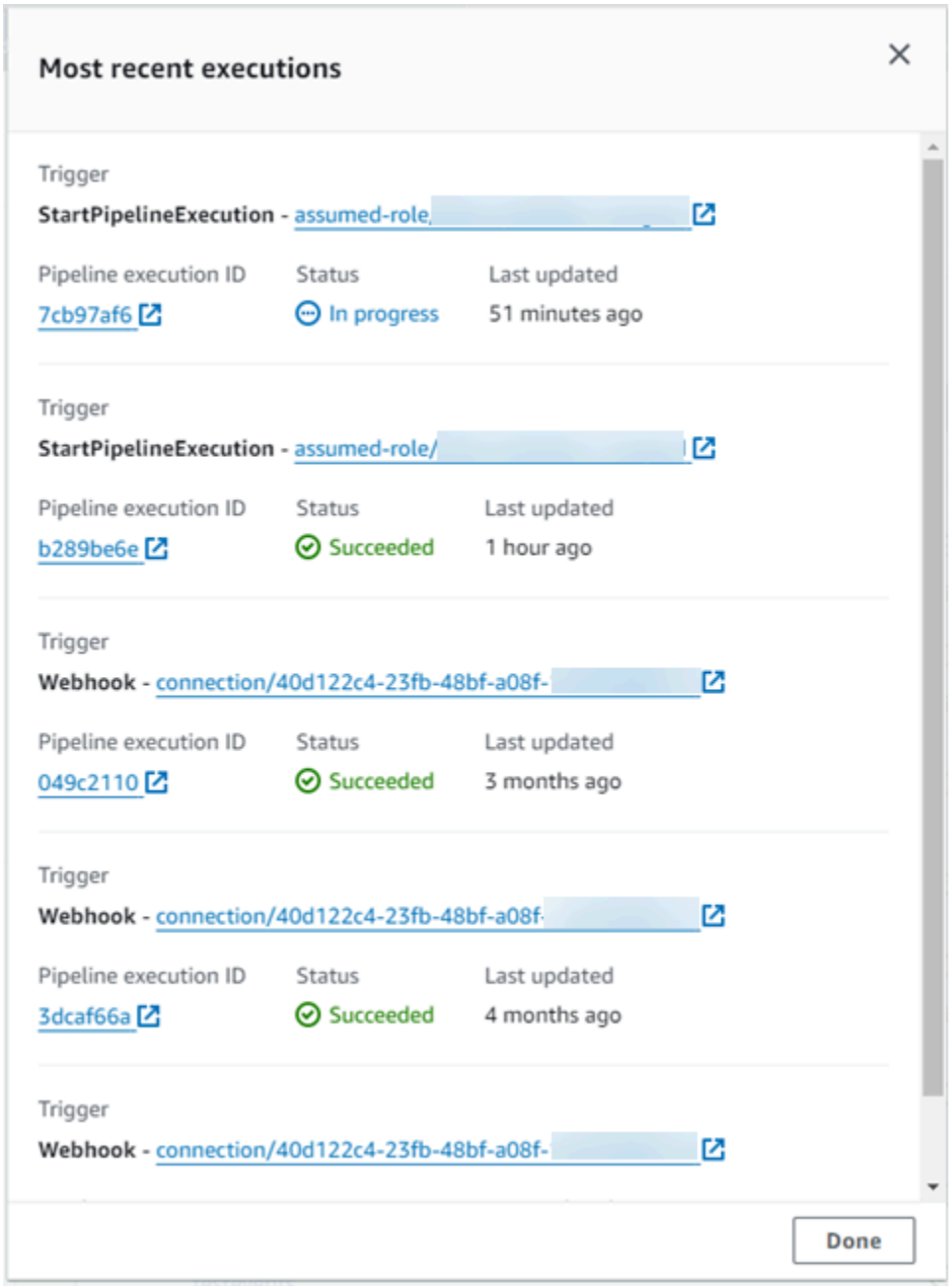
1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Seite „Pipelines“ wird angezeigt. Eine Liste all Ihrer Pipelines für diese Region wird angezeigt.

   Name, Typ, Status, Version, Erstellungsdatum und Datum der letzten Änderung aller mit Ihrem AWS Konto verknüpften Pipelines werden zusammen mit der zuletzt gestarteten Ausführungszeit angezeigt.
2. Der Status der fünf letzten Ausführungen wird angezeigt.

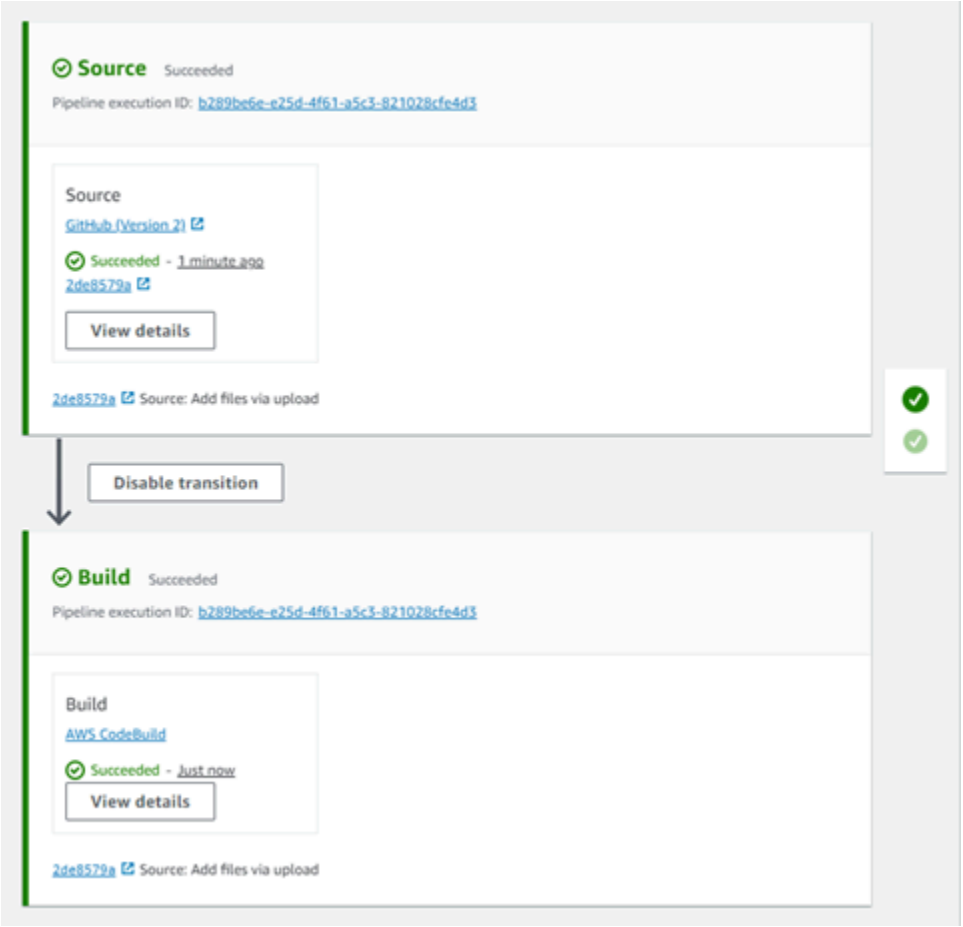


Wählen Sie neben einer bestimmten Zeile die Option Details anzeigen aus, um ein Detaildialogfeld mit den letzten Ausführungen anzuzeigen.



Um Details zu den letzten Ausführungen für die Pipeline anzuzeigen, können Sie auch den Selektor neben Ihrer Pipeline auswählen und dann Verlauf anzeigen wählen. Die früheren Ausführungen können Sie in den Revisionsdetails zu den Quell-Artefakten sehen (z. B. Ausführungs-IDs, Status, Start- und Endzeiten, Dauer und Commit-IDs und -Nachrichten).

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. B. CodeDeploy
- Wann die Aktion zuletzt ausgeführt wurde.
- Ob die Aktion erfolgreich war oder fehlgeschlagen ist.
- Links zu weiteren Informationen zur letzten Ausführung der Aktion (sofern verfügbar).
- Details zu den Quellrevisionen, die die letzte Pipeline-Ausführung in der Phase durchlaufen, oder, bei CodeDeploy Bereitstellungen, zu den neuesten Quellrevisionen, die auf Ziel-Instances bereitgestellt wurden.
- Eine Schaltfläche „Details anzeigen“, mit der ein Dialogfeld mit Details zur Ausführung der Aktion, zu Protokollen und zur Aktionskonfiguration geöffnet wird.

> **Note**
>
> Die Registerkarte „Protokolle“ ist für CodeBuild AWS CloudFormation Aktionen verfügbar, die im Konto der Pipeline ausgeführt wurden.

4. Um Details zum Anbieter der Aktion anzuzeigen, wählen Sie den Anbieter aus. Wenn Sie beispielsweise in der vorherigen Beispielpipeline entweder die Phase Staging oder die Produktionsphase wählen CodeDeploy , wird die CodeDeploy Konsolenseite für die für diese Phase konfigurierte Bereitstellungsgruppe angezeigt.
5. Um zu sehen, wird der Fortschritt einer Aktion neben einer laufenden Aktion angezeigt (gekennzeichnet durch die Meldung In Bearbeitung). Wenn die Aktion in Bearbeitung ist, werden der inkrementelle Fortschritt und die Schritte oder Aktionen angezeigt.
6. Um für die manuelle Genehmigung konfigurierte Aktionen zu genehmigen oder abzulehnen, wählen Sie Review aus.
7. Um nicht erfolgreich abgeschlossene Aktionen in einer Stufe zu wiederholen, klicken Sie auf Retry.
8. Der Status der letzten Ausführung der Aktion, einschließlich der Ergebnisse dieser Aktion (Erfolgreich oder Fehlgeschlagen), wird angezeigt.

Aktionsdetails in einer Pipeline (Konsole) anzeigen

Sie können Details für eine Pipeline anzeigen, einschließlich Details zu Aktionen in jeder Phase.

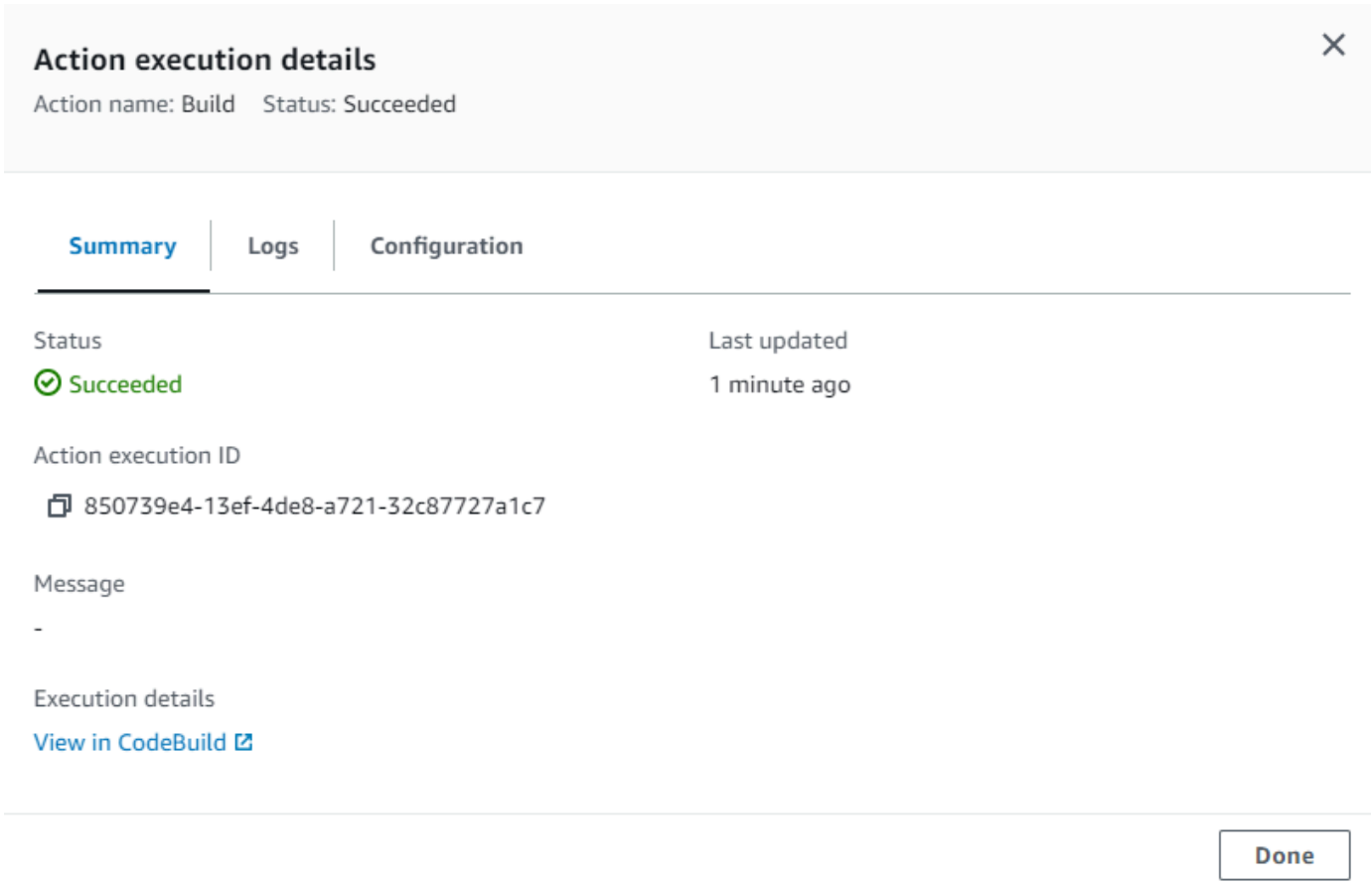
> **Note**
>
> Nach einer Stunde wird die detaillierte Anzeige der Pipeline in Ihrem Browser nicht mehr automatisch aktualisiert. Um aktuelle Informationen anzuzeigen, müssen Sie die Seite aktualisieren.

Um Aktionsdetails in einer Pipeline anzuzeigen

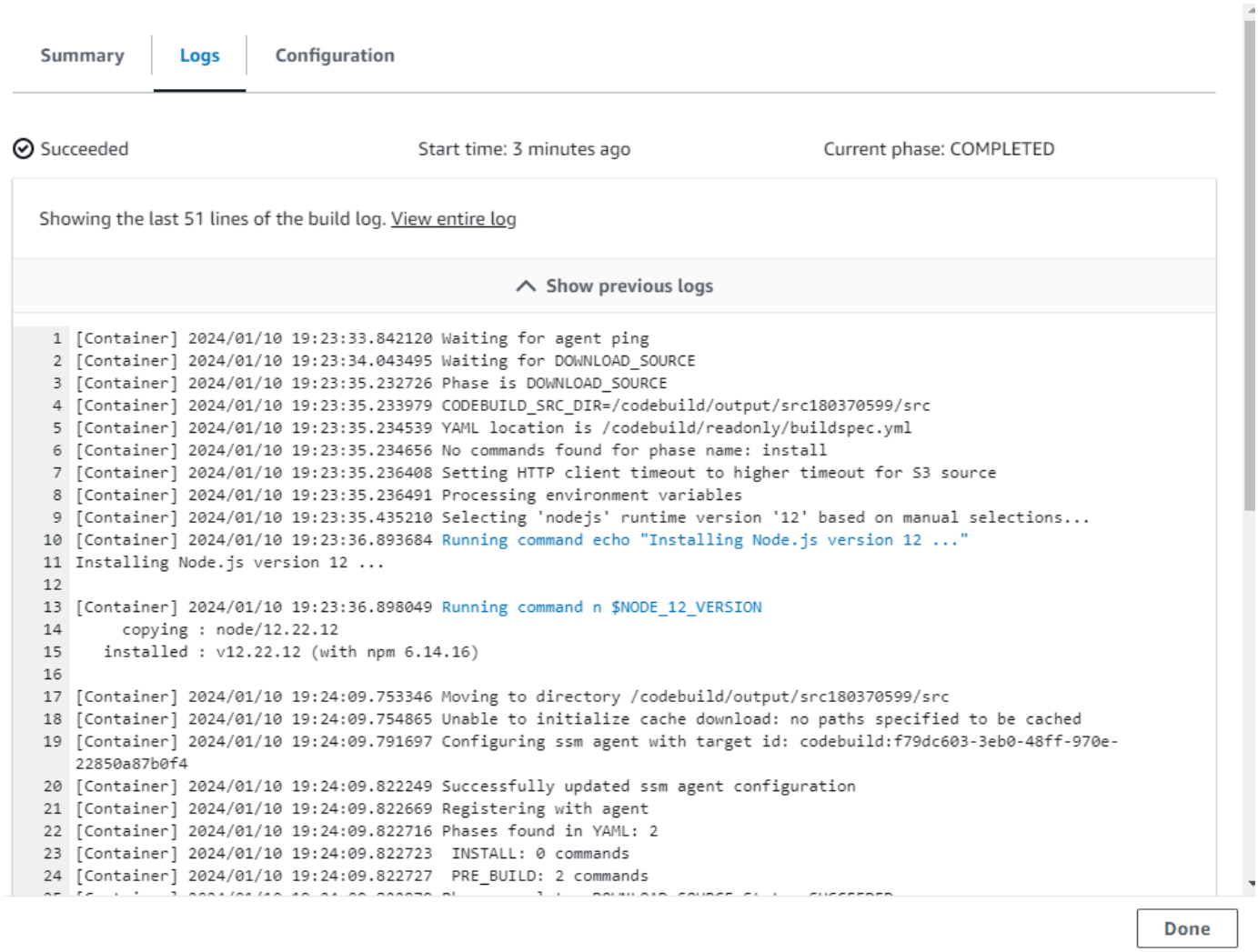
1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Die Seite „Pipelines“ wird angezeigt.
2. Wählen Sie bei jeder Aktion Details anzeigen aus, um ein Dialogfeld mit Details zur Ausführung der Aktion, den Protokollen und der Aktionskonfiguration zu öffnen.

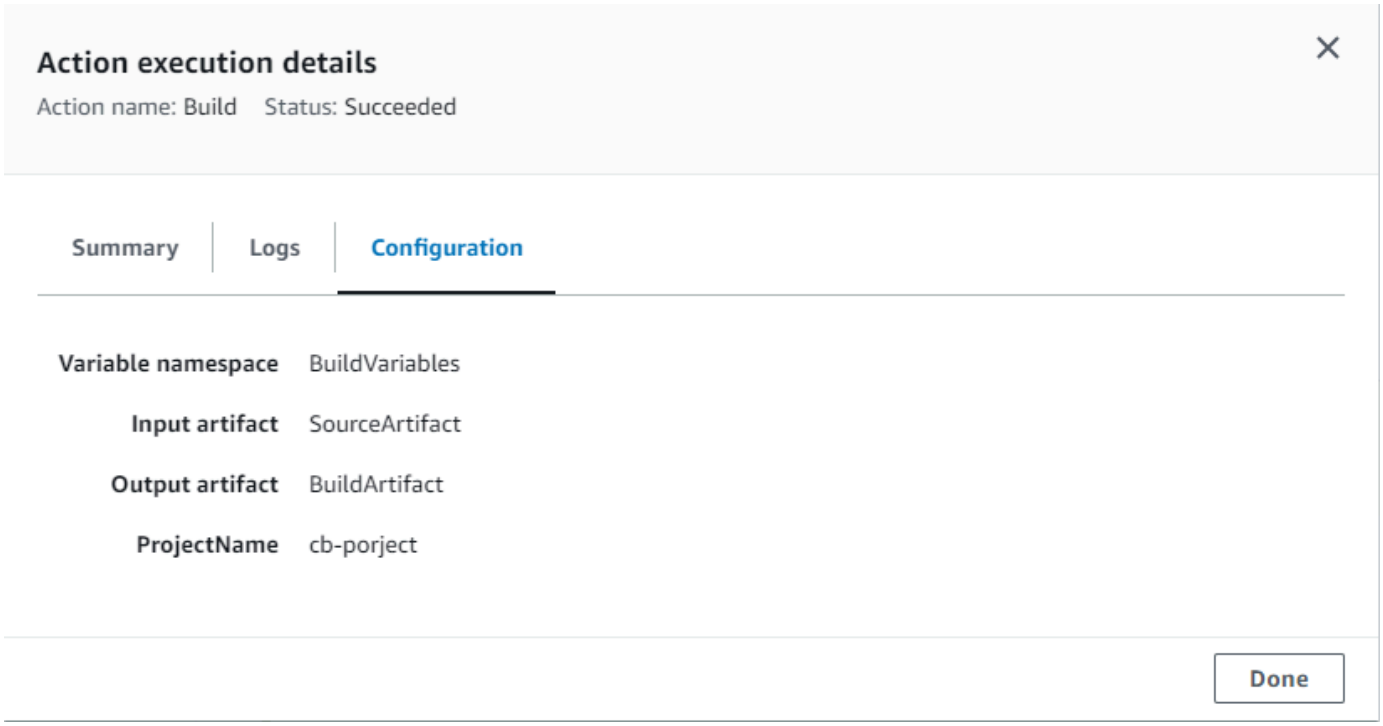
   > **Note**
   >
   > Die Registerkarte Protokolle ist für AWS CloudFormation Aktionen CodeBuild und Aktionen verfügbar.
3. Um die Aktionsübersicht für eine Aktion in einer Phase einer Pipeline anzuzeigen, wählen Sie Details zur Aktion anzeigen und dann die Registerkarte Zusammenfassung aus.



4. Um die Aktionsprotokolle für eine Aktion mit Protokollen anzuzeigen, wählen Sie Details zur Aktion anzeigen und dann die Registerkarte Protokolle aus.



5. Um die Konfigurationsdetails für eine Aktion anzuzeigen, wählen Sie die Registerkarte Konfiguration.



Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen

Sie können die Konsole verwenden, um Pipeline-Einstellungen wie den Pipeline-ARN, den Dienstrollen-ARN und den Pipeline-Artefaktspeicher anzuzeigen.

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

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie den Namen Ihrer Pipeline und dann im linken Navigationsbereich Einstellungen aus. Die Seite zeigt Folgendes:

   - Der Name der Pipeline
   - Die Pipeline Amazon Resource Name (ARN)

     Der Pipeline-ARN wird in folgendem Format erstellt:

     arn:aws:codepipeline: Region: Konto: Pipeline-Name

     Pipeline-Beispiel-ARN:

     `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

- Die CodePipeline Servicerolle ARN für Ihre Pipeline
- Die Pipeline-Version
- Der Name und der Speicherort des Artefaktspeichers für die Pipeline

Anzeigen von der Pipeline-Details und des Verlaufs (CLI)

Sie können die folgenden Befehle verwenden, um weitere Details zu Ihren Pipelines und Pipeline-Ausführungen anzuzeigen:

- list-pipelines-Befehl, um eine Zusammenfassung aller Pipelines in Ihrem AWS-Konto 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 Befehlszeile (Windows) und verwenden Sie denAWS 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.

Löschen einer 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 eine Pipeline mittels der AWS CodePipeline-Konsole oder des Befehls delete-pipeline in der AWS CLI löschen.

Themen

- Löschen einer Pipeline (Konsole)
- Löschen einer Pipeline (CLI)

Löschen einer Pipeline (Konsole)

So löschen Sie eine Pipeline

1. Melden Sie sich beim anAWS Management ConsoleÖffnen Sie die CodePipeline-Konsole unterhttp://console.aws.amazon.com/codesuite/codepipeline/homeaus.

   Es werden Name und Status aller Ihrem AWS-Konto zugeordneten Pipelines 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 mehr rückgängig gemacht werden.

Löschen einer Pipeline (CLI)

So verwenden Sie denAWS CLIUm eine Pipeline manuell manuell zu löschen, verwenden Sie diePipeline löschenbefehl.

> **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 die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix) und verwenden Sie dieAWS CLISo führen Sie das aus:delete-pipelineGeben Sie dabei den Namen der Pipeline an, die Sie löschen möchten. Um eine Pipeline mit dem Namen zu löschen*MyFirstPipeline*:

   ```
   aws codepipeline delete-pipeline --name MyFirstPipeline
   ```

   Mit diesem Befehl wird kein Inhalt zurückgegeben.

2. Löschen Sie alle Ressourcen, die Sie nicht mehr benötigen.

   > **Note**
   >
   > Das Löschen einer Pipeline bewirkt nicht Löschung die in der Pipeline verwendeten Ressourcen (z. B. die CodeDeploy- oder Elastic Beanstalk-Anwendung, die Sie zur Bereitstellung Ihres Codes verwendet haben, oder — falls Sie Ihre Pipeline über die CodePipeline-Konsole erstellt haben — den Amazon S3 S3-Bucket, den CodePipeline zum Speichern der Artefakte Ihrer Pipelines erstellt hat). 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 zur Erstellung einer Pipeline nutzen, erstellt CodePipeline einen Amazon S3 S3-Bucket zur Speicherung aller Artefakte für Ihre Pipelines. Wenn Sie alle Pipelines gelöscht haben, führen Sie die Schritte unter Löschen eines Buckets aus.

Erstellen einer Pipeline in CodePipeline, die Ressourcen aus einem anderen AWS-Konto verwendet

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 gleichen 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 vorherige Aktion im selben Konto wie die Aktion erzeugt
>
>   Anders gesagt, ist es nicht möglich, ein Artefakt von einem Konto an ein anderes Konto zu übergeben, wenn keines der Konten ein Pipeline-Konto ist.
> - Für die folgenden Aktionstrypen werden keine kontoübergreifenden Aktionen unterstützt:
>   - Jenkins-Build-Aktionen

In diesem Beispiel müssen Sie einen 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-Schlüssel können Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN verwenden.

> **Note**
>
> Aliase werden nur in dem zum Erstellen des KMS-Schlüssels verwendeten Konto erkannt. Für kontoübergreifende Aktionen können Sie zum Identifizieren des Schlüssels nur die Schlüssel-ID oder den Schlüssel-ARN vewenden. Bei kontoübergreifenden Aktionen wird die

Rolle des anderen Kontos (AccountB) verwendet, sodass bei Angabe der Schlüssel-ID der Schlüssel des anderen Kontos (AccountB) verwendet wird.

In dieser Anleitung und den entsprechenden Beispielen ist *AccountA* das Konto, das ursprünglich zur Erstellung der Pipeline verwendet wurde. Es hat Zugriff auf den Amazon S3 S3-Bucket zum Speichern von Pipeline-Artefakten und die vonAWS CodePipeline. *AccountB* ist das Konto, das ursprünglich für die Erstellung der CodeDeploy-Anwendung, der Bereitstellungsgruppe und der Service-Rolle verwendet wurde, die von CodeDeploy verwendet werden.

Damit *AccountA* eine Pipeline bearbeiten kann, um die von *AccountB* erstellte CodeDeploy Anwendung zu verwenden, muss *AccountA*:

- Die ARN oder Konto-ID von *AccountB* anfordern (in dieser Anleitung ist die ID von *AccountB* *012ID_ACCOUNT_B*).
- *Erstellen oder verwenden Sie einen vom AWS KMS Kunden verwalteten Schlüssel in der Region für die Pipeline und gewähren Sie Berechtigungen zur Verwendung dieses Schlüssels für die Servicerolle (CodePipeline_Service_Role) und AccountB.*
- Erstellen Sie eine Amazon S3 S3-Bucket-Richtlinie, die *AccountB* Zugriff auf den Amazon S3 S3-Bucket gewährt (z. B. *codepipeline-us-east-2-1234567890*).
- Erstellen Sie eine Richtlinie, die es *AccountA* ermöglicht, eine von *AccountB* konfigurierte Rolle anzunehmen, und fügen Sie diese Richtlinie der Servicerolle hinzu (*CodePipeline_Service_Role*).
- Die Pipeline bearbeiten, sodass diese den kundenverwalteten AWS KMS-Schlüssel und nicht den Standardschlüssel verwendet.

Damit *AccountB* einer in *AccountA* erstellten Pipeline Zugriff auf seine Ressourcen gestattet, muss *AccountB*:

- Die ARN oder Konto-ID von *AccountA* anfordern (in dieser Anleitung ist die ID von *AccountA* *012ID_ACCOUNT_A*).
- Erstellen Sie eine Richtlinie, die auf die Amazon EC2 EC2-Instance-Rolle angewendet wirdCodeDeploy, für die der Zugriff auf den Amazon S3 S3-Bucket konfiguriert ist (*codepipeline-us-east-2-1234567890*).

- *Erstellen Sie eine Richtlinie, die auf die für die konfigurierte Amazon EC2 EC2-Instance-Rolle angewendet wird und CodeDeploy die den Zugriff auf den vom AWS KMS Kunden verwalteten Schlüssel ermöglicht, der zur Verschlüsselung der Pipeline-Artefakte in AccountA verwendet wird.*
- Konfigurieren und fügen Sie eine IAM-Rolle (*CrossAccount_Role) mit einer Vertrauensbeziehungsrichtlinie hinzu, die es AccountA ermöglicht, die Rolle* anzunehmen.
- Erstellen Sie eine Richtlinie, die den Zugriff auf die für die Pipeline erforderlichen Bereitstellungsressourcen ermöglicht, und hängen Sie sie an *CrossAccount_Role* an.
- *Erstellen Sie eine Richtlinie, die den Zugriff auf den Amazon S3 S3-Bucket ermöglicht (codepipeline-us-east-2-1234567890), und hängen Sie sie an _Role an. CrossAccount*

Themen

- Voraussetzung: Erstellen eines AWS KMS-Verschlüsselungsschlüssels
- Schritt 1: Einrichten von Kontorichtlinien und Rollen
- Schritt 2: Bearbeiten der Pipeline

Voraussetzung: Erstellen eines AWS KMS-Verschlüsselungsschlüssels

Kundenverwaltete Schlüssel sind wie alle AWS KMS-Schlüssel regionsspezifisch. Sie müssen Ihren kundenverwalteten AWS KMS-Schlüssel in der gleichen Region erstellen, in der die Pipeline erstellt wurde (beispielsweise `us-east-2`).

So erstellen Sie einen kundenverwalteten Schlüssel in AWS KMS

1. Melden Sie sich AWS Management Console mit *AccountA bei der an* und öffnen Sie die AWS KMS Konsole.
2. Wählen Sie links Customer managed keys (Vom Kunden verwaltete Schlüssel) aus.
3. Klicken Sie auf Create key. Lassen Sie unter Configure key (Schlüssel konfigurieren) die Standardeinstellung Symmetric (Symmetrisch) ausgewählt und wählen Sie Next (Weiter) aus.
4. Geben Sie im Feld Alias einen Alias ein, der für diesen Schlüssel verwendet werden soll (z. B. *PipelineName-Key*). Geben Sie optional eine Beschreibung und die Tags für den Schlüssel ein und wählen Sie dann Next (Weiter) aus.

5. Wählen Sie unter „Wichtige Administratorberechtigungen definieren“ die Rolle oder Rollen aus, die Sie als Administratoren für diesen Schlüssel fungieren möchten, und klicken Sie dann auf Weiter.
6. Wählen Sie unter Definieren von Schlüsselverwendungsberechtigungen 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 für *AccountB* ein, um den ARN abzuschließen, und wählen Sie Next (Weiter) aus.
7. Überprüfen Sie die Richtlinie in Preview Key Policy (Vorschau der Schlüsselrichtlinie) und wählen Sie dann Finish (Beenden) aus.
8. Wählen Sie aus der Liste der Schlüssel den Alias Ihres Schlüssels aus und kopieren Sie dessen ARN (z. B. ***arn:aws:kms:us-east-2:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE***). Sie benötigen diese Informationen, wenn Sie Ihre Pipeline bearbeiten und Richtlinien konfigurieren.

Schritt 1: Einrichten von Kontorichtlinien und Rollen

Nach der Erstellung des AWS KMS-Schlüssels müssen Sie Richtlinien erstellen und anfügen, die den kontoübergreifenden Zugriff aktivieren. Dies erfordert Aktionen sowohl von *AccountA* als auch von *AccountB*.

Themen

- Konfigurieren von Richtlinien und Rollen in dem Konto, das die Pipeline erstellen wird (AccountA)
- Konfigurieren von Richtlinien und Rollen in dem Konto, das die AWS-Ressource besitzt (AccountB)

Konfigurieren von Richtlinien und Rollen in dem Konto, das die Pipeline erstellen wird (***AccountA***)

Um eine Pipeline zu erstellen, die CodeDeploy Ressourcen verwendet, die einem anderen AWS Konto zugeordnet sind, muss *AccountA* Richtlinien sowohl für den Amazon S3 S3-Bucket, der zum Speichern von Artefakten verwendet wird, als auch für die Servicerolle für konfigurieren. CodePipeline

Um eine Richtlinie für den Amazon S3 S3-Bucket zu erstellen, die Zugriff auf AccountB (Konsole) gewährt

1. Melden Sie sich AWS Management Console bei der *AccountA* öffnen Sie die Amazon S3 S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie in der Liste der Amazon S3 S3-Buckets den Amazon S3 S3-Bucket aus, in dem Artefakte für Ihre Pipelines gespeichert werden. *Dieser Bucket hat einen Namencodepipeline-region-1234567EXAMPLE, wobei Region die Region ist, in der AWS Sie die Pipeline erstellt haben, und 1234567EXAMPLE eine zehnstellige Zufallszahl ist, die sicherstellt, dass der Bucket-Name eindeutig ist (z. B. -2-1234567890). codepipeline-us-east*
3. Wählen Sie auf der Detailseite für den Amazon S3 S3-Bucket die Option Eigenschaften aus.
4. Erweitern Sie im Eigenschaftenbereich das Feld Permissions und wählen Sie Add bucket policy aus.

   > **Note**
   >
   > Wenn Ihrem Amazon S3 S3-Bucket bereits eine Richtlinie zugeordnet ist, wählen Sie Bucket-Richtlinie bearbeiten. Danach können Sie die Anweisungen im folgenden Beispiel zur vorhandenen Richtlinie hinzufügen. So fügen Sie eine neue Richtlinie hinzu. Wählen Sie den Link und befolgen Sie die Anweisungen im AWS-Richtliniengenerator. Weitere Informationen finden Sie unter Übersicht über IAM-Richtlinien.

5. Geben Sie die folgende Richtlinie in das Fenster Bucket Policy Editor ein. Dies ermöglicht *AccountB* den Zugriff auf die Pipeline-Artefakte und bietet *AccountB* die Möglichkeit zum Hinzufügen von Ausgabe-Artefakten, falls diese von einer Aktion (z. B. eine benutzerdefinierte Quell- oder Build-Aktion) erstellt werden.

   Im folgenden Beispiel ist für *AccountB* der ARN *012ID_ACCOUNT_B*. Der ARN für den Amazon S3 S3-Bucket ist *codepipeline-us-east-2-1234567890*. Ersetzen Sie diese ARNs durch den ARN für das Konto, das Sie den Zugriff zulassen möchten, und den ARN für den Amazon S3 S3-Bucket:

```
{
 "Version": "2012-10-17",
 "Id": "SSEAndSSLPolicy",
 "Statement": [
{
```

```
  "Sid": "DenyUnEncryptedObjectUploads",
  "Effect": "Deny",
  "Principal": "*",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
  "Condition": {
 "StringNotEquals": {
 "s3:x-amz-server-side-encryption": "aws:kms"
 }
   }
 },
{
  "Sid": "DenyInsecureConnections",
  "Effect": "Deny",
  "Principal": "*",
  "Action": "s3:*",
  "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
  "Condition": {
 "Bool": {
    "aws:SecureTransport": false
  }
 }
  },
{
  "Sid": "",
  "Effect": "Allow",
  "Principal": {
 "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
 },
  "Action": [
           "s3:Get*",
           "s3:Put*"
      ],
  "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
},
{
  "Sid": "",
  "Effect": "Allow",
  "Principal": {
      "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
  },
  "Action": "s3:ListBucket",
  "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890"
}
```

```
    ]
}
```

6. Wählen Sie Save und schließen Sie dann den Richtlinien-Editor.
7. Wählen Sie die zum Speichern der Berechtigungen für den Amazon S3 S3-Bucket.

So erstellen Sie eine Richtlinie für die Service-Rolle für CodePipeline (Konsole)

1. Melden Sie sich AWS Management Console bei der *AccountA* öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Roles (Rollen) aus.
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, damit *AccountB* die Rolle übernehmen kann. Im folgenden Beispiel ist *012ID_ACCOUNT_B* der ARN für *AccountB*:

```
{
   "Version": "2012-10-17",
   "Statement": {
       "Effect": "Allow",
       "Action": "sts:AssumeRole",
       "Resource": [
           "arn:aws:iam::012ID_ACCOUNT_B:role/*"
       ]
   }
 }
```

6. Wählen Sie Review policy (Richtlinie prüfen).
7. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Create Policy (Richtlinie erstellen) aus.

Konfigurieren von Richtlinien und Rollen in dem Konto, das die AWS-Ressource besitzt (*AccountB*)

Wenn Sie eine Anwendung, Bereitstellung und Bereitstellungsgruppe in erstellenCodeDeploy, erstellen Sie auch eine Amazon EC2 EC2-Instance-Rolle. (Diese Rolle wird für Sie erstellt, wenn Sie den Assistenten für die Ausführung der Bereitstellungsanleitung verwenden. Sie können sie jedoch auch manuell erstellen.) Damit eine in *AccountA* erstellte Pipeline CodeDeploy-Ressourcen verwendet, die in *AccountB* erstellt wurden, müssen Sie:

- Konfigurieren Sie eine Richtlinie für die Instance-Rolle, die ihr den Zugriff auf den Amazon S3 S3-Bucket ermöglicht, in dem Pipeline-Artefakte gespeichert sind.
- Eine zweite Rolle in *AccountB* erstellen, die für den kontoübergreifenden Zugriff konfiguriert ist.

  Diese zweite Rolle muss nicht nur Zugriff auf den Amazon S3 S3-Bucket in *AccountA* haben, sie muss auch eine Richtlinie enthalten, die den Zugriff auf die CodeDeploy Ressourcen ermöglicht, und eine Vertrauensbeziehungsrichtlinie, die es *AccountA* ermöglicht, die Rolle anzunehmen.

> **Note**
>
> Diese Richtlinien sind für die Einrichtung von CodeDeploy-Ressourcen, die in einer Pipeline verwendet werden sollen, die mit einem anderen AWS-Konto erstellt wurde. Andere AWS-Konten erfordern Richtlinien, die für ihre Ressourcenanforderungen spezifisch sind.

So erstellen Sie eine Richtlinie für die Amazon EC2 EC2-Instance-Rolle, für die CodeDeploy (Konsole) konfiguriert ist

1. Melden Sie sich AWS Management Console bei der *AccountB* öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Roles (Rollen) aus.
3. Wählen Sie in der Liste der Rollen unter Rollenname den Namen der Servicerolle aus, die als Amazon EC2 EC2-Instance-Rolle für die CodeDeploy Anwendung verwendet wird. Der Rollenname kann unterschiedlich sein. Von einer Bereitstellungsgruppe kann mehr als eine Instance-Rolle verwendet werden. Weitere Informationen finden Sie unter Erstellen eines 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 S3-Bucket zu gewähren, der von AccountA zum Speichern von Artefakten für Pipelines verwendet wird (in diesem Beispiel -2-1234567890): codepipeline-us-east*

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "s3:Get*"
       ],
       "Resource": [
         "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
       ]
     },
     {
       "Effect": "Allow",
       "Action": [
         "s3:ListBucket"
       ],
       "Resource": [
         "arn:aws:s3:::codepipeline-us-east-2-1234567890"
       ]
     }
   ]
 }
```

6. Wählen Sie Review policy (Richtlinie prüfen).
7. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Create Policy (Richtlinie erstellen) aus.
8. Erstellen Sie eine zweite Richtlinie für AWS KMS, wobei ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** der ARN des vom Kunden verwalteten Schlüssels ist, der in *AccountA* erstellt und so konfiguriert wurde, dass *AccountB* zu seiner Nutzung berechtigt ist:

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

```
        "Effect": "Allow",
        "Action": [
           "kms:DescribeKey",
           "kms:GenerateDataKey*",
           "kms:Encrypt",
           "kms:ReEncrypt*",
           "kms:Decrypt"
         ],
        "Resource": [
           "arn:aws:kms:us-
east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE"
         ]
      }
   ]
}
```

> **Important**
>
> Sie müssen die Konto-ID von *AccountA* in dieser Richtlinie als Teil des Ressourcen-ARN für den AWS KMS-Schlüssel verwenden (wie abgebildet). Ansonsten funktioniert die Richtlinie nicht.

9. Wählen Sie Review policy (Richtlinie prüfen).
10. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Create Policy (Richtlinie erstellen) aus.

Erstellen Sie nun eine IAM-Rolle, die für den kontenübergreifenden Zugriff verwendet werden soll, und konfigurieren Sie sie so, dass *AccountA* die Rolle übernehmen kann. Diese Rolle muss Richtlinien enthalten, die den Zugriff auf die CodeDeploy Ressourcen und den Amazon S3 S3-Bucket ermöglichen, die zum Speichern von Artefakten in *AccountA* verwendet werden.

So konfigurieren Sie die kontenübergreifende Rolle in IAM

1. Melden Sie sich AWS Management Console bei der *AccountB* öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam.
2. Wählen Sie im Navigationsbereich Roles aus. Wählen Sie Create role (Rolle erstellen) aus.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option Anderes AWS Konto aus. Geben Sie unter Konten angeben, die diese Rolle verwenden können, unter Konto-ID die

AWS Konto-ID für das Konto ein, das die Pipeline erstellen soll, in CodePipeline (*AccountA*), und wählen Sie dann Weiter: Berechtigungen aus.

> **Note**
>
> In diesem Schritt wird die Vertrauensstellungsrichtlinie zwischen *AccountA* und *AccountB* erstellt.

4. Wählen Sie unter Berechtigungsrichtlinien anhängen die Option AmazonS3 ReadOnlyAccess 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_Rolle*). Sie können diese Rolle beliebig benennen, solange sie den Namenskonventionen in IAM entspricht. Geben Sie der Rolle einen Namen, der ihren Zweck eindeutig beschreibt. Wählen Sie Create Role aus.
6. Wählen Sie aus der Liste der Rollen 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:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:GetApplicationRevision",
        "codedeploy:RegisterApplicationRevision"
      ],
```

```
      "Resource": "*"
    }
  ]
}
```

9. Wählen Sie Review policy (Richtlinie prüfen).
10. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Create Policy (Richtlinie erstellen) aus.
11. Wählen Sie in der Registerkarte Permissions (Berechtigungen) die Option Add inline policy (Inline-Richtlinie hinzufügen).
12. *Wählen Sie die Registerkarte **JSON** und geben Sie die folgende Richtlinie ein, damit diese Rolle Eingabeartefakte aus dem Amazon S3 S3-Bucket in AccountA abrufen und Ausgabeartefakte in diesen ablegen kann:*

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "s3:GetObject*",
         "s3:PutObject",
         "s3:PutObjectAcl"
       ],
       "Resource": [
         "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
       ]
     }
   ]
}
```

13. Wählen Sie Review policy (Richtlinie prüfen).
14. Geben Sie unter Name einen Namen für diese Richtlinie ein. Wählen Sie Create Policy (Richtlinie erstellen) aus.
15. Suchen Sie auf der Registerkarte Berechtigungen ReadOnlyAccess in der Liste der Richtlinien unter Richtlinienname nach AmazonS3 und klicken Sie neben der Richtlinie auf das Löschsymbol (X). Wählen Sie nach Aufforderung Detach aus.

Schritt 2: Bearbeiten der Pipeline

Sie können nicht die CodePipeline-Konsole verwenden, um eine Pipeline zu erstellen oder zu bearbeiten, die mit einem anderen AWS-Konto verknüpfte Ressourcen verwendet. Sie können die Konsole jedoch verwenden, um die allgemeine Struktur der Pipeline zu erstellen und anschließend die AWS CLI verwenden, um die Pipeline zu bearbeiten und die Ressourcen hinzuzufügen. Alternativ können Sie die Struktur einer vorhandenen Pipeline verwenden und dieser die Ressourcen manuell hinzufügen.

So fügen Sie die Ressourcen hinzu, die einem anderen AWS-Konto zugeordnet sind (AWS CLI)

1. Führen Sie die Eingabeaufforderung (Windows) oder das get-pipeline Terminal (Linux, macOS oder Unix). 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, wobei *codepipeline-us-east-2-1234567890* der Name des Amazon S3 S3-Buckets ist, der zum Speichern von Artefakten für die Pipeline verwendet wird, und der ARN des vom Kunden verwalteten Schlüssels ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** ist, den Sie gerade erstellt haben:

   ```
   {
     "artifactStore”: {
       "location": "codepipeline-us-east-2-1234567890",
       "type": "S3",
       "encryptionKey": {
         "id": "arn:aws:kms:us-
   east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE",
         "type": "KMS"
       }
     },
   ```

3. Fügen Sie in einer Phase eine Bereitstellungsaktion hinzu, um die mit *AccountB* verknüpften CodeDeploy Ressourcen zu verwenden, einschließlich der `roleArn` Werte für die kontenübergreifende Rolle, die Sie erstellt haben (*CrossAccount_Role*).

   Das folgende Beispiel zeigt JSON, das eine Bereitstellungsaktion mit dem Namen hinzufügt *ExternalDeploy*. Es verwendet die in CodeDeployAccountB *in einer Stufe mit dem Namen* Staging *erstellten* -Ressourcen. Im folgenden Beispiel lautet der ARN für *AccountB 012ID_ACCOUNT_B*:

```
,
            {
                "name": "Staging",
                "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "MyAppBuild"
                            }
                        ],
                        "name": "ExternalDeploy",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "AccountBApplicationName",
                            "DeploymentGroupName": "AccountBApplicationGroupName"
                        },
                        "runOrder": 1,
                        "roleArn":
 "arn:aws:iam::012ID_ACCOUNT_B:role/CrossAccount_Role"
                    }
                ]
            }
```

> **Note**
>
> Dies ist nicht der JSON-Code für die gesamte Pipeline. Es handelt sich nur um die Struktur für die Aktion in einer Stufe.

4. Sie müssen die `metadata`-Zeilen aus der Datei entfernen, damit der Befehl update-pipeline sie verwenden kann. Entfernen Sie den Abschnitt aus der Pipeline-Struktur in der JSON-Datei (die `"metadata": { }`-Zeilen und die Fehler `"created"`, `"pipelineARN"` und `"updated"`).

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

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

   Speichern Sie die Datei.

5. Führen Sie den Befehl update-pipeline aus, um die Änderungen zu übernehmen. Geben Sie die Pipeline-JSON-Datei dabei folgendermaßen an:

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

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

   Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

So testen Sie die Pipeline mit Ressourcen aus einem anderen AWS-Konto

1. Führen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix) oder eine Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix): start-pipeline-execution

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

Weitere Informationen finden Sie unter Manuelles Starten einer Pipeline.

2. Melden Sie sich AWS Management Console mit *AccountA* bei der an 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.

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, bis eine Erfolgsmeldung für die Aktion mit Ressource aus einem anderen AWS-Konto angezeigt wird.

> **Note**
>
> Sie erhalten eine Fehlermeldung, wenn Sie versuchen, Details für eine Aktion anzuzeigen, während Sie mit *AccountA* angemeldet sind. Melden Sie sich ab und melden Sie sich dann mit *AccountB* an, um die Bereitstellungsdetails in CodeDeploy anzuzeigen.

Migrieren Sie Abfrage-Pipelines, um die ereignisbasierte Änderungserkennung zu nutzen

AWS CodePipelineunterstützt die vollständige, end-to-end kontinuierliche Bereitstellung, was das Starten Ihrer Pipeline bei jeder Codeänderung einschließt. Es gibt zwei unterstützte Methoden, um Ihre Pipeline nach einer Codeänderung zu starten: ereignisbasierte Änderungserkennung und Abfrage. Wir empfehlen, die ereignisbasierte Änderungserkennung für Pipelines zu verwenden.

Verwenden Sie die hier aufgeführten Verfahren, um Ihre Abfrage-Pipelines auf die ereignisbasierte Methode zur Änderungserkennung für Ihre Pipeline zu migrieren (zu aktualisieren).

Die empfohlene Methode zur ereignisbasierten Änderungserkennung für Pipelines wird durch die Pipeline-Quelle bestimmt, z. B. CodeCommit In diesem Fall müsste die Polling-Pipeline beispielsweise auf die ereignisbasierte Änderungserkennung mit migrieren. EventBridge

Wie migriert man Polling-Pipelines

Um Abfrage-Pipelines zu migrieren, ermitteln Sie Ihre Abfrage-Pipelines und legen Sie dann die empfohlene Methode zur ereignisbasierten Änderungserkennung fest:

- Verwenden Sie die unter beschriebenen Schritte, um Ihre Abfrage-Pipelines Polling-Pipelines in Ihrem Konto anzeigen zu ermitteln.
- Suchen Sie in der Tabelle nach Ihrem Pipeline-Quelltyp und wählen Sie dann das Verfahren mit der Implementierung aus, die Sie für die Migration Ihrer Polling-Pipeline verwenden möchten. Jeder Abschnitt enthält mehrere Methoden für die Migration, z. B. die Verwendung der CLI oderAWS CloudFormation.

Migrieren von Pipelines zur empfohlenen Methode zur Änderungserkennung		
Pipeline-Quelle	Empfohlene ereignisbasierte Erkennungsmethode	Migrationsverfahren
AWS CodeCommit	EventBridge (empfohlen).	Siehe Migrieren Sie Abfrage-Pipelines mit einer Quelle CodeCommit .
Amazon S3	EventBridge und Bucket für Ereignisbenachrichtigungen aktiviert (empfohlen).	Siehe Migrieren Sie Abfrage-Pipelines mit einer für Ereignisse aktivierten S3-Quelle.
Amazon S3	EventBridge und eine AWS CloudTrail Spur.	Siehe Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail .
GitHub Version 1	Verbindungen (empfohlen)	Siehe Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Verbindungen.
GitHub Version 1	Webhooks	Siehe Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Webhooks.

> **Important**
>
> Für entsprechende Aktualisierungen der Konfiguration von Pipeline-Aktionen, wie z. B. Pipelines mit einer Aktion der GitHub Version 1, müssen Sie den `PollForSourceChanges` Parameter in der Konfiguration Ihrer Source-Aktion explizit auf false setzen, um zu verhindern, dass eine Pipeline Abfragen durchführt. Daher ist es möglich, eine Pipeline fälschlicherweise sowohl mit ereignisbasierter Änderungserkennung als auch mit Abfrage zu konfigurieren, indem Sie beispielsweise eine EventBridge Regel konfigurieren und gleichzeitig den Parameter weglassen. `PollForSourceChanges` Dies führt zu duplizierten Pipelineausführungen, wobei die Pipeline bei der maximalen Gesamtzahl von Abfrage-Pipelines berücksichtigt wird. Standardmäßig handelt es sich hierbei um einen sehr viel niedrigeren Wert als für ereignisbasierte Pipelines. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline.

Polling-Pipelines in Ihrem Konto anzeigen

Verwenden Sie als ersten Schritt eines der folgenden Skripts, um zu ermitteln, welche Pipelines in Ihrem Konto für Abfragen konfiguriert sind. Dies sind die Pipelines für die Migration zur ereignisbasierten Änderungserkennung.

Abfrage-Pipelines in Ihrem Konto anzeigen (Skript)

Gehen Sie wie folgt vor, um mithilfe eines Skripts die Pipelines in Ihrem Konto zu ermitteln, die Polling verwenden.

1. Öffnen Sie ein Terminalfenster und führen Sie dann einen der folgenden Schritte aus:

   - Führen Sie den folgenden Befehl aus, um ein neues Skript mit dem Namen PollingPipelinesExtractor.sh zu erstellen.

     ```
     vi PollingPipelinesExtractor.sh
     ```

   - Um ein Python-Skript zu verwenden, führen Sie den folgenden Befehl aus, um ein neues Python-Skript namens PollingPipelinesExtractor.py zu erstellen.

     ```
     vi PollingPipelinesExtractor.py
     ```

2. Kopieren Sie den folgenden Code und fügen Sie ihn in das PollingPipelinesExtractorSkript ein. Führen Sie eine der folgenden Aktionen aus:

   - Kopieren Sie den folgenden Code und fügen Sie ihn in das PollingPipelinesExtractor.sh-Skript ein.

```
#!/bin/bash

set +x

POLLING_PIPELINES=()
LAST_EXECUTED_DATES=()
NEXT_TOKEN=null
HAS_NEXT_TOKEN=true
if [[ $# -eq 0 ]] ; then
    echo 'Please provide region name'
    exit 0
fi
REGION=$1


while [ "$HAS_NEXT_TOKEN" != "false" ]; do
    if [ "$NEXT_TOKEN" != "null" ];
        then
            LIST_PIPELINES_RESPONSE=$(aws codepipeline list-pipelines --region
 $REGION --next-token $NEXT_TOKEN)
        else
            LIST_PIPELINES_RESPONSE=$(aws codepipeline list-pipelines --region
 $REGION)
    fi
    LIST_PIPELINES=$(jq -r '.pipelines[].name' <<< "$LIST_PIPELINES_RESPONSE")
    NEXT_TOKEN=$(jq -r '.nextToken' <<< "$LIST_PIPELINES_RESPONSE")
    if [ "$NEXT_TOKEN" == "null" ];
        then
            HAS_NEXT_TOKEN=false
    fi

    for pipline_name in $LIST_PIPELINES
    do
        PIPELINE=$(aws codepipeline get-pipeline --name $pipline_name --region
 $REGION)
        HAS_POLLABLE_ACTIONS=$(jq '.pipeline.stages[].actions[] |
 select(.actionTypeId.category == "Source") | select(.actionTypeId.owner
```

```
 == ("ThirdParty","AWS")) | select(.actionTypeId.provider ==
 ("GitHub","S3","CodeCommit")) | select(.configuration.PollForSourceChanges ==
 ("true",null))' <<< "$PIPELINE")
        if [ ! -z "$HAS_POLLABLE_ACTIONS" ];
        then
            POLLING_PIPELINES+=("$pipline_name")
            PIPELINE_EXECUTIONS=$(aws codepipeline list-pipeline-executions --
pipeline-name $pipline_name --region $REGION)
            LAST_EXECUTION=$(jq -r '.pipelineExecutionSummaries[0]' <<<
 "$PIPELINE_EXECUTIONS")
            if [ "$LAST_EXECUTION" != "null" ];
                then
                    LAST_EXECUTED_TIMESTAMP=$(jq -r '.startTime' <<<
 "$LAST_EXECUTION")
                    LAST_EXECUTED_DATE="$(date -r ${LAST_EXECUTED_TIMESTAMP%.*})"
                else
                    LAST_EXECUTED_DATE="Not executed in last year"
            fi
            LAST_EXECUTED_DATES+=("$LAST_EXECUTED_DATE")
        fi
    done

done

fileName=$REGION-$(date +%s)
printf "| %-30s | %-30s |\n" "Polling Pipeline Name" "Last Executed Time"
printf "| %-30s | %-30s |\n" "_____________________" "___________________"
for i in "${!POLLING_PIPELINES[@]}"; do
  printf "| %-30s | %-30s |\n" "${POLLING_PIPELINES[i]}"
 "${LAST_EXECUTED_DATES[i]}"
  printf "${POLLING_PIPELINES[i]}," >> $fileName.csv
done

printf "\nSaving Polling Pipeline Names to file $fileName.csv."
```

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

```
import boto3
import sys
import time
import math
```

```
hasNextToken = True
nextToken = ""
pollablePipelines = []
lastExecutedTimes = []
if len(sys.argv) == 1:
    raise Exception("Please provide region name.")
session = boto3.Session(profile_name='default', region_name=sys.argv[1])
codepipeline = session.client('codepipeline')

def is_pollable_action(action):
    actionTypeId = action['actionTypeId']
    configuration = action['configuration']
    return actionTypeId['owner'] in {"AWS", "ThirdParty"}
 and actionTypeId['provider'] in {"GitHub", "CodeCommit",
 "S3"} and ('PollForSourceChanges' not in configuration or
 configuration['PollForSourceChanges'] == 'true')

def has_pollable_actions(pipeline):
    hasPollableAction = False
    pipelineDefinition = codepipeline.get_pipeline(name=pipeline['name'])
['pipeline']
    for action in pipelineDefinition['stages'][0]['actions']:
        hasPollableAction = is_pollable_action(action)
        if hasPollableAction:
            break
    return hasPollableAction

def get_last_executed_time(pipelineName):

 pipelineExecutions=codepipeline.list_pipeline_executions(pipelineName=pipelineName)
['pipelineExecutionSummaries']
    if pipelineExecutions:
        return pipelineExecutions[0]['startTime'].strftime("%A %m/%d/%Y, %H:%M:
%S")
    else:
        return "Not executed in last year"

while hasNextToken:
    if nextToken=="":
        list_pipelines_response = codepipeline.list_pipelines()
    else:
        list_pipelines_response =
 codepipeline.list_pipelines(nextToken=nextToken)
    if 'nextToken' in list_pipelines_response:
```

```
        nextToken = list_pipelines_response['nextToken']
    else:
        hasNextToken= False
    for pipeline in list_pipelines_response['pipelines']:
        if has_pollable_actions(pipeline):
            pollablePipelines.append(pipeline['name'])
            lastExecutedTimes.append(get_last_executed_time(pipeline['name']))

fileName="{region}-
{timeNow}.csv".format(region=sys.argv[1],timeNow=math.trunc(time.time()))
file = open(fileName, 'w')

print ("{:<30} {:<30} {:<30}".format('Polling Pipeline Name', '|','Last Executed
 Time'))
print ("{:<30} {:<30} {:<30}".format('_____________________',
 '|','___________________'))
for i in range(len(pollablePipelines)):
    print("{:<30} {:<30} {:<30}".format(pollablePipelines[i], '|',
 lastExecutedTimes[i]))
    file.write("{pipeline},".format(pipeline=pollablePipelines[i]))
file.close()
print("\nSaving Polling Pipeline Names to file
 {fileName}".format(fileName=fileName))
```

3. Für jede Region, in der Sie Pipelines haben, müssen Sie das Skript für diese Region ausführen. Gehen Sie wie folgt vor, um das Skript auszuführen:

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

     ```
     ./PollingPipelinesExtractor.sh us-west-2
     ```

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

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

   In der folgenden Beispielausgabe des Skripts gab die Region us-west-2 eine Liste von Polling-Pipelines zurück und zeigt die letzte Ausführungszeit für jede Pipeline an.

```
 % ./pollingPipelineExtractor.sh us-west-2

| Polling Pipeline Name | Last Executed Time           |
| _____________________ | ____________________         |
| myCodeBuildPipeline   | Wed Mar 8 09:35:49 PST 2023  |
| myCodeCommitPipeline  | Mon Apr 24 22:32:32 PDT 2023 |
| TestPipeline          | Not executed in last year    |

Saving list of polling pipeline names to us-west-2-1682496174.csv...%
```

Analysieren Sie die Skriptausgabe und aktualisieren Sie für jede Pipeline in der Liste die Abfragequelle auf die empfohlene ereignisbasierte Methode zur Erkennung von Änderungen.

> **Note**
>
> Ihre Abfrage-Pipelines werden durch die Aktionskonfiguration der Pipeline für den Parameter bestimmt. `PollForSourceChanges` Wenn in der Konfiguration der Pipeline-Quelle der `PollForSourceChanges` Parameter weggelassen wurde, wird CodePipeline standardmäßig Ihr Repository nach Quelländerungen abgefragt. Dieses Verhalten ist dasselbe, als ob `PollForSourceChanges` es enthalten und auf true gesetzt wäre. Weitere Informationen finden Sie in den Konfigurationsparametern für die Quellaktion Ihrer Pipeline, z. B. in den Konfigurationsparametern der Amazon S3 S3-Quellaktion unterQuellaktion für Amazon S3.

Beachten Sie, dass dieses Skript auch eine .csv-Datei generiert, die die Liste der Abfrage-Pipelines in Ihrem Konto enthält, und die CSV-Datei im aktuellen Arbeitsordner speichert.

Migrieren Sie Abfrage-Pipelines mit einer Quelle CodeCommit

Sie können Ihre Polling-Pipeline migrieren, um sie EventBridge zur Erkennung von Änderungen in Ihrem CodeCommit Quell-Repository oder Ihrem Amazon S3 S3-Quell-Bucket zu verwenden.

CodeCommit-- Bei einer Pipeline mit einer CodeCommit Quelle ändern Sie die Pipeline so, dass die Änderungserkennung automatisiert wird. EventBridge Wählen Sie aus den folgenden Methoden, um die Migration zu implementieren:

- Konsole: Migrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole)
- CLI: Migrieren von Abfrage-Pipelines (CodeCommit Quelle) (CLI)

- AWS CloudFormation: Migrieren Sie Abfrage-Pipelines (CodeCommit Quelle) (AWS CloudFormationVorlage)

Migrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole)

Sie können die CodePipeline Konsole verwenden, um Ihre Pipeline zu aktualisieren, um Änderungen in Ihrem CodeCommit Quell-Repository oder Ihrem Amazon S3 S3-Quell-Bucket zu erkennen. EventBridge

> **Note**
>
> Wenn Sie die Konsole verwenden, um eine Pipeline zu bearbeiten, die über ein CodeCommit Quell-Repository oder einen Amazon S3 S3-Quell-Bucket verfügt, werden die Regel und die IAM-Rolle für Sie erstellt. Wenn Sie die AWS CLI Pipeline bearbeiten, müssen Sie die EventBridge Regel und die IAM-Rolle selbst erstellen. Weitere Informationen finden Sie unter CodeCommit -Quellaktionen und EventBridge.

Führen Sie die folgenden Schritte aus, um eine Pipeline zu bearbeiten, die periodische Prüfungen verwendet. Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen einer Pipeline in CodePipeline.

So bearbeiten Sie die Quellphase einer Pipeline

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

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Auf diese Weise wird eine detaillierte Ansicht der Pipeline geöffnet (einschließlich des Status der Aktionen in den einzelnen Stufen der Pipeline).
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie in der Stufe Edit stage (Stufe bearbeiten) das Bearbeitungssymbol für die Quellaktion aus.
5. Erweitern Sie die Optionen zur Änderungserkennung und wählen Sie CloudWatch Ereignisse verwenden, um meine Pipeline automatisch zu starten, wenn eine Änderung eintritt (empfohlen).

Es wird eine Meldung mit der EventBridge Regel angezeigt, die für diese Pipeline erstellt werden soll. Wählen Sie Aktualisieren aus.

Wenn Sie eine Pipeline aktualisieren, die über eine Amazon S3 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. Veröffentlichen Sie zum Testen der Aktion eine Änderung mit der AWS CLI, indem Sie einen Commit für eine Änderung der in der Quellstufe der Pipeline festgelegten Quelle durchführen.

Migrieren von Abfrage-Pipelines (CodeCommit Quelle) (CLI)

Gehen Sie wie folgt vor, um eine Pipeline zu bearbeiten, die Abfragen (regelmäßige Prüfungen) verwendet, um eine EventBridge Regel zum Starten der Pipeline zu verwenden. Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen einer Pipeline in CodePipeline.

Um eine ereignisgesteuerte Pipeline mit zu erstellen CodeCommit, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und erstellen dann die folgenden Ressourcen:

- EventBridge Ereignis
- IAM-Rolle, damit dieses Ereignis Ihre Pipeline starten kann

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

```
aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem beliebigen Texteditor und bearbeiten Sie die Quellstufe, indem Sie den Parameter `PollForSourceChanges` in `false` ändern, wie in diesem Beispiel gezeigt.

   Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

```
"configuration": {
    "PollForSourceChanges": "false",
    "BranchName": "main",
    "RepositoryName": "MyTestRepo"
},
```

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

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

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

   Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

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

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

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den **`start-pipeline-execution`**-Befehl, um Ihre Pipeline manuell zu starten.

Um eine EventBridge Regel mit CodeCommit als Ereignisquelle und CodePipeline als Ziel zu erstellen

1. Fügen Sie Berechtigungen hinzu EventBridge , die CodePipeline zum Aufrufen der Regel verwendet werden sollen. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie zu erstellen, die es ermöglicht, die Servicerolle EventBridge zu übernehmen. Geben Sie der Vertrauensrichtlinie den Namen `trustpolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie in diesem Beispiel für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Verwenden Sie den folgenden Befehl, um die Berechtigungsrichtlinie `CodePipeline-Permissions-Policy-for-EB` der Rolle `Role-for-MyRule` anzufügen.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen dieser Richtlinie zur Rolle werden Berechtigungen für erstellt EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

Warum nehme ich diese Änderung vor? Dieser Befehl aktiviert AWS CloudFormation, um das Ereignis zu erstellen.

Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „MyCodeCommitRepoRule“ erstellt.

```
aws events put-rule --name "MyCodeCommitRepoRule" --event-pattern "{\"source\":
[\"aws.codecommit\"],\"detail-type\":[\"CodeCommit Repository State Change\"],
\"resources\":[\"repository-ARN\"],\"detail\":{\"referenceType\":[\"branch\"],
\"referenceName\":[\"main\"]}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-
MyRule"
```

3. Um es CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die folgenden Parameter an:

   - Der Parameter `--rule` wird für den `rule_name` verwendet, den Sie mit put-rule erstellt haben.
   - Der Parameter `--targets` wird für die Listen-`Id` des Ziels in der Zielliste und den ARN der Ziel-Pipeline verwendet.

   Der folgende Beispielbefehl legt fest, dass für die Regel mit dem Namen `MyCodeCommitRepoRule` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dieses Ziel 1 ist. Der Beispielbefehl gibt zudem ein Beispiel ARN für die Pipeline an. Die Pipeline startet, wenn Änderungen im Repository auftreten.

   ```
   aws events put-targets --rule MyCodeCommitRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

Migrieren Sie Abfrage-Pipelines (CodeCommit Quelle) (AWS CloudFormationVorlage)

Um eine ereignisgesteuerte Pipeline mit AWS CodeCommit zu erstellen, bearbeiten Sie den Parameter `PollForSourceChanges` Ihrer Pipeline und fügen anschließend die folgenden Ressourcen Ihrer Vorlage hinzu:

- Eine Regel EventBridge
- Eine IAM-Rolle für Ihre Regel EventBridge

Wenn Sie AWS CloudFormation zum Erstellen und Verwalten Ihrer Pipelines verwenden, sind in Ihrer Vorlage Inhalte wie die folgenden enthalten.

> **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
        }]
    },
```

Aktualisieren der AWS CloudFormation-Vorlage Ihrer Pipeline und Erstellen einer EventBridge -Regel

1. Verwenden Sie die AWS CloudFormation-Ressource `AWS::IAM::Role` unter `Resources` in der Vorlage, um die IAM-Rolle zu konfigurieren, die Ihrem Ereignis das Starten Ihrer Pipeline erlaubt. Dieser Eintrag erstellt eine Rolle mit zwei Richtlinien:

   - Die erste Richtlinie ermöglicht die Übernahme der Rolle.
   - Die zweite Richtlinie stellt Berechtigungen zum Starten der Pipeline bereit.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::IAM::Role` Ressource können AWS CloudFormation Sie Berechtigungen für EventBridge erstellen. Diese Ressource wird Ihrem AWS CloudFormation-Stack hinzugefügt.

YAML

```
  EventRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
 'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
```

JSON

```
"EventRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "events.amazonaws.com"
            ]
          },
```

```
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "eb-pipeline-execution",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": "codepipeline:StartPipelineExecution",
              "Resource": {
                "Fn::Join": [
                  "",
                  [
                    "arn:aws:codepipeline:",
                    {
                      "Ref": "AWS::Region"
                    },
                    ":",
                    {
                      "Ref": "AWS::AccountId"
                    },
                    ":",
                    {
                      "Ref": "AppPipeline"
                    }
                  ]

...
```

2. Verwenden Sie in der Vorlage unter die `AWS::Events::Rule` AWS CloudFormation Ressource`Resources`, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis, das Push-Änderungen an Ihrem Repository überwacht. Wenn eine Änderung des Repository-Status EventBridge erkannt wird, wird die Regel in `StartPipelineExecution` Ihrer Zielpipeline aufgerufen.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::Events::Rule`-Ressource kann AWS CloudFormation das Ereignis erstellen. Diese Ressource wird Ihrem AWS CloudFormation-Stack hinzugefügt.

YAML

```
  EventRule:
    Type: AWS::Events::Rule
    Properties:
      EventPattern:
        source:
          - aws.codecommit
        detail-type:
          - 'CodeCommit Repository State Change'
        resources:
          - !Join [ '', [ 'arn:aws:codecommit:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref RepositoryName ] ]
        detail:
          event:
            - referenceCreated
            - referenceUpdated
          referenceType:
            - branch
          referenceName:
            - main
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
```

JSON

```
"EventRule": {
  "Type": "AWS::Events::Rule",
  "Properties": {
    "EventPattern": {
      "source": [
        "aws.codecommit"
      ],
      "detail-type": [
        "CodeCommit Repository State Change"
      ],
```

```
      "resources": [
        {
          "Fn::Join": [
            "",
            [
              "arn:aws:codecommit:",
              {
                "Ref": "AWS::Region"
              },
              ":",
              {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "RepositoryName"
              }
            ]
          ]
        }
      ],
      "detail": {
        "event": [
          "referenceCreated",
          "referenceUpdated"
        ],
        "referenceType": [
          "branch"
        ],
        "referenceName": [
          "main"
        ]
      }
    },
    "Targets": [
      {
        "Arn": {
          "Fn::Join": [
            "",
            [
              "arn:aws:codepipeline:",
              {
                "Ref": "AWS::Region"
              },
```

```
              ":",
              {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "AppPipeline"
              }
            ]
          ]
        },
        "RoleArn": {
          "Fn::GetAtt": [
            "EventRole",
            "Arn"
          ]
        },
        "Id": "codepipeline-AppPipeline"
      }
    ]
  }
},
```

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation-Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> In vielen Fällen ist der Parameter `PollForSourceChanges` „true“, wenn Sie eine Pipeline erstellen. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls

wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

  Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

  YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: CodeCommit
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              BranchName: !Ref BranchName
              RepositoryName: !Ref RepositoryName
              PollForSourceChanges: false
            RunOrder: 1
```

  JSON

```
{
  "Name": "Source",
  "Actions": [
    {
      "Name": "SourceAction",
      "ActionTypeId": {
        "Category": "Source",
        "Owner": "AWS",
```

```
        "Version": 1,
        "Provider": "CodeCommit"
      },
      "OutputArtifacts": [
        {
          "Name": "SourceOutput"
        }
      ],
      "Configuration": {
        "BranchName": {
          "Ref": "BranchName"
        },
        "RepositoryName": {
          "Ref": "RepositoryName"
        },
        "PollForSourceChanges": false
      },
      "RunOrder": 1
    }
  ]
},
```

Example

Wenn Sie diese Ressourcen mit AWS CloudFormation erstellen, wird Ihre Pipeline ausgelöst, wenn Dateien in Ihrem Repository erstellt oder aktualisiert werden. Im Folgenden finden Sie den endgültigen Vorlagenausschnitt:

YAML

```
Resources:
  EventRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
```

```
              - events.amazonaws.com
          Action: sts:AssumeRole
    Path: /
    Policies:
      -
        PolicyName: eb-pipeline-execution
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            -
              Effect: Allow
              Action: codepipeline:StartPipelineExecution
              Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
 EventRule:
   Type: AWS::Events::Rule
   Properties:
     EventPattern:
       source:
         - aws.codecommit
       detail-type:
         - 'CodeCommit Repository State Change'
       resources:
         - !Join [ '', [ 'arn:aws:codecommit:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref RepositoryName ] ]
       detail:
         event:
           - referenceCreated
           - referenceUpdated
         referenceType:
           - branch
         referenceName:
           - main
     Targets:
       -
         Arn:
           !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
         RoleArn: !GetAtt EventRole.Arn
         Id: codepipeline-AppPipeline
 AppPipeline:
   Type: AWS::CodePipeline::Pipeline
   Properties:
     Name: codecommit-events-pipeline
```

```
      RoleArn:
        !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: CodeCommit
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                BranchName: !Ref BranchName
                RepositoryName: !Ref RepositoryName
                PollForSourceChanges: false
              RunOrder: 1


...
```

JSON

```
    "Resources": {

...

       "EventRole": {
           "Type": "AWS::IAM::Role",
           "Properties": {
               "AssumeRolePolicyDocument": {
                   "Version": "2012-10-17",
                   "Statement": [
                       {
                           "Effect": "Allow",
                           "Principal": {
                               "Service": [
                                   "events.amazonaws.com"
                               ]
                           },
```

```
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "eb-pipeline-execution",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "codepipeline:StartPipelineExecution",
                                "Resource": {
                                    "Fn::Join": [
                                        "",
                                        [
                                            "arn:aws:codepipeline:",
                                            {
                                                "Ref": "AWS::Region"
                                            },
                                            ":",
                                            {
                                                "Ref": "AWS::AccountId"
                                            },
                                            ":",
                                            {
                                                "Ref": "AppPipeline"
                                            }
                                        ]
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
        }
    },
    "EventRule": {
        "Type": "AWS::Events::Rule",
        "Properties": {
            "EventPattern": {
```

```
                    "source": [
                        "aws.codecommit"
                    ],
                    "detail-type": [
                        "CodeCommit Repository State Change"
                    ],
                    "resources": [
                        {
                            "Fn::Join": [
                                "",
                                [
                                    "arn:aws:codecommit:",
                                    {
                                        "Ref": "AWS::Region"
                                    },
                                    ":",
                                    {
                                        "Ref": "AWS::AccountId"
                                    },
                                    ":",
                                    {
                                        "Ref": "RepositoryName"
                                    }
                                ]
                            ]
                        }
                    ],
                    "detail": {
                        "event": [
                            "referenceCreated",
                            "referenceUpdated"
                        ],
                        "referenceType": [
                            "branch"
                        ],
                        "referenceName": [
                            "main"
                        ]
                    }
                },
                "Targets": [
                    {
                        "Arn": {
                            "Fn::Join": [
```

```
                            "",
                            [
                                "arn:aws:codepipeline:",
                                {
                                    "Ref": "AWS::Region"
                                },
                                ":",
                                {
                                    "Ref": "AWS::AccountId"
                                },
                                ":",
                                {
                                    "Ref": "AppPipeline"
                                }
                            ]
                        ]
                    },
                    "RoleArn": {
                        "Fn::GetAtt": [
                            "EventRole",
                            "Arn"
                        ]
                    },
                    "Id": "codepipeline-AppPipeline"
                }
            ]
        }
    },
    "AppPipeline": {
        "Type": "AWS::CodePipeline::Pipeline",
        "Properties": {
            "Name": "codecommit-events-pipeline",
            "RoleArn": {
                "Fn::GetAtt": [
                    "CodePipelineServiceRole",
                    "Arn"
                ]
            },
            "Stages": [
                {
                    "Name": "Source",
                    "Actions": [
                        {
                            "Name": "SourceAction",
```

```
                            "ActionTypeId": {
                                "Category": "Source",
                                "Owner": "AWS",
                                "Version": 1,
                                "Provider": "CodeCommit"
                            },
                            "OutputArtifacts": [
                                {
                                    "Name": "SourceOutput"
                                }
                            ],
                            "Configuration": {
                                "BranchName": {
                                    "Ref": "BranchName"
                                },
                                "RepositoryName": {
                                    "Ref": "RepositoryName"
                                },
                                "PollForSourceChanges": false
                            },
                            "RunOrder": 1
                        }
                    ]
                },

...
```

Migrieren Sie Abfrage-Pipelines mit einer für Ereignisse aktivierten S3-Quelle

Bei einer Pipeline mit einer Amazon S3 S3-Quelle ändern Sie die Pipeline so, dass die Änderungserkennung über EventBridge und mit einem Quell-Bucket, der für Ereignisbenachrichtigungen aktiviert ist, automatisiert wird. Dies ist die empfohlene Methode, wenn Sie die CLI verwenden oder Ihre Pipeline migrieren AWS CloudFormation möchten.

> **Note**
>
> Dazu gehört die Verwendung eines Buckets, der für Ereignisbenachrichtigungen aktiviert ist, sodass Sie keinen separaten CloudTrail Trail erstellen müssen. Wenn Sie die Konsole verwenden, werden eine Ereignisregel und ein CloudTrail Trail für Sie eingerichtet.

Informationen zu diesen Schritten finden Sie unterMigrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail .

- CLI: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (CLI)
- AWS CloudFormation: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (Vorlage) AWS CloudFormation

Migrieren Sie Abfrage-Pipelines mit einer für Ereignisse aktivierten S3-Quelle (CLI)

Gehen Sie wie folgt vor, um eine Pipeline zu bearbeiten, die Polling (regelmäßige Prüfungen) verwendet, um stattdessen ein Ereignis in zu verwenden. EventBridge Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen einer Pipeline in CodePipeline.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und erstellen dann die folgenden Ressourcen:

- EventBridge Ereignisregel
- IAM-Rolle, damit das EventBridge Ereignis Ihre Pipeline starten kann

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Erteilen Sie Berechtigungen EventBridge , die CodePipeline zum Aufrufen der Regel verwendet werden können. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel zum Erstellen der Vertrauensrichtlinie, mit der EventBridge die Servicerolle übernehmen kann. Geben Sie ihr den Namen `trustpolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen dieser Vertrauensrichtlinie zur Rolle werden Berechtigungen für 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`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Führen Sie den folgenden Befehl aus, um der erstellten `Role-for-MyRule`-Rolle die neue `CodePipeline-Permissions-Policy-for-EB`-Berechtigungsrichtlinie anzufügen.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „EnabledS3SourceRule“ erstellt.

```
aws events put-rule --name "EnabledS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"Object Created\"],\"detail\":{\"bucket\":{\"name\":
[\"my-bucket\"]}}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Um das Objekt CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die Parameter `--rule` und `--targets` an.

   Der folgende Befehl legt fest, dass für die Regel mit dem Namen `EnabledS3SourceRule` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dies Ziel 1 ist. Der Befehl gibt zudem ein Beispiel ARN für die Pipeline an. Die Pipeline startet, wenn Änderungen im Repository auftreten.

   ```
   aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
   AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.

2. Öffnen Sie die JSON-Datei in einem Texteditor und bearbeiten Sie die Quellphase, indem Sie den Parameter `PollForSourceChanges` für einen Bucket mit dem Namen `storage-bucketin` `false` ändern wie in diesem Beispiel gezeigt.

   Warum nehme ich diese Änderung vor? Durch Festlegen dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

   ```
   "configuration": {
       "S3Bucket": "storage-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

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

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

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

   Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

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

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

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den start-pipeline-execution-Befehl, um Ihre Pipeline manuell zu starten.

Migrieren Sie Abfrage-Pipelines mit einer für Ereignisse aktivierten S3-Quelle (Vorlage) AWS CloudFormation

Dieses Verfahren gilt für eine Pipeline, bei der im Quell-Bucket Ereignisse aktiviert sind.

Gehen Sie wie folgt vor, um Ihre Pipeline mit einer Amazon S3 S3-Quelle von der Abfrage bis zur ereignisbasierten Änderungserkennung zu bearbeiten.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und fügen dann Ihrer Vorlage die folgenden Ressourcen hinzu:

- EventBridge Regel und IAM-Rolle, damit dieses Ereignis Ihre Pipeline starten kann.

Wenn Sie AWS CloudFormation zum Erstellen und Verwalten Ihrer Pipelines verwenden, sind in Ihrer Vorlage Inhalte wie die folgenden enthalten.

> **Note**
>
> Die `Configuration`-Eigenschaft in der Quellstufe mit dem Namen `PollForSourceChanges`. Wenn diese Eigenschaft in Ihrer Vorlage nicht enthalten ist, dann wird `PollForSourceChanges` standardmäßig auf `true` festgelegt.

YAML

```
AppPipeline:
```

```
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn: !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                -
                  Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref S3SourceObjectKey
                PollForSourceChanges: true
              RunOrder: 1


...
```

JSON

```
       "AppPipeline": {
           "Type": "AWS::CodePipeline::Pipeline",
           "Properties": {
               "RoleArn": {
                   "Fn::GetAtt": ["CodePipelineServiceRole", "Arn"]
               },
               "Stages": [
                   {
                       "Name": "Source",
                       "Actions": [
                           {
                               "Name": "SourceAction",
                               "ActionTypeId": {
                                   "Category": "Source",
                                   "Owner": "AWS",
```

```
                                "Version": 1,
                                "Provider": "S3"
                            },
                            "OutputArtifacts": [
                                {
                                    "Name": "SourceOutput"
                                }
                            ],
                            "Configuration": {
                                "S3Bucket": {
                                    "Ref": "SourceBucket"
                                },
                                "S3ObjectKey": {
                                    "Ref": "SourceObjectKey"
                                },
                                "PollForSourceChanges": true
                            },
                            "RunOrder": 1
                        }
                    ]
                },


...
```

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Verwenden Sie die AWS CloudFormation-Ressource `AWS::IAM::Role` unter `Resources` in der Vorlage, um die IAM-Rolle zu konfigurieren, die Ihrem Ereignis das Starten Ihrer Pipeline erlaubt. 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 AWS CloudFormation ermöglicht das Erstellen von Berechtigungen für EventBridge. Diese Ressource wird Ihrem AWS CloudFormation-Stack hinzugefügt.

YAML

```
  EventRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
 'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]


...
```

JSON

```
  "EventRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": [
```

```
                "events.amazonaws.com"
              ]
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "Path": "/",
      "Policies": [
        {
          "PolicyName": "eb-pipeline-execution",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Action": "codepipeline:StartPipelineExecution",
                "Resource": {
                  "Fn::Join": [
                    "",
                    [
                      "arn:aws:codepipeline:",
                      {
                        "Ref": "AWS::Region"
                      },
                      ":",
                      {
                        "Ref": "AWS::AccountId"
                      },
                      ":",
                      {
                        "Ref": "AppPipeline"
                      }
                    ]
                  ]

...
```

2. Verwenden Sie die `AWS::Events::Rule` AWS CloudFormation Ressource, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis, das die Erstellung oder Löschung von Objekten in Ihrem Amazon S3 S3-Quell-Bucket überwacht. Fügen Sie darüber hinaus ein Ziel für Ihre Pipeline ein. Wenn ein Objekt erstellt wird, wird diese Regel in `StartPipelineExecution` Ihrer Zielpipeline aufgerufen.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::Events::Rule`-Ressource kann AWS CloudFormation das Ereignis erstellen. Diese Ressource wird Ihrem AWS CloudFormation-Stack hinzugefügt.

YAML

```
  EventRule:
    Type: AWS::Events::Rule
    Properties:
      EventBusName: default
      EventPattern:
        source:
          - aws.s3
        detail-type:
          - Object Created
        detail:
          bucket:
            name:
              - !Ref SourceBucket
      Name: EnabledS3SourceRule
      State: ENABLED
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline


...
```

JSON

```
  "EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
  "EventBusName": "default",
  "EventPattern": {
      "source": [
   "aws.s3"
```

```
      ],
      "detail-type": [
    "Object Created"
      ],
      "detail": {
    "bucket": {
        "name": [
      "s3-pipeline-source-fra-bucket"
        ]
        }
            }
  },
  "Name": "EnabledS3SourceRule",
        "State": "ENABLED",
        "Targets": [
        {
          "Arn": {
            "Fn::Join": [
              "",
              [
                "arn:aws:codepipeline:",
                {
                  "Ref": "AWS::Region"
                },
                ":",
                {
                  "Ref": "AWS::AccountId"
                },
                ":",
                {
                  "Ref": "AppPipeline"
                }
              ]
            ]
          },
          "RoleArn": {
            "Fn::GetAtt": [
              "EventRole",
              "Arn"
            ]
          },
          "Id": "codepipeline-AppPipeline"
        }
      ]
```

```
    }
  }
},

...
```

3. Speichern Sie Ihre aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation-Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie Ihre aktualisierte Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Um den 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 Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

  Warum nehme ich diese Änderung vor? Durch Ändern von `PollForSourceChanges` in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

  YAML

```
          Name: Source
```

```
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
```

JSON

```
{
   "Name": "SourceAction",
   "ActionTypeId": {
     "Category": "Source",
     "Owner": "AWS",
     "Version": 1,
     "Provider": "S3"
   },
   "OutputArtifacts": [
     {
       "Name": "SourceOutput"
     }
   ],
   "Configuration": {
     "S3Bucket": {
       "Ref": "SourceBucket"
     },
     "S3ObjectKey": {
       "Ref": "SourceObjectKey"
     },
     "PollForSourceChanges": false
   },
   "RunOrder": 1
 }
```

Example

Wenn Sie diese Ressourcen mit AWS CloudFormation erstellen, 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 AWS CloudFormation Vorlage für Ihre Amazon S3 S3-Pipeline erstellen. Wenn Sie die zweite Vorlage nicht erstellen, enthält Ihre Pipeline keine Funktionalität für die Änderungserkennung.

YAML

```
Parameters:
  SourceObjectKey:
    Description: 'S3 source artifact'
    Type: String
    Default: SampleApp_Linux.zip
  ApplicationName:
    Description: 'CodeDeploy application name'
    Type: String
    Default: DemoApplication
  BetaFleet:
    Description: 'Fleet configured in CodeDeploy'
    Type: String
    Default: DemoFleet

Resources:
  SourceBucket:
    Type: AWS::S3::Bucket
    Properties:
      NotificationConfiguration:
        EventBridgeConfiguration:
          EventBridgeEnabled: true
      VersioningConfiguration:
        Status: Enabled
  CodePipelineArtifactStoreBucket:
    Type: AWS::S3::Bucket
  CodePipelineArtifactStoreBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
```

```
      Bucket: !Ref CodePipelineArtifactStoreBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Sid: DenyUnEncryptedObjectUploads
            Effect: Deny
            Principal: '*'
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*' ] ]
            Condition:
              StringNotEquals:
                s3:x-amz-server-side-encryption: aws:kms
          -
            Sid: DenyInsecureConnections
            Effect: Deny
            Principal: '*'
            Action: s3:*
            Resource: !Sub ${CodePipelineArtifactStoreBucket.Arn}/*
            Condition:
              Bool:
                aws:SecureTransport: false
  CodePipelineServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - codepipeline.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: AWS-CodePipeline-Service-3
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
```

```
            Action:
              - codecommit:CancelUploadArchive
              - codecommit:GetBranch
              - codecommit:GetCommit
              - codecommit:GetUploadArchiveStatus
              - codecommit:UploadArchive
            Resource: 'resource_ARN'
          -
            Effect: Allow
            Action:
              - codedeploy:CreateDeployment
              - codedeploy:GetApplicationRevision
              - codedeploy:GetDeployment
              - codedeploy:GetDeploymentConfig
              - codedeploy:RegisterApplicationRevision
            Resource: 'resource_ARN'
          -
            Effect: Allow
            Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
            Resource: 'resource_ARN'
          -
            Effect: Allow
            Action:
              - devicefarm:ListProjects
              - devicefarm:ListDevicePools
              - devicefarm:GetRun
              - devicefarm:GetUpload
              - devicefarm:CreateUpload
              - devicefarm:ScheduleRun
            Resource: 'resource_ARN'
          -
            Effect: Allow
            Action:
              - lambda:InvokeFunction
              - lambda:ListFunctions
            Resource: 'resource_ARN'
          -
            Effect: Allow
            Action:
              - iam:PassRole
            Resource: 'resource_ARN'
          -
```

```
              Effect: Allow
              Action:
                - elasticbeanstalk:*
                - ec2:*
                - elasticloadbalancing:*
                - autoscaling:*
                - cloudwatch:*
                - s3:*
                - sns:*
                - cloudformation:*
                - rds:*
                - sqs:*
                - ecs:*
              Resource: 'resource_ARN'
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      Name: s3-events-pipeline
      RoleArn:
        !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref SourceObjectKey
                PollForSourceChanges: false
              RunOrder: 1
        -
          Name: Beta
          Actions:
            -
              Name: BetaAction
              InputArtifacts:
```

```
              - Name: SourceOutput
            ActionTypeId:
              Category: Deploy
              Owner: AWS
              Version: 1
              Provider: CodeDeploy
            Configuration:
              ApplicationName: !Ref ApplicationName
              DeploymentGroupName: !Ref BetaFleet
            RunOrder: 1
    ArtifactStore:
      Type: S3
      Location: !Ref CodePipelineArtifactStoreBucket
 EventRole:
   Type: AWS::IAM::Role
   Properties:
     AssumeRolePolicyDocument:
       Version: 2012-10-17
       Statement:
         -
           Effect: Allow
           Principal:
             Service:
               - events.amazonaws.com
           Action: sts:AssumeRole
     Path: /
     Policies:
       -
         PolicyName: eb-pipeline-execution
         PolicyDocument:
           Version: 2012-10-17
           Statement:
             -
               Effect: Allow
               Action: codepipeline:StartPipelineExecution
               Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
 EventRule:
   Type: AWS::Events::Rule
   Properties:
     EventBusName: default
     EventPattern:
       source:
         - aws.s3
```

```
        detail-type:
          - Object Created
        detail:
          bucket:
            name:
              - !Ref SourceBucket
      Name: EnabledS3SourceRule
      State: ENABLED
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
 'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
```

JSON

```
{
    "Parameters": {
        "SourceObjectKey": {
            "Description": "S3 source artifact",
            "Type": "String",
            "Default": "SampleApp_Linux.zip"
        },
        "ApplicationName": {
            "Description": "CodeDeploy application name",
            "Type": "String",
            "Default": "DemoApplication"
        },
        "BetaFleet": {
            "Description": "Fleet configured in CodeDeploy",
            "Type": "String",
            "Default": "DemoFleet"
        }
    },
    "Resources": {
        "SourceBucket": {
            "Type": "AWS::S3::Bucket",
              "Properties": {
                "NotificationConfiguration": {
                    "EventBridgeConfiguration": {
```

```
                    "EventBridgeEnabled": true
                }
            },
            "VersioningConfiguration": {
                "Status": "Enabled"
            }
        }
    },
    "CodePipelineArtifactStoreBucket": {
        "Type": "AWS::S3::Bucket"
    },
    "CodePipelineArtifactStoreBucketPolicy": {
        "Type": "AWS::S3::BucketPolicy",
        "Properties": {
            "Bucket": {
                "Ref": "CodePipelineArtifactStoreBucket"
            },
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "DenyUnEncryptedObjectUploads",
                        "Effect": "Deny",
                        "Principal": "*",
                        "Action": "s3:PutObject",
                        "Resource": {
                            "Fn::Join": [
                                "",
                                [
                                    {
                                        "Fn::GetAtt": [
                                            "CodePipelineArtifactStoreBucket",
                                            "Arn"
                                        ]
                                    },
                                    "/*"
                                ]
                            ]
                        },
                        "Condition": {
                            "StringNotEquals": {
                                "s3:x-amz-server-side-encryption": "aws:kms"
                            }
                        }
```

```
                    },
                    {
                        "Sid": "DenyInsecureConnections",
                        "Effect": "Deny",
                        "Principal": "*",
                        "Action": "s3:*",
                        "Resource": {
                            "Fn::Join": [
                                "",
                                [
                                    {
                                        "Fn::GetAtt": [
                                            "CodePipelineArtifactStoreBucket",
                                            "Arn"
                                        ]
                                    },
                                    "/*"
                                ]
                            ]
                        },
                        "Condition": {
                            "Bool": {
                                "aws:SecureTransport": false
                            }
                        }
                    }
                ]
            }
        }
    },
    "CodePipelineServiceRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "codepipeline.amazonaws.com"
                            ]
                        },
                        "Action": "sts:AssumeRole"
```

```
                }
            ]
        },
        "Path": "/",
        "Policies": [
            {
                "PolicyName": "AWS-CodePipeline-Service-3",
                "PolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": [
                                "codecommit:CancelUploadArchive",
                                "codecommit:GetBranch",
                                "codecommit:GetCommit",
                                "codecommit:GetUploadArchiveStatus",
                                "codecommit:UploadArchive"
                            ],
                            "Resource": "resource_ARN"
                        },
                        {
                            "Effect": "Allow",
                            "Action": [
                                "codedeploy:CreateDeployment",
                                "codedeploy:GetApplicationRevision",
                                "codedeploy:GetDeployment",
                                "codedeploy:GetDeploymentConfig",
                                "codedeploy:RegisterApplicationRevision"
                            ],
                            "Resource": "resource_ARN"
                        },
                        {
                            "Effect": "Allow",
                            "Action": [
                                "codebuild:BatchGetBuilds",
                                "codebuild:StartBuild"
                            ],
                            "Resource": "resource_ARN"
                        },
                        {
                            "Effect": "Allow",
                            "Action": [
                                "devicefarm:ListProjects",
```

```
                "devicefarm:ListDevicePools",
                "devicefarm:GetRun",
                "devicefarm:GetUpload",
                "devicefarm:CreateUpload",
                "devicefarm:ScheduleRun"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:ListFunctions"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticbeanstalk:*",
                "ec2:*",
                "elasticloadbalancing:*",
                "autoscaling:*",
                "cloudwatch:*",
                "s3:*",
                "sns:*",
                "cloudformation:*",
                "rds:*",
                "sqs:*",
                "ecs:*"
            ],
            "Resource": "resource_ARN"
        }
    ]
}
}
]
```

```
            }
        },
        "AppPipeline": {
            "Type": "AWS::CodePipeline::Pipeline",
            "Properties": {
                "Name": "s3-events-pipeline",
                "RoleArn": {
                    "Fn::GetAtt": [
                        "CodePipelineServiceRole",
                        "Arn"
                    ]
                },
                "Stages": [
                    {
                        "Name": "Source",
                        "Actions": [
                            {
                                "Name": "SourceAction",
                                "ActionTypeId": {
                                    "Category": "Source",
                                    "Owner": "AWS",
                                    "Version": 1,
                                    "Provider": "S3"
                                },
                                "OutputArtifacts": [
                                    {
                                        "Name": "SourceOutput"
                                    }
                                ],
                                "Configuration": {
                                    "S3Bucket": {
                                        "Ref": "SourceBucket"
                                    },
                                    "S3ObjectKey": {
                                        "Ref": "SourceObjectKey"
                                    },
                                    "PollForSourceChanges": false
                                },
                                "RunOrder": 1
                            }
                        ]
                    },
                    {
                        "Name": "Beta",
```

```
                    "Actions": [
                        {
                            "Name": "BetaAction",
                            "InputArtifacts": [
                                {
                                    "Name": "SourceOutput"
                                }
                            ],
                            "ActionTypeId": {
                                "Category": "Deploy",
                                "Owner": "AWS",
                                "Version": 1,
                                "Provider": "CodeDeploy"
                            },
                            "Configuration": {
                                "ApplicationName": {
                                    "Ref": "ApplicationName"
                                },
                                "DeploymentGroupName": {
                                    "Ref": "BetaFleet"
                                }
                            },
                            "RunOrder": 1
                        }
                    ]
                }
            ],
            "ArtifactStore": {
                "Type": "S3",
                "Location": {
                    "Ref": "CodePipelineArtifactStoreBucket"
                }
            }
        }
    },
    "EventRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
```

```
                            "Service": [
                                "events.amazonaws.com"
                            ]
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "eb-pipeline-execution",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "codepipeline:StartPipelineExecution",
                                "Resource": {
                                    "Fn::Join": [
                                        "",
                                        [
                                            "arn:aws:codepipeline:",
                                            {
                                                "Ref": "AWS::Region"
                                            },
                                            ":",
                                            {
                                                "Ref": "AWS::AccountId"
                                            },
                                            ":",
                                            {
                                                "Ref": "AppPipeline"
                                            }
                                        ]
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
        }
    },
```

```
        "EventRule": {
            "Type": "AWS::Events::Rule",

            "Properties": {
                "EventBusName": "default",
          "EventPattern": {
              "source": [
            "aws.s3"
          ],
                "detail-type": [
        "Object Created"
          ],
          "detail": {
        "bucket": {
                        "name": [
                            {
                            "Ref": "SourceBucket"
                            }
                        ]
                    }
                }
    },
    "Name": "EnabledS3SourceRule",
          "State": "ENABLED",
                "Targets": [
                    {
                        "Arn": {
                            "Fn::Join": [
                                "",
                                [
                                    "arn:aws:codepipeline:",
                                    {
                                        "Ref": "AWS::Region"
                                    },
                                    ":",
                                    {
                                        "Ref": "AWS::AccountId"
                                    },
                                    ":",
                                    {
                                        "Ref": "AppPipeline"
                                    }
                                ]
                            ]
```

```
                        },
                        "RoleArn": {
                            "Fn::GetAtt": [
                                "EventRole",
                                "Arn"
                            ]
                        },
                        "Id": "codepipeline-AppPipeline"
                    }
                ]
            }
        }
    }
    }
}
```

Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem S3-Trail CloudTrail

Bei einer Pipeline mit einer Amazon S3 S3-Quelle ändern Sie die Pipeline so, dass die Änderungserkennung automatisiert wird EventBridge. Wählen Sie aus den folgenden Methoden, um die Migration zu implementieren:

- Konsole: Migrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole)
- CLI: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (CLI)
- AWS CloudFormation: Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (Vorlage) AWS CloudFormation

Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (CLI)

Gehen Sie wie folgt vor, um eine Pipeline zu bearbeiten, die Polling (regelmäßige Prüfungen) verwendet, um stattdessen ein Ereignis in zu verwenden. EventBridge Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen einer Pipeline in CodePipeline.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und erstellen dann die folgenden Ressourcen:

- AWS CloudTrailTrail-, 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

So erstellen Sie einen AWS CloudTrail-Trail und aktivieren die Protokollierung

Wenn Sie die AWS CLI verwenden, um einen Trail zu erstellen, rufen Sie den Befehl create-trail auf und geben Sie Folgendes an:

- Den Trail-Namen.
- Der Bucket, auf den Sie bereits die Bucket-Richtlinie für AWS CloudTrail angewendet haben.

Weitere Informationen finden Sie unter Erstellen eines Trails mit der AWS Befehlszeilenschnittstelle.

1. Rufen Sie den Befehl create-trail auf und beziehen Sie die Parameter `--name` und `--s3-bucket-name` ein.

   Warum nehme ich diese Änderung vor? Dadurch wird der für Ihren S3-Quell-Bucket erforderliche CloudTrail-Trail erstellt.

   Der folgende Befehl verwendet `--name` und `--s3-bucket-name` zum Erstellen eines Trails mit dem Namen `my-trail` und eines Buckets mit dem Namen `myBucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name myBucket
   ```

2. Rufen Sie den Befehl start-logging auf und beziehen Sie den `--name`-Parameter ein.

   Warum nehme ich diese Änderung vor? Dieser Befehl startet die CloudTrail Protokollierung für Ihren Quell-Bucket und sendet Ereignisse an EventBridge.

   Beispiel:

   Im folgenden Befehl wird `--name` verwendet, um die Protokollierung auf einem Trail mit der Bezeichnung `my-trail` zu starten.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Rufen Sie den Befehl put-event-selectors auf und beziehen Sie die Parameter `--trail-name` und `--event-selectors` ein. Verwenden Sie Event-Selektoren, um anzugeben, dass Ihr Trail

Datenereignisse für Ihren Quell-Bucket protokollieren und die Ereignisse an die EventBridge Regel senden soll.

Warum nehme ich diese Änderung vor? Dieser Befehl filtert Ereignisse.

Beispiel:

Im folgenden Beispielbefehl werden `--trail-name` und `--event-selectors` verwendet, um die Verwaltung von Datenereignissen für einen Quell-Bucket und einen Präfix namens `myBucket/myFolder` anzugeben.

```
aws cloudtrail put-event-selectors --trail-name my-trail --event-selectors
 '[{ "ReadWriteType": "WriteOnly", "IncludeManagementEvents":false,
 "DataResources": [{ "Type": "AWS::S3::Object", "Values": ["arn:aws:s3:::myBucket/
myFolder/file.zip"] }] }]'
```

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Erteilen Sie Berechtigungen EventBridge , die CodePipeline zum Aufrufen der Regel verwendet werden können. Weitere Informationen finden Sie unter Verwenden ressourcenbasierter Richtlinien für Amazon. EventBridge

   a. Verwenden Sie das folgende Beispiel, um die Vertrauensrichtlinie zu erstellen, damit EventBridge Sie die Servicerolle übernehmen können. Geben Sie ihr den Namen `trustpolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Verwenden Sie den folgenden Befehl, um die `Role-for-MyRule`-Rolle zu erstellen und die Vertrauensrichtlinie anzufügen.

Warum nehme ich diese Änderung vor? Durch das Hinzufügen dieser Vertrauensrichtlinie zur Rolle werden Berechtigungen für erstellt EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Erstellen Sie die JSON-Datei der Berechtigungsrichtlinie wie hier für die Pipeline mit dem Namen `MyFirstPipeline` gezeigt. Geben Sie der Berechtigungsrichtlinie den Namen `permissionspolicyforEB.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Führen Sie den folgenden Befehl aus, um der erstellten `Role-for-MyRule`-Rolle die neue `CodePipeline-Permissions-Policy-for-EB`-Berechtigungsrichtlinie anzufügen.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Rufen Sie den Befehl „put-rule“ auf und beziehen Sie die Parameter „`--name`“, „`--event-pattern`“ und „`--role-arn`“ ein.

Mit dem folgenden Beispielbefehl wird eine Regel mit dem Namen „`MyS3SourceRule`“ erstellt.

```
aws events put-rule --name "MyS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"AWS API Call via CloudTrail\"],\"detail\":
```

```
{\"eventSource\":[\"s3.amazonaws.com\"],\"eventName\":[\"CopyObject\",\"PutObject
\",\"CompleteMultipartUpload\"],\"requestParameters\":{\"bucketName\":[\"my-bucket
\"],\"key\":[\"my-key\"]}}}
 --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Um das Objekt CodePipeline als Ziel hinzuzufügen, rufen Sie den put-targets Befehl auf und geben Sie die `--targets` Parameter `--rule` und an.

   Der folgende Befehl legt fest, dass für die Regel mit dem Namen `MyS3SourceRule` die Ziel-`Id` aus der Nummer 1 besteht. Dies bedeutet, dass in einer Liste mit Zielen für die Regel dies Ziel 1 ist. Der Befehl gibt zudem ein Beispiel ARN für die Pipeline an. Die Pipeline startet, wenn Änderungen im Repository auftreten.

   ```
   aws events put-targets --rule MyS3SourceRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.
2. Öffnen Sie die JSON-Datei in einem Texteditor und bearbeiten Sie die Quellphase, indem Sie den Parameter `PollForSourceChanges` für einen Bucket mit dem Namen `storage-bucketin` `false` ändern wie in diesem Beispiel gezeigt.

Warum nehme ich diese Änderung vor? Durch Festlegen dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

```
"configuration": {
    "S3Bucket": "storage-bucket",
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

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

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

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

   Speichern Sie die Datei.

4. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

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

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

   Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den start-pipeline-execution-Befehl, um Ihre Pipeline manuell zu starten.

Migrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (Vorlage) AWS CloudFormation

Gehen Sie wie folgt vor, um Ihre Pipeline mit einer Amazon S3 S3-Quelle von der Abfrage bis zur ereignisbasierten Änderungserkennung zu bearbeiten.

Um eine ereignisgesteuerte Pipeline mit Amazon S3 zu erstellen, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und fügen dann Ihrer Vorlage die folgenden Ressourcen hinzu:

- EventBridge erfordert, dass alle Amazon S3 S3-Ereignisse protokolliert werden müssen. Sie müssen eine AWS CloudTrail Trail-, Bucket- und Bucket-Richtlinie erstellen, die Amazon S3 verwenden kann, um die auftretenden Ereignisse zu protokollieren. Weitere Informationen finden Sie unter Datenereignisse für Trails protokollieren und Managementereignisse für Trails protokollieren.
- EventBridge Regel und IAM-Rolle, damit dieses Ereignis unsere Pipeline starten kann.

Wenn Sie AWS CloudFormation zum Erstellen und Verwalten Ihrer Pipelines verwenden, sind in Ihrer Vorlage Inhalte wie die folgenden enthalten.

> **Note**
>
> Die `Configuration`-Eigenschaft in der Quellstufe mit dem Namen `PollForSourceChanges`. Wenn diese Eigenschaft in Ihrer Vorlage nicht enthalten ist, dann wird `PollForSourceChanges` standardmäßig auf `true` festgelegt.

YAML

```
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn: !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                -
                  Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref S3SourceObjectKey
                PollForSourceChanges: true
              RunOrder: 1


...
```

JSON

```
       "AppPipeline": {
           "Type": "AWS::CodePipeline::Pipeline",
           "Properties": {
               "RoleArn": {
                   "Fn::GetAtt": ["CodePipelineServiceRole", "Arn"]
               },
               "Stages": [
                   {
                       "Name": "Source",
                       "Actions": [
                           {
                               "Name": "SourceAction",
```

```
                            "ActionTypeId": {
                                "Category": "Source",
                                "Owner": "AWS",
                                "Version": 1,
                                "Provider": "S3"
                            },
                            "OutputArtifacts": [
                                {
                                    "Name": "SourceOutput"
                                }
                            ],
                            "Configuration": {
                                "S3Bucket": {
                                    "Ref": "SourceBucket"
                                },
                                "S3ObjectKey": {
                                    "Ref": "SourceObjectKey"
                                },
                                "PollForSourceChanges": true
                            },
                            "RunOrder": 1
                        }
                    ]
                },


...
```

Um eine EventBridge Regel mit Amazon S3 als Ereignisquelle und CodePipeline als Ziel zu erstellen und die Berechtigungsrichtlinie anzuwenden

1. Verwenden Sie die AWS CloudFormation-Ressource `AWS::IAM::Role` unter `Resources` in der Vorlage, um die IAM-Rolle zu konfigurieren, die Ihrem Ereignis das Starten Ihrer Pipeline erlaubt. 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 AWS CloudFormation ermöglicht das Erstellen von Berechtigungen für EventBridge. Diese Ressource wird Ihrem AWS CloudFormation-Stack hinzugefügt.

YAML

```
  EventRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
 'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]


...
```

JSON

```
  "EventRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
```

```
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": [
                "events.amazonaws.com"
              ]
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "Path": "/",
      "Policies": [
        {
          "PolicyName": "eb-pipeline-execution",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Action": "codepipeline:StartPipelineExecution",
                "Resource": {
                  "Fn::Join": [
                    "",
                    [
                      "arn:aws:codepipeline:",
                      {
                        "Ref": "AWS::Region"
                      },
                      ":",
                      {
                        "Ref": "AWS::AccountId"
                      },
                      ":",
                      {
                        "Ref": "AppPipeline"
                      }
                    ]
                  ]

...
```

2. Verwenden Sie die `AWS::Events::Rule` AWS CloudFormation Ressource, um eine EventBridge Regel hinzuzufügen. Dieses Ereignismuster erzeugt ein Ereignis`CopyObject`, `PutObject` das Ihren `CompleteMultipartUpload` Amazon S3 S3-Quell-Bucket überwacht. Fügen Sie darüber hinaus ein Ziel für Ihre Pipeline ein. Wenn `CopyObject`, `PutObject` oder `CompleteMultipartUpload` auftritt, ruft diese Rolle `StartPipelineExecution` in Ihrer Ziel-Pipeline auf.

   Warum nehme ich diese Änderung vor? Durch das Hinzufügen der `AWS::Events::Rule`-Ressource kann AWS CloudFormation das Ereignis erstellen. Diese Ressource wird Ihrem AWS CloudFormation-Stack hinzugefügt.

   YAML

```
EventRule:
  Type: AWS::Events::Rule
  Properties:
    EventPattern:
      source:
        - aws.s3
      detail-type:
        - 'AWS API Call via CloudTrail'
      detail:
        eventSource:
          - s3.amazonaws.com
        eventName:
          - CopyObject
          - PutObject
          - CompleteMultipartUpload
        requestParameters:
          bucketName:
            - !Ref SourceBucket
          key:
            - !Ref SourceObjectKey
    Targets:
      -
        Arn:
          !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
        RoleArn: !GetAtt EventRole.Arn
        Id: codepipeline-AppPipeline
```

```
...
```

JSON

```
"EventRule": {
  "Type": "AWS::Events::Rule",
  "Properties": {
    "EventPattern": {
      "source": [
        "aws.s3"
      ],
      "detail-type": [
        "AWS API Call via CloudTrail"
      ],
      "detail": {
        "eventSource": [
          "s3.amazonaws.com"
        ],
        "eventName": [
          "CopyObject",
          "PutObject",
          "CompleteMultipartUpload"
        ],
        "requestParameters": {
          "bucketName": [
            {
              "Ref": "SourceBucket"
            }
          ],
          "key": [
            {
              "Ref": "SourceObjectKey"
            }
          ]
        }
      }
    },
    "Targets": [
      {
        "Arn": {
          "Fn::Join": [
            "",
```

```
              [
                "arn:aws:codepipeline:",
                {
                  "Ref": "AWS::Region"
                },
                ":",
                {
                  "Ref": "AWS::AccountId"
                },
                ":",
                {
                  "Ref": "AppPipeline"
                }
              ]
            ]
          },
          "RoleArn": {
            "Fn::GetAtt": [
              "EventRole",
              "Arn"
            ]
          },
          "Id": "codepipeline-AppPipeline"
        }
      ]
    }
  }
},

...
```

3. Fügen Sie diesen Ausschnitt zu Ihrer ersten Vorlage hinzu, um Stack-übergreifende Funktionalität zu ermöglichen:

   YAML

```
Outputs:
  SourceBucketARN:
    Description: "S3 bucket ARN that Cloudtrail will use"
    Value: !GetAtt SourceBucket.Arn
    Export:
      Name: SourceBucketARN
```

JSON

```
  "Outputs" : {
    "SourceBucketARN" : {
      "Description" : "S3 bucket ARN that Cloudtrail will use",
      "Value" : { "Fn::GetAtt": ["SourceBucket", "Arn"] },
      "Export" : {
        "Name" : "SourceBucketARN"
      }
    }

...
```

4. Speichern Sie Ihre aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation-Konsole.
5. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
6. Laden Sie Ihre aktualisierte Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
7. Wählen Sie Execute (Ausführen).

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf `false` fest.

Warum nehme ich diese Änderung vor? Durch Ändern von `PollForSourceChanges` in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
```

JSON

```
{
   "Name": "SourceAction",
   "ActionTypeId": {
     "Category": "Source",
     "Owner": "AWS",
     "Version": 1,
     "Provider": "S3"
   },
   "OutputArtifacts": [
     {
       "Name": "SourceOutput"
     }
   ],
   "Configuration": {
     "S3Bucket": {
       "Ref": "SourceBucket"
     },
```

```
      "S3ObjectKey": {
        "Ref": "SourceObjectKey"
      },
      "PollForSourceChanges": false
    },
    "RunOrder": 1
  }
```

Um eine zweite Vorlage für die CloudTrail Ressourcen Ihrer Amazon S3 S3-Pipeline zu erstellen

- Verwenden Sie in einer separaten Vorlage unter `Resources` die AWS CloudFormation-Ressourcen `AWS::S3::Bucket`, `AWS::S3::BucketPolicy` und `AWS::CloudTrail::Trail`, um eine einfache Bucket-Definition und einen Trail für CloudTrail bereitzustellen.

  Warum nehme ich diese Änderung vor? Angesichts des aktuellen Limits von fünf Trails pro Konto muss der CloudTrail Trail separat erstellt und verwaltet werden. (Siehe Grenzwerte unter AWS CloudTrail.) Sie können jedoch viele Amazon S3 S3-Buckets in einen einzigen Trail aufnehmen, sodass Sie den Trail einmal erstellen und dann bei Bedarf Amazon S3 S3-Buckets für andere Pipelines hinzufügen können. Fügen Sie den folgenden Code in Ihre zweite Beispielvorlagendatei ein.

  YAML

```
################################################################################
# Prerequisites:
#   - S3 SourceBucket and SourceObjectKey must exist
################################################################################

Parameters:
  SourceObjectKey:
    Description: 'S3 source artifact'
    Type: String
    Default: SampleApp_Linux.zip

Resources:
  AWSCloudTrailBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref AWSCloudTrailBucket
```

```
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Sid: AWSCloudTrailAclCheck
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:GetBucketAcl
            Resource: !GetAtt AWSCloudTrailBucket.Arn
          -
            Sid: AWSCloudTrailWrite
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt AWSCloudTrailBucket.Arn, '/
AWSLogs/', !Ref 'AWS::AccountId', '/*' ] ]
            Condition:
              StringEquals:
                s3:x-amz-acl: bucket-owner-full-control
  AWSCloudTrailBucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
  AwsCloudTrail:
    DependsOn:
      - AWSCloudTrailBucketPolicy
    Type: AWS::CloudTrail::Trail
    Properties:
      S3BucketName: !Ref AWSCloudTrailBucket
      EventSelectors:
        -
          DataResources:
            -
              Type: AWS::S3::Object
              Values:
                - !Join [ '', [ !ImportValue SourceBucketARN, '/', !Ref
 SourceObjectKey ] ]
          ReadWriteType: WriteOnly
          IncludeManagementEvents: false
      IncludeGlobalServiceEvents: true
      IsLogging: true
```

```
      IsMultiRegionTrail: true


...
```

JSON

```
{
  "Parameters": {
    "SourceObjectKey": {
      "Description": "S3 source artifact",
      "Type": "String",
      "Default": "SampleApp_Linux.zip"
    }
  },
  "Resources": {
    "AWSCloudTrailBucket": {
      "Type": "AWS::S3::Bucket",
        "DeletionPolicy": "Retain"
    },
    "AWSCloudTrailBucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {
          "Ref": "AWSCloudTrailBucket"
        },
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "AWSCloudTrailAclCheck",
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "cloudtrail.amazonaws.com"
                ]
              },
              "Action": "s3:GetBucketAcl",
              "Resource": {
                "Fn::GetAtt": [
                  "AWSCloudTrailBucket",
                  "Arn"
                ]
```

```
            }
          },
          {
            "Sid": "AWSCloudTrailWrite",
            "Effect": "Allow",
            "Principal": {
              "Service": [
                "cloudtrail.amazonaws.com"
              ]
            },
            "Action": "s3:PutObject",
            "Resource": {
              "Fn::Join": [
                "",
                [
                  {
                    "Fn::GetAtt": [
                      "AWSCloudTrailBucket",
                      "Arn"
                    ]
                  },
                  "/AWSLogs/",
                  {
                    "Ref": "AWS::AccountId"
                  },
                  "/*"
                ]
              ]
            },
            "Condition": {
              "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control"
              }
            }
          }
        ]
      }
    }
  },
  "AwsCloudTrail": {
    "DependsOn": [
      "AWSCloudTrailBucketPolicy"
    ],
    "Type": "AWS::CloudTrail::Trail",
```

```
      "Properties": {
        "S3BucketName": {
          "Ref": "AWSCloudTrailBucket"
        },
        "EventSelectors": [
          {
            "DataResources": [
              {
                "Type": "AWS::S3::Object",
                "Values": [
                  {
                    "Fn::Join": [
                      "",
                      [
                        {
                          "Fn::ImportValue": "SourceBucketARN"
                        },
                        "/",
                        {
                          "Ref": "SourceObjectKey"
                        }
                      ]
                    ]
                  }
                ]
              }
            ],
            "ReadWriteType": "WriteOnly",
            "IncludeManagementEvents": false
          }
        ],
        "IncludeGlobalServiceEvents": true,
        "IsLogging": true,
        "IsMultiRegionTrail": true
      }
    }
  }
}

...
```

Example

Wenn Sie diese Ressourcen mit AWS CloudFormation erstellen, 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 AWS CloudFormation Vorlage für Ihre Amazon S3 S3-Pipeline erstellen. Wenn Sie die zweite Vorlage nicht erstellen, enthält Ihre Pipeline keine Funktionalität für die Änderungserkennung.

YAML

```
Resources:
  SourceBucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
  CodePipelineArtifactStoreBucket:
    Type: AWS::S3::Bucket
  CodePipelineArtifactStoreBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref CodePipelineArtifactStoreBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Sid: DenyUnEncryptedObjectUploads
            Effect: Deny
            Principal: '*'
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*' ] ]
            Condition:
              StringNotEquals:
                s3:x-amz-server-side-encryption: aws:kms
          -
            Sid: DenyInsecureConnections
            Effect: Deny
```

```
            Principal: '*'
            Action: s3:*
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*' ] ]
            Condition:
              Bool:
                aws:SecureTransport: false
  CodePipelineServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - codepipeline.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: AWS-CodePipeline-Service-3
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action:
                  - codecommit:CancelUploadArchive
                  - codecommit:GetBranch
                  - codecommit:GetCommit
                  - codecommit:GetUploadArchiveStatus
                  - codecommit:UploadArchive
                Resource: 'resource_ARN'
              -
                Effect: Allow
                Action:
                  - codedeploy:CreateDeployment
                  - codedeploy:GetApplicationRevision
                  - codedeploy:GetDeployment
                  - codedeploy:GetDeploymentConfig
                  - codedeploy:RegisterApplicationRevision
                Resource: 'resource_ARN'
```

```
            -
              Effect: Allow
              Action:
                - codebuild:BatchGetBuilds
                - codebuild:StartBuild
              Resource: 'resource_ARN'
            -
              Effect: Allow
              Action:
                - devicefarm:ListProjects
                - devicefarm:ListDevicePools
                - devicefarm:GetRun
                - devicefarm:GetUpload
                - devicefarm:CreateUpload
                - devicefarm:ScheduleRun
              Resource: 'resource_ARN'
            -
              Effect: Allow
              Action:
                - lambda:InvokeFunction
                - lambda:ListFunctions
              Resource: 'resource_ARN'
            -
              Effect: Allow
              Action:
                - iam:PassRole
              Resource: 'resource_ARN'
            -
              Effect: Allow
              Action:
                - elasticbeanstalk:*
                - ec2:*
                - elasticloadbalancing:*
                - autoscaling:*
                - cloudwatch:*
                - s3:*
                - sns:*
                - cloudformation:*
                - rds:*
                - sqs:*
                - ecs:*
              Resource: 'resource_ARN'
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
```

```
    Properties:
      Name: s3-events-pipeline
      RoleArn:
        !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref SourceObjectKey
                PollForSourceChanges: false
              RunOrder: 1
        -
          Name: Beta
          Actions:
            -
              Name: BetaAction
              InputArtifacts:
                - Name: SourceOutput
              ActionTypeId:
                Category: Deploy
                Owner: AWS
                Version: 1
                Provider: CodeDeploy
              Configuration:
                ApplicationName: !Ref ApplicationName
                DeploymentGroupName: !Ref BetaFleet
              RunOrder: 1
      ArtifactStore:
        Type: S3
        Location: !Ref CodePipelineArtifactStoreBucket
  EventRole:
    Type: AWS::IAM::Role
    Properties:
```

```
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
 EventRule:
    Type: AWS::Events::Rule
    Properties:
      EventPattern:
        source:
          - aws.s3
        detail-type:
          - 'AWS API Call via CloudTrail'
        detail:
          eventSource:
            - s3.amazonaws.com
          eventName:
            - PutObject
            - CompleteMultipartUpload
          resources:
            ARN:
              - !Join [ '', [ !GetAtt SourceBucket.Arn, '/', !Ref
SourceObjectKey ] ]
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
```

```
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline

Outputs:
  SourceBucketARN:
    Description: "S3 bucket ARN that Cloudtrail will use"
    Value: !GetAtt SourceBucket.Arn
    Export:
      Name: SourceBucketARN
```

JSON

```
    "Resources": {
        "SourceBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "VersioningConfiguration": {
                    "Status": "Enabled"
                }
            }
        },
        "CodePipelineArtifactStoreBucket": {
            "Type": "AWS::S3::Bucket"
        },
        "CodePipelineArtifactStoreBucketPolicy": {
            "Type": "AWS::S3::BucketPolicy",
            "Properties": {
                "Bucket": {
                    "Ref": "CodePipelineArtifactStoreBucket"
                },
                "PolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Sid": "DenyUnEncryptedObjectUploads",
                            "Effect": "Deny",
                            "Principal": "*",
                            "Action": "s3:PutObject",
                            "Resource": {
                                "Fn::Join": [
                                    "",
                                    [
                                        {
```

```
                                        "Fn::GetAtt": [
                                            "CodePipelineArtifactStoreBucket",
                                            "Arn"
                                        ]
                                    },
                                    "/*"
                                ]
                            ]
                        },
                        "Condition": {
                            "StringNotEquals": {
                                "s3:x-amz-server-side-encryption": "aws:kms"
                            }
                        }
                    },
                    {
                        "Sid": "DenyInsecureConnections",
                        "Effect": "Deny",
                        "Principal": "*",
                        "Action": "s3:*",
                        "Resource": {
                            "Fn::Join": [
                                "",
                                [
                                    {
                                        "Fn::GetAtt": [
                                            "CodePipelineArtifactStoreBucket",
                                            "Arn"
                                        ]
                                    },
                                    "/*"
                                ]
                            ]
                        },
                        "Condition": {
                            "Bool": {
                                "aws:SecureTransport": false
                            }
                        }
                    }
                ]
            }
        }
    },
```

```
        "CodePipelineServiceRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "codepipeline.amazonaws.com"
                                ]
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "Path": "/",
                "Policies": [
                    {
                        "PolicyName": "AWS-CodePipeline-Service-3",
                        "PolicyDocument": {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "codecommit:CancelUploadArchive",
                                        "codecommit:GetBranch",
                                        "codecommit:GetCommit",
                                        "codecommit:GetUploadArchiveStatus",
                                        "codecommit:UploadArchive"
                                    ],
                                    "Resource": "resource_ARN"
                                },
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "codedeploy:CreateDeployment",
                                        "codedeploy:GetApplicationRevision",
                                        "codedeploy:GetDeployment",
                                        "codedeploy:GetDeploymentConfig",
                                        "codedeploy:RegisterApplicationRevision"
                                    ],
```

```
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:BatchGetBuilds",
                "codebuild:StartBuild"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "devicefarm:ListProjects",
                "devicefarm:ListDevicePools",
                "devicefarm:GetRun",
                "devicefarm:GetUpload",
                "devicefarm:CreateUpload",
                "devicefarm:ScheduleRun"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:ListFunctions"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticbeanstalk:*",
                "ec2:*",
                "elasticloadbalancing:*",
                "autoscaling:*",
```

```
                                "cloudwatch:*",
                                "s3:*",
                                "sns:*",
                                "cloudformation:*",
                                "rds:*",
                                "sqs:*",
                                "ecs:*"
                            ],
                            "Resource": "resource_ARN"
                        }
                    ]
                }
            }
        ]
    }
},
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
    "Properties": {
        "Name": "s3-events-pipeline",
        "RoleArn": {
            "Fn::GetAtt": [
                "CodePipelineServiceRole",
                "Arn"
            ]
        },
        "Stages": [
            {
                "Name": "Source",
                "Actions": [
                    {
                        "Name": "SourceAction",
                        "ActionTypeId": {
                            "Category": "Source",
                            "Owner": "AWS",
                            "Version": 1,
                            "Provider": "S3"
                        },
                        "OutputArtifacts": [
                            {
                                "Name": "SourceOutput"
                            }
                        ],
                        "Configuration": {
```

```
                            "S3Bucket": {
                                "Ref": "SourceBucket"
                            },
                            "S3ObjectKey": {
                                "Ref": "SourceObjectKey"
                            },
                            "PollForSourceChanges": false
                        },
                        "RunOrder": 1
                    }
                ]
            },
            {
                "Name": "Beta",
                "Actions": [
                    {
                        "Name": "BetaAction",
                        "InputArtifacts": [
                            {
                                "Name": "SourceOutput"
                            }
                        ],
                        "ActionTypeId": {
                            "Category": "Deploy",
                            "Owner": "AWS",
                            "Version": 1,
                            "Provider": "CodeDeploy"
                        },
                        "Configuration": {
                            "ApplicationName": {
                                "Ref": "ApplicationName"
                            },
                            "DeploymentGroupName": {
                                "Ref": "BetaFleet"
                            }
                        },
                        "RunOrder": 1
                    }
                ]
            }
        ],
        "ArtifactStore": {
            "Type": "S3",
            "Location": {
```

```
                    "Ref": "CodePipelineArtifactStoreBucket"
                }
            }
        }
    },
    "EventRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "events.amazonaws.com"
                            ]
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "eb-pipeline-execution",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "codepipeline:StartPipelineExecution",
                                "Resource": {
                                    "Fn::Join": [
                                        "",
                                        [
                                            "arn:aws:codepipeline:",
                                            {
                                                "Ref": "AWS::Region"
                                            },
                                            ":",
                                            {
                                                "Ref": "AWS::AccountId"
                                            },
```

```
                                        ":",
                                        {
                                            "Ref": "AppPipeline"
                                        }
                                    ]
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    }
},
"EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
        "EventPattern": {
            "source": [
                "aws.s3"
            ],
            "detail-type": [
                "AWS API Call via CloudTrail"
            ],
            "detail": {
                "eventSource": [
                    "s3.amazonaws.com"
                ],
                "eventName": [
                    "PutObject",
                    "CompleteMultipartUpload"
                ],
                "resources": {
                    "ARN": [
                        {
                            "Fn::Join": [
                                "",
                                [
                                    {
                                        "Fn::GetAtt": [
                                            "SourceBucket",
                                            "Arn"
                                        ]
                                    },
```

```
                                        "/",
                                        {
                                            "Ref": "SourceObjectKey"
                                        }
                                    ]
                                ]
                            }
                        ]
                    }
                }
            },
            "Targets": [
                {
                    "Arn": {
                        "Fn::Join": [
                            "",
                            [
                                "arn:aws:codepipeline:",
                                {
                                    "Ref": "AWS::Region"
                                },
                                ":",
                                {
                                    "Ref": "AWS::AccountId"
                                },
                                ":",
                                {
                                    "Ref": "AppPipeline"
                                }
                            ]
                        ]
                    },
                    "RoleArn": {
                        "Fn::GetAtt": [
                            "EventRole",
                            "Arn"
                        ]
                    },
                    "Id": "codepipeline-AppPipeline"
                }
            ]
        }
    }
},
```

```
    "Outputs" : {
        "SourceBucketARN" : {
            "Description" : "S3 bucket ARN that Cloudtrail will use",
            "Value" : { "Fn::GetAtt": ["SourceBucket", "Arn"] },
            "Export" : {
                "Name" : "SourceBucketARN"
            }
        }
    }
}


...
```

Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Verbindungen

Sie können eine Quellaktion der GitHub Version 1 migrieren, um Verbindungen für Ihr externes Repository zu verwenden. Dies ist die empfohlene Methode zur Erkennung von Änderungen für Pipelines mit einer Quellaktion der GitHub Version 1.

Für eine Pipeline mit einer Quellaktion der GitHub Version 1 empfehlen wir, die Pipeline so zu ändern, dass eine Aktion der GitHub Version 2 verwendet wird, sodass die Änderungserkennung über AWS CodeStar Verbindungen automatisiert wird. Weitere Informationen zum Arbeiten mit Verbindungen finden Sie unterGitHub Verbindungen.

Stellen Sie eine Verbindung zu GitHub (Konsole) her

Sie können die Konsole verwenden, um eine Verbindung zu herzustellen GitHub.

Schritt 1: Ersetzen Sie Ihre GitHub Aktion aus Version 1

Verwenden Sie die Pipeline-Bearbeitungsseite, um Ihre Aktion aus Version 1 durch eine GitHub Aktion aus Version 2 GitHub zu ersetzen.

Um Ihre GitHub Aktion aus Version 1 zu ersetzen

1. Melden Sie sich bei der CodePipeline Konsole an.

2. Wählen Sie Ihre Pipeline und dann Bearbeiten aus. Wählen Sie in Ihrer Quellstufe die Option Phase bearbeiten aus. Es wird eine Meldung angezeigt, in der empfohlen wird, Ihre Aktion zu aktualisieren.
3. Wählen Sie unter Aktionsanbieter die Option GitHub (Version 2) aus.
4. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Option Connect GitHub. Fahren Sie mit Schritt 2 fort: Verbindung herstellen zu GitHub.
   - Wenn Sie bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Verbindung aus. Fahren Sie mit Schritt 3 fort: Speichern Sie die Quellaktion für Ihre Verbindung.

Schritt 2: Stellen Sie eine Verbindung her zu GitHub

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die GitHub Seite Connect angezeigt.

Um eine Verbindung herzustellen zu GitHub

1. Unter GitHub Verbindungseinstellungen wird Ihr Verbindungsname unter Verbindungsname angezeigt.

   Wählen Sie unter GitHub Apps eine App-Installation aus oder wählen Sie Neue App installieren, um eine zu erstellen.

   > **Note**
   >
   > Sie installieren eine App für alle Verbindungen mit einem bestimmten Anbieter. Wenn Sie die GitHub App bereits installiert haben, wählen Sie sie aus und überspringen Sie diesen Schritt.

2. Wenn die Autorisierungsseite für GitHub angezeigt wird, melden Sie sich mit Ihren Anmeldeinformationen an und wählen Sie dann, ob Sie fortfahren möchten.
3. Auf der App-Installationsseite wird eine Meldung angezeigt, dass die AWS CodeStar App versucht, eine Verbindung zu Ihrem GitHub Konto herzustellen.

> **Note**
>
> Sie installieren die App nur einmal für jedes GitHub Konto. Wenn Sie die App schon einmal installiert haben, können Sie Configure (Konfiguration) wählen und mit einer Änderungsseite für die App-Installation fortfahren. Alternativ kommen Sie über die Schaltfläche „Back“ (Zurück) zur Konsole zurück.

4. Wählen Sie auf der AWS CodeStarInstallationsseite die Option Installieren aus.
5. Auf der GitHub Seite Connect wird die Verbindungs-ID für Ihre neue Installation angezeigt. Wählen Sie Connect (Verbinden) aus.

Schritt 3: Speichern Sie Ihre GitHub Quellaktion

Vervollständigen Sie Ihre Aktualisierungen auf der Seite Aktion bearbeiten, um Ihre neue Quellaktion zu speichern.

Um Ihre GitHub Quellaktion zu speichern

1. Geben Sie unter Repository den Namen Ihres Drittanbieter-Repositorys ein. Geben Sie im Feld Branch den Branch ein, in dem Ihre Pipeline Quelländerungen erkennen soll.

> **Note**
>
> Geben Sie im Feld Repository `owner-name/repository-name` wie in diesem Beispiel gezeigt ein:
>
> ```
> my-account/my-repository
> ```

2. Wählen Sie unter Ausgabeartefaktformat das Format für Ihre Artefakte aus.
   - Um die Ausgabeartefakte der GitHub Aktion mit der Standardmethode zu speichern, wählen Sie CodePipelineStandard. Die Aktion greift auf die Dateien aus dem GitHub Repository zu und speichert die Artefakte in einer ZIP-Datei im Pipeline-Artefaktspeicher.
   - Um eine JSON-Datei zu speichern, die einen URL-Verweis auf das Repository enthält, damit Downstream-Aktionen Git-Befehle direkt ausführen können, wählen Sie Full clone (Vollständiger Klon). Diese Option kann nur von CodeBuild nachgelagerten Aktionen verwendet werden.

Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservice-Rolle aktualisieren, wie unter beschriebenHinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder 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.

3. Unter Ausgabeartefakte können Sie den Namen des Ausgabeartefakts für diese Aktion beibehalten, z. B. `SourceArtifact` Wählen Sie Fertig, um die Aktionsseite Bearbeiten zu schließen.
4. Wählen Sie „Fertig“, um die Seite zur Bearbeitung der Phase zu schließen. Wählen Sie Speichern, um die Seite zur Bearbeitung der Pipeline zu schließen.

Verbindung herstellen zu GitHub (CLI)

Sie können das AWS Command Line Interface (AWS CLI) verwenden, um eine Verbindung zu herzustellen GitHub.

Verwenden Sie dazu den Befehl create-connection.

> **Important**
>
> Eine Verbindung, die über die AWS CLI oder AWS CloudFormation erstellt wird, ist standardmäßig im Status `PENDING` (Ausstehend). Nachdem Sie mit der CLI oder mit AWS CloudFormation eine Verbindung erstellt haben, bearbeiten Sie die Verbindung mit der Konsole, um ihren Status auf `AVAILABLE` zu setzen.

Um eine Verbindung herzustellen zu GitHub

1. Öffnen Sie die Eingabeaufforderung (Windows) oder das Terminal (Linux, macOS oder Unix). Verwenden Sie die AWS CLI, um den Befehl create-connection auszuführen, und geben Sie die Parameter `--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.

Migrieren Sie Polling-Pipelines für eine Quellaktion der GitHub Version 1 zu Webhooks

Sie können Ihre Pipeline migrieren, um mithilfe von Webhooks Änderungen in Ihrem Quell-Repository zu erkennen. GitHub Diese Migration zu Webhooks gilt nur für die Aktion GitHub Version 1.

- Konsole: Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (Konsole)
- CLI: Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (CLI)
- AWS CloudFormation: Aktualisierungspipelines für Push-Ereignisse (Quellaktionen der GitHub Version 1) (AWS CloudFormationVorlage)

Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (Konsole)

Sie können die CodePipeline Konsole verwenden, um Ihre Pipeline so zu aktualisieren, dass Webhooks verwendet werden, um Änderungen in Ihrem Quell-Repository zu erkennen. CodeCommit

Gehen Sie wie folgt vor, um eine Pipeline zu bearbeiten, die stattdessen Polling (regelmäßige Prüfungen) verwendet EventBridge . Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen einer Pipeline in CodePipeline.

Wenn Sie die Konsole verwenden, wird der Parameter `PollForSourceChanges` für Ihre Pipeline für Sie geändert. Der GitHub Webhook wird für Sie erstellt und registriert.

So bearbeiten Sie die Quellphase einer Pipeline

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

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Auf diese Weise wird eine detaillierte Ansicht der Pipeline geöffnet (einschließlich des Status der Aktionen in den einzelnen Stufen der Pipeline).
3. Wählen Sie auf der Pipelinedetails-Seite Edit aus.
4. Wählen Sie in der Stufe Edit stage (Stufe bearbeiten) das Bearbeitungssymbol für die Quellaktion aus.
5. Erweitern Sie die Optionen zur Änderungserkennung und wählen Sie Amazon CloudWatch Events verwenden, um meine Pipeline automatisch zu starten, wenn eine Änderung eintritt (empfohlen).

   Es wird eine Meldung mit dem Hinweis angezeigt, dass ein Webhook CodePipeline erstellt AWS CodePipeline wird GitHub , um Quellenänderungen zu erkennen: erstellt einen Webhook für Sie. Sie können sich in den folgenden Optionen abmelden. Wählen Sie Aktualisieren aus. CodePipeline Erzeugt zusätzlich zum Webhook Folgendes:

   - Ein Geheimnis, das nach dem Zufallsprinzip generiert wurde und zur Autorisierung der Verbindung verwendet wird. GitHub
   - Die Webhook-URL wird generiert, wobei der öffentliche Endpunkt für die Region verwendet wird.

   CodePipeline registriert den Webhook bei. GitHub Damit wird die URL für den Empfang von Repository-Ereignissen abonniert.
6. Wenn Sie die Bearbeitung der Pipeline abgeschlossen haben, wählen Sie Save pipeline changes aus, um zur Übersichtsseite zurückzukehren.

   Es wird eine Meldung mit dem Namen des für Ihre Pipeline zu erstellenden Webhooks angezeigt. Wählen Sie Save and continue aus.
7. Veröffentlichen Sie zum Testen der Aktion eine Änderung mit der AWS CLI, indem Sie einen Commit für eine Änderung der in der Quellstufe der Pipeline festgelegten Quelle durchführen.

Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (CLI)

Führen Sie die folgenden Schritte aus, um eine Pipeline, die Abfragen (periodische Prüfungen) verwendet, zu bearbeiten, damit sie stattdessen einen Webhook verwendet. Wenn Sie eine Pipeline erstellen möchten, informieren Sie sich unter Erstellen einer Pipeline in CodePipeline.

Um eine ereignisgesteuerte Pipeline zu erstellen, bearbeiten Sie den Parameter `PollForSourceChanges` Ihrer Pipeline und erstellen anschließend die folgenden Ressourcen manuell:

- GitHub Webhook und Autorisierungsparameter

So erstellen und registrieren Sie Ihren Webhook

> **Note**
>
> Wenn Sie die CLI oder AWS CloudFormation zum Erstellen einer Pipeline verwenden und einen Webhook hinzufügen, müssen Sie die periodischen Prüfungen deaktivieren. Um die periodischen Prüfungen zu deaktivieren, müssen Sie explizit den `PollForSourceChanges`-Parameter hinzufügen und auf „false“ setzen, wie in der abschließenden Prozedur unten. Andernfalls ist die Standardeinstellung für eine CLI- oder AWS CloudFormation-Pipeline die, dass `PollForSourceChanges` „true“ ist und nicht in der Pipeline-Struktur-Ausgabe angezeigt wird. Weitere Hinweise zu PollForSourceChanges Standardwerten finden Sie unter. Standardeinstellungen für den Parameter PollForSourceChanges

1. Erstellen Sie in einem Texteditor eine JSON-Datei für den Webhook, den Sie erstellen möchten, und speichern Sie sie. Verwenden Sie diese Beispieldatei für einen Webhook mit dem Namen `my-webhook`:

```
{
    "webhook": {
        "name": "my-webhook",
  "targetPipeline": "pipeline_name",
  "targetAction": "source_action_name",
  "filters": [{
      "jsonPath": "$.ref",
      "matchEquals": "refs/heads/{Branch}"
```

```
  }],
  "authentication": "GITHUB_HMAC",
  "authenticationConfiguration": {
      "SecretToken": "secret"
  }
    }
}
```

2. Rufen Sie den Befehl put-webhook auf und beziehen Sie die Parameter `--cli-input` und `--region` ein.

   Der folgende Beispielbefehl erstellt einen Webhook mit der JSON-Datei `webhook_json`.

```
aws codepipeline put-webhook --cli-input-json file://webhook_json.json --region
 "eu-central-1"
```

3. In der in diesem Beispiel gezeigten Ausgabe für einen Webhook mit dem Namen `my-webhook` werden die URL und der ARN zurückgegeben.

```
{
    "webhook": {
        "url": "https://webhooks.domain.com/
trigger111111111EXAMPLE1111111111111111",
        "definition": {
            "authenticationConfiguration": {
                "SecretToken": "secret"
            },
            "name": "my-webhook",
            "authentication": "GITHUB_HMAC",
            "targetPipeline": "pipeline_name",
            "targetAction": "Source",
            "filters": [
                {
                    "jsonPath": "$.ref",
                    "matchEquals": "refs/heads/{Branch}"
                }
            ]
        },
        "arn": "arn:aws:codepipeline:eu-central-1:ACCOUNT_ID:webhook:my-webhook"
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
```

```
    }]
}
```

In diesem Beispiel wird dem Webhook Tagging hinzugefügt, indem dem Webhook der Tag-Schlüssel `Project` und der Wert `ProjectA` hinzugefügt werden. Weitere Informationen zum Markieren von Ressourcen in finden Sie CodePipeline unter. Markieren von Ressourcen

4. Rufen Sie den Befehl register-webhook-with-third-party auf und beziehen Sie den `--webhook-name`-Parameter ein.

   Der folgende Beispielbefehl registriert einen Webhook mit dem Namen `my-webhook`.

   ```
   aws codepipeline register-webhook-with-third-party --webhook-name my-webhook
   ```

So bearbeiten Sie den Parameter Ihrer Pipeline PollForSourceChanges

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

1. Führen Sie den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Für eine Pipeline mit dem Namen `MyFirstPipeline` würden Sie beispielsweise den folgenden Befehl verwenden:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.
2. Öffnen Sie die JSON-Datei in einem beliebigen Texteditor und bearbeiten Sie die Quellstufe, indem Sie den Parameter `PollForSourceChanges` ändern oder hinzufügen. In diesem Beispiel für ein Repository mit dem Namen `UserGitHubRepo` ist der Parameter auf `false` festgelegt.

Warum nehme ich diese Änderung vor? Wenn Sie diesen Parameter ändern, werden regelmäßige Überprüfungen deaktiviert, sodass Sie nur die ereignisbasierte Änderungserkennung verwenden können.

```
"configuration": {
    "Owner": "name",
    "Repo": "UserGitHubRepo",
    "PollForSourceChanges": "false",
    "Branch": "main",
    "OAuthToken": "****"
},
```

3. Falls Sie mit der Pipeline-Struktur arbeiten, die mithilfe des get-pipeline-Befehls abgerufen wurde, müssen Sie die Struktur in der JSON-Datei bearbeiten, indem Sie die `metadata`-Zeilen aus der Datei entfernen. Andernfalls kann der update-pipeline-Befehl sie nicht nutzen. Entfernen Sie den Abschnitt `"metadata"` aus der Pipeline-Struktur in der JSON-Datei, einschließlich `{ }` und der Felder `"created"`, `"pipelineARN"` und `"updated"`.

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

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

   Speichern Sie die Datei.

4. Führen Sie den Befehl update-pipeline aus, um die Änderungen zu übernehmen. Geben Sie die Pipeline-JSON-Datei dabei folgendermaßen an:

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

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

Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück.

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den start-pipeline-execution-Befehl, um Ihre Pipeline manuell zu starten.

Aktualisierungspipelines für Push-Ereignisse (Quellaktionen der GitHub Version 1) (AWS CloudFormationVorlage)

Gehen Sie wie folgt vor, um Ihre Pipeline (mit einer GitHub Quelle) von regelmäßigen Überprüfungen (Polling) bis hin zur ereignisbasierten Änderungserkennung mithilfe von Webhooks zu aktualisieren.

Um eine ereignisgesteuerte Pipeline mit zu erstellenAWS CodeCommit, bearbeiten Sie den `PollForSourceChanges` Parameter Ihrer Pipeline und fügen dann Ihrer Vorlage eine GitHub Webhook-Ressource hinzu.

Wenn Sie AWS CloudFormation zum Erstellen und Verwalten Ihrer Pipelines verwenden, sind in Ihrer Vorlage Inhalte wie die folgenden enthalten.

> **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, Ihre Anmeldeinformationen mithilfe von AWS Secrets Manager 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 AWS CloudFormation-Vorlage, um Ihre Anmeldeinformationen zu speichern.

Wenn Sie die CLI oder AWS CloudFormation zum Erstellen einer Pipeline verwenden und einen Webhook hinzufügen, müssen Sie die periodischen Prüfungen deaktivieren.

> **Note**
>
> Um die periodischen Prüfungen zu deaktivieren, müssen Sie explizit den `PollForSourceChanges`-Parameter hinzufügen und auf „false“ setzen, wie in der abschließenden Prozedur unten. Andernfalls ist die Standardeinstellung für eine CLI- oder AWS CloudFormation-Pipeline die, dass `PollForSourceChanges` „true“ ist und nicht in der Pipeline-Struktur-Ausgabe angezeigt wird. Weitere Informationen zu PollForSourceChanges Standardwerten finden Sie unter. Standardeinstellungen für den Parameter PollForSourceChanges

1. Fügen Sie in der Vorlage unter `Resources` Ihre Parameter hinzu:

   YAML

   ```
   Parameters:
           GitHubOwner:
             Type: String

   ...
   ```

   JSON

   ```
   {
    "Parameters": {
   ```

```
  "BranchName": {
   "Description": "GitHub branch name",
   "Type": "String",
   "Default": "main"
  },
  "GitHubOwner": {
   "Type": "String"
  },

...
```

2. Fügen Sie einen Webhook mithilfe der `AWS::CodePipeline::Webhook` AWS CloudFormation-Ressource hinzu.

> **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 Benutzer, dem der zugeordnet ist, die erforderlichen Bereiche festlegen `OAuthToken` kann. GitHub Das Token und der Webhook erfordern die folgenden Bereiche: GitHub

- `repo`: Wird für die vollständige Kontrolle in Bezug auf das Lesen und Einfügen von Artefakten aus öffentlichen und privaten Repositorys in eine Pipeline verwendet.
- `admin:repo_hook`: Wird für die vollständige Kontrolle in Bezug auf Repository-Hooks verwendet.

Andernfalls wird ein GitHub 404-Fehler zurückgegeben. Weitere Informationen zum zurückgegebenen Fehler 404 erhalten Sie unter https://help.github.com/articles/about-webhooks.

YAML

```
AppPipelineWebhook:
  Type: AWS::CodePipeline::Webhook
  Properties:
    Authentication: GITHUB_HMAC
    AuthenticationConfiguration:
```

```
        SecretToken:
 {{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
      Filters:
        -
          JsonPath: "$.ref"
          MatchEquals: refs/heads/{Branch}
      TargetPipeline: !Ref AppPipeline
      TargetAction: SourceAction
      Name: AppPipelineWebhook
      TargetPipelineVersion: !GetAtt AppPipeline.Version
      RegisterWithThirdParty: true


...
```

JSON

```
"AppPipelineWebhook": {
    "Type": "AWS::CodePipeline::Webhook",
    "Properties": {
        "Authentication": "GITHUB_HMAC",
        "AuthenticationConfiguration": {
            "SecretToken":
 "{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}"
        },
        "Filters": [{
            "JsonPath": "$.ref",
            "MatchEquals": "refs/heads/{Branch}"
        }],
        "TargetPipeline": {
            "Ref": "AppPipeline"
        },
        "TargetAction": "SourceAction",
        "Name": "AppPipelineWebhook",
        "TargetPipelineVersion": {
            "Fn::GetAtt": [
              "AppPipeline",
              "Version"
            ]
        },
        "RegisterWithThirdParty": true
    }
},
```

```
...
```

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation-Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Um den PollForSourceChanges Parameter Ihrer Pipeline zu bearbeiten

> **Important**
>
> Wenn Sie eine Pipeline mit dieser Methode erstellen, ist der Parameter `PollForSourceChanges` standardmäßig „true“, wenn er nicht ausdrücklich auf „false“ gesetzt wird. Wenn Sie ereignisbasierte Erkennung hinzufügen, müssen Sie den Parameter Ihrer Ausgabe hinzufügen und Ihn auf „false“ setzen, um die Abfrage zu deaktivieren. Andernfalls wird Ihre Pipeline bei einer einzigen Quelländerung zweimal gestartet. Details hierzu finden Sie unter Standardeinstellungen für den Parameter PollForSourceChanges .

- Ändern Sie in der Vorlage `PollForSourceChanges` in `false`. Wenn Sie `PollForSourceChanges` nicht in Ihre Pipeline-Definition einbezogen haben, fügen Sie das Objekt hinzu und legen es auf "false" fest.

  Warum nehme ich diese Änderung vor? Durch Ändern dieses Parameters in `false` werden periodische Prüfungen deaktiviert. Sie können daher nur die ereignisbasierte Erkennung von Änderungen verwenden.

  YAML

```
        Name: Source
        Actions:
          -
            Name: SourceAction
```

```
              ActionTypeId:
                Category: Source
                Owner: ThirdParty
                Version: 1
                Provider: GitHub
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                Owner: !Ref GitHubOwner
                Repo: !Ref RepositoryName
                Branch: !Ref BranchName
                OAuthToken:
{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
                PollForSourceChanges: false
              RunOrder: 1
```

JSON

```
{
   "Name": "Source",
   "Actions": [{
 "Name": "SourceAction",
 "ActionTypeId": {
     "Category": "Source",
     "Owner": "ThirdParty",
     "Version": 1,
     "Provider": "GitHub"
},
"OutputArtifacts": [{
    "Name": "SourceOutput"
}],
"Configuration": {
    "Owner": {
  "Ref": "GitHubOwner"
    },
    "Repo": {
  "Ref": "RepositoryName"
    },
    "Branch": {
        "Ref": "BranchName"
    },
    "OAuthToken":
"{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}",
```

```
        PollForSourceChanges: false
    },
    "RunOrder": 1
       }]
```

Example

Wenn Sie diese Ressourcen mit erstellenAWS 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

...
```

CodePipeline-Servicerolle erstellen

Wenn Sie eine Pipeline erstellen, erstellen Sie eine Servicerolle oder verwenden eine vorhandene Servicerolle.

Sie können die CodePipeline Konsole oder die verwendenAWS CLI, um eine CodePipeline Servicerolle zu erstellen. Eine Servicerolle ist erforderlich, um eine Pipeline zu erstellen, und die Pipeline ist immer dieser Servicerolle zugeordnet.

Bevor Sie eine Pipeline mit der AWS CLI erstellen, müssen Sie eine CodePipeline Servicerolle für Ihre Pipeline erstellen. Eine AWS CloudFormation Beispielvorlage mit der angegebenen Servicerolle und Richtlinie finden Sie in den Tutorials unterTutorial: Erstellen einer Pipeline, die Variablen aus AWS CloudFormation-Bereitstellungsaktionen verwendet.

Die Servicerolle ist keine AWS verwaltete Rolle, sondern wurde ursprünglich für die Pipelineerstellung erstellt. Wenn dann neue Berechtigungen zur Servicerollenrichtlinie 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 der CodePipeline Servicerolle.

Erstellen der CodePipeline-Servicerolle (Konsole)

Wenn Sie die Konsole verwenden, um eine Pipeline zu erstellen, erstellen Sie die CodePipeline Servicerolle mit dem Assistenten zur Pipelineerstellung.

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

   Wählen Sie Create pipeline (Pipeline erstellen), und führen Sie die Aktionen auf der Seite Step 1: Choose pipeline settings (Schritt 1: Auswahl der Pipelineeinstellungen) im Assistenten zur Pipelineerstellung durch.

   > **Note**
   >
   > Der Name einer erstellten Pipeline kann nicht nachträglich geändert werden. Weitere Informationen zu Einschränkungen finden Sie unter Kontingente in AWS CodePipeline.

2. Wählen Sie unter Servicerolle die Option Neue Servicerolle aus, CodePipeline um eine neue Servicerolle in IAM erstellen zu können.
3. Führen Sie die Pipeline-Erstellung durch. Ihre Pipeline-Servicerolle kann in Ihrer Liste der IAM-Rollen angezeigt werden, und Sie können den einer Pipeline zugeordneten Dienstrollen-ARN einsehen, indem Sie den `get-pipeline` Befehl mit der AWS CLI ausführen.

Erstellen der CodePipeline-Servicerolle (CLI)

Bevor Sie eine Pipeline mit der AWS CLI oder erstellenAWS 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, führen Sie die folgenden Schritte aus, 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, nur Administratorbenutzern das Erstellen einer beliebigen Servicerolle zu erlauben. 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 der Rollenrichtlinie einfügen. Dieses Beispiel verwendet VIM.

```
vim TrustPolicy.json
```

2. Fügen Sie den folgenden JSON-Text in die Datei ein.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codepipeline.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Geben Sie den folgenden VIM-Befehl ein, um die Datei zu speichern und zu beenden:

```
:wq
```

3. Geben Sie in einem Terminalfenster den folgenden Befehl ein, um eine Datei mit dem Namen zu erstellen`RolePolicy.json`, in die Sie die JSON-Datei der Rollenrichtlinie einfügen. Dieses Beispiel verwendet VIM.

```
vim RolePolicy.json
```

4. Fügen Sie den folgenden JSON-Text in die Datei ein. Stellen Sie sicher, dass Sie die Berechtigungen so weit wie möglich einschränken, indem Sie den Amazon-Ressourcennamen (ARN) für Ihre Pipeline in das `Resource` Feld mit der Richtlinienerklärung eingeben.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:CancelUploadArchive",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetUploadArchiveStatus",
        "codecommit:UploadArchive"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetApplicationRevision",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
```

```
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "devicefarm:ListProjects",
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:ListFunctions"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
```

```
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "cloudformation:*",
        "rds:*",
        "sqs:*",
        "ecs:*"
      ],
      "Resource": "resource_ARN"
    }
  ]
}
```

Geben Sie den folgenden VIM-Befehl ein, um die Datei zu speichern und zu beenden:

```
:wq
```

5. Geben Sie den folgenden Befehl ein, um die Rolle zu erstellen und die Richtlinie anzufügen. 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 anzufügen. 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 einzusehen, geben Sie den folgenden Befehl für die benannte Rolle ein`MyRole`:

```
aws iam get-role --role-name MyRole
```

8. Verwenden Sie die Servicerolle ARN, wenn Sie Ihre Pipeline mit der AWS CLI oder AWS CloudFormation erstellen.

Markieren CodePipeline

Tags sind mit AWS-Ressourcen verknüpfte Schlüssel-Wert-Paare. Sie können Tags auf Ihre Pipelines in anwenden. Weitere Informationen zum CodePipeline-Ressourcen-Tagging, zu Anwendungsfällen, Einschränkungen für Tag-Schlüssel und -Werte sowie unterstützte Ressourcentypen finden Sie unterMarkieren von Ressourcenaus.

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 beim anAWS Management Consoleund öffnen Sie die CodePipeline-Konsole unterhttp://console.aws.amazon.com/codesuite/codepipeline/homeaus.
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 beim anAWS Management Consoleund öffnen Sie die CodePipeline-Konsole unterhttp://console.aws.amazon.com/codesuite/codepipeline/homeaus.
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 beim anAWS Management Consoleund öffnen Sie die CodePipeline-Konsole unterhttp://console.aws.amazon.com/codesuite/codepipeline/homeaus.
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 Submit (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 beim anAWS Management Consoleund öffnen Sie die CodePipeline-Konsole unterhttp://console.aws.amazon.com/codesuite/codepipeline/homeaus.
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 Submit (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)

Zum Markieren von Pipelines können Sie die Konsole oder AWS CLI verwenden.

Informationen darüber, wie Sie beim Erstellen einer Pipeline ein Tag hinzufügen können, finden Sie unter Erstellen einer Pipeline in CodePipeline.

Bei diesen Schritten wird davon ausgegangen, dass Sie bereits eine aktuelle Version der AWS CLI installiert oder eine Aktualisierung auf die aktuelle Version vorgenommen haben. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface.

Führen Sie am Terminal oder in der Befehlszeile den Befehl tag-resource aus und geben Sie dabei den Amazon-Ressourcennamen (ARN) der Pipeline an, für die Sie Tags hinzufügen möchten, sowie den Schlüssel und Wert des hinzuzufügenden Tags. Sie können einer Pipeline mehrere Tags hinzufügen. Geben Sie beispielsweise Folgendes an, um eine Pipeline mit dem Namen *MyPipeline* mit zwei Tags zu markieren, einem Tag-Schlüssel mit dem Namen *DeploymentEnvironment* und dem Tag-Wert *Test* und einem Tag-Schlüssel mit dem Namen *IscontainerBased* und dem Tag-Wert *true*:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tags key=Project,value=ProjectA key=IscontainerBased,value=true
```

Bei erfolgreicher Ausführung gibt dieser Befehl nichts zurück.

Anzeigen von Tags für eine Pipeline (CLI)

Führen Sie die folgenden Schritte aus, um über die AWS CLI 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. Geben Sie beispielsweise Folgendes an, um eine Liste der Tag-Schlüssel und Tag-Werte für eine Pipeline namens *MyPipeline* mit dem ARN-Wert `arn:aws:codepipeline:`*us-west-2*`:`*account-id*`:`*MyPipeline* anzuzeigen:

```
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)

Führen Sie die folgenden Schritte aus, um über die AWS CLI 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)

Führen Sie die folgenden Schritte aus, um über die AWS CLI 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. Geben Sie beispielsweise Folgendes an, um mehrere Tags für eine Pipeline namens *MyPipeline* mit den Tag-Schlüsseln *Project* und *IscontainerBased* zu entfernen:

```
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. In den Benachrichtigungsregeln werden sowohl die Ereignisse als auch das Amazon SNS SNS-Thema festgelegt, das zum Senden von Benachrichtigungen verwendet wird. Weitere Informationen finden Sie unter Was sind Benachrichtigungen?.

Sie können die Konsole oder die AWS CLI verwenden, um Benachrichtigungsregeln für AWS CodePipeline zu erstellen.

So erstellen Sie eine Benachrichtigungsregel (Konsole):

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie Pipelines (Pipelines) und anschließend die Pipeline aus, der Sie Benachrichtigungen hinzufügen möchten.
3. Wählen Sie auf der Seite für Pipelines Notify (Benachrichtigen) und anschließend Create notification rule (Benachrichtigungsregel erstellen) aus. Sie können auch zur Seite Settings (Einstellungen) für die Pipeline navigieren und Create notification rule (Benachrichtigungsregel erstellen)auswählen.
4. Geben Sie unter Notification name (Benachrichtigungsname) einen Namen für die Regel ein.
5. Wählen Sie unter Detailtyp die Option Basic aus, wenn Sie möchten, dass nur die Informationen, die Amazon zur Verfügung gestellt wurden, in der Benachrichtigung EventBridge enthalten sind. Wählen Sie „Vollständig“, wenn Sie Informationen, die Amazon zur Verfügung gestellt wurden, EventBridge und Informationen, die möglicherweise vom CodePipeline oder vom Benachrichtigungsmanager bereitgestellt wurden, einbeziehen möchten.

   Weitere Informationen finden Sie unter Informationen zu Inhalten und Sicherheit von Benachrichtigungen.
6. Wählen Sie unter Events that trigger notifications (Ereignisse, die Benachrichtigungen auslösen) die Ereignisse aus, für die Sie Benachrichtigungen senden möchten. Weitere Informationen finden Sie unter Ereignisse für Benachrichtigungsregeln in Pipelines.
7. Führen Sie unter Targets (Ziele) einen der folgenden Schritte aus:

   - Wenn Sie bereits eine Ressource zur Verwendung mit Benachrichtigungen konfiguriert haben, wählen Sie unter Choose target type (Zieltyp wählen) entweder AWS Chatbot(Slack) oder SNS topic (SNS-Thema)aus. Wählen Sie unter Ziel auswählen den Namen des Clients (für einen in konfigurierten Slack-ClientAWS Chatbot) oder den Amazon-Ressourcennamen (ARN) des Amazon SNS-Themas (für Amazon SNS SNS-Themen, die bereits mit der für Benachrichtigungen erforderlichen Richtlinie konfiguriert wurden).
   - Wenn Sie keine Ressource für die Verwendung mit Benachrichtigungen konfiguriert haben, wählen Sie Create target (Ziel erstellen) und dann SNS topic (SNS-Thema) aus. Geben Sie

nach codestar-notifications- einen Namen für das Thema an und wählen Sie dann Create (Erstellen).

> **Note**
>
> - Wenn Sie das Amazon-SNS-Thema im Rahmen des Erstellens der Benachrichtigungsregel erstellen, wird die Richtlinie, die es ermöglicht, Ereignisse in dem Thema zu veröffentlichen, für Sie angewendet. Durch die Verwendung eines Themas, das für Benachrichtigungsregeln erstellt wurde, kann sichergestellt werden, dass Sie das Thema nur für die Benutzer abonnieren, die Benachrichtigungen zu dieser Ressource erhalten sollen.
> - Sie können keinen AWS Chatbot-Client als Teil der Erstellung einer Benachrichtigungsregel erstellen. Wenn Sie AWS Chatbot (Slack) wählen, sehen Sie eine Schaltfläche, die Sie zur Konfiguration eines Clients in AWS Chatbot führt. Wenn Sie diese Option auswählen, wird die AWS Chatbot-Konsole geöffnet. Weitere Informationen finden Sie unter Konfigurieren der Integration zwischen Benachrichtigungen und AWS Chatbot.
> - Wenn Sie ein vorhandenes Amazon SNS SNS-Thema als Ziel verwenden möchten, müssen Sie die erforderliche Richtlinie für AWS CodeStar Benachrichtigungen zusätzlich zu allen anderen Richtlinien hinzufügen, die möglicherweise für dieses Thema existieren. Weitere Informationen finden Sie unter Konfigurieren vorhandener Amazon SNS-Themen für Benachrichtigungen und Informationen zu Inhalten und Sicherheit von Benachrichtigungen.

8. Um die Erstellung der Regel abzuschließen, wählen Sie Submit (Absenden) aus.
9. Sie müssen das Amazon SNS SNS-Thema für die Regel abonnieren, bevor sie Benachrichtigungen erhalten können. Weitere Informationen finden Sie unter Amazon SNS SNS-Themen, die Ziele sind für Benutzer abonnieren. Sie können auch die Integration zwischen Benachrichtigungen einrichten und AWS Chatbot Benachrichtigungen an Amazon Chime-Chatrooms oder Slack-Kanäle senden. Weitere Informationen finden Sie unter Konfigurieren der Integration zwischen Benachrichtigungen und AWS Chatbot.

So erstellen Sie eine Benachrichtigungsregel (AWS CLI):

1. Führen Sie in einem Terminal oder einer Eingabeaufforderung den Befehl create-notification rule aus, um das JSON-Skelett zu generieren:

   ```
   aws codestar-notifications create-notification-rule --generate-cli-skeleton
    > rule.json
   ```

   Sie können die Datei beliebig benennen. In diesem Beispiel trägt die Datei den Namen *rule.json*.

2. Öffnen Sie die JSON-Datei in einem Texteditor, und bearbeiten Sie sie so, dass sie die Ressource, die Ereignistypen und das gewünschte Ziel für die Regel enthält. *Das folgende Beispiel zeigt eine Benachrichtigungsregel, die nach einer Pipeline benannt ist,* ***MyNotificationRule*** *die MyDemoPipelinein einem AWS Konto mit der ID 123456789012 benannt ist.* Benachrichtigungen werden mit dem vollständigen Detailtyp an ein Amazon SNS SNS-Thema namens *codestar-notifications* gesendet, MyNotificationTopic wenn Pipeline-Ausführungen beginnen:

   ```
   {
       "Name": "MyNotificationRule",
       "EventTypeIds": [
           "codepipeline-pipeline-pipeline-execution-started"
       ],
       "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyDemoPipeline",
       "Targets": [
           {
               "TargetType": "SNS",
               "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
   notifications-MyNotificationTopic"
           }
       ],
       "Status": "ENABLED",
       "DetailType": "FULL"
   }
   ```

   Speichern Sie die Datei.

3. Führen Sie unter Verwendung der soeben bearbeiteten Datei am Terminal oder in der Befehlszeile erneut den Befehl create-notification-rule aus, um die Benachrichtigungsregel zu erstellen:

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. Bei Erfolg gibt der Befehl den ARN der Benachrichtigungsregel zurück, der ähnlich wie im Folgenden dargestellt aussieht:

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

Arbeiten mit Auslösern in CodePipeline

Mit Auslösern können Sie Ihre Pipeline so konfigurieren, dass sie mit einem bestimmten Ereignistyp oder einem gefilterten Ereignistyp beginnt, z. B. wenn eine Änderung an einer bestimmten Verzweigung oder Pull-Anforderung erkannt wird. Auslöser sind für Quellaktionen mit Verbindungen konfigurierbar, die die `CodeStarSourceConnection` Aktion in verwenden CodePipeline, z. B. GitHub, Bitbucket und GitLab.

Quellaktionen wie CodeCommit und S3 verwenden die Änderungserkennung, wie in diesem Abschnitt zum Starten von Pipelines beschrieben.

Sie können Ihrer Pipeline einen Auslöser hinzufügen und den Auslöser so konfigurieren, dass er nach bestimmten Ereignissen filtert

Sie geben Auslöser über die Konsole oder CLI an.

Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen

Sie können Filter für Pipeline-Auslöser so konfigurieren, dass Pipeline-Ausführungen für verschiedene Git-Ereignisse gestartet werden, z. B. Tag- oder Branch-Push, Änderungen in bestimmten Dateipfaden, eine Pull-Anforderung, die in einer bestimmten Verzweigung geöffnet wird usw. Sie können die AWS CodePipeline Konsole oder die update-pipeline Befehle create-pipeline und in der verwendenAWS CLI, um Auslöserfilter zu konfigurieren.

Sie können Filter für die folgenden Auslösertypen angeben:

- Push

  Ein Push-Auslöser startet eine Pipeline, wenn eine Änderung in Ihr Quell-Repository übertragen wird. Bei der Ausführung wird der Commit aus der Verzweigung verwendet, an die Sie übertragen (d. h. der Zielzweig). Sie können Push-Auslöser nach Verzweigungen, Dateipfaden oder Git-Tags filtern.
- Pull-Anforderung

  Ein Pull-Anforderungsauslöser startet eine Pipeline, wenn eine Pull-Anforderung in Ihrem Quell-Repository geöffnet, aktualisiert oder geschlossen wird. Die Ausführung verwendet den Commit aus der Quellverzweigung, aus der Sie abrufen (d. h. der Quellverzweigung). Sie können Pull-Anforderungsauslöser nach Verzweigungen und Dateipfaden filtern.

> **Note**
>
> Die unterstützten Ereignistypen für Pull-Anforderungen werden geöffnet, aktualisiert oder geschlossen (zusammengeführt). Alle anderen Pull-Anforderungsereignisse werden ignoriert.

Mit der Pipeline-Definition können Sie verschiedene Filter innerhalb derselben Push-Auslöser-Konfiguration kombinieren. Weitere Informationen zur Pipeline-Definition finden Sie unter Auslöserfilterung in der Pipeline-JSON (CLI). Gültige Kombinationen sind:

- tags
- Verzweigungen
- Verzweigungen + Dateipfade

Sie geben Filtertypen wie folgt an:

- Kein Filter

  Diese Auslöserkonfiguration startet Ihre Pipeline bei jedem Push an den Standardzweig, der als Teil der Aktionskonfiguration angegeben ist.

- Filter angeben

  Sie fügen einen Filter hinzu, der Ihre Pipeline auf einem bestimmten Filter startet, z. B. auf den Verzweigungsnamen für einen Code-Push, und ruft den genauen Commit ab. Dadurch wird die Pipeline auch so konfiguriert, dass sie bei Änderungen nicht automatisch gestartet wird.

- Keine Änderungen erkennen

  Dadurch wird kein Auslöser hinzugefügt und die Pipeline wird bei keiner Änderung automatisch gestartet.

Die folgende Tabelle enthält gültige Filteroptionen für jeden Ereignistyp. Die Tabelle zeigt auch, welche Auslöserkonfigurationen standardmäßig für die automatische Änderungserkennung in der Aktionskonfiguration auf „true“ oder „false“ gesetzt sind.

Auslöserkonfiguration	Ereignistyp	Filteroptionen	Änderungen erkennen
Hinzufügen eines Auslösers – kein Filter	Keine	Keine	true
Hinzufügen eines Auslösers – Filtern nach Code-Push	Push-Ereignis	Git-Tags, Verzweigungen, Dateipfade	false
Hinzufügen eines Auslösers – Filtern nach Pull-Anforderungen	Pull-Anforderungen	Zweige, Dateipfade	false
Kein Auslöser – nicht erkennen	Keine	Keine	false

> **Note**
>
> Dieser Auslösertyp verwendet die automatische Änderungserkennung (als `Webhook` Auslösertyp). Die Quellaktionsanbieter, die diesen Auslösertyp verwenden, sind Verbindungen, die für Code-Push konfiguriert sind (Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltet).

Für die Filterung werden Muster für reguläre Ausdrücke im Glob-Format unterstützt, wie unter beschriebenArbeiten mit Glob-Mustern in der Syntax.

> **Note**
>
> In bestimmten Fällen beginnt die Pipeline bei Pipelines mit Auslösern, 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 Auslöserfilterung nach Dateipfaden verwenden, beginnen möglicherweise nicht bei der Erstellung von Verzweigungen.

Themen

- Überlegungen zu Auslöserfiltern
- Beispiele für Auslöserfilter
- Filtern nach Push-Ereignissen (Konsole)
- Filtern nach Pull-Anforderungen (Konsole)
- Auslöserfilterung in der Pipeline-JSON (CLI)
- Auslöserfilterung in AWS CloudFormation Vorlagen

Überlegungen zu Auslöserfiltern

Bei der Verwendung von Auslösern gelten die folgenden Überlegungen.

- Bei einem Auslöser mit Verzweigungs- und Dateipfadfiltern wird die Pipeline beim ersten Pushen der Verzweigung nicht ausgeführt, da kein Zugriff auf die Liste der Dateien besteht, die für die neu erstellte Verzweigung geändert wurden.
- Das Zusammenführen einer Pull-Anforderung kann zwei Pipeline-Ausführungen auslösen, wenn sich die Konfigurationen Push- (Branches-Filter) und Pull-Anforderungs-Auslöser (Branches-Filter) schneiden.

Beispiele für Auslöserfilter

Bei einer Git-Konfiguration mit Filtern für Push- und Pull-Anforderungsereignistypen können die angegebenen Filter miteinander in Konflikt geraten. Im Folgenden finden Sie Beispiele für gültige Filterkombinationen für Push- und Pull-Anforderungsereignisse.

Wenn Kunden Filter in einem einzigen Konfigurationsobjekt kombinieren, verwenden diese Filter eine AND-Operation, was bedeutet, dass nur eine vollständige Übereinstimmung die Pipeline startet. Das folgende Beispiel zeigt die Git-Konfiguration:

```
{
  "filePaths": {
    "includes": ["common/**/*.js"]
  },
  "branches": {
    "includes": ["feature/**"]
  }
```

```
}
```

Mit der obigen Git-Konfiguration zeigt dieses Beispiel ein Ereignis, das die Pipeline-Ausführung startet, da der AND-Vorgang erfolgreich ist.

```
{
  "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "common/app.js"
      ]
      ...
    }
  ]
}
```

Dieses Beispiel zeigt ein Ereignis, das die Pipeline-Ausführung nicht startet, da die Verzweigung filtern kann, der Dateipfad jedoch nicht.

```
{
   "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "src/Main.java"
      ]
      ...
    }
  ]
}
```

Gleichzeitig verwenden Triggerkonfigurationsobjekte innerhalb des Push-Arrays eine ODER-Operation. Auf diese Weise können Sie mehrere Auslöser konfigurieren, um die Ausführung für dieselbe Pipeline zu starten. Eine Liste der Felddefinitionen in der JSON-Struktur finden Sie unter Auslöserfilterung in der Pipeline-JSON (CLI).

Filtern nach Push-Ereignissen (Konsole)

Sie können die Konsole verwenden, um Filter für Push-Ereignisse hinzuzufügen und Verzweigungen oder Dateipfade einzuschließen oder auszuschließen.

Filtern nach Push-Ereignissen (Konsole)

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

   Es werden Name und Status aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Andernfalls führen Sie diese Schritte im Pipeline-Erstellungsassistenten aus.
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 Auslöser bearbeiten aus. Wählen Sie Filter angeben aus.
5. Wählen Sie unter Ereignistyp die Option Push aus den folgenden Optionen aus.

   - Wählen Sie Push aus, um die Pipeline zu starten, wenn eine Änderung in Ihr Quell-Repository übertragen wird. Wenn Sie dies auswählen, können die Felder Filter für Verzweigungen und Dateipfade oder Git-Tags angeben.
   - Wählen Sie Pull-Anforderung aus, um die Pipeline zu starten, wenn eine Pull-Anforderung in Ihrem Quell-Repository geöffnet, aktualisiert oder geschlossen wird. Wenn Sie dies auswählen, können die Felder Filter für Zielzweige und Dateipfade angeben.
6. Wählen Sie unter Filtertyp eine der folgenden Optionen aus.

   - Wählen Sie Verzweigung, um die Verzweigungen in Ihrem Quell-Repository anzugeben, die der Auslöser überwacht, um zu wissen, wann eine Workflow-Ausführung gestartet werden soll. Geben Sie unter Einschließen Muster für Verzweigungsnamen im Glob-Format ein, die Sie für die Auslöserkonfiguration angeben möchten, um Ihre Pipeline bei Änderungen in den angegebenen Verzweigungen zu starten. Geben Sie unter Ausschließen die Regex-Muster für Verzweigungsnamen im Glob-Format ein, die Sie für die Auslöserkonfiguration angeben möchten, um sie zu ignorieren und Ihre Pipeline nicht bei Änderungen in den angegebenen Verzweigungen zu starten. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

> **Note**
>
> Wenn Ein- und Ausschließen beide dasselbe Muster haben, wird standardmäßig das Muster ausgeschlossen.

Sie können Regex-Muster im Glob-Format verwenden, um Ihre Verzweigungsnamen zu definieren. Verwenden Sie beispielsweise , `main.*` um alle Verzweigungen abzugleichen, die mit beginnen`main.*`. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

Geben Sie für einen Push-Auslöser die Verzweigungen an, die Sie an übertragen, d. h. die Zielverzweigungen. Geben Sie für einen Pull-Anforderungsauslöser Zielzweige an, für die Sie die Pull-Anforderung öffnen.

- (Optional) Geben Sie unter Dateipfade die Dateipfade für Ihren Auslöser an. Geben Sie die Namen in Einschließen und Ausschließen ein.

  Sie können Regex-Muster im Glob-Format verwenden, um Ihre Dateinamen zu definieren. Verwenden Sie beispielsweise , `prod.*` um alle Dateipfade abzugleichen, die mit beginnen`prod.*`. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.
- Wählen Sie Tags, um die Pipeline-Auslöserkonfiguration so zu konfigurieren, dass sie mit Git-Tags beginnt. Geben Sie unter Einschließen Muster für Tag-Namen im Glob-Format ein, die Sie für die Auslöserkonfiguration angeben möchten, um Ihre Pipeline bei der Veröffentlichung des angegebenen Tags zu starten. Geben Sie unter Ausschließen die Regex-Muster für Tag-Namen im Glob-Format ein, die Sie angeben möchten, damit die Auslöserkonfiguration ignoriert und Ihre Pipeline bei der Veröffentlichung des angegebenen Tags nicht startet. Wenn Ein- und Ausschließen beide dasselbe Tag-Muster haben, wird standardmäßig das Tag-Muster ausgeschlossen.

Filtern nach Pull-Anforderungen (Konsole)

Sie können die Konsole verwenden, um Filter für Pull-Anforderungen mit bestimmten Ereignissen hinzuzufügen und Verzweigungen oder Dateipfade einzuschließen oder auszuschließen.

Filtern nach Pull-Anforderungen (Konsole)

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

   Es werden Name und Status aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, die Sie bearbeiten möchten. Andernfalls führen Sie diese Schritte im Pipeline-Erstellungsassistenten aus.
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 Auslöser bearbeiten aus. Wählen Sie Filter angeben aus.
5. Wählen Sie unter Ereignistyp die Option Pull-Anforderung aus den folgenden Optionen aus.

   - Wählen Sie Push aus, um die Pipeline zu starten, wenn eine Änderung in Ihr Quell-Repository übertragen wird. Wenn Sie dies auswählen, können die Felder Filter für Verzweigungen und Dateipfade oder Git-Tags angeben.
   - Wählen Sie Pull-Anforderung aus, um die Pipeline zu starten, wenn eine Pull-Anforderung für die angegebenen Zielzweige geöffnet, aktualisiert oder geschlossen wird. Wenn Sie dies auswählen, können die Felder Filter für Verzweigungen und Dateipfade angeben.

     Sie können optional die folgenden zu filternden Pull-Anforderungsereignisse angeben:

     - Pull-Anforderung wurde erstellt
     - Es wird eine neue Revision vorgenommen, um die Anforderung abzurufen
     - Pull-Anforderung wird geschlossen
6. Wählen Sie unter Filtertyp eine der folgenden Optionen aus.

   - Wählen Sie Verzweigung, um die Verzweigungen in Ihrem Quell-Repository anzugeben, die der Auslöser überwacht, um zu wissen, wann eine Workflow-Ausführung gestartet werden soll. Geben Sie unter Einschließen Muster für Verzweigungsnamen im Glob-Format ein, die Sie für die Auslöserkonfiguration angeben möchten, um Ihre Pipeline bei Änderungen in den angegebenen Verzweigungen zu starten. Geben Sie unter Ausschließen die Regex-Muster für Verzweigungsnamen im Glob-Format ein, die Sie für die Auslöserkonfiguration angeben möchten, um zu ignorieren und Ihre Pipeline nicht bei Änderungen in den angegebenen Verzweigungen zu starten. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

> **Note**
>
> Wenn Ein- und Ausschließen beide dasselbe Muster haben, wird standardmäßig das Muster ausgeschlossen.

Sie können Regex-Muster im Glob-Format verwenden, um Ihre Verzweigungsnamen zu definieren. Verwenden Sie beispielsweise , `main.*` um alle Verzweigungen abzugleichen, die mit beginnen`main.*`. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

Geben Sie für einen Push-Auslöser die Verzweigungen an, die Sie an übertragen, d. h. die Zielverzweigungen. Geben Sie für einen Pull-Anforderungsauslöser Zielzweige an, für die Sie die Pull-Anforderung öffnen.

- (Optional) Geben Sie unter Dateipfade die Dateinamen für Ihren Auslöser an. Geben Sie die Namen in Einschließen und Ausschließen ein.

  Sie können Regex-Muster im Glob-Format verwenden, um Ihre Dateipfadnamen zu definieren. Verwenden Sie beispielsweise , `prod.*` um alle Dateipfade abzugleichen, die mit beginnen`prod.*`. Weitere Informationen finden Sie unter Arbeiten mit Glob-Mustern in der Syntax.

Auslöserfilterung in der Pipeline-JSON (CLI)

Sie können das Pipeline-JSON aktualisieren, um Filter für Auslöser hinzuzufügen.

Um die AWS CLI zum Erstellen oder Aktualisieren Ihrer Pipeline zu verwenden, verwenden Sie den `update-pipeline` Befehl `create-pipeline` oder .

Die folgende JSON-Beispielstruktur bietet eine Referenz für die Felddefinitionen unter `create-pipeline`.

```
{
    "pipeline": {
        "name": "MyServicePipeline",
        "triggers": [
            {
                "provider": "Connection",
```

```
                "gitConfiguration": {
                    "sourceActionName": "ApplicationSource",
                    "push": [
                        {
                            "filePaths": {
                                "includes": [
                                    "projectA/**",
                                    "common/**/*.js"
                                ],
                                "excludes": [
                                    "**/README.md",
                                    "**/LICENSE",
                                    "**/CONTRIBUTING.md"
                                ]
                            },
                            "branches": {
                                "includes": [
                                    "feature/**",
                                    "release/**"
                                ],
                                "excludes": [
                                    "mainline"
                                ]
                            },
                            "tags": {
                                "includes": [
                                    "release-v0", "release-v1"
                                ],
                                "excludes": [
                                    "release-v2"
                                ]
                            }
                        }
                    ],
                    "pullRequest": [
                        {
                            "events": [
                                "CLOSED"
                            ],
                            "branches": {
                                "includes": [
                                    "feature/**",
                                    "release/**"
                                ],
```

```
                                "excludes": [
                                    "mainline"
                                ]
                            },
                            "filePaths": {
                                "includes": [
                                    "projectA/**",
                                    "common/**/*.js"
                                ],
                                "excludes": [
                                    "**/README.md",
                                    "**/LICENSE",
                                    "**/CONTRIBUTING.md"
                                ]
                            }
                        }
                    ]
                }
            }
        ],
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "ApplicationSource",
                        "configuration": {
                            "BranchName": "mainline",
                            "ConnectionArn": "arn:aws:codestar-connections:eu-
central-1:111122223333:connection/fe9ff2e8-ee25-40c9-829e-65f8EXAMPLE",
                            "FullRepositoryId": "monorepo-example",
                            "OutputArtifactFormat": "CODE_ZIP"
                        }
                    }
                ]
            }
        ]
    }
}
```

Die Felder in der JSON-Struktur sind wie folgt definiert:

- `sourceActionName`: Der Name der Pipeline-Quellaktion mit der Git-Konfiguration.

- `push`: Push-Ereignisse mit Filterung. Diese Ereignisse verwenden eine ODER-Operation zwischen verschiedenen Push-Filtern und eine AND-Operation innerhalb von Filtern.
  - `branches`: Die Verzweigungen, nach denen gefiltert werden soll. Verzweigungen verwenden eine AND-Operation zwischen Einschließen und Ausschließen.
    - `includes`: Muster, nach denen nach Zweigen gefiltert werden soll, die eingeschlossen werden. Umfasst die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Verzweigungen gefiltert werden soll, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.
  - `filePaths`: Die Dateinamen, nach denen gefiltert werden soll.
    - `includes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die enthalten sein werden. Umfasst die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.
  - `tags`: Die Tag-Namen, nach denen gefiltert werden soll.
    - `includes`: Muster, nach denen nach Tags gefiltert werden soll, die enthalten sein werden. Umfasst die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Tags gefiltert werden sollen, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.
- `pullRequest`: Abrufen von Anforderungsereignissen mit Filterung nach Pull-Anforderungsereignissen und Pull-Anforderungsfiltern.
  - `events`: Filtert nach offenen, aktualisierten oder geschlossenen Pull-Anforderungsereignissen, wie angegeben.
  - `branches`: Die Verzweigungen, nach denen gefiltert werden soll. Verzweigungen verwenden eine AND-Operation zwischen Einschließen und Ausschließen.
    - `includes`: Muster, nach denen nach Zweigen gefiltert werden soll, die eingeschlossen werden. Umfasst die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Verzweigungen gefiltert werden soll, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.
  - `filePaths`: Die Dateinamen, nach denen gefiltert werden soll.
    - `includes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die enthalten sein werden. Umfasst die Verwendung einer ODER-Operation.
    - `excludes`: Muster, nach denen nach Dateipfaden gefiltert werden soll, die ausgeschlossen werden. Ausschlüsse verwenden eine ODER-Operation.

Auslöserfilterung in AWS CloudFormation Vorlagen

Sie können die Pipeline-Ressource in aktualisierenAWS CloudFormation, um Auslöserfilterung hinzuzufügen.

Der folgende Beispiel-Vorlagenausschnitt enthält eine YAML-Referenz für Auslöserfelddefinitionen. Eine Liste der Felddefinitionen finden Sie unter Auslöserfilterung in der Pipeline-JSON (CLI).

```
pipeline:
  name: MyServicePipeline
  executionMode: PARALLEL
  triggers:
    - provider: CodeConnection
      gitConfiguration:
        sourceActionName: ApplicationSource
        push:
          - filePaths:
              includes:
                - projectA/**
                - common/**/*.js
              excludes:
                - '**/README.md'
                - '**/LICENSE'
                - '**/CONTRIBUTING.md'
            branches:
              includes:
                - feature/**
                - release/**
              excludes:
                - mainline
          - tags:
              includes:
                - release-v0
                - release-v1
              excludes:
                - release-v2
        pullRequest:
          - events:
              - CLOSED
            branches:
              includes:
                - feature/**
                - release/**
```

```
              excludes:
                - mainline
            filePaths:
              includes:
                - projectA/**
                - common/**/*.js
              excludes:
                - '**/README.md'
                - '**/LICENSE'
                - '**/CONTRIBUTING.md'
  stages:
    - name: Source
      actions:
        - name: ApplicationSource
          configuration:
            BranchName: mainline
            ConnectionArn: arn:aws:codestar-connections:eu-
central-1:111122223333:connection/fe9ff2e8-ee25-40c9-829e-65f85EXAMPLE
            FullRepositoryId: monorepo-example
            OutputArtifactFormat: CODE_ZIP
```

Verwalten von Ausführungen in CodePipeline

Um den Pipeline-Fortschritt zu analysieren, können Sie Fehlerprotokolle anzeigen, den Pipeline- und Aktionsausführungsverlauf anzeigen und fehlgeschlagene Phasen oder Aktionen wiederholen.

Themen

- Hinrichtungen anzeigen in CodePipeline
- Festlegen des Pipeline-Ausführungsmodus
- Wiederholen Sie eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase

Hinrichtungen anzeigen in CodePipeline

Sie können die AWS CodePipeline Konsole oder die verwenden, AWS CLI um den Ausführungsstatus und den Ausführungsverlauf anzuzeigen und fehlgeschlagene Phasen oder Aktionen erneut zu versuchen.

Themen

- Ausführungsverlauf einer Pipeline anzeigen (Konsole)
- 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. Der Ausführungsverlauf wird bis zu 12 Monate lang aufbewahrt.

> **Note**
>
> Der detaillierte Ausführungsverlauf ist für die Ausführungen verfügbar, die am oder nach dem 21. Februar 2019 erfolgten.

Sie können die Konsole verwenden, um den Ausführungsverlauf in einer Pipeline anzuzeigen, einschließlich Status, Quellversionen und Timingdetails für jede Ausführung.

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

   Die Namen aller Pipelines für Ihr AWS-Konto sowie deren Status werden angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline.
3. Wählen Sie View history (Verlauf anzeigen).
4. Zeigen Sie Status, Quellrevisionen, Änderungsdetails und Auslöser für jede Ausführung Ihrer Pipeline an.
5. Wählen Sie eine Ausführung aus. In der Detailansicht werden Ausführungsdetails, die Registerkarte Zeitleiste, die Registerkarte Visualisierung und die Registerkarte Variablen angezeigt. Variablenwerte für Variablen auf Pipeline-Ebene werden zum Zeitpunkt der Pipeline-Ausführung aufgelöst und können im Ausführungsverlauf für jede Ausführung eingesehen werden.

   > **Note**
   >
   > Ausgabevariablen von Pipeline-Aktionen können auf der Registerkarte Ausgabevariablen unter dem Verlauf für jede Aktionsausführung angezeigt werden.

Anzeige des Ausführungsstatus (Konsole)

Sie können den Pipeline-Status unter Status auf der Ausführungsverlaufsseite anzeigen. Wählen Sie einen Ausführungs-ID-Link, und zeigen Sie dann den Aktionsstatus an.

Die folgenden Status gelten für Pipelines, Phasen und Aktionen:

> **Note**
>
> Die folgenden Pipeline-Status gelten auch für eine Pipeline-Ausführung, bei der es sich um eine eingehende Ausführung handelt. Informationen zur Anzeige einer eingehenden Ausführung und ihres Status finden Sie unter. Eine eingehende Ausführung anzeigen (Konsole)

Status auf Pipelineebene

Pipelinestatus	Beschreibung
InProgress	Die Pipelineausführung wird derzeit ausgeführt.
Wird angehalten	Die Pipeline-Ausführung wird aufgrund einer Anforderung mit "Anhalten und beenden" oder "Anhalten und warten" angehalten.
Angehalten	Der Anhaltevorgang ist abgeschlossen, und die Pipeline-Ausführung wird angehalten.
Erfolgreich	Die Pipelineausführung wurde erfolgreich abgeschlossen.
Ersetzt	Während auf den Abschluss der nächsten Stufe dieser Pipelinea usführung gewartet wurde, wurde stattdessen eine neuere Pipelinea usführung über die Pipeline fortgesetzt.
Fehlgeschlagen	Die Pipelineausführung wurde nicht erfolgreich abgeschlossen.

Status auf Stufenebene

Stufenstatus	Beschreibung
InProgress	Die Stufe wird derzeit ausgeführt.
Wird angehalten	Die Ausführung der Phase wird aufgrund einer Anforderung zum Anhalten und Warten oder zum Anhalten und Beenden der Pipeline-Ausführung gestoppt.

Stufenstatus	Beschreibung
Angehalten	Der Anhaltevorgang ist abgeschlossen, und die Phasenausführung wird angehalten.
Erfolgreich	Die Stufe wurde erfolgreich abgeschlossen.
Fehlgeschlagen	Die Stufe wurde nicht erfolgreich abgeschlossen.

Status auf Aktionsebene

Aktionsstatus	Beschreibung
InProgress	Die Aktion wird derzeit ausgeführt.
Abandoned (Abgebrochen)	Die Aktion wird aufgrund einer Anforderung mit "Anhalten und Beenden" für die Pipeline-Ausführung beendet.
Erfolgreich	Die Aktion wurde erfolgreich abgeschlossen.
Fehlgeschlagen	Bei Genehmigungsaktionen bedeutet der Status FAILED (FEHLGESCHLAGEN), dass die Aktion entweder vom Prüfer abgelehnt wurde oder aufgrund einer falschen Aktionskonfiguration fehlgeschlagen ist.

Eine eingehende Ausführung anzeigen (Konsole)

Sie können die Konsole verwenden, um den Status und die Details einer eingehenden Ausführung anzuzeigen. Wenn der Übergang aktiviert ist oder die Phase verfügbar wird, wird eine eingehende Ausführung `InProgress` fortgesetzt und tritt in die Phase ein. Eine eingehende Ausführung mit einem `Stopped` Status tritt nicht in die Phase ein. Der Status einer eingehenden Ausführung ändert sich in, `Failed` wenn die Pipeline bearbeitet wird. Wenn Sie eine Pipeline bearbeiten, werden alle laufenden Ausführungen nicht fortgesetzt, und der Ausführungsstatus ändert sich auf. `Failed`

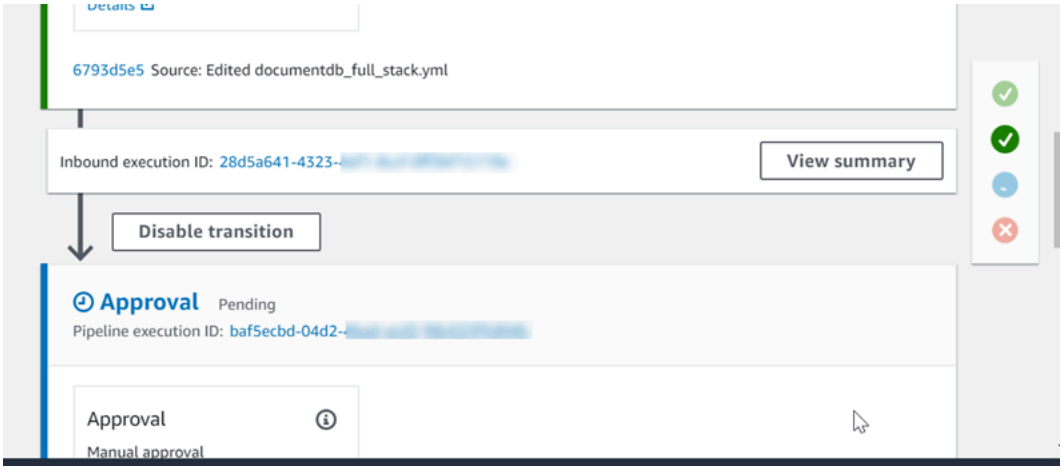
Wenn Sie keine eingehende Ausführung sehen, gibt es in einer deaktivierten Phase des Übergangs keine ausstehenden Ausführungen.

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

Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.

2. Wählen Sie den Namen der Pipeline, für die Sie die eingehende Ausführung anzeigen möchten, und führen Sie einen der folgenden Schritte aus:

   - Wählen Sie View (Anzeige). Im Pipeline-Diagramm können Sie im Feld Inbound Execution ID vor Ihrer deaktivierten Transition die Ausführungs-ID für den eingehenden Datenverkehr einsehen.



     Wählen Sie „Zusammenfassung anzeigen“, um Ausführungsdetails wie die Ausführungs-ID, den Quell-Trigger und den Namen der nächsten Phase anzuzeigen.

   - Wählen Sie die Pipeline aus und klicken Sie auf Verlauf anzeigen.

Anzeigen von Revisionen der Pipeline-Ausführungsquelle (Konsole)

Sie können Einzelheiten zu Quellartefakten (Ausgabeartefakten, die aus der ersten Phase einer Pipeline stammen) anzeigen, die bei der Ausführung einer Pipeline verwendet werden. Dazu gehören IDs wie z. B. Commit-IDs, Kommentare zu Zugängen und bei Verwendung der CLI die Versionsnummern der Pipeline-Build-Aktionen. Bei einigen Revisionstypen können Sie die URL des Commits anzeigen und öffnen. Quell-Revisionen setzen sich folgendermaßen zusammen:

- Zusammenfassung: Zusammengefasste Informationen über die letzte Revision des Artefakts. Für GitHub und CodeCommit Repositorys, die Commit-Nachricht. Für Amazon S3 S3-Buckets oder -Aktionen der vom Benutzer bereitgestellte Inhalt eines codepipeline-artifact-revision-summary Schlüssels, der in den Objektmetadaten angegeben ist.
- revisionUrl: Die Revisions-URL für die Artefakt-Revision (z. B. die externe Repository-URL).

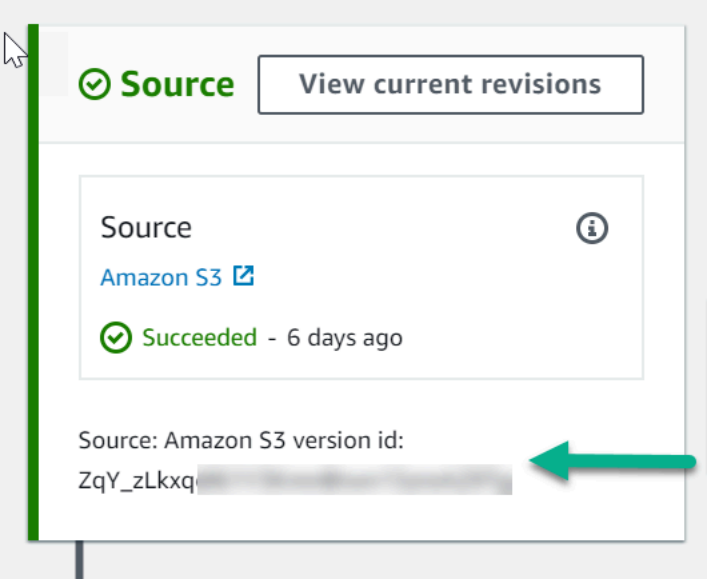
- revisionId: Die Versions-ID für die Artefakt-Revision. Für eine Quellenänderung in einem CodeCommit GitHub OR-Repository ist dies beispielsweise die Commit-ID. Bei Artefakten, die in GitHub oder CodeCommit Repositorien gespeichert sind, ist die Commit-ID mit einer Commit-Detailseite verknüpft.

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

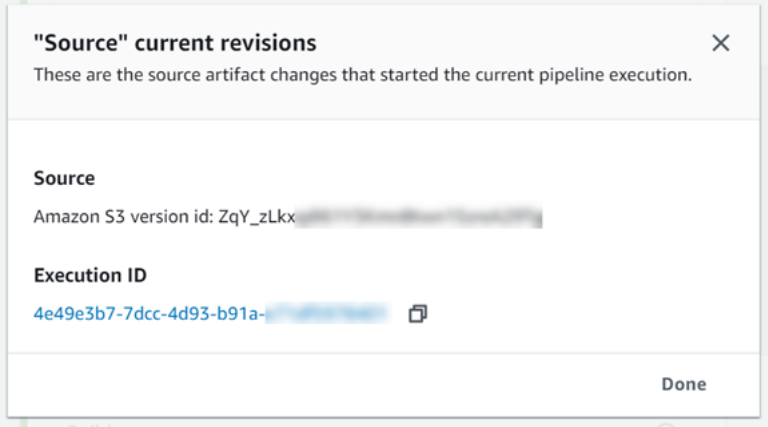
   Die Namen aller Pipelines, die mit Ihrer verknüpft sindAWS-Konto, werden angezeigt.

2. Wählen Sie den Namen der Pipeline aus, für die Sie die Quellrevision anzeigen möchten. Führen Sie eine der folgenden Aktionen aus:

   - Wählen Sie View history (Verlauf anzeigen). Unter Source revisions (Quell-Revisionen) wird die Quelländerung für jede Ausführung aufgeführt.
   - Suchen Sie eine Aktion, für die Sie Details anzeigen möchten, und suchen Sie dann die Quellrevisionsinformationen unten in ihrer Stufe:



   Wählen Sie View current revisions (Aktuelle Versionen anzeigen) aus, um Quellinformationen anzuzeigen. Mit Ausnahme von Artefakten, die in Amazon S3 S3-Buckets gespeichert sind, sind Identifikatoren wie Commit-IDs in dieser Informationsdetailansicht mit Quellinformationsseiten für die Artefakte verknüpft.



Anzeigen von Aktionsausführungen (Konsole)

Sie können Aktionsdetails für eine Pipeline anzeigen, wie etwa die Aktionsausführungs-ID, Eingabeartefakte, Ausgabeartefakte und den Status. Sie können Aktionsdetails anzeigen, indem Sie in der Konsole eine Pipeline und dann eine Ausführungs-ID auswählen.

> **Note**
>
> Der detaillierte Ausführungsverlauf ist für die Ausführungen verfügbar, die am oder nach dem 21. Februar 2019 erfolgten.

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

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie den Namen der Pipeline aus, für die Sie die Aktionsdetails anzeigen möchten, und wählen Sie dann View history (Verlauf anzeigen).
3. Wählen Sie unter Execution ID (Ausführungs-ID) die Ausführungs-ID, für die Sie die Details der Aktionsausführung anzeigen möchten.
4. Auf der Registerkarte Timeline (Zeitleiste) sehen Sie die folgenden Informationen:

   a. Wählen Sie unter Action name (Aktionsname) den Link zum Öffnen der Aktion, wo Sie Status, Phasenname, Aktionsname, Konfigurationsdaten und Artefaktinformationen finden.

   b. Wählen Sie unter Provider (Anbieter) den link zur Anzeige der Details zum Aktionsanbieter. Wenn Sie beispielsweise in der vorherigen Beispielpipeline entweder die Phase Staging

oder die Produktionsphase 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 Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie den Namen der Pipeline aus, für die Sie die Aktionsdetails anzeigen möchten, und wählen Sie dann View history (Verlauf anzeigen).
3. Wählen Sie unter Execution ID (Ausführungs-ID) die Ausführungs-ID, für die Sie die Aktionsdetails anzeigen möchten.
4. Wählen Sie auf der Registerkarte Timeline unter Action name (Aktionsname) den Link zum Öffnen einer Detailseite für die Aktion.
5. Sehen Sie sich auf der Detailseite auf der Registerkarte Ausführung den Status und den Zeitpunkt der Ausführung der Aktion an.
6. Sehen Sie sich auf der Registerkarte Konfiguration die Ressourcenkonfiguration für die Aktion an (z. B. den Namen des CodeBuild Build-Projekts).
7. Sehen Sie sich auf der Registerkarte Artefakte die Artefaktdetails unter Artefakttyp und Artefaktanbieter an. Wählen Sie den Link unter Artifact name (Artefaktname) um die Artefakte im Artefaktspeicher anzuzeigen.
8. Sehen Sie sich auf der Registerkarte Ausgabevariablen die aufgelösten Variablen von Aktionen in der Pipeline für die Aktionsausführung an.

Anzeigen von der Pipeline-Details und des Verlaufs (CLI)

Sie können die folgenden Befehle verwenden, um weitere Details zu Ihren Pipelines und Pipeline-Ausführungen anzuzeigen:

- list-pipelines-Befehl, um eine Zusammenfassung aller Pipelines in Ihrem AWS-Konto 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.

Themen

- Anzeigen des Ausführungsverlauf (CLI)
- Anzeigen des Ausführungsstatus (CLI)
- Ausführungsstatus eingehender Nachrichten anzeigen (CLI)
- Anzeigen von Quellrevisionen (CLI)
- Anzeigen von Aktionsausführungen (CLI)

Anzeigen des Ausführungsverlauf (CLI)

Sie können den Ausführungsverlauf einer Pipeline anzeigen.

- Um Details zu vorherigen Ausführungen einer Pipeline anzuzeigen, führen Sie den Befehl list-pipeline-executions unter Angabe des eindeutigen Namens der Pipeline aus. 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.

  Das folgende Beispiel zeigt die zurückgegebenen Daten für eine Pipeline mit dem Namen *MyFirstPipeline* mit drei Ausführungen:

```
{
    "pipelineExecutionSummaries": [
        {
            "lastUpdateTime": 1496380678.648,
            "pipelineExecutionId": "7cf7f7cb-3137-539g-j458-d7eu3EXAMPLE",
            "startTime": 1496380258.243,
            "status": "Succeeded"
        },
        {
            "lastUpdateTime": 1496591045.634,
            "pipelineExecutionId": "3137f7cb-8d494hj4-039j-d84l-d7eu3EXAMPLE",
            "startTime": 1496590401.222,
            "status": "Succeeded"
        },
        {
            "lastUpdateTime": 1496946071.6456,
            "pipelineExecutionId": "4992f7jf-7cf7-913k-k334-d7eu3EXAMPLE",
            "startTime": 1496945471.5645,
            "status": "Succeeded"
        }
    ]
}
```

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"
            }
        ]
    }
}
```

Anzeigen des Ausführungsstatus (CLI)

Sie können die CLI verwenden, um den Pipeline-, Phasen- und Aktionsstatus anzuzeigen.

- Um Informationen über den aktuellen Status einer Pipeline anzuzeigen, führen Sie den Befehl get-pipeline-state unter Angabe des eindeutigen Namens der Pipeline aus. 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"
        }
    ]
}
```

Ausführungsstatus eingehender Nachrichten anzeigen (CLI)

Sie können die CLI verwenden, um den Ausführungsstatus eingehender Nachrichten anzuzeigen. Wenn der Übergang aktiviert ist oder die Phase verfügbar wird, wird eine eingehende Ausführung `InProgress` fortgesetzt und tritt in die Phase ein. Eine eingehende Ausführung mit einem `Stopped` Status tritt nicht in die Phase ein. Der Status einer eingehenden Ausführung ändert sich in, `Failed` wenn die Pipeline bearbeitet wird. Wenn Sie eine Pipeline bearbeiten, werden alle laufenden Ausführungen nicht fortgesetzt, und der Ausführungsstatus ändert sich auf. `Failed`

- Um Informationen über den aktuellen Status einer Pipeline anzuzeigen, führen Sie den Befehl get-pipeline-state unter Angabe des eindeutigen Namens der Pipeline aus. 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. In der Ausgabe wird auch die Pipeline-Ausführungs-ID in jeder Phase angezeigt und es wird angegeben, ob es für eine Phase mit deaktiviertem Übergang eine eingehende Ausführungs-ID gibt.

Das folgende Beispiel zeigt die zurückgegebenen Daten für eine zweistufige Pipeline mit dem Namen *MyFirstPipeline*, wobei in der ersten Phase ein aktivierter Übergang und eine erfolgreiche Pipelineausführung angezeigt werden und in der zweiten Phase mit dem Namen `Beta` ein deaktivierter Übergang und eine Ausführungs-ID für eingehenden Datenverkehr angezeigt werden. Die eingehende Ausführung kann den Status `InProgress`, `Stopped`, 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
}
```

Anzeigen von Quellrevisionen (CLI)

Sie können Einzelheiten zu Quellartefakten (Ausgabeartefakten, die aus der ersten Phase einer Pipeline stammen) anzeigen, die bei der Ausführung einer Pipeline verwendet werden. Diese Informationen umfassen IDs wie z. B. Commit-IDs, Kommentare zu Zugängen, Zeit seit Erstellung oder Aktualisierung des Artefakts und bei Verwendung der CLI die Versionsnummern der Build-

Aktionen. Bei einigen Revisionstypen können Sie die URL des Commits für die Artefaktversion ansehen und öffnen. Quell-Revisionen setzen sich folgendermaßen zusammen:

- Zusammenfassung: Zusammengefasste Informationen über die letzte Revision des Artefakts. Für GitHub und AWS CodeCommit Repositorys, die Commit-Nachricht. Für Amazon S3 S3-Buckets oder -Aktionen der vom Benutzer bereitgestellte Inhalt eines codepipeline-artifact-revision-summary Schlüssels, der in den Objektmetadaten angegeben ist.
- revisionUrl: Die Commit-ID für die Artefaktrevision. Bei Artefakten, die in GitHub oder AWS CodeCommit Repositorys gespeichert sind, ist die Commit-ID mit einer Commit-Detailseite verknüpft.

Sie können mithilfe des Befehls get-pipeline-execution Informationen über die letzten Quellrevisionen, die in die Ausführung einer Pipeline eingeschlossen wurden, anzeigen lassen. Nach der ersten Ausführung des Befehls get-pipeline-state zum Abruf von Details zu allen Phasen in einer Pipeline identifizieren Sie die Ausführungs-ID, die für eine Phase gilt, für die Sie Quell-Revisionsdetails abrufen möchten. Anschließend verwenden Sie die Ausführungs-ID im Befehl get-pipeline-execution. (Da Phasen in einer Pipeline möglicherweise während verschiedener Pipeline-Ausführungen zuletzt erfolgreich abgeschlossen wurden, können sie unterschiedliche Ausführungs-IDs aufweisen.)

Das heißt, wenn Sie Details über Artefakte in der aktuellen Staging-Phase einsehen möchten, führen Sie den Befehl get-pipeline-state aus, ermitteln Sie die aktuelle Ausführungs-ID der Staging-Phase und führen Sie dann den Befehl get-pipeline-execution anhand dieser Ausführungs-ID aus.

So zeigen Sie Quellrevisionen in einer Pipeline an

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder eine Befehlszeile (Windows) und verwenden Sie dieAWS 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"
            }
            //More revisions might be listed here
        ],
        "pipelineExecutionId": "3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE",
        "pipelineName": "MyFirstPipeline",
        "pipelineVersion": 2,
        "status": "Succeeded"
    }
}
```

Anzeigen von Aktionsausführungen (CLI)

Sie können Aktionsausführungsdetails für eine Pipeline anzeigen, wie etwa die Aktionsausführungs-ID, Eingabeartefakte, Ausgabeartefakte sowie das Ausführungsergebnis und den Status. Sie geben den Ausführungs-ID-Filter an, um eine Liste der Aktionen in einer Pipeline-Ausführung auszugeben:

> **Note**
>
> Der detaillierte Ausführungsverlauf ist für die Ausführungen verfügbar, die am oder nach dem 21. Februar 2019 erfolgten.

- Gehen Sie zum Anzeigen der Aktionsausführungen für ein Pipeline wie folgt vor:
  - Um Details zu allen Ausführungen in einer Pipeline anzuzeigen, führen Sie den Befehl list-action-executions unter Angabe des eindeutigen Namens der Pipeline aus. 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. Wenn Sie beispielsweise die Aktionsausführungen für eine *Ausführungs-ID* anzeigen möchten, geben Sie Folgendes ein:

```
aws codepipeline list-action-executions --pipeline-name MyFirstPipeline --filter
 pipelineExecutionId=Execution_ID
```

- Das folgende Beispiel zeigt einen Teil der Beispielausgabe für diesen Befehl:

```
{
    "actionExecutionDetails": [
        {
            "stageName": "Beta",
            "pipelineVersion": 8,
            "actionName": "Build",
```

```
            "status": "Failed",
            "lastUpdateTime": 1552958312.034,
            "input": {
                "configuration": {
                    "ProjectName": "java-project"
                },
                "region": "us-east-1",
                "actionTypeId": {
                    "owner": "AWS",
                    "category": "Build",
                    "provider": "CodeBuild",
                    "version": "1"
                },
                "inputArtifacts": [
                    {
                        "s3location": {
                            "bucket": "codepipeline-us-east-1-ID",
                            "key": "MyFirstPipeline/MyApp/Object.zip"
                        },
                        "name": "MyApp"
                    }
                ]
            },


. . .
```

Festlegen des Pipeline-Ausführungsmodus

Sie können den Ausführungsmodus für Ihre Pipeline festlegen, um anzugeben, wie mehrere Ausführungen verarbeitet werden.

Weitere Informationen zu Pipeline-Ausführungsmodi finden Sie unter So funktionieren Pipeline-Ausführungen.

> **Important**
>
> Bei Pipelines im PARALLEL-Modus wird der aktualisierte Status beim Bearbeiten des Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED nicht als PARALLEL

angezeigt. Weitere Informationen finden Sie unter Pipelines, die aus dem PARALLEL-Modus geändert wurden, zeigen einen vorherigen Ausführungsmodus an.

Important

Bei Pipelines im PARALLEL-Modus wird beim Bearbeiten des Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED die Pipeline-Definition für die Pipeline in jedem Modus nicht aktualisiert. Weitere Informationen finden Sie unter Pipelines im PARALLEL-Modus haben eine veraltete Pipeline-Definition, wenn sie beim Wechsel in den Modus QUEUED oder SUPERSEDED bearbeitet werden.

Festlegen des Pipeline-Ausführungsmodus (Konsole)

Sie können die Konsole verwenden, um den Pipeline-Ausführungsmodus festzulegen.

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

   Die Namen und Status aller Pipelines, die Ihrem AWS Konto zugeordnet 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 die Option Bearbeiten: Pipeline-Eigenschaften aus.
5. Wählen Sie den Modus für Ihre Pipeline aus.
   - Ersetzt
   - Warteschlange (Pipeline-Typ V2 erforderlich)
   - Parallel (Pipeline-Typ V2 erforderlich)
6. Wählen Sie auf der Seite Bearbeiten die Option Fertig aus.

Festlegen des Pipeline-Ausführungsmodus (CLI)

Um die AWS CLI zum Festlegen des Pipeline-Ausführungsmodus zu verwenden, verwenden Sie den `update-pipeline` Befehl `create-pipeline` oder .

1. Öffnen Sie eine Terminalsitzung (Linux, macOS oder Unix) oder Eingabeaufforderung (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 den Pipeline-Ausführungsmodus widerspiegelt, den Sie festlegen möchten, z. B. QUEUED.

   ```
   "executionMode": "QUEUED"
   ```

   Das folgende Beispiel zeigt, wie Sie den Ausführungsmodus in einer Beispielpipeline mit zwei Phasen 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.

Wiederholen Sie eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase

Sie können eine Phase wiederholen, bei der ein Fehler aufgetreten ist, ohne eine Pipeline erneut von Anfang an ausführen zu müssen. Sie tun dies, indem Sie entweder die fehlgeschlagenen Aktionen in einer Phase erneut versuchen oder indem Sie alle Aktionen in der Phase wiederholen, beginnend mit der ersten Aktion in der Phase. Wenn Sie die fehlgeschlagenen Aktionen in einer Phase wiederholen, funktionieren alle Aktionen, die noch ausgeführt werden, weiterhin, und fehlgeschlagene Aktionen werden erneut ausgelöst. Wenn Sie eine fehlgeschlagene Phase von der ersten Aktion in der Phase wiederholen, können in der Phase keine Aktionen ausgeführt werden. Bevor eine Phase erneut versucht werden kann, müssen entweder alle Aktionen fehlgeschlagen sein oder einige Aktionen sind fehlgeschlagen und andere erfolgreich.

> **Important**
>
> Beim erneuten Versuch einer fehlgeschlagenen Phase werden alle Aktionen der Phase von der ersten Aktion in der Phase wiederholt, und bei einem erneuten Versuch, fehlgeschlagene Aktionen durchzuführen, werden alle fehlgeschlagenen Aktionen in der Phase wiederholt. Dadurch werden Ausgabeartefakte 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, erscheint entweder die Schaltfläche Phase wiederholen oder Fehlgeschlagene Aktionen wiederholen auf der Stufe, die erneut versucht werden kann.

Wenn Sie die AWS CLI verwenden, können Sie den get-pipeline-state Befehl verwenden, um festzustellen, ob Aktionen fehlgeschlagen sind.

> **Note**
>
> In den folgenden Fällen können Sie eine Phase möglicherweise nicht erneut versuchen:
>
> - Alle Aktionen in der Phase waren erfolgreich, sodass die Phase nicht den Status Fehlgeschlagen hat.
> - Die gesamte Pipeline-Struktur hat sich geändert, nachdem die Phase gescheitert war.
> - Es wird bereits ein anderer Wiederholungsversuch in der Phase ausgeführt.

Themen

- Versuchen Sie es erneut mit einer fehlgeschlagenen Phase (Konsole)
- Eine fehlgeschlagene Phase erneut versuchen (CLI)

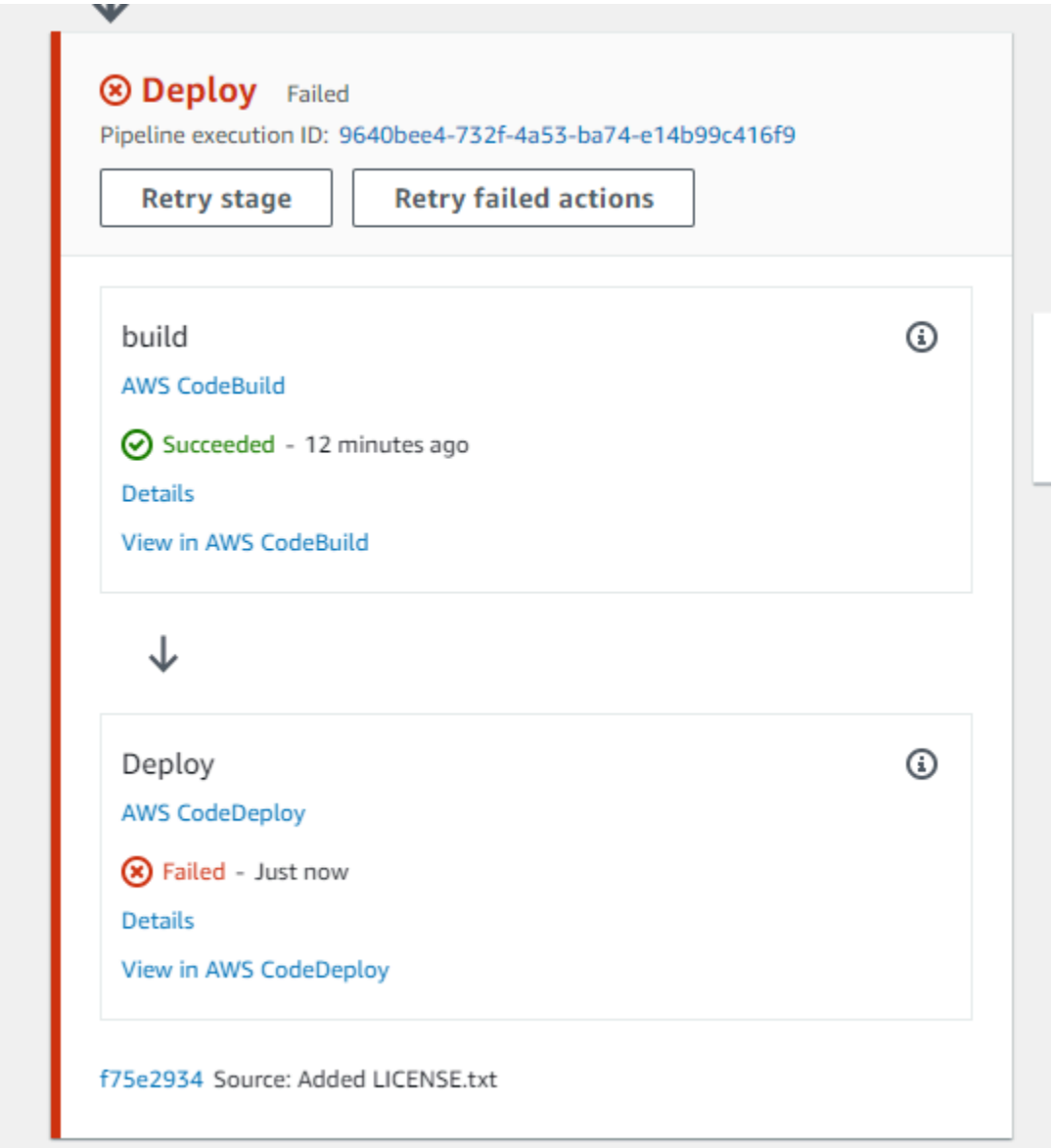
Versuchen Sie es erneut mit einer fehlgeschlagenen Phase (Konsole)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase erneut zu versuchen: Konsole

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

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline.
3. Suchen Sie die Phase mit der fehlgeschlagenen Aktion und wählen Sie dann eine der folgenden Optionen aus:

   - Um alle Aktionen in der Phase erneut zu versuchen, wählen Sie Phase wiederholen aus.
   - Um nur fehlgeschlagene Aktionen in der Phase erneut zu versuchen, wählen Sie Fehlgeschlagene Aktionen wiederholen.



Wenn alle wiederholten Aktionen in der Stufe erfolgreich abgeschlossen sind, wird die Pipeline weiter ausgeführt.

Eine fehlgeschlagene Phase erneut versuchen (CLI)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase erneut zu versuchen - CLI

Um alle Aktionen oder alle fehlgeschlagenen Aktionen erneut AWS CLI zu versuchen, führen Sie den retry-stage-execution Befehl mit den folgenden Parametern aus:

```
--pipeline-name <value>
--stage-name <value>
--pipeline-execution-id <value>
--retry-mode ALL_ACTIONS/FAILED_ACTIONS
```

> **Note**
>
> Die Werte, für die Sie verwenden können, `retry-mode` sind `FAILED_ACTIONS` und`ALL_ACTIONS`.

1. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer retry-stage-executionBefehlszeile (Windows) aus, wie im folgenden Beispiel für eine Pipeline mit dem Namen gezeigt`MyPipeline`.

   ```
   aws codepipeline retry-stage-execution --pipeline-name MyPipeline --stage-name
    Deploy --pipeline-execution-id b59babff-5f34-EXAMPLE --retry-mode FAILED_ACTIONS
   ```

   Die Ausgabe gibt die Ausführungs-ID zurück:

   ```
   {
       "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
   }
   ```

2. Sie können den Befehl auch mit einer JSON-Eingabedatei ausführen. Sie erstellen zunächst eine JSON-Datei, die die Pipeline, die Stufe mit den fehlgeschlagenen Aktionen und die letzte Pipeline-Ausführung in dieser Stufe ermittelt. Führen Sie den Befehl retry-stage-execution mit dem Parameter `--cli-input-json` aus. Am einfachsten können Sie die für die JSON-Datei benötigten Details über den Befehl get-pipeline-state abrufen.

   a. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer get-pipeline-stateBefehlszeile (Windows) in einer Pipeline aus. Für eine Pipeline mit dem Namen würden Sie MyFirstPipeline beispielsweise etwas Ähnliches wie das Folgende eingeben:

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

Die Ausgabe des Befehls umfasst die Pipeline-Statusinformationen für jede einzelne Stufe. Im folgenden Beispiel zeigt die Ausgabe, dass eine oder mehrere Aktionen in der Staging-Stufe fehlgeschlagen sind:

```
{
    "updated": 1427245911.525,
    "created": 1427245911.525,
    "pipelineVersion": 1,
    "pipelineName": "MyFirstPipeline",
    "stageStates": [
        {
            "actionStates": [...],
            "stageName": "Source",
            "latestExecution": {
                "pipelineExecutionId": "9811f7cb-7cf7-SUCCESS",
                "status": "Succeeded"
            }
        },
        {
            "actionStates": [...],
            "stageName": "Staging",
            "latestExecution": {
                "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
                "status": "Failed"
            }
        }
    ]
}
```

b. Erstellen Sie in einem Texteditor eine Datei im JSON-Format, in der Sie Folgendes festhalten:

- Der Name der Pipeline mit den fehlgeschlagenen Aktionen
- Der Name der Stufe mit den fehlgeschlagenen Aktionen
- Die ID der letzten Pipeline-Ausführung in der Stufe
- Der Retry-Modus.

Für das vorherige MyFirstPipeline Beispiel würde Ihre Datei etwa so aussehen:

```
{
```

```
    "pipelineName": "MyFirstPipeline",
    "stageName": "Staging",
    "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
    "retryMode": "FAILED_ACTIONS"
}
```

c. Speichern Sie die Datei mit einem Namen wie **`retry-failed-actions.json`**.

d. Rufen Sie die beim Ausführen des Befehls retry-stage-execution erstellte Datei auf. Beispiele:

> **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.

Arbeiten mit Aktionen in CodePipeline

In AWS CodePipeline ist eine Aktion ein 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 unterstützt sechs Arten von Maßnahmen:

- Quelle
- Entwicklung
- Test
- Bereitstellen
- Genehmigung
- Aufrufen

Informationen zu den ProduktenAWS-Service und Dienstleistungen von Partnern, die Sie je nach Aktionstyp in Ihre Pipeline integrieren können, finden Sie unterIntegrationen mit CodePipeline Aktionstypen.

Themen

- Arbeiten mit Aktionstypen
- Erstellen und Hinzufügen einer benutzerdefinierten Aktion in CodePipeline
- Markieren einer benutzerdefinierten Aktion in CodePipeline
- Aufrufen einer AWS Lambda-Funktion in einer Pipeline in CodePipeline
- Eine fehlgeschlagene Aktion in einer Phase wiederholen
- Genehmigungsaktionen verwalten in CodePipeline
- Hinzufügen einer regionsübergreifenden Aktion in CodePipeline
- Arbeiten mit Variablen

Arbeiten mit Aktionstypen

Aktionstypen sind vorkonfigurierte Aktionen, die Sie als Anbieter für Kunden erstellen, indem Sie eines der unterstützten Integrationsmodelle in verwendenAWS CodePipeline.

Sie können Aktionstypen anfordern, anzeigen und aktualisieren. Wenn der Aktionstyp für Ihr Konto als Eigentümer erstellt wurde, können Sie den verwenden,AWS CLI um die Eigenschaften und die Struktur Ihres Aktionstyps anzuzeigen oder zu aktualisieren. Wenn Sie der Anbieter oder Eigentümer des Aktionstyps sind, können Ihre Kunden die Aktion auswählen und sie zu ihren Pipelines hinzufügen, sobald sie verfügbar istCodePipeline.

> **Note**
>
> Sie erstellen Aktionen mit`custom` im`owner` Feld, um sie mit einem Jobworker auszuführen. Sie erstellen sie nicht mit einem Integrationsmodell. Hinweise zu benutzerdefinierten Aktionen finden Sie unterErstellen und Hinzufügen einer benutzerdefinierten Aktion in CodePipeline.

Komponenten des Aktionstyps

Die folgenden Komponenten bilden einen Aktionstyp.

- Aktionstyp-ID — Die ID besteht aus der Kategorie, dem Besitzer, dem Anbieter und der Version. Das folgende Beispiel zeigt eine Aktionstyp-ID mit einem Besitzer von`ThirdParty`, einer Kategorie von`Test`, einem benannten`TestProvider` Anbieter und einer Version von`1`.

```
{
    "Category": "Test",
    "Owner": "ThirdParty",
    "Provider": "TestProvider",
    "Version": "1"
},
```

- Executor-Konfiguration — Das Integrationsmodell oder die Aktions-Engine, die bei der Erstellung der Aktion angegeben wurde. Wenn Sie den Executor für einen Aktionstyp angeben, wählen Sie einen von zwei Typen aus:
  - Lambda: Der Besitzer des Aktionstyps schreibt die Integration als Lambda-Funktion, die vonCodePipeline immer dann aufgerufen wird, wenn ein Job für die Aktion verfügbar ist.
  - JobWorker: Der Besitzer des Aktionstyps schreibt die Integration als Job Worker, der nach verfügbaren Jobs in den Kunden-Pipelines abfragt. Der Job Worker führt dann den Job aus und sendet das Auftragsergebnis mithilfeCodePipeline vonCodePipeline APIs zurück.

> **Note**
>
> Das Integrationsmodell für Arbeitnehmer ist nicht das bevorzugte Integrationsmodell.

- Eingabe- und Ausgabeartefakte: Grenzwerte für die Artefakte, die der Besitzer des Aktionstyps für Kunden der Aktion festlegt.
- Berechtigungen: Die Berechtigungsstrategie, die Kunden bestimmt, die auf den Aktionstyp eines Drittanbieters zugreifen können. Die verfügbaren Berechtigungsstrategien hängen vom gewählten Integrationsmodell für den Aktionstyp ab.
- URLs: Deeplinks zu Ressourcen, mit denen der Kunde interagieren kann, z. B. die Konfigurationsseite des Besitzers des Aktionstyps.

Themen

- Beantragen Sie einen Aktionstyp
- Hinzufügen eines verfügbaren Aktionstyps zu einer Pipeline (Konsole)
- Einen Aktionstyp anzeigen
- Einen Aktionstyp aktualisieren

Beantragen Sie einen Aktionstyp

Wenn ein neuerCodePipeline Aktionstyp von einem Drittanbieter angefordert wird, wird der Aktionstyp für den Aktionstyp in erstelltCodePipeline, und der Besitzer 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 dasCodePipeline Serviceteam.

Bevor Sie Ihre Aktionsdefinitionsdatei, die Executor-Ressourcen und die Aktionstyp-Anforderung für dasCodePipeline Team erstellen, müssen Sie ein Integrationsmodell auswählen.

Schritt 1: Wählen Sie Ihr Integrationsmodell

Wählen Sie Ihr Integrationsmodell und erstellen Sie dann die Konfiguration für dieses Modell. Nachdem Sie das Integrationsmodell ausgewählt haben, müssen Sie Ihre Integrationsressourcen konfigurieren.

- Für das Lambda-Integrationsmodell erstellen Sie eine Lambda-Funktion und fügen Berechtigungen hinzu. Fügen Sie Ihrer Integrator-Lambda-Funktion Berechtigungen hinzu, um demCodePipeline Dienst Berechtigungen zu geben, ihn mithilfe desCodePipeline Dienstprinzips aufzurufen:`codepipeline.amazonaws.com`. Die Berechtigungen können mithilfe derAWS CloudFormation oder der Befehlszeile durchgeführt werden.
  - Beispiel für das Hinzufügen von Berechtigungen mitAWS 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 Integrationsmodell für Job Worker erstellen Sie eine Integration mit einer Liste zulässiger Konten, über die der Jobworker mithilfe derCodePipeline APIs nach Jobs abfragt.

Schritt 2: Erstellen einer Aktionstyp-Definitionsdatei

Sie definieren einen Aktionstyp in einer Aktionstypdefinitionsdatei mit JSON. In der Datei geben Sie die Aktionskategorie, das Integrationsmodell, das zur Verwaltung des Aktionstyps verwendet wird, und die Konfigurationseigenschaften an.

> **Note**
>
> Nachdem Sie eine öffentliche Aktion erstellt haben, können Sie die Aktionstypeigenschaft unter`properties` von`optional` bis nicht ändern`required`. Sie können das auch nicht ändern`owner`.

Weitere Informationen zu den Parametern der Definitionsdatei für Aktionstypen finden Sie unter ActionTypeDeclarationund UpdateActionTypein der CodePipelineAPI-Referenz.

Die Definitionsdatei für Aktionstypen enthält acht Abschnitte:

- `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`Lambda` oder`job worker`. Sie können nur entweder`jobWorkerExecutorConfiguration` oder angeben`lambdaExecutorConfiguration`, basierend auf Ihrem Executor-Typ.
  - `configuration`: Ressourcen für die Konfiguration des Aktionstyps, basierend auf dem gewählten Integrationsmodell. Verwenden Sie für das Lambda-Integrationsmodell die Lambda-Funktion ARN. Verwenden Sie für das Integrationsmodell für Jobworker das Konto oder die Liste der Konten, von denen aus der Jobworker arbeitet.
  - `jobTimeout`: Das Timeout für den Auftrag. Eine Aktionsausführung kann aus mehreren Jobs bestehen. Dies ist das Timeout für einen einzelnen Job und nicht für die gesamte Aktionsausführung.

    > **Note**
    >
    > Für das Lambda-Integrationsmodell beträgt das maximale Timeout 15 Minuten.

  - `policyStatementsTemplate`: Die Richtlinienerklärung, die die Berechtigungen imCodePipeline Kundenkonto angibt, die für die erfolgreiche Ausführung einer Aktion erforderlich sind.
  - `type`: Das Integrationsmodell, das zum Erstellen und Aktualisieren des Aktionstyps verwendet wurde, entweder`Lambda` oder`JobWorker`.
- `id`: Die Kategorie, der Besitzer, der Anbieter und die Versions-ID für den Aktionstyp:
  - `category`: Die Art von Aktion in den Stufen Quelle, Build, Deploy, Test, Invoke oder Approval.
  - `provider`: Der Anbieter des aufgerufenen Aktionstyps, z. B. die Anbieterfirma oder der Produktname. Der Anbietername wird bei der Erstellung des Aktionstyps angegeben.
  - `owner`: Der Ersteller des aufgerufenen Aktionstyps:`AWS` oder`ThirdParty`.
  - `version`: Eine Zeichenfolge, die verwendet wird, um den Aktionstyp zu versionieren. Stellen Sie für die erste Version die Versionsnummer auf 1 ein.
- `inputArtifactDetails`: Die Anzahl der Artefakte, die in der vorherigen Phase der Pipeline zu erwarten sind.

- `outputArtifactDetails`: Die Anzahl der Artefakte, die vom Ergebnis der Aktionstypphase zu erwarten sind.
- `permissions`: Details zur Identifizierung der Konten mit Berechtigungen zur Verwendung des Aktionstyps.
- `properties`: Die Parameter, die für die Ausführung Ihrer Projektaufgaben erforderlich sind.
  - `description`: Die Beschreibung der Eigenschaft, die Benutzern angezeigt wird.
  - `optional`: Ob die Konfigurationseigenschaft optional ist.
  - `noEcho`: Ob der vom Kunden eingegebene Feldwert im Protokoll weggelassen wird. Wenn`true`, dann wird der Wert redigiert, wenn er mit einerGetPipeline API-Anfrage zurückgegeben wird.
  - `key`: Ob die Konfigurationseigenschaft ein Schlüssel ist.
  - `queryable`: Ob die Immobilie für Umfragen genutzt 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 URLsCodePipeline 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 derCodePipeline Konsole angezeigte URL auf die Seite enthält, auf der Kunden die Konfiguration der externen Aktion aktualisieren oder ändern können.
  - `thirdPartyConfigurationUrl`: URL einer Seite, auf der sich Benutzer für einen externen Dienst anmelden und die von diesem Dienst bereitgestellte Aktion zunächst konfigurieren können.

Der folgende Code zeigt ein Beispiel für eine Aktionstypdefinition.

```
{
   "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 Ihrer Integration beiCodePipeline

Um Ihren Aktionstyp bei zu registrierenCodePipeline, wenden Sie sich mit Ihrer Anfrage an dasCodePipeline Serviceteam.

DasCodePipeline Serviceteam registriert die Integration des neuen Aktionstyps, indem es Änderungen an der Service-Codebasis vornimmt. CodePipelineregistriert zwei neue Aktionen: eine öffentliche 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 verwalten.

Um eine Anfrage für eine Lambda-Integration zu registrieren

- Senden Sie über das folgende Formular eine Anfrage an dasCodePipeline 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 einen Antrag auf Eingliederung von Arbeitnehmern zu registrieren

- Senden Sie über das folgende Formular eine Anfrage an dasCodePipeline 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: Aktiviere deine neue Integration

Wenden Sie sich an dasCodePipeline Serviceteam, wenn Sie bereit sind, die neue Integration öffentlich zu verwenden.

Hinzufügen eines verfügbaren Aktionstyps zu einer Pipeline (Konsole)

Sie fügen Ihren Aktionstyp zu einer Pipeline hinzu, damit Sie ihn testen können. Sie können dies tun, indem Sie eine neue Pipeline erstellen oder eine vorhandene bearbeiten.

> **Note**
>
> Wenn es sich bei Ihrem Aktionstyp um eine Aktion der Kategorie Quelle, 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 derCodePipeline Konsole aus zu einer vorhandenen Pipeline hinzuzufügen

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie dieCodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie in der Liste der Pipelines die Pipeline aus, der Sie den Aktionstyp hinzufügen möchten.
3. Wählen Sie auf der Seite mit der Übersichtsansicht der Pipeline die Option Bearbeiten aus.
4. Wählen Sie, ob Sie die Phase bearbeiten möchten. Wählen Sie in der Phase, in der Sie Ihren Aktionstyp hinzufügen möchten, die Option Aktionsgruppe hinzufügen aus. Die Seite „Aktion bearbeiten“ wird angezeigt.
5. Geben Sie auf der Seite Aktion bearbeiten unter 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 Artefaktnamen 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 Mengen definiert werden.
8. Wählen Sie Connect 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. Ihre Schritte variieren je nach Aktionskategorie, Website und Konfiguration, beinhalten jedoch in der Regel eine Abschlussaktion, die den Kunden zur Seite Aktion bearbeiten zurückbringt.

10. Auf der Seite AktionCodePipeline bearbeiten werden die zusätzlichen Konfigurationsfelder für die Aktion angezeigt. Die angezeigten Felder sind 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 Aktionsseite Bearbeiten für Ihre Aktion ein Feld mit der Bezeichnung Host angezeigt.

11. Geben Sie im Feld Ausgabeartefakte den Artefaktnamen 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 Mengen 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, nehmen Sie eine Änderung an der Quelle vor, die in der Quellphase der Pipeline angegeben ist, oder folgen Sie den Schritten unter Manuelles Starten einer Pipeline.

Um eine Pipeline mit Ihrem Aktionstyp zu erstellen, folgen Sie den Schritten unterErstellen einer Pipeline in CodePipeline und wählen Sie Ihren Aktionstyp aus beliebig vielen Stufen aus, die Sie testen möchten.

Einen Aktionstyp anzeigen

Sie können die CLI verwenden, um Ihren Aktionstyp anzuzeigen. Verwenden Sie denget-action-type Befehl, um Aktionstypen anzuzeigen, die mithilfe eines Integrationsmodells erstellt wurden.

Um einen Aktionstyp anzuzeigen

1. Erstellen Sie eine JSON-Eingabedatei und geben Sie der Datei einen Namen`file.json`. Fügen Sie Ihre Aktionstyp-ID im JSON-Format hinzu, wie im folgenden Beispiel gezeigt.

```
{
    "category": "Test",
    "owner": "ThirdParty",
    "provider": "TestProvider",
    "version": "1"
}
```

2. Führen Sie den Befehl in einem Terminalfenster oder über dieget-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": []
    }
}
```

Einen Aktionstyp aktualisieren

Sie können die CLI verwenden, um Aktionstypen zu bearbeiten, die mit einem Integrationsmodell erstellt wurden.

Für einen öffentlichen Aktionstyp können Sie den Besitzer nicht aktualisieren, Sie können optionale Eigenschaften nicht in erforderlich ändern und Sie können nur neue optionale Eigenschaften hinzufügen.

1. Verwenden Sie den`get-action-type` Befehl, um die Struktur für Ihren Aktionstyp abzurufen. Kopieren Sie die Struktur.
2. Erstellen Sie eine JSON-Eingabedatei und geben Sie ihr einen Namen`action.json`. Fügen Sie die Aktionstypstruktur ein, die Sie im vorherigen Schritt kopiert haben. Aktualisieren Sie alle Parameter, die Sie ändern möchten. Sie können auch optionale Parameter hinzufügen.

   Weitere Informationen zu den Parametern für die Eingabedatei finden Sie in der Aktionsdefinitionsdatei unterSchritt 2: Erstellen einer Aktionstyp-Definitionsdatei.

   Das folgende Beispiel veranschaulicht, wie Sie einen mit dem Lambda-Integrationsmodell erstellten Aktionstyp aktualisieren. In diesem Beispiel werden die folgenden Änderungen durchgeführt:

   - Ändert den`provider` Namen in`TestProvider1`.
   - Fügen Sie ein Job-Timeout-Limit von 900 Sekunden hinzu.
   - Fügt eine Aktionskonfigurationseigenschaft mit dem Namen`Host` hinzu, die dem Kunden mithilfe der Aktion angezeigt wird.

   ```
   {
       "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 über die Befehlszeile denupdate-action-type Befehl aus

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

Dieser Befehl gibt die Ausgabe des Aktionstyps zurück, die Ihren aktualisierten Parametern entspricht.

Erstellen und Hinzufügen einer benutzerdefinierten Aktion in CodePipeline

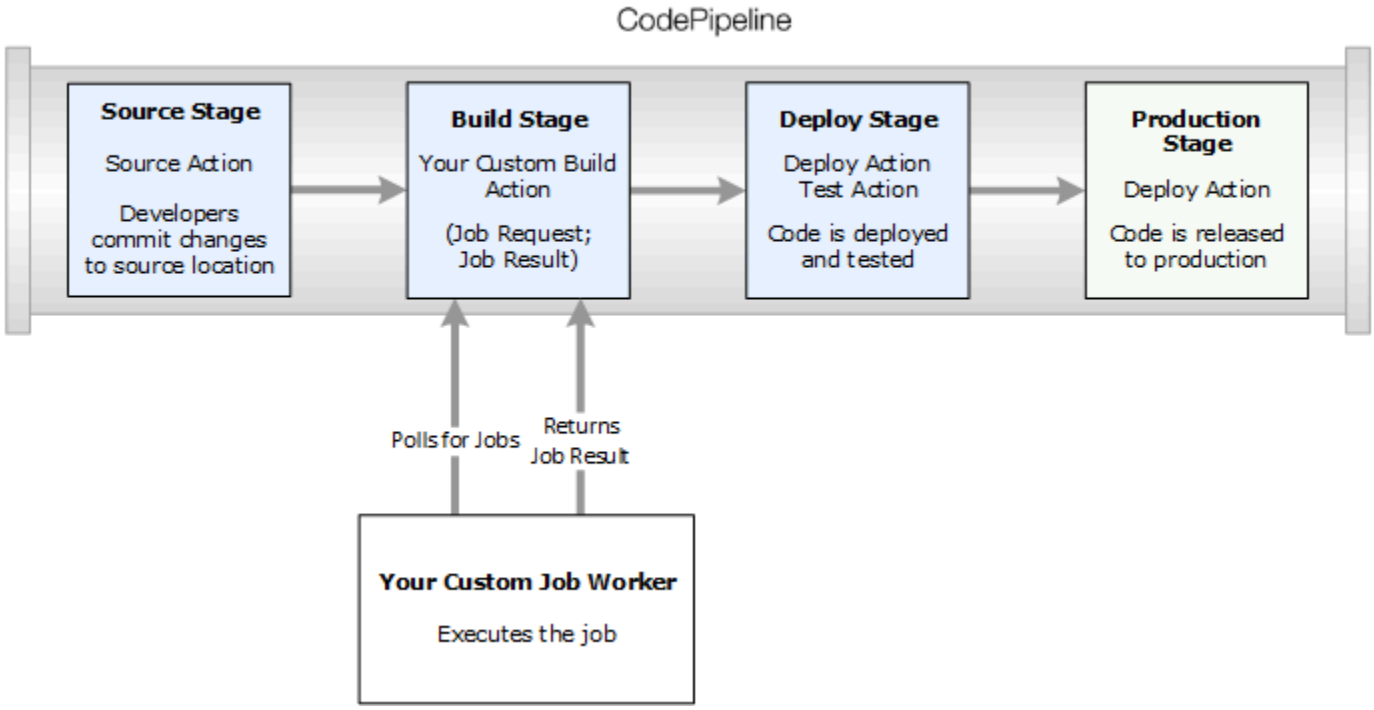
AWS CodePipeline enthält eine Reihe von Aktionen, die Ihnen beim Konfigurieren von Entwicklungs-, Test- und Bereitstellungsressourcen für Ihr automatisiertes Freigabeverfahren helfen. Falls Ihr Freigabeverfahren Aktivitäten beinhaltet, die nicht durch die Standardaktionen abgedeckt sind (wie z. B. ein intern entwickelter Erstellungsprozess oder eine Testsuite), können Sie dafür eine benutzerdefinierte Aktion erstellen und diese Ihrer Pipeline hinzufügen. Sie können die AWS CLI verwenden, um benutzerdefinierte Aktionen in Pipelines zu erstellen, die mit Ihrem AWS-Konto verknüpft sind.

Sie können benutzerdefinierte Aktionen für die folgenden AWS CodePipeline-Aktionskategorien erstellen:

- Eine benutzerdefinierte Build-Aktionen für die Erstellung oder Umwandlung von Elementen
- Eine benutzerdefinierte Deploy-Aktion zur Bereitstellung der Elemente für eine(n) oder mehrere Server, Webseiten oder Repositorys
- Eine benutzerdefinierte Testaktion für die Konfiguration und Ausführung von automatisierten Tests
- Eine benutzerdefinierte Aufrufaktionen für das Ausführen von Funktionen

Wenn Sie eine benutzerdefinierte Aktion erstellen, müssen Sie auch einen Job Worker erstellen, derCodePipeline nach Stellenanfragen für diese benutzerdefinierte Aktion fragt, den Job ausführt und das Statusergebnis an zurückgibtCodePipeline. 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 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 Erfolgsergebnis eingeht, stellt die Pipeline die Revision und ihre Artefakte für die nächste Aktion bereit. Wenn ein Fehler zurückgegeben wird, stellt die Pipeline die Revision für die nächste Aktion in der Pipeline nicht 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

So erstellen Sie eine benutzerdefinierte Aktion mit der AWS CLI

1. Öffnen Sie einen Texteditor und erstellen Sie eine JSON-Datei mit der Aktion, dem Aktionsanbieter und allen erforderlichen Einstellungen für Ihre benutzerdefinierte Aktion. Ihre JSON-Datei für eine benutzerdefinierte Aktion mit nur einer Eigenschaft könnte beispielsweise wie folgt aussehen:

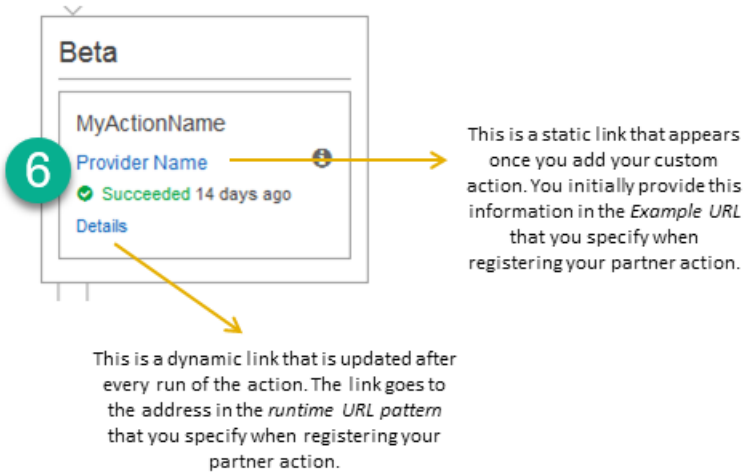
```
{
    "category": "Build",
    "provider": "My-Build-Provider-Name",
    "version": "1",
    "settings": {
        "entityUrlTemplate": "https://my-build-instance/job/{Config:ProjectName}/",
        "executionUrlTemplate": "https://my-build-instance/job/
{Config:ProjectName}/lastSuccessfulBuild/{ExternalExecutionId}/"
    },
    "configurationProperties": [{
        "name": "ProjectName",
        "required": true,
        "key": true,
        "secret": false,
        "queryable": false,
        "description": "The name of the build project must be provided when this
 action is added to the pipeline.",
        "type": "String"
    }],
    "inputArtifactDetails": {
        "maximumCount": integer,
        "minimumCount": integer
    },
    "outputArtifactDetails": {
        "maximumCount": integer,
        "minimumCount": integer
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
    }]
}
```

In diesem Beispiel wird der benutzerdefinierten Aktion Tagging hinzugefügt, indem der benutzerdefinierten Aktion der Tag-Schlüssel `Project` und der Wert `ProjectA` hinzugefügt werden. Weitere Informationen zum Markieren von Ressourcen in CodePipeline finden Sie unter Markieren von Ressourcen.

Die JSON-Datei enthält zwei Eigenschaften, `entityUrlTemplate` und `executionUrlTemplate`. Solange die Konfigurationseigenschaft erforderlich und nicht geheim ist, können Sie über das Format `{Config:name}` in den URL-Vorlagen auf einen Namen in den Konfigurationseigenschaften der benutzerdefinierten Aktion verweisen. Im obigen Beispiel bezieht sich der`entityUrlTemplate` Wert beispielsweise auf die Konfigurationseigenschaft *`ProjectName`*.

- `entityUrlTemplate`: Der statische Link mit Informationen über die Dienstanbieter für die Aktion. Im Beispiel umfasst das Build-System einen statischen Link zu den einzelnen Build-Projekten. Das Linkformat variiert je nach Build-Anbieter oder Dienstanbieter (beispielsweise bei anderen Aktionstypen wie Test). Sie müssen dieses Linkformat so angeben, dass der Benutzer beim Hinzufügen der benutzerdefinierten Aktion mit dem Link eine Webseite im Browser öffnen kann, die die Einzelheiten für das Build-Projekt (oder die Testumgebung) enthält.
- `executionUrlTemplate`: Der dynamische Link, der mit Informationen zur aktuellen oder letzten Ausführung der Aktion aktualisiert wird. Wenn Ihr benutzerdefinierter Auftragsworker den Status eines Auftrags aktualisiert (z. B. bei Erfolg, Fehler oder während der Ausführung), stellt er zusätzlich eine `externalExecutionId` bereit, die zur Vervollständigung des Links verwendet wird. Dieser Link kann verwendet werden, um Details zum Ausführen einer Aktion bereitzustellen.

Wenn Sie die Aktion beispielsweise in der Pipeline anzeigen, sehen Sie die beiden folgenden Links:



1

Dieser statische Link wird angezeigt, nachdem Sie Ihre benutzerdefinierte Aktion hinzugefügt haben. Er verweist auf die Adresse in `entityUrlTemplate`, die Sie bei der Erstellung Ihrer benutzerdefinierten Aktion angegeben haben.

2

Dieser dynamische Link wird nach jeder Ausführung der Aktion aktualisiert und verweist auf die Adresse in `executionUrlTemplate`, die Sie bei der Erstellung Ihrer benutzerdefinierten Aktion angegeben haben.

Weitere Informationen zu diesen Linktypen sowie zu`RevisionURLTemplate` und`ThirdPartyURL` finden Sie unter ActionTypeSettingsund CreateCustomActionTypein der CodePipelineAPI-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. Führen Sie den Befehl aws codepipeline create-custom-action-type mithilfe der AWS CLI unter Angabe des Namens der gerade erstellten JSON-Datei aus.

Um beispielsweise eine benutzerdefinierte Build-Aktion zu erstellen, gehen Sie wie folgt vor:

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

```
aws codepipeline create-custom-action-type --cli-input-json
 file://MyCustomAction.json
```

5. Dieser Befehl gibt die gesamte Struktur der erstellten, benutzerdefinierten Aktion sowie die für Sie hinzugefügte `JobList`-Aktionskonfigurationseigenschaft zurück. Wenn Sie die benutzerdefinierte Aktion zu einer Pipeline hinzufügen, können Sie `JobList` nutzen, um die Projekte vom Anbieter anzugeben, für die Sie Aufträge abfragen können. Ohne entsprechende Konfiguration werden bei der Abfrage von Aufträgen durch Ihren benutzerdefinierten Auftragsworker alle verfügbaren Aufträge zurückgegeben.

   Der vorherige Befehl könnte beispielsweise eine Struktur ähnlich der Folgenden zurückgeben:

```
{
    "actionType": {
        "inputArtifactDetails": {
            "maximumCount": 1,
            "minimumCount": 1
       },
       "actionConfigurationProperties": [
            {
                "secret": false,
                "required": true,
                "name": "ProjectName",
                "key": true,
                "description": "The name of the build project must be provided when
 this action is added to the pipeline."
            }
        ],
        "outputArtifactDetails": {
            "maximumCount": 0,
            "minimumCount": 0
        },
```

```
        "id": {
            "category": "Build",
            "owner": "Custom",
            "version": "1",
            "provider": "My-Build-Provider-Name"
        },
        "settings": {
            "entityUrlTemplate": "https://my-build-instance/job/
{Config:ProjectName}/",
            "executionUrlTemplate": "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild/{ExternalExecutionId}/"
        }
    }
}
```

> **Note**
>
> Als Teil der Ausgabe descreate-custom-action-type Befehls enthält derid Abschnitt`"owner": "Custom"`. CodePipeline weist benutzerdefinierten 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 Jobworker erforderlich, derCodePipeline nach Stellenanfragen für die benutzerdefinierte Aktion fragt, den Job ausführt und das Statusergebnis an zurückgibtCodePipeline. 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 Auftragsworker für Ihre benutzerdefinierte Aktion in CodePipeline zu entwickeln, brauchen Sie eine Strategie für die Integration der Benutzer- und Berechtigungsverwaltung.

Die einfachste Strategie besteht darin, die Infrastruktur hinzuzufügen, die Sie für Ihren Custom Job Worker benötigen, indem Sie Amazon EC2 EC2-Instances mit einer IAM-Instance-Rolle erstellen, sodass Sie die Ressourcen, die Sie für Ihre Integration benötigen, einfach skalieren können. Sie können die werkseitige Integration mit AWS verwenden, um die Interaktion zwischen dem benutzerdefinierten Auftragsworker und CodePipeline zu vereinfachen.

So Sie Amazon EC2 Instances Amazon EC2

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 EC2-Instances. Informationen unter Erste Schritte

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 dem Identitätsverbund können Sie sicheren Zugriff aufAWS Ressourcen gewährenCodePipeline, 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. Informationen zu IIIIIIIIIIIIII Weitere Informationen finden Sie unter Manage Federation.

2. Sehen Sie sich die Beispiele in Scenarios for Granting Temporary Access an, um das Szenario für den temporären Zugriff zu ermitteln, das am besten für Ihre benutzerdefinierte Aktion geeignet ist.
3. Sehen Sie sich die für Ihre Infrastruktur relevanten Codebeispiele zum Identitätsverbund an. Diese sind beispielsweise:

   - Identity Federation Sample Application for an Active Directory Use Case
4. Erste Schritte bei der Konfiguration eines Identitätsverbunds. Weitere Informationen finden Sie unter Identity Providers and Federation im IAM-Benutzerhandbuch.

Erstellen Sie eine der folgenden Optionen, um sie unter Ihrem zu verwenden,AWS-Konto wenn Sie Ihre benutzerdefinierte Aktion und Ihren Job Worker ausführen.

Benutzer benötigen programmgesteuerten Zugriff, wenn sie außerhalb der AWS Management Console mit AWS interagieren möchten. Die Vorgehensweise, um programmgesteuerten Zugriff zu gewähren, hängt davon ab, welcher Benutzertyp auf AWS zugreift:

Wählen Sie eine der folgenden Optionen aus, um den Benutzern programmgesteuerten Zugriff zu gewähren.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anfragen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Konfigurieren der AWS CLI für die Verwendung von AWS IAM Identity Center im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs, Tools und AWS-APIs finden Sie unter IAM-Ident

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		ity-Center-Authentifizierung im Referenzhandbuch zu AWS-SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anfragen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmgesteuerte Anfragen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Authentifizierung mit IAM-Benutzer-Anmeldeinformationen im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs und Tools finden Sie unter Authentifizierung mit langfristigen Anmeldeinformationen im Referenzhandbuch zu AWS-SDKs und Tools. • Informationen zu AWS-APIs finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

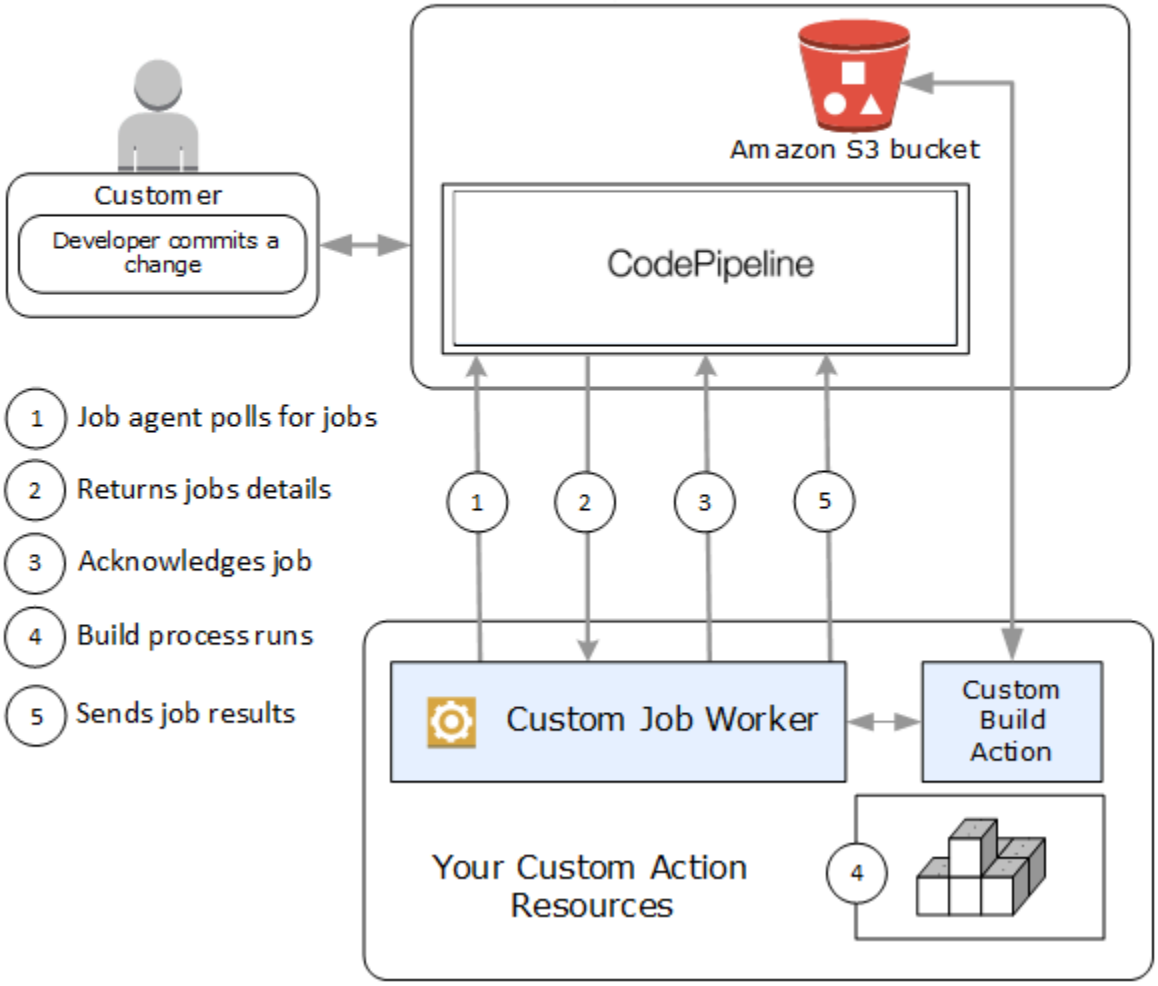
Es folgt ein Beispiel für eine Richtlinie, die Sie für die Verwendung mit Ihrem benutzerdefinierten Auftragsworker erstellen können. Diese Richtlinie dient lediglich als Beispiel und wird unverändert bereitgestellt.

```
{
  "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 Berechtigungsverwaltung gewählt haben, sollten Sie überlegen, wie Ihr Auftragsworker mit CodePipeline interagiert. Das folgende allgemeine Diagramm zeigt den Arbeitsablauf einer benutzerdefinierten Aktion und eines Jobworkers für einen Build-Prozess.



1. Ihr Auftragsworker fragt in CodePipeline Aufträge mit `PollForJobs` ab.
2. Wenn eine Pipeline durch eine Veränderung in der Quellphase ausgelöst wird (wenn beispielsweise ein Entwickler eine Änderung vornimmt), wird das automatisierte Freigabeverfahren gestartet. Der Vorgang wird fortgesetzt, bis Ihre benutzerdefinierte Aktion konfiguriert wurde. Wenn er in dieser Phase Ihre Aktion erreicht, stellt CodePipeline einen Auftrag in die Warteschlange. 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 Jobarbeiter ruft an`AcknowledgeJob`, umCodePipeline eine Auftragsbestätigung zu senden. CodePipelinegibt eine Bestätigung zurück, die angibt, dass der Jobworker den Job fortsetzen soll (`InProgress`), oder, falls mehr als ein Jobworker nach Jobs abfragt und ein anderer Jobworker

die Stelle bereits in Anspruch genommen hat, wird eine`InvalidNonceException` Fehlerantwort zurückgegeben. Nach der Bestätigung `InProgress` wartet CodePipeline auf die Ergebnisse.

4. Der Job Worker initiiert Ihre benutzerdefinierte Aktion für die Revision, und dann wird Ihre Aktion ausgeführt. Zusammen mit allen anderen Aktionen gibt Ihre benutzerdefinierte Aktion ein Ergebnis an den Job Worker zurück. Im Beispiel einer benutzerdefinierten Build-Aktion ruft die Aktion Artefakte aus dem Amazon S3 S3-Bucket ab, erstellt sie und schiebt erfolgreich erstellte Artefakte zurück in den Amazon S3 S3-Bucket.
5. Während die Aktion ausgeführt wird, kann der Job Worker`PutJobSuccessResult` mit einem Fortsetzungstoken (der Serialisierung des vom Job-Worker generierten Status des Jobs, z. B. einer Build-ID im JSON-Format oder einem Amazon S3 S3-Objektschlüssel) sowie den`ExternalExecutionId` Informationen, die zum Füllen des Links verwendet werden, aufrufen`executionUrlTemplate`. Dadurch wird die Konsolenansicht der Pipeline während des Vorgangs mit einem funktionierenden Link zu spezifischen Aktionsdetails versehen. Dieses Vorgehen ist zwar nicht erforderlich, jedoch eine bewährte Methode, da sie Benutzern das Abrufen des Status der benutzerdefinierten Aktion ermöglicht, während diese ausgeführt wird.

   Sobald `PutJobSuccessResult` aufgerufen wird, gilt der Auftrag als abgeschlossen. In CodePipeline wird ein neuer Auftrag erstellt, bei dem das Fortsetzungstoken miteinbezogen wird. 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. Als letzter Schritt wird eine Erfolgs- oder Fehlermeldung an CodePipeline gesendet, ohne dass ein Fortsetzungstoken zur Anwendung kommt.

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 Job Worker 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:

- Abfragen von Aufträgen in CodePipeline Aufträge mit `PollForJobs`
- Bestätigen von Aufträgen und Zurückgeben von Ergebnissen an CodePipeline mit `AcknowledgeJob`, `PutJobSuccessResult` und `PutJobFailureResult`.
- Abrufen von Artefakten aus dem Amazon S3-Bucket und/oder Einfügen von Artefakten in den Amazon S3 S3-Bucket für die Pipeline. Um Artefakte aus dem Amazon S3 S3-Bucket herunterzuladen, müssen Sie einen Amazon S3 S3-Client erstellen, der Signature Version 4-Signatur (Sig V4) verwendet. Sig V4 ist erforderlich fürAWS KMS.

  Um Artefakte in den Amazon S3 S3-Bucket hochzuladen, müssen Sie die Amazon S3`PutObject` S3-Anfrage zusätzlich so konfigurieren, dass sie Verschlüsselung verwendet. Derzeit wird nurAWS Key Management Service (AWS KMS) für die Verschlüsselung unterstützt. AWS KMSNutzungenAWS KMS keys. Um zu wissen, ob ein vom Kunden verwalteter SchlüsselVon AWS verwalteter Schlüssel oder ein vom Kunden verwalteter Schlüssel zum Hochladen von Artefakten verwendet werden soll, muss Ihr Custom Job Worker sich 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 verwendenAWS

KMS. Wenn die Schlüsseleigenschaft Null ist, verwenden Sie dieVon AWS verwalteter Schlüssel. CodePipelineverwendet die,Von AWS verwalteter Schlüssel sofern nicht anders konfiguriert.

Ein Beispiel, das zeigt, wie dieAWS KMS Parameter in Java oder.NET erstellt werden, finden Sie unter Angeben derAWS Key Management Service in Amazon S3 mithilfe derAWS SDKs. Weitere Informationen zum Amazon S3 S3-Bucket fürCodePipeline finden Sie unterCodePipeline - Konzepte.

Ein komplexeres Beispiel für einen Custom Job Worker finden Sie unterGitHub. Dieses Beispiel ist ein Open Source-Beispiel und wird unverändert bereitgestellt.

- Beispiel für einen Job Worker fürCodePipeline: Laden Sie das Beispiel aus demGitHub 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 diese auswählen, wenn Sie den Assistenten „Pipeline erstellen“ verwenden, indem Sie eine vorhandene Pipeline bearbeiten und die benutzerdefinierte Aktion hinzufügen oder indem Sie dieAWS CLI, die SDKs oder die APIs verwenden.

> **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 verwendenAWS CLI, um einer vorhandenen Pipeline eine benutzerdefinierte Aktion hinzuzufügen.

1. Öffnen Sie eine Eingabeaufforderung (Windows) und verwenden Sie die Eingabeaufforderung (Windows) und verwenden Sie die Eingabeaufforderung (Windows) und verwenden Sie dieget-pipeline Eingabeaufforderung (Windows) und verwenden Sie die Eingabeaufforderung (Windows) und verwenden Sie die Eingabeaufforderung (Windows) und verwenden Sie den 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 soeben 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.

Markieren einer benutzerdefinierten Aktion in CodePipeline

Tags sind mit AWS-Ressourcen verknüpfte Schlüssel-Wert-Paare. Sie können die Konsole oder die CLI verwenden, um auf Ihre benutzerdefinierten Aktionen in CodePipeline Tags anzuwenden. Weitere Informationen zum CodePipeline-Ressourcen-Tagging, Anwendungsfälle, Einschränkungen für Tag-Schlüssel und -Werte sowie unterstützte Ressourcentypen finden Sie unterMarkieren von Ressourcenaus.

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

Führen Sie die folgenden Schritte aus, um mithilfe der AWS CLI einer benutzerdefinierten Aktion ein Tag hinzuzufügen. Informationen zum Hinzufügen eines Tags zu einer benutzerdefinierten Aktion während der Erstellung finden Sie unter Erstellen und Hinzufügen einer benutzerdefinierten Aktion 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 markieren, einen Tag-Schlüssel mit dem Namen *TestActionType* und dem Tag-Wert *UnitTest*, und einen Tag-Schlüssel mit dem Namen *ApplicationName* und dem Tag-Wert *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

Führen Sie die folgenden Schritte aus, um mithilfe der AWS CLI 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

Führen Sie die folgenden Schritte aus, um mithilfe der AWS CLI ein Tag für eine benutzerdefinierte Aktion zu bearbeiten. Sie können den Wert für einen vorhandenen Schlüssel ändern oder einen anderen Schlüssel hinzufügen. Sie können Tags auch aus einer benutzerdefinierten Aktion entfernen wie im nächsten Abschnitt gezeigt.

Führen Sie im Terminal oder in der Befehlszeile den Befehl tag-resource aus und geben Sie dabei den Amazon-Ressourcennamen (ARN) der benutzerdefinierten Aktion, für die Sie ein Tag aktualisieren möchten, sowie den Schlüssel und den Wert des Tags an, das Sie aktualisieren möchten:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version --tags
 key=TestActionType,value=IntegrationTest
```

Entfernen von Tags aus einer benutzerdefinierten Aktion

Führen Sie die folgenden Schritte aus, um mithilfe der AWS CLI ein Tag aus einer benutzerdefinierten Aktion zu entfernen. Wenn Sie Tags aus der zugehörigen Ressource entfernen, werden die Tags gelöscht.

> **Note**
>
> Wenn Sie eine benutzerdefinierte Aktion löschen, werden alle Tag-Zuordnungen aus der gelöschten benutzerdefinierten Aktion entfernt. Sie müssen Tags nicht entfernen, bevor Sie eine benutzerdefinierte Aktion löschen.

Führen Sie im Terminal oder in der Befehlszeile den Befehl untag-resource aus und geben Sie dabei den Amazon-Ressourcennamen (ARN) der benutzerdefinierten Aktion, aus der Sie ein Tag entfernen möchten, sowie den Schlüssel und den Wert des Tags an, das Sie entfernen möchten:

Um beispielsweise ein Tag aus einer benutzerdefinierten Aktion zu löschen, das den Tag-Schlüssel *TestActionType* hat:

```
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.

Aufrufen einer 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, dasCodePipeline an Lambda gesendet wird, da dies dazu führen kann, dass Benutzeranmeldeinformationen inCloudWatch Logs protokolliert werden. DieCodePipeline Rolle verwendet ein JSON-Ereignis, um temporäre Anmeldeinformationen an Lambda im`artifactCredentials` Feld zu übergeben. Ein Beispiel für ein Ereignis finden Sie unter JSON-Beispielereignis.

Hier sind einige Möglichkeiten, wie Lambda-Funktionen in Pipelines verwendet werden können:

- Bedarfsabhängiges Erstellen von Ressourcen in einer Phase einer Pipeline mithilfe von AWS CloudFormation und zum Löschen in einer anderen Phase
- Um Anwendungsversionen ohne Ausfallzeiten bereitzustellen, verwenden Sie eine Lambda-Funktion, die CNAME-Werte austauscht.AWS Elastic Beanstalk
- 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**
>
> Durch das Erstellen und Ausführen von Lambda-Funktionen können Kosten für IhrAWS Konto anfallen. Weitere Informationen finden Sie unter -Preisgestaltung.

In diesem Thema wird vorausgesetzt, dass Sie mit Pipelines, FunktionenAWS CodePipelineAWS 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 dieCodePipeline und Lambda-Ausführungsrollen sowie die Berechtigungen, die erforderlich sind, um die Funktion als Teil der Pipeline auszuführen.
- 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 verwendenCodePipeline, PutJobFailureResultsollte der Status der Lambda-Ausführung mit dem PutJobSuccessResultund an dieAWS Region gesendet werden, in der die Lambda-Funktion vorhanden ist, und nicht an die Region, in der sieCodePipeline vorhanden ist.

Dieses Thema enthält Beispielfunktionen, um die Flexibilität der Arbeit mit Lambda-Funktionen in folgenden Bereichen zu demonstrierenCodePipeline:

- Basic Lambda function
  - Erstellen einer grundlegenden Lambda-Funktion zur VerwendungCodePipeline.
  - Zurücksenden von Erfolgs- oder Fehlerergebnissen an CodePipeline im Link Details der Aktion.
- Beispiel einer Python-Funktion, die eine AWS CloudFormation-Vorlage verwendet
  - Verwenden von JSON-verschlüsselten Benutzerparametern für die Weiterleitung mehrerer Konfigurationswerte an die Funktion (`get_user_params`)
  - Interaktion mit ZIP-Artefakten in einem Artefakt-Bucket (`get_template`)

- Verwenden eines Fortsetzungstokens, um einen lange andauernden asynchronen Prozess zu überwachen (`continue_job_later`). Dadurch kann die Aktion fortgesetzt werden und die Funktion kann erfolgreich sein, auch wenn sie eine Laufzeit von fünfzehn Minuten überschreitet (ein Limit in Lambda).

Jede Beispielfunktion beinhaltet Informationen zu den Berechtigungen, die Sie der Rolle hinzufügen müssen. Informationen zu Einschränkungen in AWS Lambda finden Sie unter Einschränkungen im AWS Lambda-Entwicklerhandbuch.

> **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 einer Lambda-Funktion
- Schritt 3: Fügen Sie die Lambda-Funktion zu einer Pipeline in derCodePipeline Konsole hinzu
- Schritt 4: Testen der 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 Schritten unterTutorial: Erstellen einer einfachen Pipeline (S3-Bucket), um einen Amazon S3 S3-Bucket,CodeDeploy Ressourcen und eine zweistufige Pipeline zu erstellen. Wählen Sie die Amazon Linux-Option für Ihre Instanztypen. Sie können einen

beliebigen Namen für die Pipeline verwenden. Die Schritte in diesem Thema verwenden den Namen MyLambdaTestPipeline.

2. Wählen Sie auf der Statusseite für Ihre 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 EC2-Konsole auf der Registerkarte Details für die Instance die IP-Adresse in das Feld Öffentliche IPv4-Adresse (z. B.**`192.0.2.4`**). Sie verwenden diese Adresse als Ziel der Funktion in AWS Lambda.

> **Note**
>
> Die standardmäßige Servicerollenrichtlinie fürCodePipeline 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 einer Lambda-Funktion

In diesem Schritt erstellen Sie eine Lambda-Funktion, die eine HTTP-Anfrage stellt und auf einer Webseite nach einer Textzeile 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 AWS Management Console an, und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies aus und wählen Sie dann Create Policy aus. Wählen Sie die Registerkarte JSON aus und kopieren Sie dann die folgende JSON-Richtlinie in das Feld.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
```

```
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Action": [
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
        ],
        "Effect": "Allow",
        "Resource": "*"
     }
  ]
}
```

3. Wählen Sie Review policy (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 Create Policy (Richtlinie erstellen) aus.

   > **Note**
   >
   > Dies sind die Mindestberechtigungen, die erforderlich sind, damit eine Lambda-Funktion mitCodePipeline Amazon interagieren kannCloudWatch. Wenn Sie diese Richtlinie erweitern möchten, um Funktionen zuzulassen, die mit anderenAWS Ressourcen interagieren, sollten Sie diese Richtlinie ändern, um die für diese Lambda-Funktionen erforderlichen Aktionen zuzulassen.

5. Wählen Sie auf der Seite mit dem Richtlinien-Dashboard Roles (Rollen) aus und wählen Sie dann Create role (Rolle erstellen) aus.
6. Wählen Sie auf der Seite „Rolle erstellen“ AWS-Service. 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).

So erstellen Sie die Lambda-Beispielfunktion, die mit verwendet werden sollCodePipeline

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Lambda-Konsole an https://console.aws.amazon.com/lambda.
2. Klicken Sie auf der Seite Functions (Funktionen) auf Create function (Funktion erstellen).

> **Note**
>
> Wenn Sie anstelle der Lambda-Seite eine Willkommensseite sehen, wählen Sie Get Started Now.

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 Node.js 14.x aus.
4. Wählen Sie für Role (Rolle) die Option Choose an existing role (Eine vorhandene Rolle wählen) aus. Wählen Sie Ihre Rolle unter Existing role (Vorhandene Rolle) aus und klicken Sie dann auf Create function (Funktion erstellen).

   Die Detailseite für Ihre erstellte Funktion wird geöffnet.
5. Fügen Sie den folgenden Code in das Feld Function code (Funktionscode) ein:

> **Note**
>
> Das Ereignisobjekt unter demCodePipeline Schlüssel.job enthält die Jobdetails. Ein vollständiges Beispiel für dieCodePipeline Rückkehr des JSON-Ereignisses zu Lambda finden Sie unterJSON-Beispielereignis.

```
var assert = require('assert');
var AWS = require('aws-sdk');
var http = require('http');

exports.handler = function(event, context) {
```

```
    var codepipeline = new AWS.CodePipeline();

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

    // Retrieve the value of UserParameters from the Lambda action configuration in
CodePipeline, in this case a URL which will be
    // health checked by this function.
    var url =
event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;

    // Notify CodePipeline of a successful job
    var putJobSuccess = function(message) {
        var params = {
            jobId: jobId
        };
        codepipeline.putJobSuccessResult(params, function(err, data) {
            if(err) {
                context.fail(err);
            } else {
                context.succeed(message);
            }
        });
    };

    // Notify CodePipeline of a failed job
    var putJobFailure = function(message) {
        var params = {
            jobId: jobId,
            failureDetails: {
                message: JSON.stringify(message),
                type: 'JobFailed',
                externalExecutionId: context.awsRequestId
            }
        };
        codepipeline.putJobFailureResult(params, function(err, data) {
            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
    var 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 derCodePipeline Konsole hinzu

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 anAWS Management Console und öffnen Sie dieCodePipeline 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 Edit (Bearbeiten) die Option + Add stage (Stufe hinzufügen) zum Hinzufügen einer Stufe nach der Bereitstellungsstufe mit der CodeDeploy-Aktion aus. Geben Sie einen Namen für die Stufe ein (z. B. **`LambdaStage`**) und wählen Sie Add stage (Stufe hinzufügen).

> **Note**
>
> Sie haben auch die Möglichkeit, Ihre Lambda-Aktion einer vorhandenen Phase hinzuzufügen. Zu Demonstrationszwecken fügen wir die Lambda-Funktion als einzige Aktion in einer Phase hinzu, damit Sie ihren Fortschritt einfach verfolgen können, wenn Artefakte eine Pipeline durchlaufen.

5. Wählen Sie + Add action group (Aktionsgruppe hinzufügen). Geben Sie unter Aktion bearbeiten unter Aktionsname einen Namen für Ihre Lambda-Aktion ein (z. B.**`MyLambdaAction`**). Wählen Sie in Provider (Anbieter) AWS Lambda aus. Wählen oder geben Sie im Feld Funktionsname den Namen Ihrer Lambda-Funktion ein (z. B.**`MyLambdaFunctionForAWSCodePipeline`**). Geben Sie unter Benutzerparameter die IP-Adresse für die Amazon EC2 EC2-Instance an, die Sie zuvor kopiert haben (z. B.**`http://192.0.2.4`**), und wählen Sie dann Fertig.

   > **Note**
   >
   > Dieses Thema verwendet eine IP-Adresse. In echten Szenarien sollten Sie jedoch Ihren registrierten Website-Namen verwenden (z. B. **`http://www.example.com`**). Weitere Informationen zu Ereignisdaten und -handlern in AWS Lambda finden Sie unter Programmiermodell im AWS LambdaEntwicklerhandbuch.

6. Wählen Sie auf der Seite Edit action (Aktion bearbeiten) die Option Save (Speichern).

Schritt 4: Testen der Pipeline mit der Lambda-Funktion

Um die Funktion zu testen, führen Sie die aktuelleste Änderung über die Pipeline aus.

So verwenden Sie die Konsole zum Ausführen der letzten Version eines Artefaktes in einer Pipeline

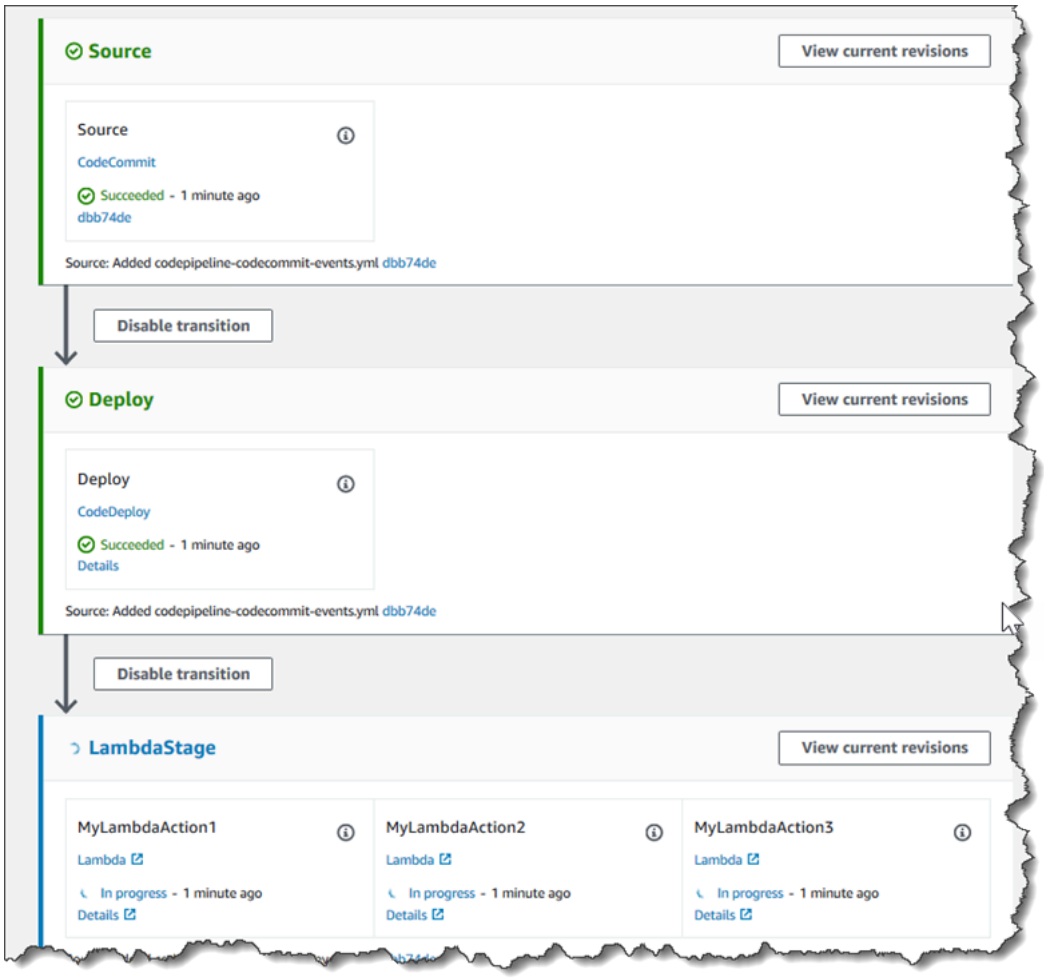
1. Wählen Sie auf der Seite mit den Pipeline-Details die Option Änderung freigeben aus. Dadurch wird die letzte Revision gestartet, die in jedem in einer Quellaktion der Pipeline angegebenen Quellspeicherort vorhanden ist.
2. Wenn die Lambda-Aktion abgeschlossen ist, wählen Sie den Link Details, um den Log-Stream für die Funktion in Amazon anzuzeigenCloudWatch, einschließlich der in Rechnung gestellten Dauer des Ereignisses. Wenn die Funktion fehlgeschlagen ist, stellt das CloudWatch-Protokoll Informationen zur Ursache bereit.

Schritt 5: Nächste Schritte

Nachdem Sie erfolgreich eine Lambda-Funktion erstellt und sie als Aktion in einer Pipeline hinzugefügt haben, können Sie Folgendes ausprobieren:

- Fügen Sie Ihrer Bühne 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 entfernenAWS Lambda, sie aus und die Rolle aus IAM zu löschen, um mögliche Gebühren zu vermeiden. Weitere Informationen finden Sie unter Bearbeiten einer Pipeline in CodePipeline, Löschen einer 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 istCodePipeline. Die Struktur dieses Ereignisses ähnelt der Antwort unter `GetJobDetails API`, aber ohne den `actionTypeId`- und `pipelineContext`-Datentyp. Zwei Aktionskonfigurationsdetails, `FunctionName` und `UserParameters`, sind sowohl im JSON-

Ereignis als auch in der Antwort auf die `GetJobDetails`-API enthalten. Die Werte in *roter Kursivschrift* sind Beispiele oder Erklärungen, keine realen Werte.

```
{
    "CodePipeline.job": {
        "id": "11111111-abcd-1111-abcd-111111abcdef",
        "accountId": "111111111111",
        "data": {
            "actionConfiguration": {
                "configuration": {
                    "FunctionName": "MyLambdaFunctionForAWSCodePipeline",
                    "UserParameters": "some-input-such-as-a-URL"
                }
            },
            "inputArtifacts": [
                {
                    "location": {
                        "s3Location": {
                            "bucketName": "the name of the bucket configured as the
 pipeline artifact store in Amazon S3, for example codepipeline-us-east-2-1234567890",
                            "objectKey": "the name of the application, for example
 CodePipelineDemoApplication.zip"
                        },
                        "type": "S3"
                    },
                    "revision": null,
                    "name": "ArtifactName"
                }
            ],
            "outputArtifacts": [],
            "artifactCredentials": {
                "secretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
                "sessionToken": "MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
```

```
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
            },
            "continuationToken": "A continuation token if continuing job",
            "encryptionKey": {
              "id": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
              "type": "KMS"
            }
        }
    }
}
```

Weitere Beispielfunktionen

Die folgenden Lambda-Beispielfunktionen demonstrieren zusätzliche Funktionen, die Sie für Ihre Pipelines verwenden könnenCodePipeline. 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

- Beispiel einer Python-Funktion, die eine AWS CloudFormation-Vorlage verwendet

Beispiel einer Python-Funktion, die eine AWS CloudFormation-Vorlage verwendet

Im folgenden Beispiel wird eine Funktion gezeigt, die einen Stack auf Grundlage einer bereitgestellten AWS CloudFormation-Vorlage erstellt. Die Vorlage erstellt einen Amazon S3 S3-Bucket. Dies erfolgt nur zu Demonstrationszwecken, um Kosten zu sparen. Im Idealfall sollten Sie den Stack löschen, bevor Sie Inhalte in den Bucket hochladen. Falls Sie Dateien in den Bucket hochladen, können Sie den Bucket nicht mehr löschen, wenn Sie den Stack löschen. In diesem Fall müssen Sie alle Inhalte des Buckets manuell löschen, bevor Sie den Bucket selbst löschen können.

In diesem Python-Beispiel wird davon ausgegangen, dass Sie über eine Pipeline verfügen, die einen Amazon S3 S3-Bucket als Quellaktion verwendet, oder dass Sie Zugriff auf einen versionierten Amazon S3 S3-Bucket haben, den Sie mit der Pipeline verwenden können. Sie erstellen die AWS CloudFormation-Vorlage, komprimieren sie und laden Sie als ZIP-Datei in den 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 oder Dateien in eine einzelne ZIP-Datei komprimieren und die ZIP-Datei in Ihren Quell-Bucket hochladen. Sie können auch eine einzelne Datei ungezippt hochladen, aber nachgelagerte Aktionen, die eine ZIP-Datei erwarten, schlagen dann fehl.

In diesem Beispiel wird Folgendes gezeigt:

- Die Verwendung von JSON-verschlüsselten Benutzerparametern für die Weiterleitung mehrerer Konfigurationswerte an die Funktion (`get_user_params`)
- Die Interaktion mit ZIP-Artefakten in einem Artefakt-Bucket (`get_template`)
- Die Verwendung eines Fortsetzungstokens, um einen lange andauernden asynchronen Prozess zu überwachen (`continue_job_later`). Dadurch kann die Aktion fortgesetzt werden und die Funktion kann erfolgreich sein, auch wenn sie eine Laufzeit von fünfzehn Minuten überschreitet (ein Limit in Lambda).

Um diese Lambda-Beispielfunktion verwenden zu können, muss die Richtlinie für die Lambda-Ausführungsrolle über`Allow` Berechtigungen in AWS CloudFormation Amazon S3 verfügen und CodePipeline, wie in dieser Beispielrichtlinie gezeigt, über folgende Berechtigungen verfügen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Action": [
                "codepipeline:PutJobSuccessResult",
                "codepipeline:PutJobFailureResult"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "cloudformation:DescribeStacks",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Um die AWS CloudFormation-Vorlage zu erstellen, öffnen Sie einen Texteditor, kopieren Sie folgenden Code und fügen Sie ihn ein:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "CloudFormation template which creates an S3 bucket",
  "Resources" : {
    "MySampleBucket" : {
      "Type" : "AWS::S3::Bucket",
      "Properties" : {
      }
    }
  },
  "Outputs" : {
    "BucketName" : {
      "Value" : { "Ref" : "MySampleBucket" },
      "Description" : "The name of the S3 bucket"
    }
  }
}
```

Speichern Sie dies als JSON-Datei mit dem Namen **`template.json`** in einem Verzeichnis mit dem Namen **`template-package`**. Erstellen Sie eine komprimierte Datei (.zip) mit diesem Verzeichnis und diesem Dateinamen**`template-package.zip`** und laden Sie die komprimierte Datei in einen versionierten Amazon S3 S3-Bucket hoch. Wenn Sie bereits einen Bucket für Ihre Pipeline konfiguriert haben, können Sie diesen verwenden. Bearbeiten Sie als Nächstes Ihre Pipeline und fügen Sie eine Quellaktion hinzu, welche die ZIP-Datei abruft. Benennen Sie die Ausgabe für diese Aktion *`MyTemplate`*. Weitere Informationen finden Sie unter Bearbeiten einer Pipeline in CodePipeline.

> **Note**
>
> Die Lambda-Beispielfunktion erwartet diese Dateinamen und die komprimierte Struktur. Sie können jedoch Ihre eigene AWS CloudFormation-Vorlage anstelle dieses Beispiels verwenden. 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 IhreAWS 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 Bestehende Rolle verwenden aus. Wählen Sie Ihre Rolle unter Existing role (Vorhandene Rolle) aus und klicken Sie dann auf Create function (Funktion erstellen).

   Die Detailseite für Ihre erstellte Funktion wird geöffnet.
5. Fügen Sie den folgenden Code in das Feld Function code (Funktionscode) ein:

```
from __future__ import print_function
from boto3.session import Session

import json
import urllib
import boto3
import zipfile
```

```
import tempfile
import botocore
import traceback

print('Loading function')

cf = boto3.client('cloudformation')
code_pipeline = boto3.client('codepipeline')

def find_artifact(artifacts, name):
    """Finds the artifact 'name' among the 'artifacts'

    Args:
        artifacts: The list of artifacts available to the function
        name: The artifact we wish to use
    Returns:
        The artifact dictionary found
    Raises:
        Exception: If no matching artifact is found

    """
    for artifact in artifacts:
        if artifact['name'] == name:
            return artifact

    raise Exception('Input artifact named "{0}" not found in event'.format(name))

def get_template(s3, artifact, file_in_zip):
    """Gets the template artifact

    Downloads the artifact from the S3 artifact store to a temporary file
    then extracts the zip and returns the file containing the CloudFormation
    template.

    Args:
        artifact: The artifact to download
        file_in_zip: The path to the file within the zip containing the template

    Returns:
        The CloudFormation template as a string

    Raises:
        Exception: Any exception thrown while downloading the artifact or unzipping
 it
```

```
    """
    tmp_file = tempfile.NamedTemporaryFile()
    bucket = artifact['location']['s3Location']['bucketName']
    key = artifact['location']['s3Location']['objectKey']

    with tempfile.NamedTemporaryFile() as tmp_file:
        s3.download_file(bucket, key, tmp_file.name)
        with zipfile.ZipFile(tmp_file.name, 'r') as zip:
            return zip.read(file_in_zip)

def update_stack(stack, template):
    """Start a CloudFormation stack update

    Args:
        stack: The stack to update
        template: The template to apply

    Returns:
        True if an update was started, false if there were no changes
        to the template since the last update.

    Raises:
        Exception: Any exception besides "No updates are to be performed."

    """
    try:
        cf.update_stack(StackName=stack, TemplateBody=template)
        return True

    except botocore.exceptions.ClientError as e:
        if e.response['Error']['Message'] == 'No updates are to be performed.':
            return False
        else:
            raise Exception('Error updating CloudFormation stack
 "{0}"'.format(stack), e)

def stack_exists(stack):
    """Check if a stack exists or not

    Args:
        stack: The stack to check

    Returns:
```

```
        True or False depending on whether the stack exists

    Raises:
        Any exceptions raised .describe_stacks() besides that
        the stack doesn't exist.

    """
    try:
        cf.describe_stacks(StackName=stack)
        return True
    except botocore.exceptions.ClientError as e:
        if "does not exist" in e.response['Error']['Message']:
            return False
        else:
            raise e

def create_stack(stack, template):
    """Starts a new CloudFormation stack creation

    Args:
        stack: The stack to be created
        template: The template for the stack to be created with

    Throws:
        Exception: Any exception thrown by .create_stack()
    """
    cf.create_stack(StackName=stack, TemplateBody=template)

def get_stack_status(stack):
    """Get the status of an existing CloudFormation stack

    Args:
        stack: The name of the stack to check

    Returns:
        The CloudFormation status string of the stack such as CREATE_COMPLETE

    Raises:
        Exception: Any exception thrown by .describe_stacks()

    """
    stack_description = cf.describe_stacks(StackName=stack)
    return stack_description['Stacks'][0]['StackStatus']
```

```
def put_job_success(job, message):
    """Notify CodePipeline of a successful job

    Args:
        job: The CodePipeline job ID
        message: A message to be logged relating to the job status

    Raises:
        Exception: Any exception thrown by .put_job_success_result()

    """
    print('Putting job success')
    print(message)
    code_pipeline.put_job_success_result(jobId=job)

def put_job_failure(job, message):
    """Notify CodePipeline of a failed job

    Args:
        job: The CodePipeline job ID
        message: A message to be logged relating to the job status

    Raises:
        Exception: Any exception thrown by .put_job_failure_result()

    """
    print('Putting job failure')
    print(message)
    code_pipeline.put_job_failure_result(jobId=job, failureDetails={'message':
 message, 'type': 'JobFailed'})

def continue_job_later(job, message):
    """Notify CodePipeline of a continuing job

    This will cause CodePipeline to invoke the function again with the
    supplied continuation token.

    Args:
        job: The JobID
        message: A message to be logged relating to the job status
        continuation_token: The continuation token

    Raises:
        Exception: Any exception thrown by .put_job_success_result()
```

```
    """

    # Use the continuation token to keep track of any job execution state
    # This data will be available when a new job is scheduled to continue the
 current execution
    continuation_token = json.dumps({'previous_job_id': job})

    print('Putting job continuation')
    print(message)
    code_pipeline.put_job_success_result(jobId=job,
 continuationToken=continuation_token)

def start_update_or_create(job_id, stack, template):
    """Starts the stack update or create process

    If the stack exists then update, otherwise create.

    Args:
        job_id: The ID of the CodePipeline job
        stack: The stack to create or update
        template: The template to create/update the stack with

    """
    if stack_exists(stack):
        status = get_stack_status(stack)
        if status not in ['CREATE_COMPLETE', 'ROLLBACK_COMPLETE',
 'UPDATE_COMPLETE']:
            # If the CloudFormation stack is not in a state where
            # it can be updated again then fail the job right away.
            put_job_failure(job_id, 'Stack cannot be updated when status is: ' +
 status)
            return

        were_updates = update_stack(stack, template)

        if were_updates:
            # If there were updates then continue the job so it can monitor
            # the progress of the update.
            continue_job_later(job_id, 'Stack update started')

        else:
            # If there were no updates then succeed the job immediately
            put_job_success(job_id, 'There were no stack updates')
```

```
    else:
        # If the stack doesn't already exist then create it instead
        # of updating it.
        create_stack(stack, template)
        # Continue the job so the pipeline will wait for the CloudFormation
        # stack to be created.
        continue_job_later(job_id, 'Stack create started')

def check_stack_update_status(job_id, stack):
    """Monitor an already-running CloudFormation update/create

    Succeeds, fails or continues the job depending on the stack status.

    Args:
        job_id: The CodePipeline job ID
        stack: The stack to monitor

    """
    status = get_stack_status(stack)
    if status in ['UPDATE_COMPLETE', 'CREATE_COMPLETE']:
        # If the update/create finished successfully then
        # succeed the job and don't continue.
        put_job_success(job_id, 'Stack update complete')

    elif status in ['UPDATE_IN_PROGRESS', 'UPDATE_ROLLBACK_IN_PROGRESS',
    'UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS', 'CREATE_IN_PROGRESS',
    'ROLLBACK_IN_PROGRESS', 'UPDATE_COMPLETE_CLEANUP_IN_PROGRESS']:
        # If the job isn't finished yet then continue it
        continue_job_later(job_id, 'Stack update still in progress')

    else:
        # If the Stack is a state which isn't "in progress" or "complete"
        # then the stack update/create has failed so end the job with
        # a failed result.
        put_job_failure(job_id, 'Update failed: ' + status)

def get_user_params(job_data):
    """Decodes the JSON user parameters and validates the required properties.

    Args:
        job_data: The job data structure containing the UserParameters string which
 should be a valid JSON structure

    Returns:
```

```
        The JSON parameters decoded as a dictionary.

    Raises:
        Exception: The JSON can't be decoded or a property is missing.

    """
    try:
        # Get the user parameters which contain the stack, artifact and file
 settings
        user_parameters = job_data['actionConfiguration']['configuration']
['UserParameters']
        decoded_parameters = json.loads(user_parameters)

    except Exception as e:
        # We're expecting the user parameters to be encoded as JSON
        # so we can pass multiple values. If the JSON can't be decoded
        # then fail the job with a helpful message.
        raise Exception('UserParameters could not be decoded as JSON')

    if 'stack' not in decoded_parameters:
        # Validate that the stack is provided, otherwise fail the job
        # with a helpful message.
        raise Exception('Your UserParameters JSON must include the stack name')

    if 'artifact' not in decoded_parameters:
        # Validate that the artifact name is provided, otherwise fail the job
        # with a helpful message.
        raise Exception('Your UserParameters JSON must include the artifact name')

    if 'file' not in decoded_parameters:
        # Validate that the template file is provided, otherwise fail the job
        # with a helpful message.
        raise Exception('Your UserParameters JSON must include the template file
 name')

    return decoded_parameters

def setup_s3_client(job_data):
    """Creates an S3 client

    Uses the credentials passed in the event by CodePipeline. These
    credentials can be used to access the artifact bucket.

    Args:
```

```
        job_data: The job data structure

    Returns:
        An S3 client with the appropriate credentials

    """
    key_id = job_data['artifactCredentials']['accessKeyId']
    key_secret = job_data['artifactCredentials']['secretAccessKey']
    session_token = job_data['artifactCredentials']['sessionToken']

    session = Session(aws_access_key_id=key_id,
        aws_secret_access_key=key_secret,
        aws_session_token=session_token)
    return session.client('s3',
 config=botocore.client.Config(signature_version='s3v4'))

def lambda_handler(event, context):
    """The Lambda function handler

    If a continuing job then checks the CloudFormation stack status
    and updates the job accordingly.

    If a new job then kick of an update or creation of the target
    CloudFormation stack.

    Args:
        event: The event passed by Lambda
        context: The context passed by Lambda

    """
    try:
        # Extract the Job ID
        job_id = event['CodePipeline.job']['id']

        # Extract the Job Data
        job_data = event['CodePipeline.job']['data']

        # Extract the params
        params = get_user_params(job_data)

        # Get the list of artifacts passed to the function
        artifacts = job_data['inputArtifacts']

        stack = params['stack']
```

```
        artifact = params['artifact']
        template_file = params['file']

        if 'continuationToken' in job_data:
            # If we're continuing then the create/update has already been triggered
            # we just need to check if it has finished.
            check_stack_update_status(job_id, stack)
        else:
            # Get the artifact details
            artifact_data = find_artifact(artifacts, artifact)
            # Get S3 client to access artifact with
            s3 = setup_s3_client(job_data)
            # Get the JSON template file out of the artifact
            template = get_template(s3, artifact_data, template_file)
            # Kick off a stack update or create
            start_update_or_create(job_id, stack, template)

    except Exception as e:
        # If any other exceptions which we didn't expect are raised
        # then fail the job and log the exception message.
        print('Function failed due to exception.')
        print(e)
        traceback.print_exc()
        put_job_failure(job_id, 'Function exception: ' + str(e))

    print('Function complete.')
    return "Complete."
```

6. Belassen Sie Handler auf dem Standardwert und belassen Sie Role bei dem Namen, den Sie zuvor ausgewählt oder erstellt haben**`CodePipelineLambdaExecRole`**.
7. Ersetzen Sie unter Basic settings (Grundlegende Einstellungen) im Feld Timeout den Standardwert von 3 Sekunden durch **`20`**.
8. Wählen Sie Speichern.
9. Bearbeiten Sie die Pipeline in der CodePipeline-Konsole, um die Funktion als Aktion in einer Stufe in Ihrer Pipeline hinzuzufügen. Wählen Sie Bearbeiten für die Pipeline-Phase, die Sie ändern möchten, und wählen Sie Aktionsgruppe hinzufügen. Geben Sie auf der Seite Aktion bearbeiten unter Aktionsname einen Namen für Ihre Aktion ein. Wählen Sie unter Aktionsanbieter Lambda aus.

   Wählen Sie unter Eingabeartefakte aus`MyTemplate`. In UserParametersmüssen Sie eine JSON-Zeichenfolge mit drei Parametern angeben:

- Stack name
- AWS CloudFormation-Vorlagename und -Pfad zur Datei
- Eingabeartefakt

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 *MyTemplate*in zu erstellen 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 unter 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.

Eine fehlgeschlagene Aktion in einer Phase wiederholen

Sie können eine fehlgeschlagene Phase erneut versuchen, ohne eine Pipeline erneut von Anfang an ausführen zu müssen. Sie tun dies, indem Sie entweder die fehlgeschlagenen Aktionen in einer Phase erneut versuchen oder indem Sie alle Aktionen in der Phase wiederholen, beginnend mit der ersten Aktion in der Phase. Wenn Sie die fehlgeschlagenen Aktionen in einer Phase wiederholen, funktionieren alle Aktionen, die noch ausgeführt werden, weiterhin, und fehlgeschlagene Aktionen werden erneut ausgelöst. Wenn Sie eine fehlgeschlagene Phase von der ersten Aktion in der Phase wiederholen, können in der Phase keine Aktionen ausgeführt werden. Bevor eine Phase erneut versucht werden kann, müssen entweder alle Aktionen fehlgeschlagen sein oder einige Aktionen sind fehlgeschlagen und andere erfolgreich.

> **Important**
>
> Beim erneuten Versuch einer fehlgeschlagenen Phase werden alle Aktionen der Phase von der ersten Aktion in der Phase wiederholt, und bei einem erneuten Versuch, fehlgeschlagene Aktionen durchzuführen, werden alle fehlgeschlagenen Aktionen in der Phase wiederholt. Dadurch werden Ausgabeartefakte zuvor erfolgreicher Aktionen in derselben Ausführung außer Kraft gesetzt.

Artefakte können zwar überschrieben werden, der Ausführungsverlauf zuvor erfolgreicher Aktionen wird jedoch beibehalten.

Wenn Sie die Konsole verwenden, um eine Pipeline anzuzeigen, erscheint entweder die Schaltfläche Phase wiederholen oder Fehlgeschlagene Aktionen wiederholen auf der Stufe, die erneut versucht werden kann.

Wenn Sie die AWS CLI verwenden, können Sie den get-pipeline-state Befehl verwenden, um festzustellen, ob Aktionen fehlgeschlagen sind.

Note

In den folgenden Fällen können Sie eine Phase möglicherweise nicht erneut versuchen:

- Alle Aktionen in der Phase waren erfolgreich, sodass die Phase nicht den Status Fehlgeschlagen hat.
- Die gesamte Pipeline-Struktur hat sich geändert, nachdem die Phase gescheitert war.
- Es wird bereits ein anderer Wiederholungsversuch in der Phase ausgeführt.

Themen

- Wiederholen fehlgeschlagener Aktionen (Konsole)
- Wiederholen fehlgeschlagener Aktionen (CLI)

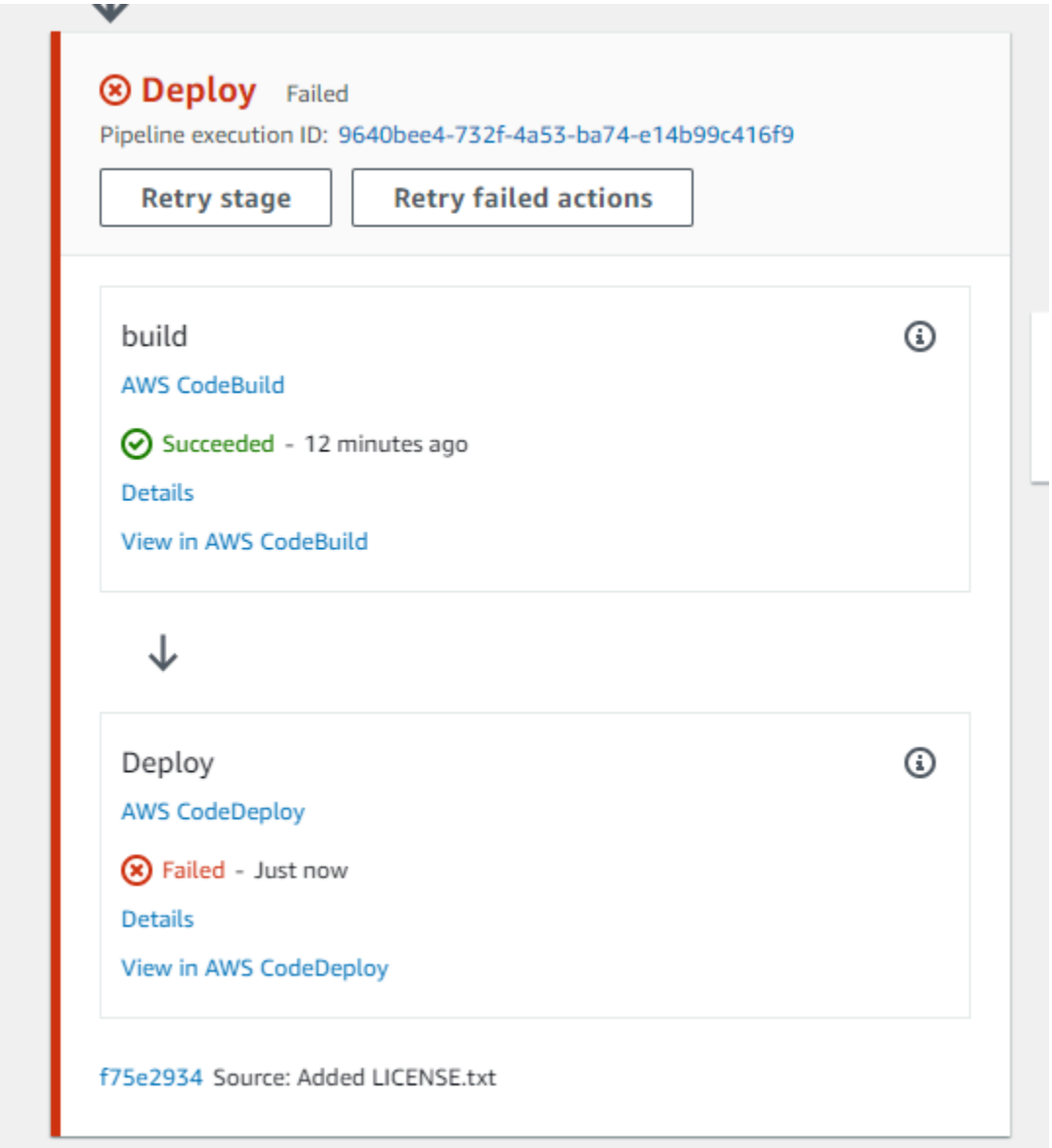
Wiederholen fehlgeschlagener Aktionen (Konsole)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase erneut zu versuchen: Konsole

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

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline.
3. Suchen Sie die Phase mit der fehlgeschlagenen Aktion und wählen Sie dann eine der folgenden Optionen aus:

- Um alle Aktionen in der Phase erneut zu versuchen, wählen Sie Phase wiederholen aus.
- Um nur fehlgeschlagene Aktionen in der Phase erneut zu versuchen, wählen Sie Fehlgeschlagene Aktionen wiederholen.



Wenn alle wiederholten Aktionen in der Stufe erfolgreich abgeschlossen sind, wird die Pipeline weiter ausgeführt.

Wiederholen fehlgeschlagener Aktionen (CLI)

Um eine fehlgeschlagene Phase oder fehlgeschlagene Aktionen in einer Phase erneut zu versuchen - CLI

Um alle Aktionen oder alle fehlgeschlagenen Aktionen erneut AWS CLI zu versuchen, führen Sie den retry-stage-execution Befehl mit den folgenden Parametern aus:

```
--pipeline-name <value>
--stage-name <value>
--pipeline-execution-id <value>
--retry-mode ALL_ACTIONS/FAILED_ACTIONS
```

> **Note**
>
> Die Werte, für die Sie verwenden können, `retry-mode` sind `FAILED_ACTIONS` und`ALL_ACTIONS`.

1. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer [retry-stage-execution](#)Befehlszeile (Windows) aus, wie im folgenden Beispiel für eine Pipeline mit dem Namen gezeigt`MyPipeline`.

   ```
   aws codepipeline retry-stage-execution --pipeline-name MyPipeline --stage-name
    Deploy --pipeline-execution-id b59babff-5f34-EXAMPLE --retry-mode FAILED_ACTIONS
   ```

   Die Ausgabe gibt die Ausführungs-ID zurück:

   ```
   {
       "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
   }
   ```

2. Sie können den Befehl auch mit einer JSON-Eingabedatei ausführen. Sie erstellen zunächst eine JSON-Datei, die die Pipeline, die Stufe mit den fehlgeschlagenen Aktionen und die letzte Pipeline-Ausführung in dieser Stufe ermittelt. Führen Sie den Befehl retry-stage-execution mit dem Parameter `--cli-input-json` aus. Am einfachsten können Sie die für die JSON-Datei benötigten Details über den Befehl get-pipeline-state abrufen.

a. Führen Sie den Befehl an einem Terminal (Linux, macOS oder Unix) oder einer get-pipeline-stateBefehlszeile (Windows) in einer Pipeline aus. Für eine Pipeline mit dem Namen würden Sie MyFirstPipeline beispielsweise etwas Ähnliches wie das Folgende eingeben:

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

Die Ausgabe des Befehls umfasst die Pipeline-Statusinformationen für jede einzelne Stufe. Im folgenden Beispiel zeigt die Ausgabe, dass eine oder mehrere Aktionen in der Staging-Stufe fehlgeschlagen sind:

```
{
    "updated": 1427245911.525,
    "created": 1427245911.525,
    "pipelineVersion": 1,
    "pipelineName": "MyFirstPipeline",
    "stageStates": [
        {
            "actionStates": [...],
            "stageName": "Source",
            "latestExecution": {
                "pipelineExecutionId": "9811f7cb-7cf7-SUCCESS",
                "status": "Succeeded"
            }
        },
        {
            "actionStates": [...],
            "stageName": "Staging",
            "latestExecution": {
                "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
                "status": "Failed"
            }
        }
    ]
}
```

b. Erstellen Sie in einem Texteditor eine Datei im JSON-Format, in der Sie Folgendes festhalten:

- Der Name der Pipeline mit den fehlgeschlagenen Aktionen
- Der Name der Stufe mit den fehlgeschlagenen Aktionen

- Die ID der letzten Pipeline-Ausführung in der Stufe
- Der Retry-Modus.

Für das vorherige MyFirstPipeline Beispiel würde Ihre Datei etwa so aussehen:

```
{
    "pipelineName": "MyFirstPipeline",
    "stageName": "Staging",
    "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
    "retryMode": "FAILED_ACTIONS"
}
```

c. Speichern Sie die Datei mit einem Namen wie **`retry-failed-actions.json`**.

d. Rufen Sie die beim Ausführen des Befehls retry-stage-execution erstellte Datei auf. Beispiele:

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

```
aws codepipeline retry-stage-execution --cli-input-json file://retry-failed-
actions.json
```

e. Um die Ergebnisse des Wiederholungsversuchs anzuzeigen, öffnen Sie entweder die CodePipeline Konsole und wählen Sie die Pipeline aus, die die fehlgeschlagenen Aktionen enthält, oder verwenden Sie den get-pipeline-state Befehl erneut. Weitere Informationen finden Sie unter Pipelines und Details anzeigen in CodePipeline.

Genehmigungsaktionen verwalten in CodePipeline

In AWS CodePipeline können Sie einer Phase in einer Pipeline eine Genehmigungsaktion hinzufügen, wenn Sie möchten, dass die Pipeline-Ausführung an einem bestimmten Punkt angehalten wird, damit jemand mit den erforderlichen AWS Identity and Access Management-Berechtigungen die Aktion erlauben oder ablehnen kann.

Wird die Aktion genehmigt, wird die Pipeline-Ausführung fortgesetzt. Wenn die Aktion abgelehnt wird — oder wenn niemand die Aktion innerhalb von sieben Tagen, nachdem die Pipeline die Aktion erreicht und gestoppt hat, genehmigt oder ablehnt — ist das Ergebnis dasselbe wie eine fehlgeschlagene Aktion, und die Pipeline-Ausführung wird nicht fortgesetzt.

Es gibt die folgenden Gründe für die Verwendung manueller Genehmigungen:

- Sie möchten, dass jemand eine Code- oder Change-Management-Prüfung durchführt, bevor eine Überarbeitung für die nächste Stufe einer Pipeline erlaubt wird.
- Sie möchten, dass jemand einen manuellen Qualitätssicherungstest bei der letzten Version einer Anwendung durchführt oder die Integrität eines Build-Artefakts bestätigt, bevor eine Veröffentlichung stattfindet.
- Sie möchten, dass jemand einen neuen oder aktualisierten Text prüft, bevor er auf einer Firmenwebsite veröffentlicht wird.

Konfigurationsoptionen für manuelle Genehmigungsaktionen in CodePipeline

CodePipeline bietet drei Konfigurationsoptionen, mit denen Sie Genehmiger über die Genehmigungsaktion informieren können.

Genehmigungsbenachrichtigungen veröffentlichen Sie können eine Genehmigungsaktion so konfigurieren, dass eine Nachricht in einem Amazon Simple Notification Service-Thema veröffentlicht wird, wenn die Pipeline bei der Aktion stoppt. Amazon SNS übermittelt die Nachricht an jeden Endpunkt, der das Thema abonniert hat. Sie müssen ein Thema verwenden, das in derselben AWS Region wie die Pipeline erstellt wurde und die Genehmigungsaktion enthalten wird. Wenn Sie ein Thema erstellen, sollten Sie ihm einen Namen geben, der seinen Zweck erkennen lässt, wie z. B. `MyFirstPipeline-us-east-2-approval`.

Wenn Sie Genehmigungsbenachrichtigungen zu Amazon SNS-Themen veröffentlichen, können Sie aus Formaten wie E-Mail- oder SMS-Empfängern, SQS-Warteschlangen, HTTP/HTTPS-Endpunkten oder AWS Lambda Funktionen wählen, die Sie mit Amazon SNS aufrufen. Informationen zu Amazon SNS SNS-Themenbenachrichtigungen finden Sie in den folgenden Themen:

- Was ist Amazon Simple Notification Service?
- Ein Thema in Amazon SNS erstellen
- Senden von Amazon SNS-Nachrichten an Amazon SQS-Warteschlagen

- Abonnieren einer Warteschlange für ein Amazon SNS-Thema
- Senden von Amazon SNS-Nachrichten an HTTP/HTTPS-Endpunkte
- Aufrufen von Lambda-Funktionen mit Amazon SNS-Benachrichtigungen

Unter JSON-Datenformat für manuelle Genehmigungsbenachrichtigungen in CodePipeline finden Sie die Struktur der JSON-Daten, die für eine Genehmigungsaktionsbenachrichtigung generiert werden.

Festlegen einer URL zur Prüfung – Als Teil der Konfiguration der Genehmigungsaktion können Sie eine URL angeben, die geprüft werden soll. Bei der URL kann es sich um einen Link zu einer Webanwendung handeln, die Genehmiger testen sollen, oder um eine Seite mit weiteren Informationen zu Ihrer Genehmigungsanfrage. Die URL ist in der Benachrichtigung enthalten, die im Amazon SNS SNS-Thema veröffentlicht wird. Genehmiger können die Konsole oder CLI zum Anzeigen verwenden.

Eingeben von Kommentaren für Genehmiger Beim Erstellen einer Genehmigungsaktion können Sie ebenfalls Kommentare hinzufügen, die für die Empfänger der Benachrichtigungen oder für die Personen, die die Aktion in der Konsole oder CLI-Reaktion sehen, angezeigt werden.

Keine Konfigurationsoptionen – Sie können auch keine dieser drei Optionen konfigurieren. Sie brauchen sie möglicherweise nicht, wenn Sie beispielsweise jemanden direkt darüber benachrichtigen, dass die Aktion von ihm geprüft werden kann, oder wenn Sie einfach nur möchten, dass die Pipeline so lange anhält, bis Sie sich dazu entschieden haben, selbst die Aktion zu genehmigen.

Setup- und Workflow-Übersicht 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 SNS-Benachrichtigungen verwenden, stellen Sie sicher, dass die Servicerolle, die Sie in Ihrem CodePipeline Betrieb verwenden, berechtigt ist, auf Amazon SNS SNS-Ressourcen zuzugreifen.
3. (Optional) Wenn Sie Amazon SNS SNS-Benachrichtigungen verwenden, erstellen Sie ein Amazon SNS SNS-Thema und fügen ihm einen oder mehrere Abonnenten oder Endpunkte hinzu.

4. Wenn Sie die AWS CLI verwenden, um die Pipeline zu erstellen, oder nachdem Sie die CLI oder Konsole zum Erstellen der Pipeline verwendet haben, fügen Sie einer Phase in der Pipeline eine Genehmigungsaktion hinzu.

   Wenn Sie Benachrichtigungen verwenden, nehmen Sie den Amazon-Ressourcennamen (ARN) des Amazon SNS-Themas in die Konfiguration der Aktion auf. (Ein ARN ist eine eindeutige Kennung für eine Amazon-Ressource. ARNs für Amazon SNS SNS-Themen sind wie *arn:aws:sns:us-east* - 2:80398 EXAMPLE: strukturiert. MyApprovalTopic Weitere Informationen finden Sie unter Amazon-Ressourcennamen (ARNs) und AWS-Service Namespaces in der.) Allgemeine Amazon Web Services-Referenz
5. Die Pipeline stoppt, wenn sie die Genehmigungsaktion erreicht. Wenn ein Amazon SNS SNS-Themen-ARN in der Konfiguration der Aktion enthalten war, wird eine Benachrichtigung unter dem Amazon SNS SNS-Thema veröffentlicht, und allen Abonnenten des Themas oder abonnierten Endpunkten wird eine Nachricht mit einem Link zur Überprüfung der Genehmigungsaktion in der Konsole zugestellt.
6. Ein Genehmiger untersucht die Ziel-URL und prüft Kommentare, sofern vorhanden.
7. Mithilfe der Konsole, der CLI oder dem SDK stellt der Genehmiger einen Übersichtskommentar bereit und sendet eine Antwort:
   - Genehmigt: Die Pipeline-Ausführung wird fortgesetzt.
   - Abgelehnt: Der Stufenstatus wird in "Fehlgeschlagen" geändert und die Pipeline-Ausführung wird nicht fortgesetzt.

   Wenn innerhalb von sieben Tagen keine Antwort übermittelt wird, wird die Aktion als "fehlgeschlagen" markiert.

Erteilen Sie einem IAM-Benutzer Genehmigungsberechtigungen in CodePipeline

Bevor IAM-Benutzer in Ihrer Organisation Genehmigungsaktionen genehmigen oder ablehnen können, müssen ihnen Berechtigungen für den Zugriff auf Pipelines und die Aktualisierung des Status von Genehmigungsaktionen erteilt werden. Sie können Berechtigungen für den Zugriff auf alle Pipelines und Genehmigungsaktionen in Ihrem Konto erteilen, indem Sie die `AWSCodePipelineApproverAccess` verwaltete Richtlinie einem IAM-Benutzer, einer Rolle oder einer Gruppe zuordnen. Oder Sie können eingeschränkte Berechtigungen gewähren, indem Sie die einzelnen Ressourcen angeben, auf die ein IAM-Benutzer, eine IAM-Rolle oder eine Gruppe zugreifen kann.

> **Note**
>
> Die in diesem Thema beschriebenen Berechtigungen gewähren einen äußerst begrenzten Zugriff. Um einem Benutzer, einer Rolle oder einer Gruppe mehr zu erlauben, als Genehmigungsaktionen anzunehmen oder abzulehnen, können Sie weitere verwaltete Richtlinien anfügen. Informationen zu den verfügbaren verwalteten Richtlinien finden Sie unter. CodePipeline AWS Von verwaltete Richtlinien für AWS CodePipeline

Gewähren von Genehmigungsberechtigungen für alle Pipelines und Genehmigungsaktionen

Verwenden Sie für Benutzer, die Genehmigungsaktionen in ausführen müssen CodePipeline, die `AWSCodePipelineApproverAccess` verwaltete Richtlinie.

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

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center-Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

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

Festlegen von Genehmigungsberechtigungen für bestimmte Pipelines und Genehmigungsaktionen

Verwenden Sie für Benutzer, die Genehmigungsaktionen in ausführen müssen CodePipeline, die folgende benutzerdefinierte Richtlinie. Geben Sie in der folgenden Richtlinie die einzelnen

Ressourcen an, auf die ein Benutzer zugreifen kann. Die folgende Richtlinie gewährt Benutzern beispielsweise die Befugnis, nur die in der `MyFirstPipeline` Pipeline angegebene Aktion `MyApprovalAction` in der Region USA Ost (Ohio) (us-east-2) zu genehmigen oder abzulehnen:

> **Note**
>
> Die `codepipeline:ListPipelines` Berechtigung ist nur erforderlich, wenn IAM-Benutzer auf das CodePipeline Dashboard zugreifen müssen, um diese Pipeline-Liste anzuzeigen. Wenn kein Konsolenzugriff erforderlich ist, können Sie `codepipeline:ListPipelines` weglassen.

So verwenden Sie den JSON-Richtlinieneditor zum Erstellen einer Richtlinie

1. Melden Sie sich bei der AWS Management Console an 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.

> **Note**
>
> Sie können jederzeit zwischen den Editoroptionen Visual und JSON wechseln. Wenn Sie jedoch Änderungen vornehmen oder im Visual-Editor Weiter wählen, strukturiert IAM Ihre Richtlinie möglicherweise um, um sie für den visuellen Editor zu optimieren. Weitere Informationen finden Sie unter Richtlinienrestrukturierung im IAM-Benutzerhandbuch.

7. Geben Sie auf der Seite Prüfen und erstellen unter Richtlinienname einen Namen und unter Beschreibung (optional) eine Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie Permissions defined in this policy (In dieser Richtlinie definierte Berechtigungen), um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden.
8. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.

Gewähren Sie Amazon SNS SNS-Berechtigungen für eine CodePipeline Servicerolle

Wenn Sie Amazon SNS verwenden möchten, um Benachrichtigungen zu Themen zu veröffentlichen, bei denen Genehmigungsaktionen überprüft werden müssen, muss der Servicerolle, die Sie in Ihrem CodePipeline Betrieb verwenden, die Berechtigung zum Zugriff auf die Amazon SNS SNS-

Ressourcen erteilt werden. Sie können die IAM-Konsole verwenden, um diese Berechtigung zu Ihrer Servicerolle hinzuzufügen.

Geben Sie in der folgenden Richtlinie die Richtlinie für die Veröffentlichung mit SNS an. Für die folgende Richtlinie können Sie ihr einen Namen geben`SNSPublish`. Verwenden Sie die folgende Richtlinie, indem Sie sie Ihrer Servicerolle zuordnen.

> **Important**
>
> Stellen Sie sicher, dass Sie in der AWS Management Console mit den Kontoinformationen angemeldet sind, die Sie in Erste Schritte mit CodePipeline verwendet haben.

```
{
    "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 AWS Management Console an 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 AWS Management Console erstellen oder bearbeiten, können Sie eine JSON- oder YAML-Richtlinienvorlage generieren, die Sie in AWS CloudFormation-Vorlagen verwenden können.

   Wählen Sie dazu im Richtlinien-Editor Aktionen und anschließend CloudFormationVorlage generieren aus. Weitere Informationen zu AWS CloudFormation finden Sie unter AWS Identity and Access Management-Ressourcentyp-Referenz im AWS CloudFormation-Benutzerhandbuch.
8. Wenn Sie mit dem Hinzufügen von Berechtigungen zur Richtlinie fertig sind, wählen Sie Next (Weiter) aus.
9. Geben Sie auf der Seite Prüfen und erstellen unter Richtlinienname einen Namen und unter Beschreibung (optional) eine Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie Permissions defined in this policy (In dieser Richtlinie definierte Berechtigungen), um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden.
10. (Optional) Fügen Sie der Richtlinie Metadaten hinzu, indem Sie Tags als Schlüssel-Wert-Paare anfügen. Weitere Informationen zur Verwendung von Tags in IAM finden Sie unter Markieren von IAM-Ressourcen im IAM-Benutzerhandbuch.
11. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.

Hinzufügen einer manuellen Genehmigungsaktion zu einer Pipeline in CodePipeline

Sie können einer Phase in einer CodePipeline Pipeline an dem Punkt, an dem die Pipeline gestoppt werden soll, eine Genehmigungsaktion hinzufügen, sodass jemand die Aktion manuell genehmigen oder ablehnen kann.

> **Note**
>
> Genehmigungsaktionen können nicht zu Quellenstufen hinzugefügt werden. Quellenstufen können nur Quellenaktionen enthalten.

Wenn Sie Amazon SNS verwenden möchten, um Benachrichtigungen zu versenden, wenn eine Genehmigungsaktion zur Überprüfung bereit ist, müssen Sie zunächst die folgenden Voraussetzungen erfüllen:

- Erteilen Sie Ihrer CodePipeline Servicerolle die Erlaubnis, auf Amazon SNS SNS-Ressourcen zuzugreifen. Weitere Informationen finden Sie unter Gewähren Sie Amazon SNS SNS-Berechtigungen für eine CodePipeline Servicerolle.
- Erteilen Sie einer oder mehreren IAM-Identitäten in Ihrer Organisation die Erlaubnis, den Status einer Genehmigungsaktion zu aktualisieren. Weitere Informationen finden Sie unter Erteilen Sie einem IAM-Benutzer Genehmigungsberechtigungen in CodePipeline.

In diesem Beispiel erstellen Sie eine neue Genehmigungsphase und fügen der Phase eine manuelle Genehmigungsaktion hinzu. Sie können einer vorhandenen Phase, die andere Aktionen enthält, auch eine manuelle Genehmigungsaktion hinzufügen.

Hinzufügen einer manuellen Genehmigungsaktion zu einer CodePipeline -Pipeline (Konsole)

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 aus.

Hinzufügen einer manuellen Genehmigungsaktion zu einer CodePipeline-Pipeline (CLI)

Sie können die CLI zum Hinzufügen einer Genehmigungsaktion zu einer vorhandenen Pipeline verwenden oder wenn Sie eine Pipeline erstellen. Hierzu schließen Sie eine Genehmigungsaktion mit dem Typ „manuelle Genehmigung“ in einer Stufe ein, die Sie erstellen oder bearbeiten.

Weitere Informationen zum Erstellen und Bearbeiten von Pipelines finden Sie unter Erstellen einer Pipeline in CodePipeline und Bearbeiten einer Pipeline 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
        }
    ]
```

```
}
```

Erlauben oder Ablehnen einer Genehmigungsaktion 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".

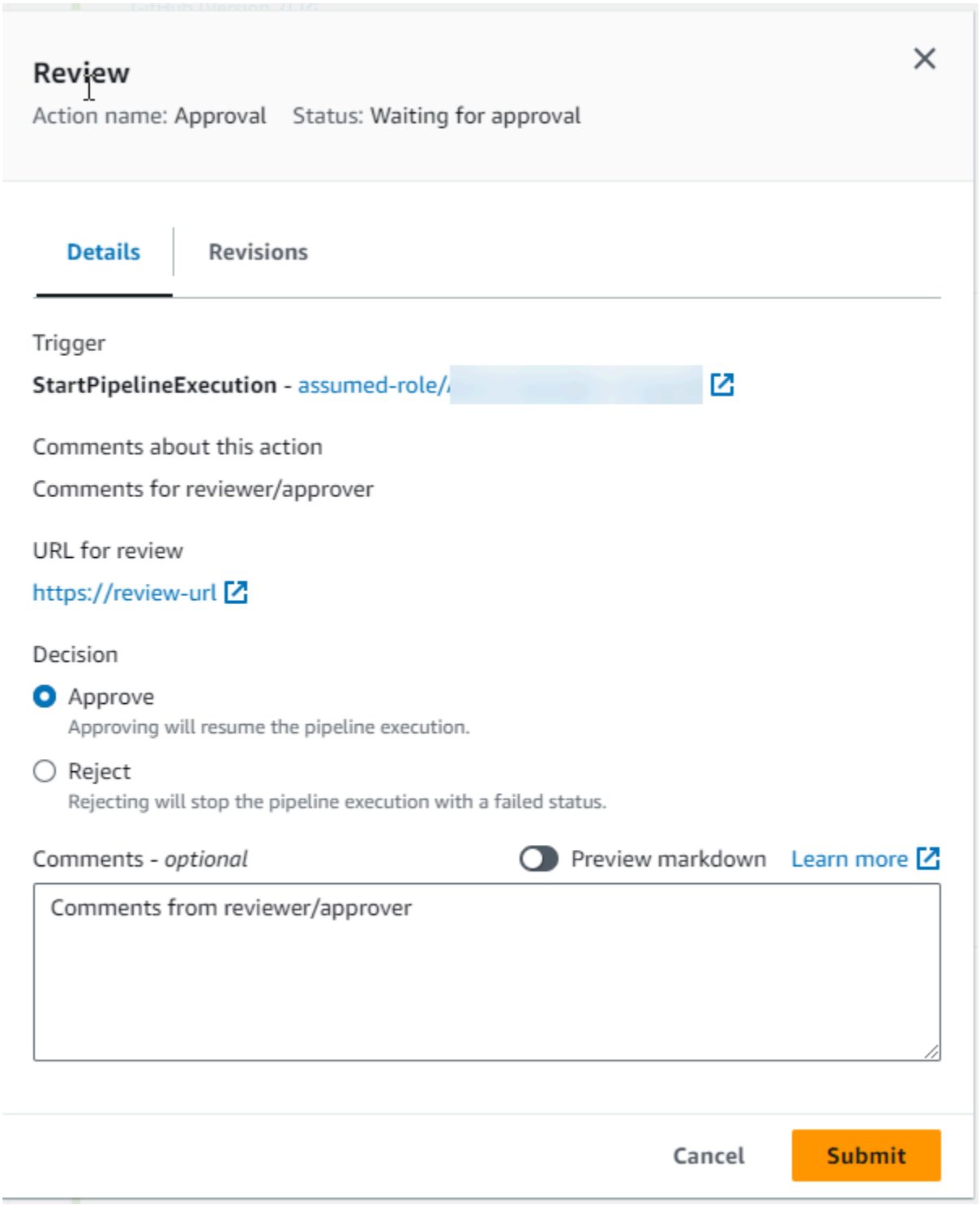
Wenn die Person, die die Genehmigungsaktion zur Pipeline hinzugefügt hat, Benachrichtigungen konfiguriert hat, erhalten Sie möglicherweise eine E-Mail mit den Pipeline-Informationen und dem Status für die Genehmigung.

Erlauben oder Ablehnen einer Genehmigungsaktion (Konsole)

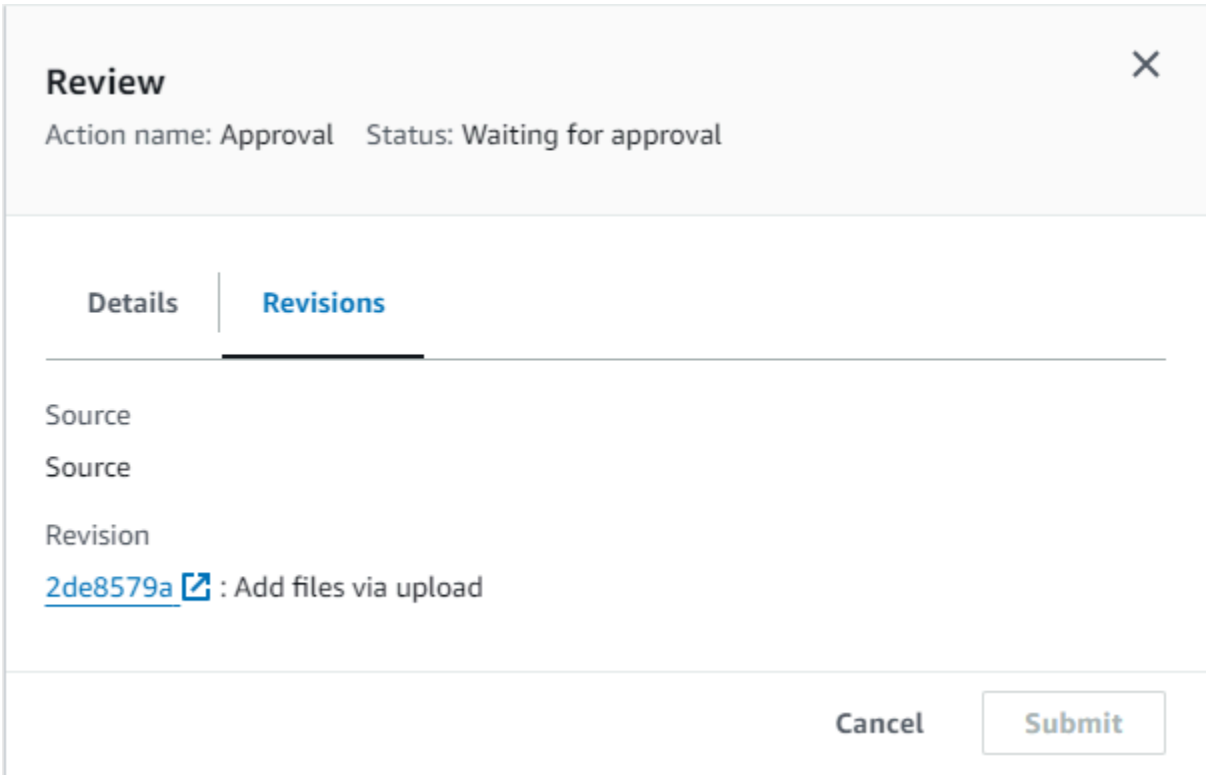
Wenn Sie eine Benachrichtigung erhalten, die einen direkten Link zu einer Genehmigungsaktion enthält, klicken Sie auf den Link Approve or reject (Genehmigen oder ablehnen), melden Sie sich bei der Konsole an und fahren Sie dann mit Schritt 7 hier fort. Führen Sie anderenfalls alle der folgenden Schritte aus.

1. Öffnen Sie die CodePipeline Konsole unter https://console.aws.amazon.com/codepipeline/.
2. Wählen Sie auf der Seite All Pipelines den Namen der Pipeline aus.
3. Suchen Sie die Stufe mit der Genehmigungsaktion. Wählen Sie Überprüfen aus.

   Das Dialogfeld „Überprüfen“ wird angezeigt. Auf der Registerkarte „Details“ werden der Inhalt und die Kommentare der Bewertung angezeigt.



Auf der Registerkarte „Revisionen“ werden die Quellversionen für die Ausführung angezeigt.



4. Sehen Sie sich auf der Registerkarte Details die Kommentare und die URL an, falls vorhanden. Die Nachricht zeigt auch die URL des von Ihnen zu prüfenden Inhalts an, sofern eingeschlossen.
5. Wenn eine URL angegeben wurde, wählen Sie in der Aktion zum Öffnen der Zielwebseite den Link zur URL für die Überprüfung aus, und überprüfen Sie dann den Inhalt.
6. Geben Sie im Überprüfungsfenster Bewertungskommentare ein, z. B. warum Sie die Aktion genehmigen oder ablehnen, und wählen Sie dann Genehmigen oder Ablehnen aus.
7. Wählen Sie Absenden aus.

Erlauben oder Ablehnen einer Genehmigungsanfrage (CLI)

Um mit der CLI auf eine Genehmigungsaktion zu reagieren, müssen Sie zuerst den Befehl get-pipeline-stateverwenden, um das Token abzurufen, das mit der letzten Ausführung der Genehmigungsaktion verknüpft ist.

1. Führen Sie an einem Terminal (Linux, macOS oder Unix) oder einer Befehlszeile (Windows) den get-pipeline-stateBefehl in der Pipeline aus, die die Genehmigungsaktion enthält. Geben Sie 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 put-approval-resultBefehl aus und geben Sie dabei den Namen der Genehmigungs-JSON-Datei an, ähnlich wie im Folgenden:

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

```
aws codepipeline put-approval-result --cli-input-json file://approvalstage-approved.json
```

JSON-Datenformat für manuelle Genehmigungsbenachrichtigungen in CodePipeline

Für Genehmigungsaktionen, die Amazon SNS-Benachrichtigungen verwenden, werden JSON-Daten über die Aktion erstellt und in Amazon SNS veröffentlicht, wenn die Pipeline stoppt. Sie können die JSON-Ausgabe verwenden, um Nachrichten an Amazon SQS SQS-Warteschlangen zu senden oder Funktionen in aufzurufen. AWS Lambda

> **Note**
>
> In dieser Anleitung wird nicht behandelt, wie Benachrichtigungen mit JSON konfiguriert werden. Weitere Informationen finden Sie unter Senden von Amazon SNS SNS-Nachrichten an Amazon SQS SQS-Warteschlangen und Aufrufen von Lambda-Funktionen mithilfe von Amazon SNS SNS-Benachrichtigungen im Amazon SNS SNS-Entwicklerhandbuch.

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."
    }
}
```

Hinzufügen einer regionsübergreifenden Aktion in CodePipeline

AWS CodePipeline enthält eine Reihe von Aktionen, die Ihnen beim Konfigurieren von Entwicklungs-, Test- und Bereitstellungsressourcen für Ihr automatisiertes Freigabeverfahren helfen. Sie können Ihrer Pipeline Aktionen hinzufügen, die sich in einer anderenAWS Region als Ihrer Pipeline befinden. Wenn an der Anbieter für eine AktionAWS-Service ist und sich dieser Aktionstyp/Anbietertyp in einer anderenAWS Region als Ihrer Pipeline befindet, handelt es sich um eine regionsübergreifende Aktion.

> **Note**
>
> Regionalübergreifende Aktionen werden unterstützt und können nur in denAWS Regionen ins Leben gerufen werden, in denenCodePipeline dies unterstützt wird. Eine Liste der unterstütztenAWS Regionen fürCodePipeline finden Sie unterKontingente in AWS CodePipeline.

Sie können die Konsole, die AWS CLI oder AWS CloudFormation verwenden, um regionsübergreifende Aktionen in Pipelines hinzuzufügen.

> **Note**
>
> Bestimmte Aktionstypen sindCodePipeline möglicherweise nur in bestimmtenAWS Regionen verfügbar. Beachten Sie auch, dass esAWS Regionen geben kann, in denen ein Aktionstyp verfügbar ist, ein bestimmterAWS 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**
>
> CodePipelineübernimmt das Kopieren von Artefakten von einerAWS Region in die anderen Regionen, wenn regionsübergreifende Aktionen ausgeführt werden.

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 AWS CLI, AWS CloudFormation oder ein SDK zur Erstellung einer Pipeline oder von regionsübergreifenden Aktionen verwenden, geben Sie den Artefakt-Bucket für jede Region an, in der Sie Aktionen haben.

> **Note**
>
> Sie müssen den Artefakt-Bucket und den Verschlüsselungsschlüssel in derselbenAWS Region wie die regionsübergreifende Aktion und in demselben Konto wie Ihre Pipeline erstellen.

Für die folgenden Aktionstypen können Sie keine regionsübergreifenden Aktionen erstellen:

- Quellaktionen
- Drittanbieteraktionen

- Benutzerdefinierte Aktionen

> **Note**
>
> Wenn Sie die regionsübergreifende Lambda-Aufrufaktion in verwendenCodePipeline, PutJobFailureResultsollte der Status der Lambda-Ausführung mit dem PutJobSuccessResultund an dieAWS Region gesendet werden, in der die Lambda-Funktion vorhanden ist, und nicht an die Region, in der sieCodePipeline vorhanden ist.

Wenn eine Pipeline eine regionsübergreifende Aktion als Teil einer Stufe enthält, repliziert CodePipeline nur die Eingabe-Artefakte der regionsübergreifenden Aktion aus der Pipeline-Region in der Region der Aktion.

> **Note**
>
> Die Pipeline-Region und die Region, in der Ihre Ressourcen zur Erkennung vonCloudWatch Ereignisänderungen verwaltet werden, bleiben unverändert. Die Region, in der Ihre Pipeline gehostet wird, ändert sich nicht.

Verwalten von regionsübergreifenden Aktionen in einer Pipeline (Konsole)

Sie können einer bestehenden Pipeline über die CodePipeline-Konsole eine regionsübergreifende Aktion hinzufügen. Zur Erstellung einer neuen Pipeline mit regionsübergreifenden Aktionen mithilfe des Assistenten zum Erstellen einer Pipeline vgl. Erstellen einer Pipeline (Konsole).

In der Konsole erstellen Sie eine regionsübergreifende Aktion in einer Pipeline-Phase, indem Sie den Anbieter der Aktion und das Feld Region auswählen, das die von Ihnen in der jeweiligen Region für diesen Anbieter erstellten Ressourcen auflistet. Wenn Sie eine regionsübergreifende Aktion hinzufügen, verwendet CodePipeline einen separaten Artefakt-Bucket in der Region der Aktion. Weitere Informationen zu regionsübergreifenden Artefakt-Buckets finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline-Phase (Konsole)

Verwenden Sie die Konsole zum Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline.

> **Note**
>
> Wenn die Pipeline ausgeführt wird, während Änderungen gespeichert werden, wird diese Ausführung nicht abgeschlossen.

So fügen Sie eine regionsübergreifende Aktion hinzu

1. Melden Sie sich bei der Konsole unter http://console.aws.amazon.com/codesuite/codepipeline an.
2. Wählen Sie Ihre Pipeline und dann Edit (Bearbeiten).
3. Wählen Sie unten im Diagramm + Add stage (Phase hinzufügen), wenn Sie eine neue Phase erstellen, oder Edit stage (Phase bearbeiten), wenn Sie die Aktion einer bestehenden Phase hinzufügen möchten.
4. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) die Option + Add action group (Aktionsgruppe hinzufügen), um eine serielle Aktion hinzuzufügen. Oder wählen Sie +Add action (Aktion hinzufügen), um eine parallele Aktion hinzuzufügen.
5. Führen Sie auf der Seite Edit action (Aktion bearbeiten) Folgendes durch:

    a. Geben Sie unter Action name (Aktionsname) einen Namen für die regionsübergreifende Aktion ein.

    b. Wählen Sie unter Action provider (Aktionsanbieter) den Anbieter der Aktion.

    c. Wählen Sie unter Region dieAWS Region aus, in der Sie die Ressource für die Aktion erstellt haben oder in der Sie sie erstellen möchten. Wenn die Region ausgewählt ist, werden die verfügbaren Ressourcen für diese Region zur Auswahl aufgelistet. Das Feld Region gibt an, wo dieAWS Ressourcen für diesen Aktionstyp und Anbietertyp erstellt werden. Dieses Feld wird nur für Aktionen angezeigt, bei denen der Aktionsanbieter ein istAWS-Service. Das Feld Region ist standardmäßig auf dasselbeAWS-Region wie Ihre Pipeline eingestellt.

    d. Wählen Sie unter Input artifacts (Eingabeartefakte) die korrekte Eingabe aus der vorherigen Phase. Wenn es sich bei der vorherigen Phase beispielsweise um eine Quellphase handelt, wählen Sie SourceArtifact.

    e. Füllen Sie alle erforderlichen Felder für den Aktionsanbieter aus, den Sie konfigurieren.

f. Wählen Sie unter Output artifacts (Ausgabeartefakte) die korrekte Ausgabe zur nächsten Phase. Wenn es sich bei der nächsten Phase beispielsweise um eine Bereitstellungsphase handelt, wählen Sie BuildArtifact.

g. Wählen Sie Speichern.

6. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) Done (Fertig).
7. Wählen Sie Speichern.

Bearbeiten einer regionsübergreifenden Aktion in einer Pipeline-Phase (Konsole)

Verwenden Sie die Konsole zur Bearbeitung einer vorhandenen regionsübergreifenden Aktion in einer Pipeline.

> **Note**
>
> Wenn die Pipeline ausgeführt wird, während Änderungen gespeichert werden, wird diese Ausführung nicht abgeschlossen.

So bearbeiten Sie eine regionsübergreifende Aktion

1. Melden Sie sich unter https://console.aws.amazon.com/codesuite/codepipeline/home. bei der Konsole an.
2. Wählen Sie Ihre Pipeline und dann Edit (Bearbeiten).
3. Wählen Sie Edit stage (Phase bearbeiten).
4. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) das Symbol zur Bearbeitung einer vorhandenen Aktion.
5. Nehmen Sie auf der Seite Edit action (Aktion bearbeiten) die erforderlichen Änderungen an den Feldern vor.
6. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) Done (Fertig).
7. Wählen Sie Speichern.

Löschen einer regionsübergreifenden Aktion aus einer Pipeline-Phase (Konsole)

Verwenden Sie die Konsole zum Löschen einer vorhandenen regionsübergreifenden Aktion aus einer Pipeline.

> **Note**
>
> Wenn die Pipeline ausgeführt wird, während Änderungen gespeichert werden, wird diese Ausführung nicht abgeschlossen.

So löschen Sie eine regionsübergreifende Aktion

1. Melden Sie sich bei der Konsole unter http://console.aws.amazon.com/codesuite/codepipeline an.
2. Wählen Sie Ihre Pipeline und dann Edit (Bearbeiten).
3. Wählen Sie Edit stage (Phase bearbeiten).
4. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) das Symbol zum Löschen einer vorhandenen Aktion.
5. Wählen Sie unter Edit: <Stage> (Bearbeiten: <Phase>) Done (Fertig).
6. Wählen Sie Speichern.

Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline (CLI)

Sie können einer bestehenden Pipeline über die AWS CLI eine regionsübergreifende Aktion hinzufügen.

Um mit der AWS CLI eine regionsübergreifende Aktion in einer Pipeline-Phase zu erstellen, fügen Sie die Konfigurationsaktion sowie ein optionales `region`-Feld hinzu. Außerdem müssen Sie bereits einen Artefakt-Bucket in der Region der Aktion erstellt haben. Anstatt den Parameter `artifactStore` der Einzelregionpipeline anzugeben, verwenden Sie den Parameter `artifactStores`, um eine Liste der Artefakt-Buckets jeder Region einzufügen.

> **Note**
>
> In dieser Anleitung und den entsprechenden Beispielen ist *`RegionA`* die Region, in der die Pipeline erstellt wird. Es hat Zugriff auf den Amazon S3 S3-Bucket der *`RegionA`*, der zum Speichern von Pipeline-Artefakten verwendet wird, und auf die von verwendete ServicerolleCodePipeline. *`RegionB`* ist die Region, in der die CodeDeploy-Anwendung, Bereitstellungsgruppe und Servicerolle, die von CodeDeploy verwendet werden, erstellt werden.

Voraussetzungen

Sie müssen Folgendes erstellt haben:

- Eine Pipeline in *RegionA*
- Ein Amazon S3 S3-Artefakt-Bucket in *RegionB*
- Die Ressourcen für Ihre Aktion, wie etwa Ihre CodeDeploy-Anwendungs- und -Bereitstellungsgruppe für eine regionsübergreifende Bereitstellungsaktion, in *RegionB*.

Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline (CLI)

Verwenden Sie die AWS CLI zum Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline.

So fügen Sie eine regionsübergreifende Aktion hinzu

1. Führen Sie für eine Pipeline in *RegionA* den Befehl get-pipeline zum Kopieren der Pipeline-Struktur in eine JSON-Datei aus. Geben Sie für eine Pipeline mit dem Namen `MyFirstPipeline` den folgenden Befehl ein:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Dieser Befehl gibt nichts zurück. Die erstellte Datei sollte jedoch in dem Verzeichnis auftauchen, in dem Sie den Befehl ausgeführt haben.
2. Fügen Sie das Feld `region` zum Hinzufügen einer neuen Phase mit Ihrer regionsübergreifenden Aktion hinzu, die die Region und Ressourcen für Ihre Aktion enthält. Das folgende JSON-Beispiel fügt eine Bereitstellungsphase mit einer regionsübergreifenden Bereitstellungsaktion, bei der der Anbieter CodeDeploy ist, einer neuen Region `us-east-1` hinzu.

   ```
   {
                   "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. Das Mapping muss einen Eintrag für jedeAWS Region enthalten, in der Sie Aktionen haben. Für jeden Eintrag im Mapping müssen sich die Ressourcen in der jeweiligenAWS Region befinden. In dem Beispiel unten ist `ID-A` die Verschlüsselungsschlüssel-ID für *`RegionA`* und `ID-B` die Verschlüsselungsschlüssel-ID für *`RegionB`*.

```
"artifactStores":{
   "RegionA":{
      "encryptionKey":{
         "id":"ID-A",
         "type":"KMS"
      },
      "location":"Location1",
      "type":"S3"
   },
   "RegionB":{
      "encryptionKey":{
         "id":"ID-B",
         "type":"KMS"
      },
      "location":"Location2",
      "type":"S3"
   }
}
```

Das folgende JSON-Beispiel zeigt den us-west-2-Bucket als `my-storage-bucket` und fügt den neuen us-east-1-Bucket mit dem Namen `my-storage-bucket-us-east-1` hinzu.

```
        "artifactStores": {
            "us-west-2": {
                "type": "S3",
                "location": "my-storage-bucket"
            },
            "us-east-1": {
                "type": "S3",
                "location": "my-storage-bucket-us-east-1"
            }
        },
```

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

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

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

   Speichern Sie die Datei.

5. Um Ihre Änderungen zu übernehmen, führen Sie den Befehl update-pipeline aus und geben Sie die Pipeline-JSON-Datei an:

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

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

   Dieser Befehl gibt die gesamte Struktur der bearbeiteten Pipeline zurück. Die Ausgabe sieht folgendermaßen oder ähnlich aus.

```
{
    "pipeline": {
        "version": 4,
        "roleArn": "ARN",
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "inputArtifacts": [],
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeCommit"
                        },
                        "outputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "configuration": {
                            "PollForSourceChanges": "false",
                            "BranchName": "main",
                            "RepositoryName": "MyTestRepo"
                        },
                        "runOrder": 1
                    }
                ]
            },
            {
                "name": "Deploy",
                "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "name": "Deploy",
                        "region": "us-east-1",
                        "actionTypeId": {
```

```
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "name",
                            "DeploymentGroupName": "name"
                        },
                        "runOrder": 1
                    }
                ]
            }
        ],
        "name": "AnyCompanyPipeline",
        "artifactStores": {
            "us-west-2": {
                "type": "S3",
                "location": "my-storage-bucket"
            },
            "us-east-1": {
                "type": "S3",
                "location": "my-storage-bucket-us-east-1"
            }
        }
    }
}
```

> **Note**
>
> Der Befehl update-pipeline stoppt die Pipeline. Wenn eine Revision über die Pipeline ausgeführt wird, wenn Sie den Befehl update-pipeline ausführen, wird diese Ausführung gestoppt. Sie müssen die Ausführung der Pipeline manuell starten, um die Revision über die aktualisierte Pipeline auszuführen. Verwenden Sie den **`start-pipeline-execution`**-Befehl, um Ihre Pipeline manuell zu starten.

6. Nachdem Sie Ihre Pipeline aktualisiert haben, wird die regionsübergreifende Aktion in der Konsole angezeigt.

Hinzufügen einer regionsübergreifenden Aktion zu einer Pipeline (AWS CloudFormation)

Sie können einer bestehenden Pipeline über die AWS CloudFormation eine regionsübergreifende Aktion hinzufügen.

So fügen Sie eine regionsübergreifende Aktion mit AWS CloudFormation hinzu

1. Fügen Sie den `Region`-Parameter der `ActionDeclaration`-Ressource in Ihrer Vorlage hinzu, wie in diesem Beispiel gezeigt:

```
ActionDeclaration:
              Type: Object
              Properties:
                ActionTypeId:
                  Type: ActionTypeId
                  Required: true
                Configuration:
                  Type: Map
                InputArtifacts:
                  Type: Array
                  ItemType:
                    Type: InputArtifact
                Name:
                  Type: String
                  Required: true
                OutputArtifacts:
                  Type: Array
```

```
                ItemType:
                  Type: OutputArtifact
              RoleArn:
                Type: String
              RunOrder:
                Type: Integer
              Region:
                Type: String
```

2. Fügen Sie unter `Mappings` die Regionskarte, wie in diesem Beispiel gezeigt, für eine Zuordnung mit dem Namen `SecondRegionMap` hinzu, die Werte für die Schlüssel `RegionA` und `RegionB` zuordnet. Fügen Sie unter der `Pipeline`-Ressource dem `artifactStore`-Feld die `artifactStores`-Zuweisung für Ihre neue regionsübergreifende Aktion wie folgt hinzu:

```
Mappings:
  SecondRegionMap:
    RegionA:
      SecondRegion: "RegionB"
    RegionB:
      SecondRegion: "RegionA"

...

          Properties:
            ArtifactStores:
              -
                Region: RegionB
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-RegionB
              -
                Region: RegionA
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-RegionA
```

Das folgende YAML-Beispiel zeigt den *RegionA*-Bucket `us-west-2` und fügt den neuen *RegionB*-Bucket `eu-central-1` hinzu:

```
Mappings:
  SecondRegionMap:
```

```
    us-west-2:
      SecondRegion: "eu-central-1"
    eu-central-1:
      SecondRegion: "us-west-2"

...

          Properties:
            ArtifactStores:
              -
                Region: eu-central-1
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-eu-central-1
              -
                Region: us-west-2
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-us-west-2
```

3. Speichern Sie die aktualisierte Vorlage auf Ihrem lokalen Computer und öffnen Sie die AWS CloudFormation-Konsole.
4. Wählen Sie Ihren Stack aus und klicken Sie auf Create Change Set for Current Stack (Änderungssatz für laufenden Stack erstellen).
5. Laden Sie die Vorlage hoch und zeigen Sie dann die in AWS CloudFormation aufgeführten Änderungen an. Dies sind die Änderungen, die am Stack vorgenommen werden sollen. Ihre neuen Ressourcen sollten in der Liste angezeigt werden.
6. Wählen Sie Execute (Ausführen).

Arbeiten mit Variablen

Einige Aktionen in CodePipeline Generieren von Variablen. 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: Verwenden von Variablen mit Lambda-Aufrufaktionen.
- Ein Tutorial mit einerAWS CloudFormation Aktion, die auf Stack-Ausgabevariablen aus einer CloudFormation Upstream-Aktion verweist, finden Sie unterTutorial: Erstellen einer Pipeline, die Variablen aus AWS CloudFormation-Bereitstellungsaktionen verwendet.
- Ein Beispiel für eine manuelle Genehmigungsaktion mit Nachrichtentext, der auf Ausgabevariablen verweist, die sich auf die CodeCommit Commit-ID und die Commit-Nachricht auflösen, finden Sie unterBeispiel: Variablen in manuellen Genehmigungen verwenden.
- Eine CodeBuild Beispielaktion mit einer Umgebungsvariablen, die sich in den GitHub Zweignamen auflöst, 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 Aktionsausgabevariablen. Eine Liste der Umgebungsvariablen CodeBuild, in denen Sie sie verwenden können, finden Sie unter Umgebungsvariablen in Build-Umgebungen im AWS CodeBuildBenutzerhandbuch.

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 verbraucht.

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.

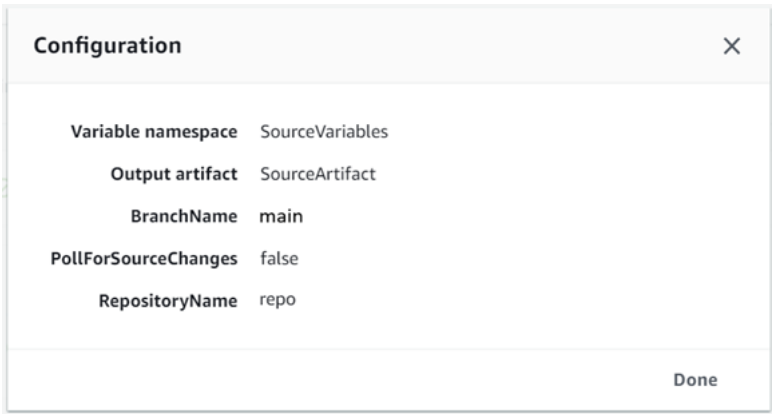
So erstellen Sie eine Pipeline mit Variablen

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie Create pipeline (Pipeline erstellen) aus. Geben Sie einen Namen für die Pipeline ein und wählen Sie anschließend Next (Weiter) aus.
3. Wählen Sie unter Quelle unter Anbieter die Option CodeCommit. Wählen Sie das CodeCommit Repository und den Zweig für die Quellaktion aus, und wählen Sie dann Weiter.
4. Wählen Sie in Build unter Provider CodeBuild. Wählen Sie einen Namen für ein vorhandenes CodeBuild Build-Projekt oder wählen Sie Projekt erstellen. 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 variablen Syntax`#{codepipeline.PipelineExecutionId}` und die Commit-ID mit der variablen Syntax ein`#{SourceVariables.CommitId}`.

   > **Note**
   >
   > Sie können in jedes Aktionskonfigurationsfeld des Assistenten Variablensyntax eingeben.

5. Wählen Sie Create (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 anAWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wählen Sie die Pipeline aus, die Sie bearbeiten möchten, und anschließend Edit (Bearbeiten). Wählen Sie für die Quellphase Edit stage (Phase bearbeiten) aus. Füge 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 anAWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Wenn die Pipeline erstellt wurde und erfolgreich ausgeführt wird, können Sie die Variablen auf der Seite Action execution details (Details der Aktionsausführung) anzeigen. Weitere Informationen finden Sie unter Anzeigen von Variablen (Konsole).

Konfigurieren von Aktionen für Variablen (CLI)

Wenn Sie den Befehl create-pipeline zum Erstellen einer Pipeline oder den Befehl update-pipeline zum Bearbeiten einer Pipeline verwenden, können Sie Variablen in der Konfiguration einer Aktion referenzieren/verwenden.

Wenn kein Namespace angegeben ist, können die von dieser Aktion erzeugten Variablen in Konfigurationen für nachgeschaltete Aktionen nicht referenziert werden.

So konfigurieren Sie eine Aktion mit einem Namespace

1. Befolgen Sie die Schritte unter Erstellen einer Pipeline in CodePipeline zum Erstellen einer Pipeline über die CLI. Starten Sie eine Eingabedatei, um den Befehl create-pipeline mit dem Parameter `--cli-input-json` bereitzustellen. Fügen Sie in der Pipelinestruktur den Parameter `namespace` hinzu und geben Sie einen Namen an, z. B. `SourceVariables`.

```
. . .
{
          "inputArtifacts": [],
          "name": "Source",
          "region": "us-west-2",
          "namespace": "SourceVariables",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeCommit"
          },
          "outputArtifacts": [

. . .
```

2. Speichern Sie die Datei mit einem Namen wie **`MyPipeline.json`**.
3. Führen Sie auf einem Terminal (Linux, macOS oder Unix) den Befehl 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 auf einem Terminal (Linux, macOS oder Unix) den Befehl 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

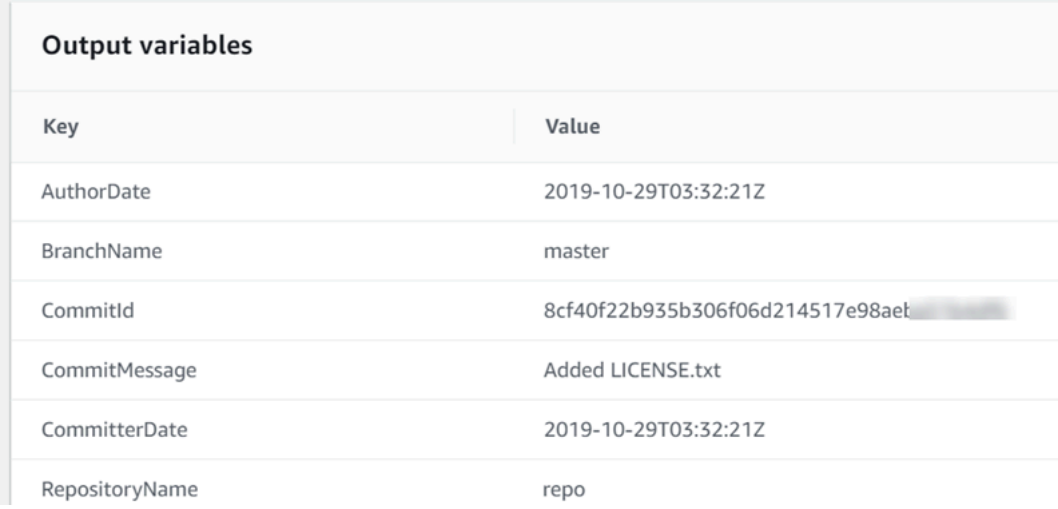
Sie können die Details der Aktionsausführung anzeigen, um die Variablen für diese Aktion anzuzeigen, spezifisch für jede Ausführung.

Anzeigen von Variablen (Konsole)

Sie können die Konsole verwenden, um Variablen für eine Aktion anzuzeigen.

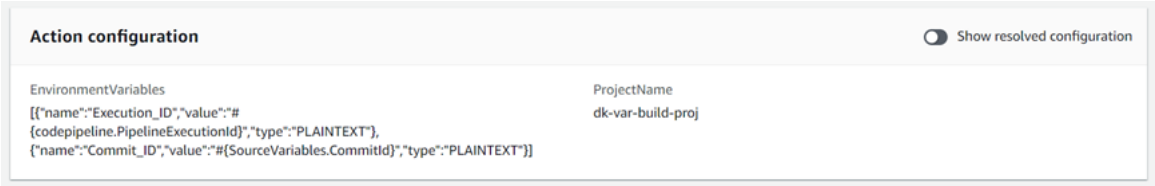
1. Melden Sie sich bei der anAWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline.
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 zur Aktionsausführung für die CodeCommit Aktion anzuzeigen. Sie können die Variablen im Bildschirm „Action detail (Aktionsdetails)“ unter Output variables (Ausgabevariablen) anzeigen.

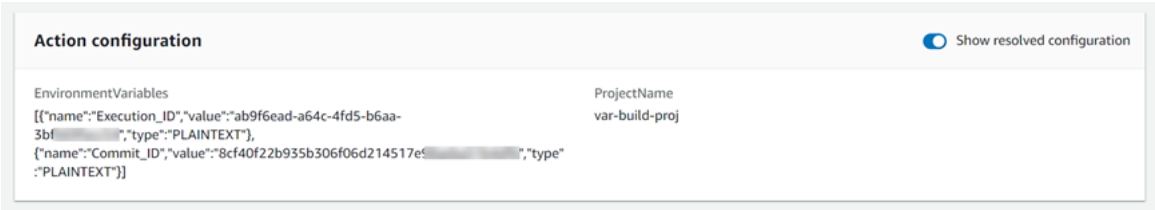


5. Nach erfolgreicher Ausführung der Pipeline können Sie die von der Build-Aktion erzeugten Variablen anzeigen. Wählen Sie View history (Verlauf anzeigen). Wählen Sie in der Aktionsliste

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 sehen:

```
{
    "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 anAWS Management Console und öffnen Sie die CodePipeline Konsole unter http://console.aws.amazon.com/codesuite/codepipeline/home.

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt. 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 die variable Syntax für`CommitId` zu Ihrer CodeCommit URL 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 unter Kommentare`CommitMessage`, in 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 für die Verwendung in Ihren Build-Befehlen erstellen.

Dieses Beispiel zeigt Ihnen, wie Sie eine Ausgabevariablensyntax aus einer GitHub Quellaktion zu einer CodeBuild Umgebungsvariablen hinzufügen. Die Syntax der Ausgabevariablen in diesem Beispiel stellt die Ausgangsvariable der GitHub Quellaktion für dar`BranchName`. Nachdem die Aktion erfolgreich ausgeführt wurde, wird die Variable aufgelöst, sodass der GitHub Zweigname angezeigt wird.

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

   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt. 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:

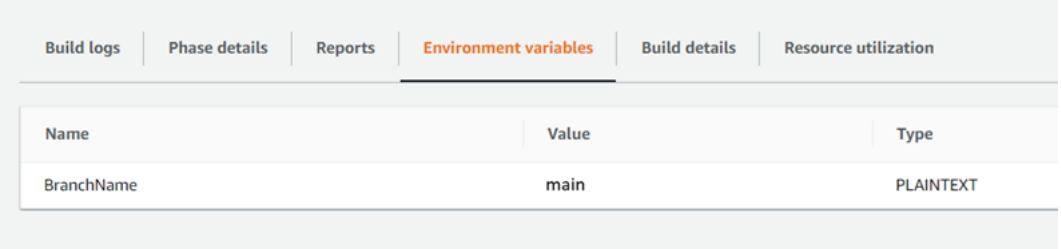
   - Geben Sie unter Name) einen Namen für Ihre Umgebungsvariable ein.
   - Geben Sie im Feld Wert die Variablensyntax für Ihre Pipeline-Ausgabevariable ein, die den Namespace enthält, der Ihrer Quellaktion zugewiesen ist. Die Syntax der Ausgabevariablen für eine GitHub Aktion mit dem Standardnamespace`SourceVariables` lautet beispielsweise`#{SourceVariables.BranchName}`.
   - Wählen Sie unter Typ die Option Klartext aus.
5. Nachdem die Pipeline erfolgreich ausgeführt wurde, können Sie sehen, wie die aufgelöste Ausgabevariable der Wert in der Umgebungsvariablen ist. Wählen Sie eine der folgenden Optionen aus:

   - CodePipeline Konsole: Wählen Sie Ihre Pipeline und dann Verlauf. Wählen Sie die letzte Pipeline-Ausführung aus.
     - Wählen Sie unter Timeline den Selektor für Quelle aus. Dies ist die Quellaktion, die GitHub Ausgabevariablen generiert. Wählen Sie Ausführungsdetails anzeigen. Sehen Sie sich unter Ausgabevariablen die Liste der Ausgabevariablen an, die durch diese Aktion generiert wurden.
     - 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. Sehen Sie sich unter Aktionskonfiguration Ihre CodeBuild Umgebungsvariablen an. Wählen Sie „Gelöste Konfiguration anzeigen“. Ihr Umgebungsvariablenwert ist die

aufgelösteBranchName Ausgabevariable aus der GitHub Quellaktion. In diesem Beispiel ist der aufgelöste Wertmain.

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 NameBranchName und der Wert die aufgelösteBranchName Ausgabevariable aus der GitHub Quellaktion. In diesem Beispiel ist der aufgelöste Wertmain.



Arbeiten mit Phasenübergängen 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 mit der AWS CodePipeline-Konsole oder der AWS CLI Übergänge zwischen Phasen in einer Pipeline deaktivieren oder aktivieren.

> **Note**
>
> Sie können eine Genehmigungsaktion verwenden, um die Ausführung einer Pipeline anzuhalten, bis die Fortsetzung manuell genehmigt wird. Weitere Informationen finden Sie unter Genehmigungsaktionen verwalten in CodePipeline .

Themen

- Deaktivieren oder Aktivieren von Übergängen (Konsole)
- Deaktivieren oder Aktivieren von Übergängen (CLI)

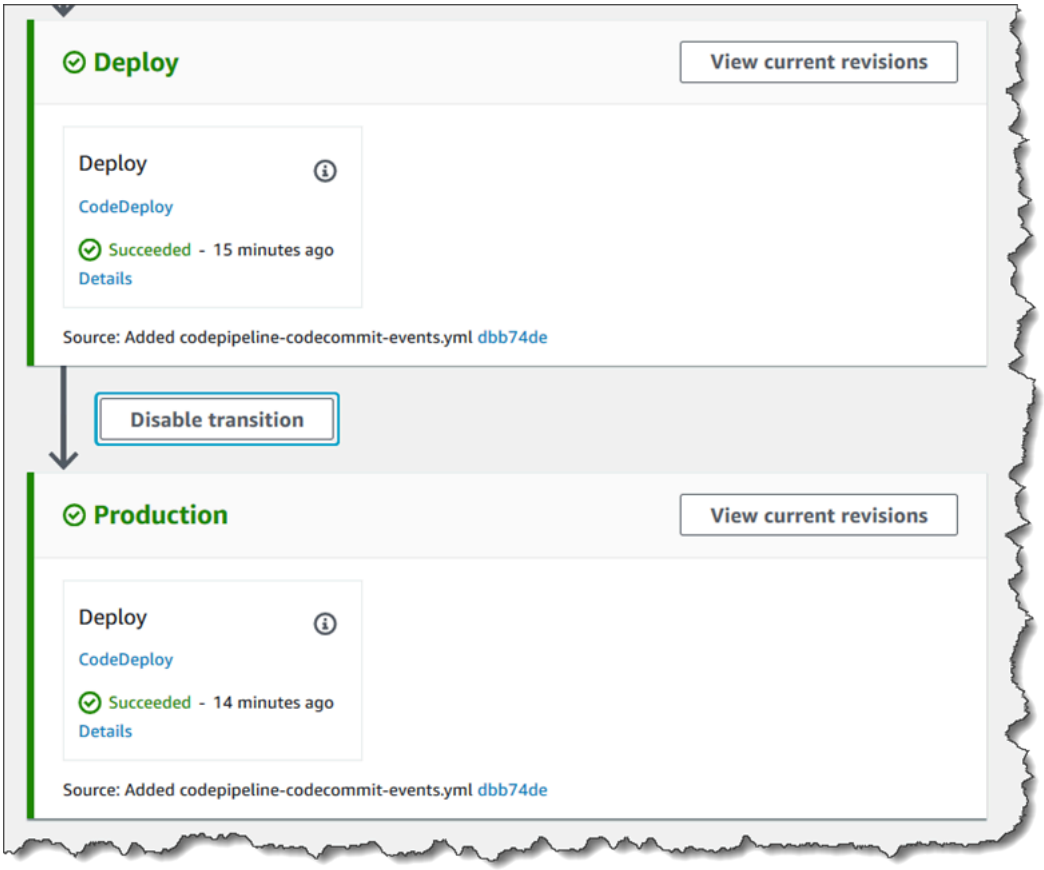
Deaktivieren oder Aktivieren von Übergängen (Konsole)

So deaktivieren oder aktivieren Sie Übergänge in einer Pipeline

1. Melden Sie sich beim anAWS Management ConsoleÖffnen Sie die CodePipeline-Konsole unterhttp://console.aws.amazon.com/codesuite/codepipeline/homeaus.

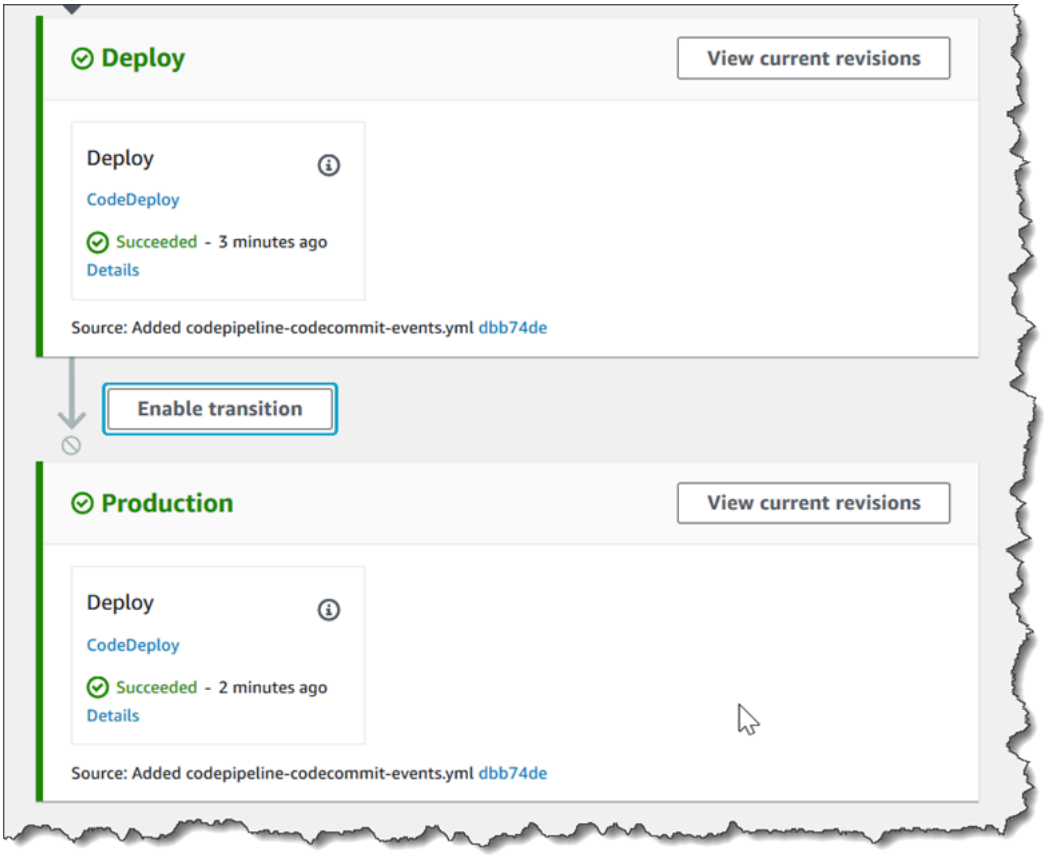
   Es werden die Namen aller Ihrem AWS-Konto zugeordneten Pipelines angezeigt.
2. Wählen Sie im Feld Name den Namen der Pipeline aus, für die Sie Übergänge aktivieren oder deaktivieren möchten. Eine detaillierte Ansicht der Pipeline wird geöffnet (einschließlich der Übergänge zwischen den Stufen der Pipeline).
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 nach der Aktivierung eines Übergangs einige Sekunden dauern, bis Änderungen in der CodePipeline-Konsole angezeigt werden.

Deaktivieren oder Aktivieren von Übergängen (CLI)

Um einen Übergang zwischen Stufen unter Verwendung der AWS CLI zu deaktivieren, führen Sie den Befehl disable-stage-transition 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 die Eingabeaufforderung (Windows) und verwenden Sie dieAWS CLISo führen Sie das aus:Deaktivieren-Stage-ÜbergangGeben Sie den Namen der Pipeline, den Namen der zu deaktivierenden Stufe, den Übergangstyp und den Grund für die Deaktivierung des Übergangs auf die 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 mit dem Namen zu deaktivieren*Staging*in einer Pipeline namens*MyFirstPipeline*Geben Sie einen Befehl ähnlich wie den folgenden ein:

   ```
   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 das aus:get-pipeline-statebefehl. Weitere Informationen finden Sie unter Pipelines anzeigen (Konsole) und Anzeigen von der Pipeline-Details und des Verlaufs (CLI).

So aktivieren Sie einen Übergang

1. Öffnen Sie ein Terminal (Linux, macOS oder Unix) oder die Eingabeaufforderung (Windows) und verwenden Sie dieAWS CLISo führen Sie das aus:Enable-Stage-ÜbergangGeben Sie den Namen der Pipeline, den Namen der zu aktivierenden Stufe und den Übergangstyp an.

   Um beispielsweise den Übergang zu einer Stufe mit dem Namen zu ermöglichen*Staging*in einer Pipeline namens*MyFirstPipeline*Geben Sie einen Befehl ähnlich wie den folgenden ein:

   ```
   aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stage-
   name Staging  --transition-type Inbound
   ```

   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 das aus:get-pipeline-statebefehl. Weitere Informationen erhalten Sie unter Pipelines anzeigen (Konsole) und Anzeigen von der Pipeline-Details und des Verlaufs (CLI).

Überwachung von Pipelines

Überwachung ist wichtig, um die Zuverlässigkeit, Verfügbarkeit und Performance von AWS CodePipeline aufrechtzuerhalten. Sie sollten Überwachungsdaten aller Bestandteile Ihrer AWS-Lösung sammeln, damit Sie auftretende Multipunkt-Fehler leichter beheben können. Bevor Sie mit der Überwachung beginnen, sollten Sie einen Überwachungsplan erstellen, der die folgenden Fragen beantwortet:

- Was sind Ihre Ziele bei der Überwachung?
- 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 zum Überwachen Ihrer CodePipeline-Pipelines und ihrer Ressourcen verwenden:

- EventBridgeEventbus-Ereignisse — Sie können CodePipeline Ereignisse in überwachenEventBridge, wodurch Änderungen in Ihrer Pipeline, Phase oder dem Status der Aktionsausführung erkannt werden. EventBridgeleitet diese Daten an Ziele wie AWS Lambda Amazon Simple Notification Service weiter. EventBridgeEreignisse sind dieselben wie die, die in Amazon CloudWatch Events erscheinen.
- Benachrichtigungen für Pipeline-Ereignisse in der Developer Tools-Konsole — Sie können CodePipeline Ereignisse mithilfe von Benachrichtigungen überwachen, die Sie in der Konsole einrichten, und dann ein Amazon Simple Notification Service-Thema und Abonnement für erstellen. Weitere Informationen finden Sie unter Was sind Benachrichtigungen im Developer Tools Console-Benutzerhandbuch.
- AWS CloudTrail— Wird verwendetCloudTrail, um API-Aufrufe zu erfassen, die von oder im Namen von CodePipeline in Ihrem AWS Konto getätigt wurden, und um die Protokolldateien an einen Amazon S3-Bucket zu senden. Sie können festlegen, dass Amazon SNS-Benachrichtigungen CloudWatch veröffentlicht werden, wenn neue Protokolldateien übermittelt 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 Pipelineausführung einzusehen.

Themen

- Überwachen von CodePipeline Ereignissen
- Platzhalter-Bucket-Referenz für Ereignisse
- Protokollierung von CodePipeline-API-Aufrufen mit AWS CloudTrail

Überwachen von CodePipeline Ereignissen

Sie können CodePipeline Ereignisse in überwachen EventBridge, die einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, software-as-a-service (SaaS)-Anwendungen und liefert AWS-Services. EventBridge leitet diese Daten an Ziele wie AWS Lambda und Amazon Simple Notification Service weiter. Diese Ereignisse sind dieselben, die in Amazon CloudWatch Events angezeigt werden, das einen Stream von Systemereignissen in nahezu Echtzeit bereitstellt, der Änderungen an - AWS Ressourcen beschreibt. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? im Amazon- EventBridge Benutzerhandbuch.

> **Note**
>
> Amazon EventBridge ist die bevorzugte Methode zur Verwaltung Ihrer Ereignisse. Amazon CloudWatch Events und EventBridge sind derselbe zugrunde liegende Service und die gleiche API, EventBridge bieten jedoch mehr Funktionen. Änderungen, die Sie entweder in - CloudWatch Ereignissen oder vornehmen EventBridge , werden in jeder Konsole angezeigt.

Ereignisse bestehen aus Regeln. Eine Regel wird konfiguriert, indem Sie Folgendes auswählen:

- Ereignismuster. Jede Regel wird als Ereignismuster mit der Quelle und dem Typ der zu überwachenden Ereignisse und Ereigniszielen ausgedrückt. Um Ereignisse zu überwachen, erstellen Sie eine Regel mit dem Service, den Sie als Ereignisquelle überwachen, z. B. CodePipeline. Sie können beispielsweise eine Regel mit einem Ereignismuster erstellen, das CodePipeline als Ereignisquelle verwendet, 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 Zielservice einrichten, um Benachrichtigungen zu senden, Statusinformationen zu erfassen, Korrekturmaßnahmen zu ergreifen, Ereignisse zu initiieren oder andere Maßnahmen zu ergreifen. Wenn Sie Ihr Ziel hinzufügen, müssen Sie auch Berechtigungen erteilen EventBridge , damit es den ausgewählten Zielservice 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 an.
- Der `version`-Eintrag unter Pipeline-`detail` zeigt die Versionsnummer der Pipelinestruktur.
- Der `execution-id`-Eintrag unter Pipeline-`detail` zeigt die Ausführungs-ID für die Pipelineausführung, die die Statusänderung verursacht hat. Weitere Informationen finden Sie im APIGetPipelineExecution-Aufruf in der APIAWS CodePipeline -Referenz zu .
- Der `pipeline-execution-attempt` Eintrag zeigt die Anzahl der Versuche oder Wiederholungen für die spezifische Ausführungs-ID an.

CodePipeline meldet ein Ereignis an , EventBridge wenn sich der Status einer Ressource in Ihren AWS-Konto ändert. Ereignisse werden auf einer garantierten at-least-once Basis für die folgenden Ressourcen ausgegeben:

- Pipeline-Ausführungen
- Stufenausführungen
- Aktionsausführungen

Ereignisse werden von EventBridge mit dem oben beschriebenen Ereignismuster und Schema ausgegeben. Bei verarbeiteten Ereignissen, z. B. Ereignissen, die Sie über Benachrichtigungen erhalten, die Sie in der Entwicklertools-Konsole konfiguriert haben, enthält die Ereignisnachricht Ereignismusterfelder mit einer gewissen Variation. Das `detail-type` Feld wird beispielsweise in konvertiert`detailType`. Weitere Informationen finden Sie im APIPutEvents-Aufruf in der Amazon EventBridge -API-Referenz .

Die folgenden Beispiele zeigen Ereignisse für CodePipeline. Wenn möglich, zeigt jedes Beispiel das Schema für ein ausgegebenes Ereignis zusammen mit dem Schema für ein verarbeitetes Ereignis.

Themen

- Detailtypen
- Ereignisse auf Pipeline-Ebene
- Ereignisse auf Stufenebene
- Ereignisse auf Aktionsebene
- Erstellen einer Regel, die eine Benachrichtigung über ein Pipeline-Ereignis 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 in konvertiert wird, `detailType` wenn Ereignisse verarbeitet werden.

Detailtyp	Status	Beschreibung
CodePipeline Statusänderung der Pipeline-Ausführung	CANCELED	Die Pipelineausführung wurde abgebrochen, da die Pipelinestruktur aktualisiert wurde.
	FEHLGESCHLAGEN	Die Pipelineausführung wurde nicht erfolgreich abgeschlossen.
	RESUMED	Eine fehlgeschlagene Pipelineausführung wurde als Reaktion auf den API-Aufruf `RetryStageExecution` erneut ausgeführt.
	STARTED	Die Pipelineausführung wird derzeit ausgeführt.

Detailtyp	Status	Beschreibung
	STOPPED	Der Anhaltevorgang ist abgeschlossen, und die Pipeline-Ausführung wird angehalten.
	STOPPING	Die Pipeline-Ausführung wird aufgrund einer Anforderung mit "Anhalten und beenden" oder "Anhalten und warten" angehalten.
	SUCCEEDED	Die Pipelineausführung wurde erfolgreich abgeschlossen.
	SUPERSEDED	Während auf den Abschluss der nächsten Stufe dieser Pipelineausführung gewartet wurde, wurde stattdess en eine neuere Pipelineausführung über die Pipeline fortgesetzt.
CodePipeline Statusänderung der Bühnenaus führung	CANCELED	Die Stufe wurde abgebrochen, da die Pipelinestruktur aktualisiert wurde.
	FEHLGESCH LAGEN	Die Stufe wurde nicht erfolgreich abgeschlossen.
	RESUMED	Eine fehlgeschlagene Stufe wurde als Reaktion auf den API-Aufruf `RetryStageExecution` erneut ausgeführt.
	STARTED	Die Stufe wird derzeit ausgeführt.
	STOPPED	Der Anhaltevorgang ist abgeschlossen, und die Phasenausführung wird angehalten.
	STOPPING	Die Ausführung der Phase wird aufgrund einer Anforderu ng zum Anhalten und Warten oder zum Anhalten und Beenden der Pipeline-Ausführung gestoppt.
	SUCCEEDED	Die Stufe wurde erfolgreich abgeschlossen.
CodePipeline Statusänderung der Aktionsau sführung	ABBRUCH	Die Aktion wird aufgrund einer Anforderung mit "Anhalten und Beenden" für die Pipeline-Ausführung beendet.

Detailtyp	Status	Beschreibung
	CANCELED	Die Aktion wurde abgebrochen, da die Pipelinestruktur aktualisiert wurde.
	FEHLGESCHLAGEN	Bei Genehmigungsaktionen bedeutet der Status FAILED (FEHLGESCHLAGEN), dass die Aktion entweder vom Prüfer abgelehnt wurde oder aufgrund einer falschen Aktionskonfiguration fehlgeschlagen ist.
	STARTED	Die Aktion wird derzeit ausgeführt.
	SUCCEEDED	Die Aktion wurde erfolgreich abgeschlossen.

Ereignisse auf Pipeline-Ebene

Ereignisse auf Pipeline-Ebene werden ausgegeben, wenn es eine Statusänderung für eine Pipeline-Ausführung gibt.

Themen

- Pipeline STARTED-Ereignis
- Pipeline-STOPPING-Ereignis
- Pipeline-Erfolgreiches Ereignis
- Pipeline SUCCEEDED (Beispiel mit Git-Tags)
- Pipeline-FEHLGESCHLAGEN-Ereignis
- Pipeline FEHLGESCHLAGEN (Beispiel mit Git-Tags)

Pipeline STARTED-Ereignis

Wenn eine Pipeline-Ausführung gestartet wird, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `"myPipeline"` in der `us-east-1` Region . Das `id` Feld steht für die Ereignis-ID und das `account` Feld für die Konto-ID, in 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,
        "pipeline-execution-attempt": 1
    }
}
```

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,
        "pipeline-execution-attempt": 1
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

Pipeline-STOPPING-Ereignis

Wenn eine Pipeline-Ausführung beendet wird, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `myPipeline` in der `us-west-2` Region .

```
{
    "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,
        "pipeline-execution-attempt": 1
        "stop-execution-comments": "Stopping the pipeline for an update"
    }
}
```

Pipeline-Erfolgreiches Ereignis

Wenn eine Pipeline-Ausführung erfolgreich ist, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `myPipeline` in der `us-east-1` Region .

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,
        "pipeline-execution-attempt": 1
    }
}
```

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,
        "pipeline-execution-attempt": 1
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

Pipeline SUCCEEDED (Beispiel mit Git-Tags)

Wenn eine Pipeline-Ausführung eine Phase hat, die erneut versucht wurde und erfolgreich war, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `myPipeline` in der `eu-central-1` Region, in der 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
    }
}
```

Pipeline-FEHLGESCHLAGEN-Ereignis

Wenn eine Pipeline-Ausführung fehlschlägt, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `"myPipeline"` in der `us-west-2` Region .

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,
        "pipeline-execution-attempt": 1
    }
```

```
}
```

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,
        "pipeline-execution-attempt": 1
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "failedActionCount": 1,
        "failedActions": [
            {
                "action": "Deploy",
                "additionalInformation": "Deployment <ID> failed"
            }
        ],
        "failedStage": "Deploy"
    }
```

Pipeline FEHLGESCHLAGEN (Beispiel mit Git-Tags)

Sofern es in der Quellphase nicht fehlschlägt, gibt es für eine Pipeline, die mit Auslösern konfiguriert ist, ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `myPipeline` in der `eu-central-1` Region, in der 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.

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,
        "pipeline-execution-attempt": 1
    }
}
```

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,
        "pipeline-execution-attempt": 1
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "failedActionCount": 1,
        "failedActions": [
            {
                "action": "Deploy",
                "additionalInformation": "Deployment <ID> failed"
            }
        ],
        "failedStage": "Deploy"
```

```
    }
```

Ereignisse auf Stufenebene

Ereignisse auf Stufenebene werden ausgegeben, wenn es eine Statusänderung für eine Stufenausführung gibt.

Themen

- Ereignis STARTED inszenieren
- Ereignis STOPPING inszenieren
- Ereignis „Phase angehalten“
- Phase wurde nach dem Wiederholungsereignis der Stufe wieder aufgenommen

Ereignis STARTED inszenieren

Wenn eine Stufenausführung gestartet wird, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `"myPipeline"` in der `us-east-1` Region für die Stufe `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",
        "execution-id": 12345678-1234-5678-abcd-12345678abcd,
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Prod",
        "state": "STARTED",
```

```
        "pipeline-execution-attempt": 1
    }
}
```

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,
        "pipeline-execution-attempt": 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 STOPPING inszenieren

Wenn eine Stufenausführung beendet wird, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `myPipeline` in der `us-west-2` Region für die Stufe `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,
        "pipeline-execution-attempt": 1
    }
}
```

Ereignis „Phase angehalten“

Wenn eine Stufenausführung gestoppt wird, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `myPipeline` in der `us-west-2` Region für die Stufe `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,
        "pipeline-execution-attempt": 1
    }
}
```

Phase wurde nach dem Wiederholungsereignis der Stufe wieder aufgenommen

Wenn eine Stufenausführung fortgesetzt wird und eine Stufe hat, die erneut versucht wurde, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet.

Wenn eine Stufe erneut versucht wurde, wird das `stage-last-retry-attempt-time` Feld angezeigt, wie im Beispiel gezeigt. Das Feld zeigt bei allen Bühnenereignissen an, wenn ein Wiederholungsversuch durchgeführt wurde.

> **Note**
>
> Das `stage-last-retry-attempt-time` Feld ist in allen nachfolgenden Bühnenereignissen vorhanden, 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
    }
}
```

Ereignisse auf Aktionsebene

Ereignisse auf Aktionsebene werden ausgegeben, wenn es eine Statusänderung für eine Aktionsausführung gibt.

Themen

- Aktion STARTED-Ereignis
- Aktion ERFOLGT-Ereignis
- Aktion FEHLGESCHLAGEN-Ereignis
- Aktion ABANDONED-Ereignis

Aktion STARTED-Ereignis

Wenn eine Aktionsausführung gestartet wird, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `myPipeline` in der `us-east-1` Region 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
        },
        "pipeline-execution-attempt": 1
        "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,
        "pipeline-execution-attempt": 1
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

Aktion ERFOLGT-Ereignis

Wenn eine Aktionsausführung erfolgreich ist, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `"myPipeline"` in der `us-west-2` Region 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,
        "pipeline-execution-attempt": 1
    }
}
```

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,
        "pipeline-execution-attempt": 1
    },
    "resources": [
```

```
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

Aktion FEHLGESCHLAGEN-Ereignis

Wenn eine Aktionsausführung fehlschlägt, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `"myPipeline"` in der `us-west-2` Region 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,
        "pipeline-execution-attempt": 1
    }
}
```

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,
        "pipeline-execution-attempt": 1
    },
```

```
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "additionalInformation": "Deployment <ID> failed"
    }
}
```

Aktion ABANDONED-Ereignis

Wenn eine Aktionsausführung abgebrochen wird, gibt sie ein Ereignis aus, das Benachrichtigungen mit dem folgenden Inhalt sendet. Dieses Beispiel bezieht sich auf die Pipeline mit dem Namen `"myPipeline"` in der `us-west-2` Region 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,
        "pipeline-execution-attempt": 1
    }
```

```
}
```

Erstellen einer Regel, die eine Benachrichtigung über ein Pipeline-Ereignis sendet

Eine Regel überwacht bestimmte Ereignisse und leitet sie dann an von Ihnen gewählte AWS Ziele weiter. Sie können eine Regel erstellen, die automatisch eine AWS Aktion ausführt, wenn eine andere AWS Aktion stattfindet, oder eine Regel, die regelmäßig eine AWS Aktion nach einem festgelegten Zeitplan ausführt.

Themen

- Senden einer Benachrichtigung, wenn sich der Pipeline-Status ändert (Konsole)
- Senden einer Benachrichtigung, wenn sich der Pipeline-Status ändert (CLI)

Senden einer Benachrichtigung, wenn sich der Pipeline-Status ändert (Konsole)

Diese Schritte zeigen, wie Sie mit der EventBridge Konsole eine Regel zum Senden von Benachrichtigungen über Änderungen in erstellen CodePipeline.

So erstellen Sie eine - EventBridge Regel, die auf Ihre Pipeline mit einer Amazon S3-Quelle abzielt

1. Öffnen Sie die Amazon- EventBridge 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 aus. Wählen Sie Regel erstellen aus.
3. Geben Sie unter 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 Services 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 Statusänderung der Pipeline-Ausführung aus.
   - Wählen Sie für eine Regel, die für Ereignisse auf Stufenebene gilt, CodePipeline Statusänderung der Stufenausführung aus.

- Wählen Sie für eine Regel, die für Ereignisse auf Aktionsebene gilt, die Option CodePipeline Statusänderung der Aktionsausführung 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 die Selektoren zulassen, können Sie auch die Option Muster bearbeiten im Fenster Ereignismuster verwenden, um ein Ereignismuster im JSON-Format zu kennzeichnen.

   > **Note**
   >
   > Falls nicht anders angegeben, wird das Ereignis für alle Pipelines/Stufen/Aktionen und Status erstellt.

   Für detailliertere Ereignismuster können Sie die folgenden Beispielereignismuster kopieren und in das Fenster Ereignismuster einfügen.

   - Example

     Verwenden Sie dieses Beispiel-Ereignismuster, um fehlgeschlagene Bereitstellungs- und Erstellungsaktionen in allen Pipelines zu erfassen.

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Deploy", "Build"]
    }
```

```
  }
}
```

- Example

  Verwenden Sie dieses Beispiel-Ereignismuster, um alle abgelehnten oder fehlgeschlagenen Genehmigungsaktionen in allen Pipelines zu erfassen.

```
{
 "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Approval"]
    }
  }
}
```

- Example

  Verwenden Sie diese Beispiel-Ereignismuster, um alle Ereignisse in den angegebenen Pipelines zu erfassen.

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Pipeline Execution State Change",
    "CodePipeline Action Execution State Change",
    "CodePipeline Stage Execution State Change"
  ],
  "detail": {
    "pipeline": ["myPipeline", "my2ndPipeline"]
  }
```

```
}
```

9. Wählen Sie Weiter aus.
10. Wählen Sie unter Zieltypen die Option -AWS Service aus.
11. Wählen Sie unter Ziel auswählen die Option ausCodePipeline. Geben Sie unter Pipeline-ARN den Pipeline-ARN für die Pipeline ein, die von 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- 2:80398EXAMPLE:MyFirstPipeline

12. So erstellen oder geben Sie eine IAM-Servicerolle an, die EventBridge Berechtigungen zum Aufrufen des Ziels erteilt, das Ihrer EventBridge Regel zugeordnet ist (in diesem Fall ist das Ziel CodePipeline):
    - Wählen Sie Neue Rolle für diese spezifische Ressource erstellen, um eine Servicerolle zu erstellen, die Ihren Pipeline-Ausführungen EventBridge Berechtigungen erteilt.
    - Wählen Sie Vorhandene Rolle verwenden, um eine Servicerolle einzugeben, die Ihren Pipeline-Ausführungen EventBridge Berechtigungen 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).

Senden einer Benachrichtigung, wenn sich der Pipeline-Status ändert (CLI)

Diese Schritte zeigen, wie Sie mit der CLI eine CloudWatch Ereignisregel erstellen, um Benachrichtigungen über Änderungen in zu senden CodePipeline.

Um eine Regel AWS CLI mit der zu erstellen, 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, mit denen Sie erstellen, die Ihrem AWS Konto CodePipeline zugeordnet sind, eindeutig sein.
- Das Ereignismuster für die Quelle und von der Regel verwendete Detailfelder. Weitere Informationen finden Sie unter Amazon EventBridge und Ereignismuster.

So erstellen Sie eine - EventBridge Regel mit CodePipeline als Ereignisquelle

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.)

   Der folgende Beispielbefehl verwendet , --event-pattern um eine Regel namens zu erstellen“`MyPipelineStateChanges`”, die das CloudWatch Ereignis ausgibt, wenn eine Pipeline-Ausführung für die Pipeline namens „myPipeline."

   ```
   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 schließen 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 des `Id` Ziels in der Liste der Ziele und dem `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 für hinzu EventBridge , um den angegebenen Zielservice zum Aufrufen der Benachrichtigung zu verwenden. Weitere Informationen finden Sie unter Verwenden von ressourcenbasierten 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. Diese Ressourcen sind die CloudWatch Ereignisregeln, die so konfiguriert sind, dass sie auf Ereignisse in der Pipeline-Quelle reagieren, z. B. eine Codeänderung im CodeCommit Repository. Wenn Sie CloudWatch Ereignisse für eine Amazon S3-Quelle verwenden, müssen Sie aktivieren, CloudTrail damit die Ereignisse protokolliert werden. CloudTrail erfordert einen S3-Bucket, in den es seine Digests senden kann. Sie können aus dem benutzerdefinierten Bucket auf die Protokolldateien für Ihre CloudWatch Ereignisressourcen zugreifen, aber Sie können nicht aus dem Platzhalter-Bucket auf die Daten zugreifen.

- Wenn Sie die -CLI oder AWS CloudFormation zum Einrichten der CloudWatch Ereignisressourcen 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 Ereignisressourcen für Sie erstellt. - CloudTrail Digests werden im Platzhalter-Bucket in der gespeichertAWS-Region, in der die Pipeline erstellt wird.

Sie können die Konfiguration ändern, wenn Sie einen anderen Bucket als den Platzhalter-Bucket verwenden möchten.

> **Note**
>
> In CloudTrail Platzhalter-Buckets geschriebene Daten laufen nach einem Tag automatisch ab und werden nicht beibehalten.

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

Diese Tabelle listet die Namen der S3-Platzhalter-Buckets auf, die Protokolldateien enthalten, die Änderungserkennungsereignisse für Pipelines mit Amazon S3-Quellaktionen verfolgen.

Name der Region	Platzhalter-Bucket-Name	Region-ID
USA Ost (Ohio)	codepipeline-cloudtrail-placeholder-bucket-us-east-2	us-east-2
US East (N. Virginia)	codepipeline-cloudtrail-placeholder-bucket-us-east-1	us-east-1
US West (N. California)	codepipeline-cloudtrail-placeholder-bucket-us-west-1	us-west-1
US West (Oregon)	codepipeline-cloudtrail-placeholder-bucket-us-west-2	us-west-2
Kanada (Zentral)	codepipeline-cloudtrail-placeholder-bucket-ca-central-1	ca-central-1
Europa (Frankfurt)	codepipeline-cloudtrail-placeholder-bucket-eu-central-1	eu-central-1
Europe (Ireland)	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-north-1	eu-north-1

Name der Region	Platzhalter-Bucket-Name	Region-ID
Asien-Pazifik (Hongkong)	codepipeline-cloudtrail-placeholder-bucket-ap-east-1	ap-east-1
Asien-Pazifik (Hyderabad)	codepipeline-cloudtrail-placeholder-bucket-ap-south-2	ap-south-2
Asien-Pazifik (Jakarta)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-3	ap-southeast-3
Asien-Pazifik (Melbourne)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-4	ap-southeast-4
Asien-Pazifik (Mumbai)	codepipeline-cloudtrail-placeholder-bucket-ap-south-1	ap-south-1
Asien-Pazifik (Osaka)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-3-prod	ap-northeast-3
Asien-Pazifik (Tokio)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-1	ap-northeast-1
Asien-Pazifik (Seoul)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-2	ap-northeast-2
Asia Pacific (Singapore)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-1	ap-southeast-1
Asia Pacific (Sydney)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-2	ap-southeast-2

Name der Region	Platzhalter-Bucket-Name	Region-ID
Asia Pacific (Tokyo)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-1	ap-northeast-1
Kanada (Zentral)	codepipeline-cloudtrail-placeholder-bucket-ca-central-1	ca-central-1
Europa (Frankfurt)	codepipeline-cloudtrail-placeholder-bucket-eu-central-1	eu-central-1
Europe (Ireland)	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-south-1	eu-south-1
Europa (Paris)	codepipeline-cloudtrail-placeholder-bucket-eu-west-3	eu-west-3
Europa (Spain)	codepipeline-cloudtrail-placeholder-bucket-eu-south-2	eu-south-2
Europa (Stockholm)	codepipeline-cloudtrail-placeholder-bucket-eu-north-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-me-south-1	me-south-1

Name der Region	Platzhalter-Bucket-Name	Region-ID
Naher Osten (VAE)	codepipeline-cloudtrail-placeholder-bucket-me-central-1	me-central-1
Südamerika (São Paulo)	codepipeline-cloudtrail-placeholder-bucket-sa-east-1	sa-east-1

Protokollierung von CodePipeline-API-Aufrufen mit AWS CloudTrail

AWS CodePipelineist in integriertAWS CloudTrail, einem Service, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem durchgeführten Aktionen AWS-Service in CodePipeline protokolliert. CloudTrail erfasst alle API-Aufrufe für CodePipeline als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der CodePipeline-Konsole und Code-Aufrufe der CodePipeline-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen für einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen fürCodePipeline. Auch wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse in der CloudTrail-Konsole in Event history (Ereignisverlauf) anzeigen. Mit den von CloudTrail gesammelten Informationen können Sie die an CodePipeline gestellte Anforderung, die IP-Adresse, von der die Anforderung gestellt wurde, den Initiator sowie den Zeitpunkt der Anforderung und weitere Angaben bestimmen.

Weitere Informationen zu CloudTrail finden Sie im AWS CloudTrail-Benutzerhandbuch.

CodePipeline-Informationen in CloudTrail

CloudTrail wird beim Erstellen Ihres Kontos auf AWS-Konto aktiviert. Die in protokolliertCodePipeline, wird diese Aktivität zusammen mit anderen AWS-Service Ereignissen in Ereignisverlauf aufgezeichnet. CloudTrail Sie können die neusten Ereignisse in Ihr AWS-Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail -API-Ereignisverlauf.

Erstellen Sie einen Trail zur laufenden Aufzeichnung der Ereignisse in IhremAWS-Konto, einschließlich Ereignissen für, erstellen Sie einen Trail zur CodePipeline laufenden Aufzeichnung der Ereignisse in Ihrem protokolliert. Ein Trail ermöglicht CloudTrail die Bereitstellung von Protokolldateien für einen Amazon S3 S3-Bucket bereitstellt. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon S3 Bucket

bereit. Darüber hinaus können Sie andere AWS-Services konfigurieren, um die in den CloudTrail-Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie unter:

- Übersicht zum Erstellen eines Trails
- Von CloudTrail unterstützte Services und Integrationen
- Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail-Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail-Protokolldateien aus mehreren Konten

Alle CodePipeline-Aktionen werden von CloudTrail protokolliert und in der CodePipeline-API-Referenz dokumentiert. Zum Beispiel werden durch Aufrufe der Aktionen `CreatePipeline`, `GetPipelineExecution` und `UpdatePipeline` Einträge in den CloudTrail-Protokolldateien generiert.

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 Anforderung mit Stamm- oder AWS Identity and Access Management (IAM) -Anmeldeinformationen ausgeführt wurde
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen verbundenen Benutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Grundlagen zu CodePipeline-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, durch die Ereignisse als Protokolldateien an den von Ihnen angegebenen Amazon-S3-Bucket übermittelt werden. CloudTrail-Protokolldateien können einen oder mehrere Einträge enthalten. Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält unter anderem Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion sowie über die Anfrageparameter. CloudTrail-Protokolleinträge sind kein geordnetes Stacktrace der öffentlichen API-Aufrufe und erscheinen daher nicht in einer bestimmten Reihenfolge.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für ein Aktualisierungspipeline-Ereignis, bei dem eine Pipeline mit dem Namen von dem Benutzer mit dem Namen bearbeitet MyFirstPipeline

wurde JaneDoe — CodePipeline mit der Konto-ID 80398EXAMPLE. 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":"awscodepipeline-demobucket-example-date",
     "S3ObjectKey":"sampleapp_linux.zip"
      }
      }
     ],
       "name":"Source"
     },
     (...)
               },
 "responseElements":{
   "pipeline":{
     "version":2,
     (...)
           },
     "requestID":"2c4af5c9-7ce8-EXAMPLE",
     "eventID":""c53dbd42-This-Is-An-Example"",
     "eventType":"AwsApiCall",
     "recipientAccountId":"80398EXAMPLE"
     }
      ]
}
```

Fehlerbehebung CodePipeline

Die folgenden Informationen helfen Ihnen möglicherweise bei der Lösung häufiger Probleme in AWS CodePipeline.

Themen

- Pipeline-Fehler: Eine mit AWS Elastic Beanstalk konfigurierte Pipeline gibt eine Fehlermeldung zurück: „Bereitstellung fehlgeschlagen. Die bereitgestellte Rolle verfügt nicht über ausreichende Berechtigungen: Service:AmazonElasticLoadBalancing“
- Bereitstellungsfehler: Eine Pipeline, die mit einer AWS Elastic Beanstalk Bereitstellungsaktion konfiguriert ist, hängt, anstatt fehlzuschlagen, wenn die Berechtigung „DescribeEvents“ fehlt
- Pipeline-Fehler: Eine Quellaktion gibt die Meldung mit unzureichenden Berechtigungen zurück: „Kann nicht auf das CodeCommit Repository zugreifenrepository-name. 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, die in einer AWS Region mit einem Bucket erstellt wurde, der in einer anderen AWS Region erstellt wurde, gibt ein „InternalError“ mit dem Code „JobFailed“ zurück
- Bereitstellungsfehler: Eine ZIP-Datei mit einer WAR-Datei wird erfolgreich für AWS Elastic Beanstalk bereitgestellt, die Anwendungs-URL meldet jedoch den Fehler „404 nicht gefunden“.
- Pipeline-Artefakt-Ordnernamen werden gekürzt
- Hinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder GitLab.com
- Hinzufügen von CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen
- Pipeline-Fehler: Eine Bereitstellung mit der CodeDeployToECS-Aktion gibt eine Fehlermeldung zurück: „Ausnahme beim Versuch, die Artefaktdatei der Aufgabendefinition zu lesen aus: <Name des Quellartefakts>“
- GitHub Quellaktion der Version 1: Die Repository-Liste zeigt verschiedene Repositorys
- GitHub Quellaktion der Version 2: Verbindung für ein Repository kann nicht hergestellt werden
- Amazon S3 error: CodePipeline service role <ARN> erhält eine S3-Zugriffsverweigerung für den S3-Bucket <BucketName>
- Pipelines mit einer Amazon S3-, Amazon-ECR- oder CodeCommit -Quelle werden nicht mehr automatisch gestartet

- Verbindungsfehler beim Herstellen einer Verbindung mit GitHub: „Ein Problem ist aufgetreten, stellen Sie sicher, dass Cookies in Ihrem Browser aktiviert sind“ oder „Ein Organisationsbesitzer muss die GitHub App installieren“
- Fehler, wenn die CloudFormationStackInstances Aktionen CloudFormationStackSet oder in einer Region nicht verfügbar sind
- Pipelines, deren Ausführungsmodus in den Modus QUEUED oder PARALLEL geändert wurde, schlagen fehl, wenn das Ausführungslimit erreicht ist
- Pipelines im PARALLEL-Modus haben eine veraltete Pipeline-Definition, wenn sie beim Wechsel in den Modus QUEUED oder SUPERSEDED bearbeitet werden
- Pipelines, die aus dem PARALLEL-Modus geändert wurden, zeigen einen vorherigen Ausführungsmodus an
- Pipelines mit Verbindungen, die Auslöserfilterung nach Dateipfaden verwenden, beginnen möglicherweise nicht bei der Erstellung von Verzweigungen
- Pipelines mit Verbindungen, die Auslöserfilterung nach Dateipfaden verwenden, werden möglicherweise nicht gestartet, wenn das Dateilimit erreicht ist
- Benötigen Sie Hilfe für ein anderes Problem?

Pipeline-Fehler: Eine mit AWS Elastic Beanstalk konfigurierte Pipeline gibt eine Fehlermeldung zurück: „Bereitstellung fehlgeschlagen. Die bereitgestellte Rolle verfügt nicht über ausreichende Berechtigungen: Service:AmazonElasticLoadBalancing“

Problem: Die Servicerolle für CodePipeline verfügt nicht über ausreichende Berechtigungen für AWS Elastic Beanstalk, einschließlich, aber nicht beschränkt auf einige Operationen in Elastic Load Balancing. Die Servicerolle für CodePipeline wurde am 6. August 2015 aktualisiert, um dieses Problem zu beheben. Kunden, die ihre Service-Rolle vor diesem Datum erstellt haben, müssen Sie Richtlinienanweisung für ihre Service-Rolle ändern, um die erforderlichen Berechtigungen hinzuzufügen.

Mögliche Fehlerbehebungen: Die einfachste Lösung besteht darin, die Richtlinienanweisung für Ihre Servicerolle zu bearbeiten, wie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle beschrieben.

Nachdem Sie die bearbeitete Richtlinie angewendet haben, führen Sie die Schritte unter aus, 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 einer AWS Elastic Beanstalk Bereitstellungsaktion konfiguriert ist, hängt, anstatt fehlzuschlagen, wenn die Berechtigung „DescribeEvents“ fehlt

Problem: Die Servicerolle für CodePipeline muss die `-"elasticbeanstalk:DescribeEvents"`Aktion für alle Pipelines enthalten, die verwendenAWS Elastic Beanstalk. Ohne diese Berechtigung bleiben AWS Elastic Beanstalk-Bereitstellungsaktionen hängen, ohne dass sie fehlschlagen oder einen Fehler zurückmelden. Wenn diese Aktion in Ihrer Servicerolle fehlt, CodePipeline hat keine Berechtigungen zum Ausführen der Pipeline-Bereitstellungsphase in AWS Elastic Beanstalk in Ihrem Namen.

Mögliche Korrekturen: Überprüfen Sie Ihre CodePipeline Servicerolle. Wenn die `"elasticbeanstalk:DescribeEvents"`-Aktion fehlt, folgen Sie den Schritten in Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle, um sie unter Verwendung der Funktion Edit Policy (Richtlinie bearbeiten) in der IAM-Konsole hinzuzufügen.

Nachdem Sie die bearbeitete Richtlinie angewendet haben, führen Sie die Schritte unter aus, 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: „Kann nicht auf das CodeCommit Repository zugreifen**`repository-name`**. Überprüfen Sie, ob die Pipeline-IAM-Rolle über ausreichende Berechtigungen für den Zugriff auf das Repository verfügt.“

Problem: Die Servicerolle für CodePipeline verfügt nicht über ausreichende Berechtigungen für CodeCommit und wurde wahrscheinlich erstellt, bevor die Unterstützung für die Verwendung

CodeCommit von 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 Korrekturen: Fügen Sie der Richtlinie Ihrer CodePipeline Servicerolle CodeCommit die erforderlichen Berechtigungen für hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.

Pipeline-Fehler: Ein Jenkins-Build oder eine Testaktion werden über eine lange Zeit ausgeführt und schlagen dann aufgrund fehlender Anmeldeinformationen oder Berechtigungen fehl.

Problem: Wenn der Jenkins-Server auf einer Amazon EC2-Instance installiert ist, wurde die Instance möglicherweise nicht mit einer Instance-Rolle erstellt, die über die für erforderlichen Berechtigungen verfügt CodePipeline. Wenn Sie einen IAM-Benutzer auf einem Jenkins-Server, einer On-Premises-Instance oder einer Amazon EC2 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 Korrekturen: Stellen Sie sicher, dass die Amazon 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 Von 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 angeben, die Sie für die Integration zwischen Jenkins und CodePipeline direkt in die Jenkins-Benutzeroberfläche konfiguriert haben. Dies ist keine empfohlene bewährte Methode. Wenn Sie dies tun müssen, müssen Sie sicherstellen, dass der Jenkins-Server geschützt ist und HTTPS statt HTTP verwendet.

Pipeline-Fehler: Eine Pipeline, die in einer AWS Region mit einem Bucket erstellt wurde, der in einer anderen AWS Region erstellt wurde, gibt ein „InternalError“ mit dem Code „JobFailed“ zurück

Problem: Der Download eines in einem Amazon S3-Bucket gespeicherten Artefakts schlägt fehl, wenn die Pipeline und der Bucket in verschiedenen AWS Regionen erstellt werden.

Mögliche Korrekturen: Stellen Sie sicher, dass sich der Amazon S3-Bucket, in dem Ihr Artefakt gespeichert ist, in derselben AWS Region befindet wie die Pipeline, die Sie erstellt haben.

Bereitstellungsfehler: Eine ZIP-Datei mit einer WAR-Datei wird erfolgreich für AWS Elastic Beanstalk bereitgestellt, die Anwendungs-URL meldet jedoch den Fehler „404 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 Lösungen: AWS Elastic Beanstalk kann eine ZIP-Datei, nicht jedoch eine WAR-Datei in einer ZIP-Datei entpacken. Geben Sie statt einer WAR-Datei in Ihrer Datei `buildspec.yml` einen Ordner an, der die Inhalte enthält, die bereitgestellt werden sollen. Beispielsweise:

```
version: 0.2

phases:
  post_build:
    commands:
      - mvn package
      - mv target/my-web-app ./
artifacts:
  files:
    - my-web-app/**/*
 discard-paths: yes
```

Ein Beispiel hierfür finden Sie unter AWS Elastic Beanstalk-Beispiel für CodeBuild.

Pipeline-Artefakt-Ordnernamen werden gekürzt

Problem: Wenn Sie Pipeline-Artefaktnamen in anzeigen CodePipeline, scheinen die Namen gekürzt zu werden. Dadurch sehen manche Namen gleich aus oder enthalten augenscheinlich nicht mehr den gesamten Pipelinenamen.

Erläuterung: CodePipeline Kürzt Artefaktnamen, um sicherzustellen, dass der vollständige Amazon S3-Pfad die Größenbeschränkungen der Richtlinien nicht überschreitet, wenn temporäre Anmeldeinformationen für Auftragsarbeitsnehmer CodePipeline generiert.

Obwohl der Artefaktname gekürzt zu sein scheint, CodePipeline weist dem Artefakt-Bucket eine Weise zu, die nicht von Artefakten mit gekürzten Namen betroffen ist. 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.

Hinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder GitLab.com

Wenn Sie eine -AWS CodeStarVerbindung in einer Quellaktion und einer - CodeBuild Aktion verwenden, gibt es zwei Möglichkeiten, das Eingabeartefakt an den Build zu übergeben:

- Der Standardwert: Die Quellaktion erzeugt eine ZIP-Datei, die den Code enthält, den CodeBuild herunterlädt.
- Git-Klon: Der Quellcode kann direkt in die Build-Umgebung heruntergeladen werden.

  Der Git-Klon-Modus ermöglicht es Ihnen, mit dem Quellcode als funktionierendes Git-Repository zu interagieren. Um diesen Modus verwenden zu können, müssen Sie Ihrer CodeBuild Umgebung Berechtigungen zur Verwendung der Verbindung erteilen.

Um Ihrer CodeBuild Servicerollenrichtlinie Berechtigungen hinzuzufügen, erstellen Sie eine vom Kunden verwaltete Richtlinie, die Sie Ihrer CodeBuild Servicerolle anfügen. Mit den folgenden Schritten wird eine Richtlinie erstellt, bei der die `UseConnection`-Berechtigung im `action`-Feld und der Verbindungs-ARN im `Resource`-Feld angegeben wird.

So verwenden Sie die Konsole zum Hinzufügen der UseConnection Berechtigungen

1. Um den Verbindungs-ARN für Ihre Pipeline zu finden, öffnen Sie die Pipeline und klicken Sie auf das (i)-Symbol in der Quellaktion. Sie fügen den Verbindungs-ARN zu Ihrer CodeBuild Servicerollenrichtlinie hinzu.

   Ein Beispiel für einen Verbindungs-ARN ist:

   ```
   arn:aws:codestar-connections:eu-central-1:123456789123:connection/
   sample-1908-4932-9ecc-2ddacee15095
   ```

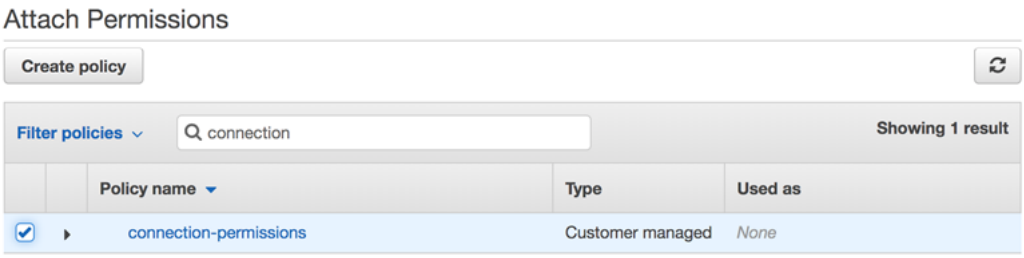
2. Um Ihre CodeBuild Servicerolle zu finden, öffnen Sie das in Ihrer Pipeline verwendete Build-Projekt und navigieren Sie zur Registerkarte Build-Details.
3. Wählen Sie den Link Service role (Servicerolle) . Dadurch wird die IAM-Konsole geöffnet, in der Sie eine neue Richtlinie hinzufügen können, die Zugriff auf Ihre Verbindung gewährt.
4. Wählen Sie in der IAM-Konsole Attach policies (Richtlinien anhängen) und dann Create policy (Richtlinie erstellen).

   Verwenden Sie die folgende Beispielrichtlinienvorlage. Fügen Sie Ihren Verbindungs-ARN in das `Resource`-Feld ein, wie in diesem Beispiel gezeigt:

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": "codestar-connections:UseConnection",
               "Resource": "insert connection ARN here"
           }
       ]
   }
   ```

   Fügen Sie auf der Registerkarte JSON Ihre Richtlinie ein.

5. Wählen Sie Richtlinie prüfen. Geben Sie einen Namen für die Richtlinie ein (beispielsweise **`connection-permissions`**) und wählen Sie dann Create policy (Richtlinie erstellen) aus.
6. Kehren Sie zu der Seite zurück, auf der Sie Berechtigungen angehängt haben, aktualisieren Sie die Richtlinienliste und wählen Sie die gerade erstellte Richtlinie aus. Wählen Sie Richtlinien anfügen.



Hinzufügen von CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen

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 Code enthält, den CodeBuild herunterlädt.
- Vollständiger Klon – Der Quellcode kann direkt in die Build-Umgebung heruntergeladen werden.

  Mit der Option Vollständiger Klon können Sie mit dem Quellcode als funktionierendes Git-Repository interagieren. Um diesen Modus zu verwenden, müssen Sie Ihrer CodeBuild Umgebung Berechtigungen hinzufügen, um aus Ihrem Repository abzurufen.

Um Ihrer CodeBuild Servicerollenrichtlinie Berechtigungen hinzuzufügen, erstellen Sie eine vom Kunden verwaltete Richtlinie, die Sie Ihrer CodeBuild Servicerolle anfügen. Mit den folgenden Schritten wird eine Richtlinie erstellt, die die `codecommit:GitPull` Berechtigung im `action` Feld angibt.

So verwenden Sie die Konsole zum Hinzufügen der GitPull Berechtigungen

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: Eine Bereitstellung mit der CodeDeployToECS-Aktion gibt eine Fehlermeldung zurück: „Ausnahme beim Versuch, die Artefaktdatei der 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 Artefakt-ZIP-Größe in der `CodeDeployToECS` Bereitstellungsaktion beträgt 3 MB. Die folgende Fehlermeldung wird zurückgegeben, wenn die Datei nicht gefunden wird oder die Artefaktgröße 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 vorhanden ist, stellt sicher, dass die komprimierte Größe kleiner als 3 MB ist.

GitHub Quellaktion der Version 1: Die Repository-Liste zeigt verschiedene Repositorys

Problem:

Nach einer erfolgreichen Autorisierung für eine Aktion der GitHub Version 1 in der CodePipeline Konsole können Sie aus einer Liste Ihrer GitHub Repositorys auswählen. Wenn die Liste die Repositorys, die Sie erwartet haben, nicht enthält, können Sie Probleme mit dem Konto beheben, das für die Autorisierung verwendet wird.

Mögliche Korrekturen: Die Liste der in der CodePipeline Konsole bereitgestellten Repositorys basiert auf der GitHub Organisation, zu der das autorisierte Konto gehört. Stellen Sie sicher, dass das Konto, mit dem Sie autorisieren, das Konto GitHub ist, das der GitHub Organisation zugeordnet ist, in der Ihr Repository erstellt wird.

GitHub Quellaktion der Version 2: Verbindung für ein Repository kann nicht hergestellt werden

Problem:

Da eine Verbindung zu einem GitHub Repository den AWS Connector für verwendet GitHub, benötigen Sie Berechtigungen für den Organisationsbesitzer oder Administratorberechtigungen für das Repository, um die Verbindung zu erstellen.

Mögliche Korrekturen: Informationen zu Berechtigungsstufen für ein GitHub Repository finden Sie unter https://docs.github.com/en/free-pro-team@latest/github/setting-up-and-managing-organizations-and-teams/permission-levels-for-an-organization.

Amazon S3 error: CodePipeline service role <ARN> erhält eine S3-Zugriffsverweigerung für den S3-Bucket <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 zum Überprüfen verfügt, tritt in Amazon S3 ein `AccessDenied` Fehler auf und die folgende Fehlermeldung wird in angezeigt CodePipeline:

CodePipeline -Servicerolle „arn:aws:iam::*`AccountID`* :role/service-role/*`RoleID`*“ erhält S3-Zugriff für den S3-Bucket „*`BucketName`*“ verweigert

Die CloudTrail Protokolle für die Aktion protokollieren auch den `AccessDenied` Fehler.

Mögliche Korrekturen: Gehen Sie wie folgt vor:

- Fügen Sie für die Richtlinie, die Ihrer CodePipeline Servicerolle zugeordnet ist, `s3:ListBucket` zur Liste der Aktionen in Ihrer Richtlinie hinzu. Anweisungen zum Anzeigen Ihrer Servicerollenrichtlinie finden Sie unter Den Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen. Bearbeiten Sie die Richtlinienanweisung für Ihre Servicerolle wie unter beschriebenHinzufügen von Berechtigungen zur CodePipeline-Servicerolle.
- Fügen Sie für die ressourcenbasierte Richtlinie, die dem Amazon S3-Artefakt-Bucket für Ihre Pipeline zugeordnet ist, auch als Artefakt-Bucket-Richtlinie bezeichnet, eine Anweisung hinzu, damit die -`s3:ListBucket`Berechtigung von Ihrer CodePipeline Servicerolle verwendet werden kann.

  So fügen Sie Ihre Richtlinie zum Artefakt-Bucket hinzu

  1. Führen Sie die Schritte unter ausDen Pipeline-ARN und die Servicerolle ARN (Konsole) anzeigen, um Ihren Artefakt-Bucket auf der Seite Pipeline-Einstellungen auszuwählen und ihn dann in der Amazon 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 vorhandene Richtlinie wie im folgenden Beispiel gezeigt. Die Bucket-Richtlinie ist eine JSON-Datei, daher müssen Sie gültiges JSON eingeben.

     Das folgende Beispiel zeigt eine Bucket-Richtlinienanweisung für einen Artefakt-Bucket, bei dem die Beispielrollen-ID für die Servicerolle *AROAEXAMPLEID* lautet.

     ```
     {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::BucketName",
         "Condition": {
             "StringLike": {
                 "aws:userid": "AROAEXAMPLEID:*"
             }
         }
     }
     ```

     Das folgende Beispiel zeigt dieselbe Bucket-Richtlinienanweisung, nachdem die Berechtigung hinzugefügt wurde.

```
{
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890",
            "Condition": {
                "StringLike": {
                    "aws:userid": "AROAEXAMPLEID:*"
                }
            }
        },
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": false
                }
            }
        }
    ]
}
```

Weitere Informationen finden Sie in den Schritten unter https://aws.amazon.com/blogs/security/writing-iam-policies-how-to-grant-access-to-an-amazon-s3-bucket/.

5. Wählen Sie Speichern.

Nachdem Sie die bearbeitete Richtlinie angewendet haben, führen Sie die Schritte unter aus, Manuelles Starten einer Pipeline um Ihre Pipeline manuell erneut auszuführen.

Pipelines mit einer Amazon S3-, Amazon-ECR- oder CodeCommit -Quelle werden nicht mehr automatisch gestartet

Problem:

Nachdem Sie eine Änderung an den Konfigurationseinstellungen für eine Aktion vorgenommen haben, die Ereignisregeln (EventBridge oder CloudWatch Ereignisse) für die Änderungserkennung verwendet, erkennt die Konsole möglicherweise keine Änderung, bei der Quellauslöser-IDs ähnlich sind und identische Anfangszeichen haben. Da die neue Ereignisregel nicht von der Konsole erstellt wird, startet die Pipeline nicht mehr automatisch.

Ein Beispiel für eine geringfügige Änderung am Ende des Parameternamens für CodeCommit wäre das Ändern Ihres CodeCommit Zweignamens `MyTestBranch-1` in `MyTestBranch-2`. Da die Änderung am Ende des Zweignamens steht, aktualisiert oder erstellt die Ereignisregel für die Quellaktion möglicherweise keine Regel für die neuen Quelleinstellungen.

Dies gilt für Quellaktionen, die CWE-Ereignisse für die Änderungserkennung verwenden, wie folgt:

Quellaktion	Parameter/Auslöser-IDs (Konsole)
Amazon ECR	Repository-Name Image-Tag
Amazon S3	Bucket S3-Objektschlüssel
CodeCommit	Repository-Name Branch-Name

Mögliche Lösungen:

Führen Sie eine der folgenden Aktionen aus:

- Ändern Sie die CodeCommit/S3/ECR-Konfigurationseinstellungen so, dass Änderungen am Anfang des Parameterwerts vorgenommen werden.

  Beispiel: Ändern Sie Ihren Verzweigungsnamen `release-branch` in `2nd-release-branch`. Vermeiden Sie eine Änderung am Ende des Namens, z. B. `release-branch-2`.

- Ändern Sie die CodeCommit/S3/ECR-Konfigurationseinstellungen für jede Pipeline.

  Beispiel: Ändern Sie Ihren Verzweigungsnamen `myRepo/myBranch` in `myDeployRepo/myDeployBranch`. Vermeiden Sie eine Änderung am Ende des Namens, z. B. `myRepo/myBranch2`.

- Verwenden Sie anstelle der Konsole die CLI oder , AWS CloudFormation um Ihre Ereignisregeln zur Erkennung von Änderungen zu erstellen und zu aktualisieren. Anweisungen zum Erstellen von Ereignisregeln für eine S3-Quellaktion finden Sie unter Amazon S3-Quellaktionen und EventBridge mit AWS CloudTrail. Anweisungen zum Erstellen von Ereignisregeln für eine Amazon-ECR-Aktion finden Sie unter Amazon-ECR-Quellaktionen und - EventBridge Ressourcen. Anweisungen zum Erstellen von Ereignisregeln für eine CodeCommit Aktion finden Sie unter CodeCommit - Quellaktionen und EventBridge.

Nachdem Sie Ihre Aktionskonfiguration in der Konsole bearbeitet haben, akzeptieren Sie die aktualisierten Ressourcen zur Erkennung von Änderungen, die von der Konsole erstellt wurden.

Verbindungsfehler beim Herstellen einer Verbindung mit GitHub: „Ein Problem ist aufgetreten, stellen Sie sicher, dass Cookies in Ihrem Browser aktiviert sind“ oder „Ein Organisationsbesitzer muss die GitHub App installieren“

Problem:

Um die Verbindung für eine GitHub Quellaktion in zu erstellen CodePipeline, müssen Sie der GitHub Organisationsbesitzer 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 Organisationsbesitzer muss die GitHub App installieren

Mögliche Korrekturen: Für Repositorys in einer GitHub Organisation muss der Organisationsbesitzer die Verbindung zum GitHub Repository herstellen. Bei Repositorys, die keiner Organisation angehören, müssen Sie der Repository-Besitzer sein.

Fehler, wenn die **`CloudFormationStackInstances`** Aktionen **`CloudFormationStackSet`** oder in einer Region nicht verfügbar sind

Problem: Das Vorhandensein von CodePipeline in einer Region bedeutet nicht, dass alle Aktionen in dieser Region verfügbar sind. Wenn beispielsweise eine Pipeline die `CloudFormationStackSet` Aktion in einer Region ausführt, in der die Aktion nicht verfügbar ist, z. B. Afrika (Kapstadt), werden die folgenden Fehler angezeigt:

```
InvalidActionDeclarationException: ActionType (Category: 'Deploy',
Provider: 'CloudFormationStackSet', Owner: 'AWS, Version: '1') in action
'Deploy' is not available in region 'AF_SOUTH_1'
```

Mögliche Korrekturen: Verweisen Sie auf den folgenden Hinweis, um die Aktion in einer Region zu verwenden, in der die Aktion verfügbar ist.

> **Note**
>
> Die `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und sind in den Regionen Asien-Pazifik (Hongkong), Europa (Zürich), Europa (Mailand), Afrika (Kapstadt) und Naher Osten (Bahrain) nicht verfügbar. Informationen zum Verweisen auf andere verfügbare Aktionen finden Sie unter Produkt- und Serviceintegrationen mit CodePipeline.

Weitere Informationen zur Aktion finden Sie unter AWS CloudFormation StackSets.

Pipelines, deren Ausführungsmodus in den Modus QUEUED oder PARALLEL 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 Korrekturen: Wenn Sie die Pipeline-Definition für den Ausführungsmodus bearbeiten, nehmen Sie die Bearbeitung getrennt von anderen Bearbeitungsaktionen vor.

Weitere Informationen zum QUEUED- oder PARALLEL-Ausführungsmodus finden Sie unter CodePipeline -Konzepte.

Pipelines im PARALLEL-Modus haben eine veraltete Pipeline-Definition, wenn sie beim Wechsel in den Modus QUEUED oder SUPERSEDED bearbeitet werden

Problem: Bei Pipelines im parallelen Modus wird die Pipeline-Definition für den PARALLEL-Modus nicht aktualisiert, wenn der Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED geändert wird. Die aktualisierte Pipeline-Definition beim Aktualisieren des PARALLEL-Modus wird nicht im SUPERSEDED- oder QUEUED-Modus verwendet

Mögliche Korrekturen: Vermeiden Sie es bei Pipelines im Parallelmodus, wenn Sie den Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED ändern, gleichzeitig die Pipeline-Definition zu aktualisieren.

Weitere Informationen zum QUEUED- oder PARALLEL-Ausführungsmodus finden Sie unter CodePipeline -Konzepte.

Pipelines, die aus dem PARALLEL-Modus geändert wurden, zeigen einen vorherigen Ausführungsmodus an

Problem: Bei Pipelines im PARALLEL-Modus zeigt der Pipeline-Status beim Bearbeiten des Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDEDED den aktualisierten Status nicht als

PARALLEL an. Wenn die Pipeline von PARALLEL in QUEUED oder SUPERSEDED geändert wurde, ist der Status für die Pipeline im SUPERSED- oder QUEUED-Modus 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, dass die Anzeige des Ausführungsmodus beim Bearbeiten des Pipeline-Ausführungsmodus auf QUEUED oder SUPERSEDED den Status PARALLEL nicht anzeigt.

Weitere Informationen zum QUEUED- oder PARALLEL-Ausführungsmodus finden Sie unter CodePipeline -Konzepte.

Pipelines mit Verbindungen, die Auslöserfilterung nach Dateipfaden verwenden, beginnen möglicherweise nicht bei der Erstellung von Verzweigungen

Beschreibung: Für Pipelines mit Quellaktionen, die Verbindungen verwenden, z. B. eine BitBucket Quellaktion, können Sie einen Auslöser mit einer Git-Konfiguration einrichten, mit der Sie nach Dateipfaden filtern können, um Ihre Pipeline zu starten. In bestimmten Fällen beginnt die Pipeline bei Pipelines mit Auslösern, die nach Dateipfaden gefiltert werden, möglicherweise nicht, wenn ein Branch mit einem Dateipfadfilter erstellt wird, da die Verbindung dadurch die geänderten Dateien nicht CodeConnections auflösen kann. Wenn die Git-Konfiguration für den Auslöser so eingerichtet ist, dass nach Dateipfaden gefiltert wird, startet die Pipeline nicht, wenn der Zweig mit dem Filter gerade im Quell-Repository erstellt wurde. Weitere Informationen zum Filtern nach Dateipfaden finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen.

Ergebnis: Pipelines in , CodePipeline die einen Dateipfadfilter für einen Zweig „B“ haben, werden beispielsweise nicht ausgelöst, wenn der Zweig „B“ erstellt wird. Wenn es keine Dateipfadfilter gibt, wird die Pipeline trotzdem gestartet.

Pipelines mit Verbindungen, die Auslöserfilterung nach Dateipfaden verwenden, werden möglicherweise nicht gestartet, wenn das Dateilimit erreicht ist

Beschreibung: Für Pipelines mit Quellaktionen, die Verbindungen verwenden, z. B. eine BitBucket Quellaktion, können Sie einen Auslöser mit einer Git-Konfiguration einrichten, mit der Sie nach

Dateipfaden filtern können, um Ihre Pipeline zu starten. CodePipeline ruft bis zu den ersten 100 Dateien ab. Wenn die Git-Konfiguration für den Auslöser daher so eingerichtet ist, dass nach Dateipfaden gefiltert wird, startet die Pipeline möglicherweise nicht, wenn es mehr als 100 Dateien gibt. Weitere Informationen zum Filtern nach Dateipfaden finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen.

Ergebnis: Wenn ein Diff beispielsweise 150 Dateien enthält, CodePipeline sucht die ersten 100 Dateien (ohne bestimmte Reihenfolge), um sie mit dem angegebenen Dateipfadfilter zu vergleichen. Wenn die Datei, die dem Dateipfadfilter entspricht, nicht zu den 100 Dateien gehört, die von abgerufen wurden CodePipeline, wird die Pipeline nicht aufgerufen

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 in AWS CodePipeline

Cloud-Sicherheit hat AWS bei höchste Priorität. Als - AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die den Anforderungen der anspruchsvollsten Organisationen entspricht.

Sicherheit ist eine geteilte Verantwortung zwischen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud – AWS ist für den Schutz der Infrastruktur verantwortlich, die AWS-Services in der ausgeführt wird AWS Cloud. stellt Ihnen AWS außerdem Services bereit, die Sie sicher nutzen können. Auditoren von Drittanbietern testen und überprüfen die Effektivität unserer Sicherheitsmaßnahmen im Rahmen der AWS -Compliance-Programme regelmäßig. Informationen zu den Compliance-Programmen, die für gelten AWS CodePipeline, finden Sie unter AWS-Service im Rahmen des Compliance-Programms .
- Sicherheit in der Cloud – Ihre Verantwortung wird durch die bestimmt AWS-Service , die Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, die Anforderungen Ihres Unternehmens und die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der geteilten Verantwortung bei der Verwendung von einsetzen können CodePipeline. Die folgenden Themen zeigen Ihnen, wie Sie konfigurieren, CodePipeline um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere verwenden AWS-Services , die Ihnen bei der Überwachung und Sicherung Ihrer - CodePipeline Ressourcen helfen.

Themen

- Datenschutz in AWS CodePipeline
- Identity and Access Management für AWS CodePipeline
- Protokollierung und Überwachung in CodePipeline
- Compliance-Validierung für AWS CodePipeline
- Ausfallsicherheit in AWS CodePipeline
- Infrastruktursicherheit in AWS CodePipeline
- Bewährte Methoden für die Gewährleistung der Sicherheit

Datenschutz in AWS CodePipeline

Das AWS Modell der geteilten VerantwortungModell gilt für den Datenschutz in AWS CodePipeline. Wie in diesem Modell beschrieben, AWS ist für den Schutz der globalen Infrastruktur verantwortlich, die alle ausführt AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir Ihnen, -Anmeldeinformationen zu schützen AWS-Konto und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor Authentifizierung (MFA).
- Verwenden Sie SSL/TLS für die Kommunikation mit - AWS Ressourcen. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API- und Benutzeraktivitätsprotokollierung mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in 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 auf AWS über eine Befehlszeilenschnittstelle oder eine API FIPS-140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit CodePipeline oder anderen AWS-Services über die Konsole, API AWS CLI oder AWS SDKs arbeiten. 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 der serverseitigen Verschlüsselung für Artefakte, die in Amazon S3 für gespeichert sind CodePipeline
- Verwenden von AWS Secrets Manager zum Nachverfolgen von Datenbankpasswörtern oder API-Schlüsseln von Drittanbietern

Richtlinie für den Datenverkehr zwischen Netzwerken

Amazon VPC ist ein AWS-Service , mit dem Sie AWS Ressourcen in einem von Ihnen definierten virtuellen Netzwerk (Virtual Private Cloud) starten können. CodePipeline unterstützt Amazon-VPC-Endpunkte AWS PrivateLink, die von unterstützt werden, einer - AWS Technologie, die die private Kommunikation zwischen der AWS-Services Verwendung einer Elastic-Network-Schnittstelle mit privaten IP-Adressen erleichtert. Das bedeutet, dass Sie eine direkte Verbindung zu CodePipeline über einen privaten Endpunkt in Ihrer VPC herstellen können, sodass der gesamte Datenverkehr innerhalb Ihrer VPC und des - AWS Netzwerks erhalten 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 mit zu verbinden CodePipeline, definieren Sie einen Schnittstellen-VPC-Endpunkt für CodePipeline. Diese Art von Endpunkt ermöglicht es Ihnen, Ihre VPC mit zu verbinden AWS-Services. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität zu , CodePipeline ohne dass ein Internet-Gateway, eine NAT-Instance (Network Address Translation) oder eine VPN-Verbindung erforderlich ist. Weitere Informationen zur Einrichtung einer VPC finden Sie im VPC-Benutzerhandbuch.

Verschlüsselung im Ruhezustand

Daten in CodePipeline werden im Ruhezustand mit verschlüsselt AWS KMS keys. Codeartefakte 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 der serverseitigen Verschlüsselung für Artefakte, die in Amazon S3 für gespeichert sind CodePipeline.

Verschlüsselung während der Übertragung

Die gesamte service-to-service Kommunikation wird während der Übertragung mit SSL/TLS verschlüsselt.

Verwaltung von Verschlüsselungsschlüsseln

Wenn Sie die Standardoption für die Verschlüsselung von Codeartefakten wählen, CodePipeline verwendet die Von AWS verwalteter Schlüssel. Sie können diesen nicht ändern oder löschen Von AWS verwalteter Schlüssel. Wenn Sie einen vom Kunden verwalteten Schlüssel in verwenden, AWS KMS um Artefakte im S3-Bucket zu verschlüsseln oder zu entschlüsseln, 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 der serverseitigen Verschlüsselung für Artefakte, die in Amazon S3 für gespeichert sind CodePipeline

Es gibt zwei Möglichkeiten, die serverseitige Verschlüsselung für Amazon S3-Artefakte zu konfigurieren:

- CodePipeline erstellt einen S3-Artefakt-Bucket und Standard, Von AWS verwalteter Schlüssel wenn Sie eine Pipeline mit dem Assistenten zum Erstellen einer Pipeline erstellen. Der Von AWS verwalteter Schlüssel wird zusammen mit Objektdaten verschlüsselt und von verwaltet AWS.
- Sie können Ihren eigenen kundenverwalteten 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 in verwenden, AWS KMS um Artefakte im S3-Bucket zu verschlüsseln oder zu entschlüsseln, können Sie diesen vom Kunden verwalteten Schlüssel nach Bedarf ändern oder rotieren.

Amazon S3 unterstützt Bucket-Richtlinien, die Sie verwenden können, wenn Sie eine serverseitige Verschlüsselung für alle in Ihrem Bucket gespeicherten Objekte benötigen. Beispielsweise verweigert die folgende Bucket-Richtlinie jedem die `s3:PutObject`-Berechtigung zum Hochladen von Objekten, wenn die Anfrage nicht den `x-amz-server-side-encryption`-Header enthält, der eine serverseitige Verschlüsselung mit SSE-KMS anfordert.

```
{
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-west-2-89050EXAMPLE/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-west-2-89050EXAMPLE/*",
            "Condition": {
                "Bool": {
```

```
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

Weitere Informationen zur serverseitigen Verschlüsselung und zu finden Sie unter Schutz von Daten mit serverseitiger Verschlüsselung und Schutz von Daten mit serverseitiger Verschlüsselung mit KMS-Schlüsseln AWS KMS, die in (SSE-KMS) gespeichert sind. AWS Key Management Service

Weitere Informationen zu AWS KMS finden Sie im AWS Key Management Service - Entwicklerhandbuch.

Themen

- Anzeigen Ihrer Von AWS verwalteter Schlüssel
- Konfigurieren der serverseitigen Verschlüsselung für S3-Buckets mit AWS CloudFormation oder der AWS CLI

Anzeigen Ihrer 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. Standardmäßig CodePipeline verwendet die serverseitige Verschlüsselung mit AWS KMS unter Verwendung von Von AWS verwalteter Schlüssel für Amazon S3 (der `aws/s3` Schlüssel). Dies Von AWS verwalteter Schlüssel wird in Ihrem AWS Konto erstellt und gespeichert. Wenn Artefakte aus dem S3-Bucket abgerufen werden, CodePipeline verwendet denselben SSE-KMS-Prozess, um das Artefakt zu entschlüsseln.

So zeigen Sie Informationen zu Ihrem an Von AWS verwalteter Schlüssel

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die - AWS KMS Konsole.
2. Wenn eine Willkommensseite angezeigt wird, wählen Sie Jetzt starten aus.
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) festgelegt ist.

Weitere Informationen zu den Regionen und Endpunkten, die für verfügbar sind CodePipeline, 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 der serverseitigen Verschlüsselung für S3-Buckets mit AWS CloudFormation oder der AWS CLI

Wenn Sie AWS CloudFormation oder verwenden, AWS CLI um eine Pipeline zu erstellen, müssen Sie die serverseitige Verschlüsselung manuell konfigurieren. Verwenden Sie das obige Beispiel für eine Bucket-Richtlinie und erstellen Sie dann Ihren eigenen vom Kunden verwalteten Schlüssel. Sie können auch Ihre eigenen Schlüssel anstelle von verwenden Von AWS verwalteter Schlüssel. Einige Gründe für die Auswahl Ihres eigenen Schlüssels sind:

- 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 einer Pipeline in CodePipeline, die Ressourcen aus einem anderen AWS-Konto verwendet.

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 kundenverwalteten Schlüssel erstellen und dann Ihre Anwendungen oder Aliase ändern, um den neuen kundenverwalteten Schlüssel zu verwenden. Sie können auch die automatische Schlüsseldrehung für einen vorhandenen kundenverwalteten Schlüssel aktivieren.

Informationen zum Rotieren Ihres kundenverwalteten 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 von AWS Secrets Manager zum Nachverfolgen von Datenbankpasswörtern oder API-Schlüsseln von Drittanbietern

Wir empfehlen Ihnen, zu verwenden, AWS Secrets Manager um Datenbankanmeldeinformationen, API-Schlüssel und andere Secrets während ihres gesamten Lebenszyklus zu rotieren, zu verwalten und abzurufen. Mit Secrets Manager können Sie fest codierte Anmeldeinformationen in Ihrem Code (einschließlich Passwörtern) durch einen API-Aufruf an Secrets Manager ersetzen, um das Secret programmgesteuert abzurufen. Weitere Informationen finden Sie unter Was ist AWS Secrets Manager? im AWS Secrets-Manager-Benutzerhandbuch.

Für Pipelines, in denen Sie Parameter übergeben, die Secrets sind (z. B. OAuth-Anmeldeinformationen), sollten AWS CloudFormation Sie dynamische Referenzen 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 Referenzen 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 Datenbankanmeldeinformationen automatisch rotieren, z. B. für die Rotation von Amazon-RDS-Secrets. Weitere Informationen finden Sie unter Rotieren Ihrer AWS Secrets-Manager-Secrets im AWS Secrets-Manager-Benutzerhandbuch.
- Anweisungen zum Hinzufügen dynamischer Verweise von Secrets Manager zu Ihren AWS CloudFormation -Vorlagen finden Sie unter https://aws.amazon.com/blogs/security/how-to-create-and-retrieve-secrets-managed-in-aws-secrets-manager-using-aws-cloudformation-template/.

Identity and Access Management für AWS CodePipeline

AWS Identity and Access Management (IAM) ist ein AWS-Service , mit dem ein Administrator den Zugriff auf - AWS Ressourcen sicher steuern kann. IAM-Administratoren steuern, wer für die Nutzung von - CodePipeline Ressourcen authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) werden kann. IAM ist ein AWS-Service , den Sie ohne zusätzliche Kosten verwenden können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- Funktionsweise AWS CodePipeline von mit IAM
- AWS CodePipeline Beispiele für identitätsbasierte -Richtlinien
- Beispiele für eine ressourcenbasierte AWS CodePipeline -Richtlinie
- Fehlerbehebung für AWS CodePipeline -Identität und -Zugriff
- CodePipeline -Berechtigungsreferenz
- Verwalten der CodePipeline Servicerolle

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, unterscheidet sich je nach Ihrer Arbeit in CodePipeline.

Service-Benutzer – Wenn Sie den CodePipeline Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie für Ihre Arbeit weitere CodePipeline Funktionen ausführen, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Unter Fehlerbehebung für AWS CodePipeline -Identität und -Zugriff finden Sie nützliche Informationen für den Fall, dass Sie keinen Zugriff auf eine Funktion in CodePipeline haben.

Service-Administrator – Wenn Sie in Ihrem Unternehmen für CodePipeline Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollständigen Zugriff auf CodePipeline. Ihre Aufgabe besteht darin, zu bestimmen, auf welche CodePipeline Funktionen und Ressourcen Ihre Service-Benutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit verwenden kann CodePipeline, finden Sie unter Funktionsweise AWS CodePipeline von mit IAM.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf CodePipeline verfassen können. Beispiele für CodePipeline identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie unter AWS CodePipeline Beispiele für identitätsbasierte -Richtlinien.

Authentifizierung mit Identitäten

Die Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten bei anmelden. Sie müssen als Root-Benutzer des AWS-Kontos, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle authentifiziert (bei angemeldet AWS) sein.

Sie können sich bei AWS als Verbundidentität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt werden. AWS IAM Identity Center (IAM Identity Center)-Benutzer, die Single-Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für Verbundidentitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie AWS über einen Verbund auf zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, um welchen Benutzertyp es sich handelt, können Sie sich bei der AWS Management Console oder im - AWS Zugriffsportal anmelden. Weitere Informationen zur Anmeldung bei AWS finden Sie unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung - Benutzerhandbuch.

Wenn Sie AWS programmgesteuert auf zugreifen, AWS stellt ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (Command Line Interface, CLI) bereit, um Ihre Anforderungen mithilfe Ihrer -Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenständigen Signieren von Anforderungen finden Sie unter Signieren von AWS API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. empfiehlt beispielsweise, AWS Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

Root-Benutzer des AWS-Kontos

Wenn Sie ein erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet und Sie melden sich mit der E-Mail-Adresse und dem Passwort an, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für

Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management Console indem Sie die Rollen wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI - oder AWS -API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert,

so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.

- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können AWS-Services Sie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff – Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) – Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen in auszuführen AWS, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anforderung AWS-Service , Anforderungen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Service eine Anfrage erhält, die Interaktionen mit anderen AWS-Services oder -Ressourcen erfordert. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle: Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

  - Serviceverknüpfte Rolle – Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einem verknüpft ist AWS-Service. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem angezeigt AWS-Konto und gehören dem Service. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.
- Anwendungen, die auf Amazon EC2 ausgeführt werden – Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und - AWS CLI oder AWS -API-Anforderungen stellen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine - AWS Rolle zuzuweisen und sie für alle ihre Anwendungen verfügbar zu machen, erstellen Sie ein Instance-Profil, das der Instance zugeordnet ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie steuern den Zugriff in , AWS indem Sie Richtlinien erstellen und sie an AWS Identitäten oder Ressourcen anfügen. Eine Richtlinie ist ein Objekt in , AWS das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können AWS JSON-Richtlinien verwenden, um anzugeben, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen

auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen über die AWS Management Console, die AWS CLI oder die AWS -API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem anfügen können AWS-Konto. Verwaltete Richtlinien umfassen AWS -verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder umfassen AWS-Services.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs)– SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in angeben AWS Organizations. AWS Organizations ist ein Service zum Gruppieren und zentralen Verwalten mehrerer AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Die SCP beschränkt Berechtigungen für Entitäten in Mitgliedskonten, einschließlich jeder Root-Benutzer des AWS-Kontos. Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations -Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Funktionsweise AWS CodePipeline von mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf zu verwalten CodePipeline, sollten Sie verstehen, welche IAM-Funktionen Sie mit verwenden können CodePipeline. Einen Überblick darüber AWS-

Services , wie CodePipeline und andere mit IAM funktionieren, finden Sie unter , AWS-Services die mit IAM funktionieren im IAM-Benutzerhandbuch.

Themen

- Identitätsbasierte CodePipeline-Richtlinien
- CodePipeline ressourcenbasierte Richtlinien
- Autorisierung auf der Basis von CodePipeline -Tags
- CodePipeline IAM-Rollen

Identitätsbasierte CodePipeline-Richtlinien

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. CodePipeline unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Aktionen

Administratoren können AWS JSON-Richtlinien verwenden, um anzugeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben in der Regel denselben Namen wie die zugehörige AWS API-Operation. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Richtlinienaktionen in CodePipeline verwenden das folgende Präfix vor der Aktion: `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.

Richtlinienanweisungen müssen entweder ein - `Action` oder -`NotAction`Element enthalten. CodePipeline definiert einen eigenen Satz von Aktionen, die Aufgaben beschreiben, die Sie mit diesem Service durchführen 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 unter Von definierte Aktionen AWS CodePipeline im IAM-Benutzerhandbuch.

Ressourcen

Administratoren können AWS JSON-Richtlinien verwenden, um anzugeben, wer Zugriff auf was hat. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

CodePipeline -Ressourcen und -Operationen

In ist CodePipelinedie primäre Ressource eine Pipeline. In einer Richtlinie verwenden Sie einen Amazon-Ressourcennamen (ARN), um die Ressource zu identifizieren, für die die Richtlinie gilt. CodePipeline unterstützt andere Ressourcen, die mit der primären Ressource verwendet werden können, z. B. Phasen, 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-Ressourcennamen (ARN) und AWS-Service Namespaces im Allgemeine Amazon Web Services-Referenz. Um den Pipeline-ARN abzurufen, der Ihrer Pipeline zugeordnet ist, 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*
Stufe	arn:aws:codepipeline:*region* :*account* :*pipeline-name* /*stage-name*
Aktion	arn:aws:codepipeline:*region* :*account* :*pipeline-name* /*stage-name* /*action-name*
Benutzerdefinierte Aktion	arn:aws:codepipeline:*region* :*account* :actiontype:*owner* /*category* /*provider* /*version*
Alle CodePipeline Ressourcen	arn:aws:codepipeline:*
Alle CodePipeline Ressourcen, die dem angegebenen Konto in der angegebenen Region gehören	arn:aws:codepipeline:*region* :*account* :*

> **Note**
>
> Die meisten Services in AWS behandeln einen Doppelpunkt (:) oder einen Schrägstrich (/) als dasselbe Zeichen in ARNs . CodePipeline 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 CodePipeline, die Berechtigungen auf Ressourcenebene unterstützen. Berechtigungen auf Ressourcenebene geben an, ob ein API-Aufruf einen Ressourcen-ARN angeben können, oder ob der API-Aufruf nur alle Ressourcen unter Verwendung des Platzhalters angeben kann. CodePipeline -Berechtigungsreferenz Eine detaillierte Beschreibung der Berechtigungen auf Ressourcenebene und eine Auflistung der CodePipeline API-Aufrufe, die Berechtigungen auf Ressourcenebene unterstützen, finden Sie unter .

Sie können beispielsweise folgendermaßen eine bestimmte Pipeline (*myPipeline*) in Ihrer Anweisung mit dem ARN angeben:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:myPipeline"
```

Sie können auch alle Pipelines angeben, die zu einem bestimmten Konto gehören, indem Sie das Platzhalterzeichen (*) wie folgt verwenden:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:*"
```

Wenn Sie alle Ressourcen angeben möchten oder wenn eine bestimmte API-Aktion keine ARNs unterstützt, verwenden Sie das Platzhalterzeichen (*) wie folgt im `Resource`-Element:

```
"Resource": "*"
```

> **Note**
>
> Wenn Sie IAM-Richtlinien erstellen, befolgen Sie die Standardsicherheitshinweise zum Erteilen der geringsten Berechtigungen, d. h. nur die Berechtigungen, die zum Ausführen einer Aufgabe erforderlich sind. Wenn ein API-Aufruf ARNs unterstützt, unterstützt er Berechtigungen auf Ressourcenebene, und Sie müssen das Platzhalterzeichen (*) nicht verwenden.

Einige CodePipeline API-Aufrufe akzeptieren mehrere Ressourcen (z. B. `GetPipeline`). Um mehrere Ressourcen in einer einzigen Anweisung anzugeben, trennen Sie die ARNs mit Komma, wie folgt:

```
"Resource": ["arn1", "arn2"]
```

CodePipeline bietet eine Reihe von Operationen für die Arbeit mit den - CodePipeline Ressourcen. Eine Liste der verfügbaren Operationen finden Sie unter CodePipeline -Berechtigungsreferenz.

Bedingungsschlüssel

Administratoren können AWS JSON-Richtlinien verwenden, um festzulegen, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen AND-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen OR Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und servicespezifische Bedingungsschlüssel. Informationen zum Anzeigen aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

CodePipeline definiert einen eigenen Satz von Bedingungsschlüsseln und unterstützt auch die Verwendung einiger globaler Bedingungsschlüssel. Informationen zum Anzeigen 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 dazu, mit welchen Aktionen und Ressourcen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von definierte Aktionen 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 CodePipeline wird jedoch ein ressourcenbasiertes Richtlinienbeispiel für den S3-Service bereitgestellt, der sich auf bezieht.

Beispiele

Beispiele für CodePipeline ressourcenbasierte Richtlinien finden Sie unter Beispiele für eine ressourcenbasierte AWS CodePipeline -Richtlinie,

Autorisierung auf der Basis von CodePipeline -Tags

Sie können Tags an CodePipeline Ressourcen anfügen oder Tags in einer Anforderung 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 Markieren 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 CodePipeline-Ressourcen.

CodePipeline IAM-Rollen

Eine IAM-Rolle ist eine Entität in Ihrem AWS Konto mit bestimmten Berechtigungen.

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 Sicherheitsanmeldeinformationen, indem Sie AWS STS -API-Operationen wie AssumeRole oder aufrufenGetFederationToken.

CodePipeline unterstützt die Verwendung temporärer Anmeldeinformationen.

Servicerollen

CodePipeline ermöglicht es einem Service, eine Servicerolle in Ihrem Namen 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 über die AWS Management Console AWS CLI, die oder die 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 aus einem anderen Konto verwendet, und die zugehörigen Beispielrichtlinien finden Sie unter Erstellen einer Pipeline in CodePipeline, die Ressourcen aus einem anderen AWS-Konto verwendet.

Themen

- Bewährte Methoden für Richtlinien
- Anzeigen von Ressourcen in der Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

- Beispiele für identitätsbasierte Richtlinien (IAM)
- Verwenden von Tags zur Steuerung des Zugriffs auf CodePipeline-Ressourcen
- Erforderliche Berechtigungen für die Verwendung der CodePipeline -Konsole
- AWS Von verwaltete Richtlinien für AWS CodePipeline
- Beispiele für vom Kunden verwaltete Richtlinien

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand CodePipeline Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Erste Schritte mit AWS -verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Berechtigungen – Um Ihren Benutzern und Workloads Berechtigungen zu erteilen, verwenden Sie die -AWS verwalteten Richtlinien, die Berechtigungen für viele häufige Anwendungsfälle 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 für Ihre Anwendungsfälle spezifisch sind. Weitere Informationen finden Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten:Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Service-Aktionen zu gewähren, wenn sie über eine bestimmte verwendet werden AWS-Service, z. B. AWS CloudFormation. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.

- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich – Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Anzeigen von Ressourcen in der Konsole

Die CodePipeline Konsole benötigt die `ListRepositories` Berechtigung zum Anzeigen einer Liste von Repositorys für Ihr AWS Konto in der AWS Region, in der Sie angemeldet sind. Die Konsole umfasst auch eine Funktion Go to resource (Zu Ressource wechseln), um schnell ohne Berücksichtigung der Groß- und Kleinschreibung nach Ressourcen zu suchen. Diese Suche wird in Ihrem AWS Konto in der AWS Region durchgeführt, in der Sie angemeldet sind. Die folgenden Ressourcen werden für die folgenden Services angezeigt:

- AWS CodeBuild: Build-Projekte
- AWS CodeCommit: Repositorys
- AWS CodeDeploy: Anwendungen
- AWS CodePipeline: Pipelines

Für diese Suche unter den Ressourcen in allen Services müssen Sie über die folgenden Berechtigungen verfügen:

- CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`

- CodePipeline: `ListPipelines`

Es werden keine Ergebnisse für die Ressourcen eines Service zurückgegeben, wenn Sie nicht über Berechtigungen für diesen Service verfügen. Auch wenn Sie über Berechtigungen zum Anzeigen von Ressourcen verfügen, werden manche Ressourcen nicht zurückgegeben, wenn die Anzeige dieser Ressourcen ausdrücklich verweigert wird (`Deny`).

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie enthält Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der AWS CLI oder AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Beispiele für identitätsbasierte Richtlinien (IAM)

Richtlinien können IAM-Identitäten angefügt werden. Sie können z. B. Folgendes tun:

- Einem Benutzer oder einer Gruppe in Ihrem Konto eine Berechtigungsrichtlinie zuweisen – Um einem Benutzer Berechtigungen zum Anzeigen von Pipelines in der CodePipeline Konsole zu erteilen, können Sie einem Benutzer oder einer Gruppe, zu der der Benutzer gehört, eine Berechtigungsrichtlinie zuweisen.
- Einer Rolle eine Berechtigungsrichtlinie zuweisen (kontoübergreifende Berechtigungen gewähren) – Sie können einer IAM-Rolle eine identitätsbasierte Berechtigungsrichtlinie zuweisen, um kontoübergreifende Berechtigungen zu erteilen. Beispielsweise kann der Administrator in Konto A eine Rolle erstellen, um einem anderen AWS Konto (z. B. Konto B) oder einem AWS-Service wie folgt kontoübergreifende Berechtigungen zu erteilen:
  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 Berechtigungen zur Übernahme der Rolle an alle Benutzer in Konto B delegieren. Dadurch können Benutzer in Konto B Ressourcen in Konto A erstellen oder darauf zugreifen. Der Prinzipal in der Vertrauensrichtlinie kann auch ein AWS-Service Prinzipal sein, wenn Sie einem AWS-Service Berechtigungen zur Übernahme der Rolle erteilen möchten.

  Weitere Informationen zum Delegieren von Berechtigungen mithilfe von IAM finden Sie unter Zugriffsverwaltung im IAM-Benutzerhandbuch.

Im Folgenden wird ein Beispiel für eine Berechtigungsrichtlinie gezeigt, die Berechtigungen zum Deaktivieren und Aktivieren von Übergängen zwischen allen Phasen in der Pipeline mit dem Namen `MyFirstPipeline` in der gewährt`us-west-2 region`:

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

```
    {
      "Effect" : "Allow",
      "Action" : [
        "codepipeline:EnableStageTransition",
        "codepipeline:DisableStageTransition"
      ],
      "Resource" : [
        "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/*"
      ]
    }
  ]
}
```

Im folgenden Beispiel ist eine Richtlinie im Konto 111222333444 zu sehen, die Benutzern das Anzeigen, jedoch nicht das Ändern der Pipeline `MyFirstPipeline` in der CodePipeline -Konsole erlaubt. 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 CodePipeline erstellt und verwaltet wurden. Weitere Informationen finden Sie unter Arbeiten mit verwalteten Richtlinien. Sie müssen diese Richtlinie an eine IAM-Rolle anfügen, die Sie für den Zugriff erstellen, z. B. eine Rolle mit dem Namen `CrossAccountPipelineViewers`:

```
{
  "Statement": [
    {
      "Action": [
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListActionExecutions",
        "codepipeline:ListActionTypes",
        "codepipeline:ListPipelines",
        "codepipeline:ListTagsForResource",
        "iam:ListRoles",
        "s3:ListAllMyBuckets",
        "codecommit:ListRepositories",
        "codedeploy:ListApplications",
        "lambda:ListFunctions",
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
```

```
        "codestar-notifications:ListTargets"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"
    },
    {
      "Action": [
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListActionExecutions",
        "codepipeline:ListActionTypes",
        "codepipeline:ListPipelines",
        "codepipeline:ListTagsForResource",
        "iam:ListRoles",
        "s3:GetBucketPolicy",
        "s3:GetObject",
        "s3:ListBucket",
        "codecommit:ListBranches",
        "codedeploy:GetApplication",
        "codedeploy:GetDeploymentGroup",
        "codedeploy:ListDeploymentGroups",
        "elasticbeanstalk:DescribeApplications",
        "elasticbeanstalk:DescribeEnvironments",
        "lambda:GetFunctionConfiguration",
        "opsworks:DescribeApps",
        "opsworks:DescribeLayers",
        "opsworks:DescribeStacks"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:DescribeNotificationRule"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "codestar-notifications:NotificationsForResource": "arn:aws:codepipeline:*"
        }
```

```
      }
    }
  ],
  "Version": "2012-10-17"
}
```

Nachdem Sie diese Richtlinie erstellt haben, erstellen Sie die IAM-Rolle im Konto 111222333444 und fügen Sie die Richtlinie dieser Rolle an. In den Vertrauensstellungen der Rolle müssen Sie das AWS Konto hinzufügen, das diese Rolle übernehmen wird. Das folgende Beispiel zeigt eine Richtlinie, mit der Benutzer aus dem Konto *111111111111* AWS Rollen übernehmen können, die im Konto 111222333444 definiert sind:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Das folgende Beispiel zeigt eine Richtlinie, die im Konto *111111111111* AWS erstellt wurde und es Benutzern ermöglicht, die Rolle namens *CrossAccountPipelineViewers* im Konto 111222333444 zu übernehmen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::111222333444:role/CrossAccountPipelineViewers"
        }
    ]
}
```

Sie können IAM-Richtlinien erstellen, um die Aufrufe und Ressourcen einzuschränken, auf die Benutzer in Ihrem Konto Zugriff haben, und diese Richtlinien dann Ihrem Administratorbenutzer anfügen. Weitere Informationen zum Erstellen von IAM-Rollen und Beispiele für IAM-Richtlinienanweisungen für CodePipelinefinden Sie unter Beispiele für vom Kunden verwaltete Richtlinien.

Verwenden von Tags zur Steuerung des Zugriffs auf CodePipeline-Ressourcen

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 Markieren von Ressourcen. Dieses Thema behandelt die Tag-basierte Zugriffskontrolle.

Wenn Sie IAM-Richtlinien entwerfen, können Sie präzise abgestufte Berechtigungen festlegen, indem Sie Zugriff auf bestimmte Ressourcen gewähren. Mit zunehmender Anzahl der Ressourcen, die Sie verwalten, wird diese Aufgabe erschwert. Durch Markieren von Ressourcen und Verwenden von Tags in Richtlinienanweisungsbedingungen lässt sich diese Aufgabe vereinfachen. Sie erteilen Massenzugriff auf eine beliebige Ressource mit einem bestimmten Tag. Anschließend wenden Sie dieses Tag während der Erstellung oder zu einem späteren Zeitpunkt wiederholt auf relevante Ressourcen an.

Markierungen können an die Ressource angehängt oder in der Anfrage an Services übergeben werden, die das Markieren unterstützen. In können CodePipelineRessourcen 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 `IfExists` am Ende eines beliebigen Bedingungsoperatornamens mit Ausnahme der Null-Bedingung hinzufügen. 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 beispielsweise verwenden, `StringEqualsIfExists` um durch Bedingungsschlüssel einzuschränken, die möglicherweise nicht auf anderen Arten von Ressourcen vorhanden sind.

Die vollständige Syntax und Semantik von Tag-Bedingungsschlüsseln finden Sie unter Steuern des Zugriffs mit Tags. Weitere Informationen zu Bedingungsschlüsseln finden Sie in den folgenden Ressourcen. Die CodePipeline Richtlinienbeispiele in diesem Abschnitt stimmen mit den folgenden Informationen zu Bedingungsschlüsseln überein und erweitern sie um Beispiele für Details CodePipeline wie die Verschachtelung von Ressourcen.

- Bedingungsoperatoren für Zeichenfolgen
- AWS-Services , die mit IAM funktionieren
- SCP-Syntax
- IAM-JSON-Richtlinienelemente: Condition
- aws:RequestTag/tag-key
- Bedingungsschlüssel 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 erteilt Benutzern die uneingeschränkte Berechtigung, jede CodePipeline Aktion für jede Ressource auszuführen.

Die folgende Richtlinie schränkt diese Leistung ein und verweigert nicht autorisierten Benutzern die Berechtigung zum Erstellen von Pipelines, in denen bestimmte Tags in der Anforderung 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 soll die Richtlinie nicht autorisierten Benutzern die Berechtigung zum Erstellen einer Pipeline mit den angegebenen Tag-Werten verweigern. Das Erstellen einer Pipeline erfordert jedoch den Zugriff auf Ressourcen zusätzlich zur Pipeline selbst (z. B. Pipeline-Aktionen und -Phasen). Da das in der Richtlinie `'Resource'` angegebene ist`'*'`, wird die Richtlinie anhand jeder Ressource mit einem ARN ausgewertet und erstellt, wenn die Pipeline erstellt

wird. Diese zusätzlichen Ressourcen verfügen nicht über den Tag-Bedingungsschlüssel, daher schlägt die `StringEquals` Prüfung fehl und dem Benutzer wird nicht die Möglichkeit gewährt, 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 lesen als: „Wenn die zu überprüfende Ressource über einen Tag-`"RequestTag/Project"`Bedingungsschlüssel verfügt, erlauben Sie die Aktion nur, 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 verwendet, um keine Tag-Änderungsaktionen zuzulassen, die dieselben Tag-Werte enthalten. Der Administrator eines Kunden muss diese IAM-Richtlinie zusätzlich zur verwalteten Benutzerrichtlinie nicht autorisierten Administratorbenutzern anfügen.

```
{
  "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: Tagging-Aktionen basierend auf Ressourcen-Tags einschränken

Die Richtlinie für `AWSCodePipeline_FullAccess` verwaltete Benutzer erteilt Benutzern die uneingeschränkte Berechtigung, jede CodePipeline Aktion für jede Ressource auszuführen.

Die folgende Richtlinie beschränkt diese Befugnis und verweigert nicht autorisierten Benutzern die Berechtigung, Aktionen für bestimmte Projekt-Pipelines auszuführen. Hierfür verweigert sie einige Aktionen, wenn die Ressource über ein Tag mit dem Namen `Project` und einem der Werte `ProjectA` oder `ProjectB` verfügt. (Der Bedingungsschlüssel `aws:ResourceTag` wird verwendet, um den Zugriff auf die Ressourcen auf der Grundlage der Tags dieser Ressourcen zu steuern.) Der Administrator eines Kunden muss diese IAM-Richtlinie nicht autorisierten IAM-Benutzern hinzufügen, zusätzlich zu der verwalteten Benutzerrichtlinie.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": ["ProjectA", "ProjectB"]
        }
      }
    }
  ]
}
```

Example 3: Zulassen von Aktionen basierend auf Tags in der Anforderung

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-Anforderung 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 `AWSCodePipeline_FullAccess` verwaltete Benutzerrichtlinie nicht zugeordnet ist. Die von verwaltete Richtlinie erteilt Benutzern die uneingeschränkte Berechtigung, jede CodePipeline Aktion für jede Ressource auszuführen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:CreatePipeline",
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Project": "ProjectA"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Example 4: Beschränken Sie Tagging-Aktionen basierend auf Ressourcen-Tags

Die Richtlinie für `AWSCodePipeline_FullAccess` verwaltete Benutzer erteilt Benutzern die uneingeschränkte Berechtigung, jede CodePipeline Aktion für jede 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 den -`aws:TagKeys`Bedingungsschlüssel verwendet, um keine Tag-Änderungsaktionen zum vollständigen Entfernen des `Project` Tags zuzulassen. Der Administrator eines Kunden muss diese IAM-Richtlinie zusätzlich zur verwalteten Benutzerrichtlinie nicht autorisierten Benutzern oder Rollen anfügen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Erforderliche Berechtigungen für die Verwendung der CodePipeline -Konsole

Um CodePipeline in der CodePipeline Konsole verwenden zu können, müssen Sie über einen Mindestsatz von Berechtigungen für die 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

- CodeBuild
- AWS CloudFormation
- AWS CodeDeploy
- AWS Elastic Beanstalk
- AWS Lambda
- AWS OpsWorks

Wenn Sie eine IAM-Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Benutzer mit dieser IAM-Richtlinie. Um sicherzustellen, dass diese Benutzer weiterhin die CodePipeline Konsole verwenden können, fügen Sie dem Benutzer auch die `AWSCodePipeline_ReadOnlyAccess` verwaltete Richtlinie an, wie unter beschriebenAWS Von verwaltete Richtlinien für AWS CodePipeline.

Sie müssen Benutzern, die Aufrufe an die AWS CLI oder die CodePipeline API durchführen, keine Mindestberechtigungen für die Konsole gewähren.

AWS Von verwaltete Richtlinien für AWS CodePipeline

Eine AWS von verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. Von AWS verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele häufige Anwendungsfälle bereitstellen, sodass Sie mit der Zuweisung von Berechtigungen für Benutzer, Gruppen und Rollen beginnen können.

Beachten Sie, dass von AWS verwaltete Richtlinien möglicherweise keine Berechtigungen mit den geringsten Berechtigungen für Ihre spezifischen Anwendungsfälle gewähren, da sie für alle - AWS Kunden verfügbar sind. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie kundenverwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in verwalteten AWS Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS von verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert, wirkt sich die Aktualisierung auf alle Prinzipalidentitäten (Benutzer, Gruppen und Rollen) aus, an die die Richtlinie angefügt ist. aktualisiert am AWS wahrscheinlichsten eine von AWS verwaltete Richtlinie, wenn ein neuer gestartet AWS-Service wird oder neue API-Operationen für vorhandene Services 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 Von verwaltete Richtlinie: **`AWSCodePipeline_FullAccess`**

Dies ist eine Richtlinie, die vollen Zugriff auf gewährt CodePipeline. Informationen zum Anzeigen des JSON-Richtliniendokuments in der IAM-Konsole finden Sie unter AWSCodePipeline_FullAccess.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `codepipeline` – Gewährt Berechtigungen für CodePipeline.
- `chatbot` – Gewährt Berechtigungen, damit Prinzipale Ressourcen in verwalten können AWS Chatbot.
- `cloudformation` – Gewährt Berechtigungen, damit Prinzipale Ressourcen-Stacks in verwalten können AWS CloudFormation.
- `cloudtrail` – Gewährt Berechtigungen, damit Prinzipale Protokollierungsressourcen in verwalten können CloudTrail.
- `codebuild` – Gewährt Berechtigungen, um Prinzipalen den Zugriff auf Build-Ressourcen in zu ermöglichen CodeBuild.
- `codecommit` – Gewährt Berechtigungen, um Prinzipalen den Zugriff auf Quellressourcen in zu ermöglichen CodeCommit.
- `codedeploy` – Gewährt Berechtigungen, um Prinzipalen den Zugriff auf Bereitstellungsressourcen in zu ermöglichen CodeDeploy.
- `codestar-notifications` – Gewährt Berechtigungen, um Prinzipalen den Zugriff auf Ressourcen in AWS CodeStar Benachrichtigungen zu ermöglichen.

- `ec2` – Gewährt Berechtigungen, damit Bereitstellungen in CodeCatalyst das Elastic Load Balancing in Amazon EC2 verwalten können.
- `ecr` – Gewährt Berechtigungen, um den Zugriff auf Ressourcen in Amazon ECR zu ermöglichen.
- `elasticbeanstalk` – Gewährt Berechtigungen, um Prinzipalen den Zugriff auf Ressourcen in Elastic Beanstalk zu ermöglichen.
- `iam` – Gewährt Berechtigungen, damit Prinzipale Rollen und Richtlinien in IAM verwalten können.
- `lambda` – Gewährt Berechtigungen, damit Prinzipale Ressourcen in Lambda verwalten können.
- `events` – Gewährt Berechtigungen, damit Prinzipale Ressourcen in CloudWatch Events verwalten können.
- `opsworks` – Gewährt Berechtigungen, damit Prinzipale Ressourcen in verwalten können AWS OpsWorks.
- `s3` – Gewährt Berechtigungen, damit Prinzipale Ressourcen in Amazon S3 verwalten können.
- `sns` – Gewährt Berechtigungen, damit Prinzipale Benachrichtigungsressourcen in Amazon SNS verwalten können.
- `states` – Gewährt Berechtigungen, damit Prinzipale Zustandsautomaten in anzeigen können AWS Step Functions. Ein Zustandsautomat besteht aus einer Sammlung von Zuständen, die Aufgaben und den Übergang zwischen Zuständen verwalten.

```
{
    "Statement": [
        {
            "Action": [
                "codepipeline:*",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStacks",
                "cloudformation:ListChangeSets",
                "cloudtrail:DescribeTrails",
                "codebuild:BatchGetProjects",
                "codebuild:CreateProject",
                "codebuild:ListCuratedEnvironmentImages",
                "codebuild:ListProjects",
                "codecommit:ListBranches",
                "codecommit:GetReferences",
                "codecommit:ListRepositories",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
```

```
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecs:ListClusters",
                "ecs:ListServices",
                "elasticbeanstalk:DescribeApplications",
                "elasticbeanstalk:DescribeEnvironments",
                "iam:ListRoles",
                "iam:GetRole",
                "lambda:ListFunctions",
                "events:ListRules",
                "events:ListTargetsByRule",
                "events:DescribeRule",
                "opsworks:DescribeApps",
                "opsworks:DescribeLayers",
                "opsworks:DescribeStacks",
                "s3:ListAllMyBuckets",
                "sns:ListTopics",
                "codestar-notifications:ListNotificationRules",
                "codestar-notifications:ListTargets",
                "codestar-notifications:ListTagsforResource",
                "codestar-notifications:ListEventTypes",
                "states:ListStateMachines"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Sid": "CodePipelineAuthoringAccess"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketPolicy",
                "s3:GetBucketVersioning",
                "s3:GetObjectVersion",
                "s3:CreateBucket",
                "s3:PutBucketPolicy"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3::*:codepipeline-*",
            "Sid": "CodePipelineArtifactsReadWriteAccess"
        },
```

```
        {
            "Action": [
                "cloudtrail:PutEventSelectors",
                "cloudtrail:CreateTrail",
                "cloudtrail:GetEventSelectors",
                "cloudtrail:StartLogging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:cloudtrail:*:*:trail/codepipeline-source-trail",
            "Sid": "CodePipelineSourceTrailReadWriteAccess"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/service-role/cwe-role-*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "events.amazonaws.com"
                    ]
                }
            },
            "Sid": "EventsIAMPassRole"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "codepipeline.amazonaws.com"
                    ]
                }
            },
            "Sid": "CodePipelineIAMPassRole"
        },
        {
```

```
            "Action": [
                "events:PutRule",
                "events:PutTargets",
                "events:DeleteRule",
                "events:DisableRule",
                "events:RemoveTargets"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:events:*:*:rule/codepipeline-*"
            ],
            "Sid": "CodePipelineEventsReadWriteAccess"
        },
        {
            "Sid": "CodeStarNotificationsReadWriteAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-notifications:CreateNotificationRule",
                "codestar-notifications:DescribeNotificationRule",
                "codestar-notifications:UpdateNotificationRule",
                "codestar-notifications:DeleteNotificationRule",
                "codestar-notifications:Subscribe",
                "codestar-notifications:Unsubscribe"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "codestar-notifications:NotificationsForResource":
"arn:aws:codepipeline:*"
                }
            }
        },
        {
            "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:SetTopicAttributes"
            ],
            "Resource": "arn:aws:sns:*:*:codestar-notifications*"
        },
        {
            "Sid": "CodeStarNotificationsChatbotAccess",
            "Effect": "Allow",
```

```
            "Action": [
                "chatbot:DescribeSlackChannelConfigurations",
                "chatbot:ListMicrosoftTeamsChannelConfigurations"
            ],
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

AWS Von verwaltete Richtlinie: AWSCodePipeline_ReadOnlyAccess

Dies ist eine Richtlinie, die schreibgeschützten Zugriff 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` – Gewährt Berechtigungen für Aktionen in CodePipeline.
- `codestar-notifications` – Gewährt Berechtigungen, um Prinzipalen den Zugriff auf Ressourcen in AWS CodeStar Benachrichtigungen zu ermöglichen.
- `s3` – Gewährt Berechtigungen, damit Prinzipale Ressourcen in Amazon S3 verwalten können.
- `sns` – Gewährt Berechtigungen, damit Prinzipale Benachrichtigungsressourcen in Amazon SNS verwalten können.

```
{
    "Statement": [
        {
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:GetPipelineExecution",
                "codepipeline:ListPipelineExecutions",
                "codepipeline:ListActionExecutions",
                "codepipeline:ListActionTypes",
```

```
                "codepipeline:ListPipelines",
                "codepipeline:ListTagsForResource",
                "s3:ListAllMyBuckets",
                "codestar-notifications:ListNotificationRules",
                "codestar-notifications:ListEventTypes",
                "codestar-notifications:ListTargets"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketPolicy"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3::*:codepipeline-*"
        },
        {
            "Sid": "CodeStarNotificationsReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-notifications:DescribeNotificationRule"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "codestar-notifications:NotificationsForResource":
 "arn:aws:codepipeline:*"
                }
            }
        }
    ],
    "Version": "2012-10-17"
}
```

AWS Von verwaltete Richtlinie: **`AWSCodePipelineApproverAccess`**

Dies ist eine Richtlinie, die die Berechtigung zum Genehmigen oder Ablehnen einer manuellen Genehmigungsaktion gewährt. Informationen zum Anzeigen des JSON-Richtliniendokuments in der IAM-Konsole finden Sie unter AWSCodePipelineApproverAccess.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `codepipeline` – Gewährt Berechtigungen für Aktionen in CodePipeline.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:GetPipelineExecution",
                "codepipeline:ListPipelineExecutions",
                "codepipeline:ListPipelines",
                "codepipeline:PutApprovalResult"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

AWS Von verwaltete Richtlinie: AWSCodePipelineCustomActionAccess

Dies ist eine Richtlinie, die die Berechtigung zum Erstellen benutzerdefinierter Aktionen in CodePipeline oder zum Integrieren von Jenkins-Ressourcen für Build- oder Testaktionen gewährt. Informationen zum Anzeigen des JSON-Richtliniendokuments in der IAM-Konsole finden Sie unter AWSCodePipelineCustomActionAccess.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `codepipeline` – Gewährt Berechtigungen für Aktionen in CodePipeline.

```
{
    "Statement": [
        {
            "Action": [
                "codepipeline:AcknowledgeJob",
                "codepipeline:GetJobDetails",
                "codepipeline:PollForJobs",
                "codepipeline:PutJobFailureResult",
                "codepipeline:PutJobSuccessResult"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

CodePipeline Von verwaltete Richtlinien und Benachrichtigungen

CodePipeline unterstützt Benachrichtigungen, die Benutzer über wichtige Änderungen an Pipelines informieren können. Verwaltete Richtlinien für CodePipeline enthalten Richtlinienanweisungen 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 zusammen mit den zugehörigen Services CodeCommit CodeBuild, CodeDeployund AWS CodeStar Benachrichtigungen. Die Richtlinie gewährt auch Berechtigungen, die Sie für die Arbeit mit anderen -Services benötigen, die in Ihre Pipelines integriert sind, z. B. Amazon S3, Elastic Beanstalk, CloudTrail Amazon EC2 und AWS CloudFormation. Benutzer mit dieser verwalteten Richtlinie können auch Amazon SNS-Themen für Benachrichtigungen erstellen und verwalten, Benutzer für Themen abonnieren und abmelden, Themen auflisten, die als Ziele für Benachrichtigungsregeln ausgewählt werden sollen, und Clients auflisten AWS Chatbot , 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 CodeStarBenachrichtigungen.

AWS CodePipeline -Aktualisierungen für - AWS verwaltete Richtlinien

Anzeigen von Details zu Aktualisierungen für - AWS verwaltete Richtlinien für , CodePipeline seit dieser Service mit der Verfolgung dieser Änderungen begonnen hat. Um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der CodePipeline Seite -Dokumentverlauf.

Änderung	Beschreibung	Datum
AWSCodePipeline_FullAccess – Aktualisierungen der vorhandenen Richtlinie	CodePipeline hat dieser Richtlinie eine Berechtigung zur Unterstützung `ListStacks` von in hinzugefügt AWS CloudFormation.	15. März 2024
AWSCodePipeline_FullAccess – Aktualisierungen der vorhandenen Richtlinie	Diese Richtlinie wurde aktualisiert, um Berechtigungen für hinzuzufügen AWS Chatbot. Weitere Informationen finden Sie unter CodePipeline Von verwaltete Richtlinien und Benachrichtigungen.	21. Juni 2023
AWSCodePipeline_FullAccess Von und AWSCodePipeline_ReadOnlyAccess verwaltete Richtlinien – Aktualisierungen der vorhandenen Richtlinie	CodePipeline hat diesen Richtlinien eine Berechtigung hinzugefügt, um einen zusätzlichen Benachrichtigungstyp mit AWS Chatbot, zu unterstützen`chatbot:ListMicrosoftTeamsChannelConfigurations` .	16. Mai 2023
AWSCodePipelineFullAccess – Veraltet	Diese Richtlinie wurde ersetzt durch `AWSCodePipeline_FullAccess` .	17. November 2022

Änderung	Beschreibung	Datum
	Nach dem 17. November 2022 kann diese Richtlinie keinen neuen Benutzern, Gruppen oder Rollen angefügt werden. Weitere Informationen finden Sie unter AWS Von verwaltete Richtlinien für AWS CodePipeline.	
AWSCodePipelineReadOnlyAccess – Veraltet	Diese Richtlinie wurde ersetzt durch `AWSCodePipeline_ReadOnlyAccess` . Nach dem 17. November 2022 kann diese Richtlinie keinen neuen Benutzern, Gruppen oder Rollen angefügt werden. Weitere Informationen finden Sie unter AWS Von verwaltete Richtlinien für AWS CodePipeline.	17. November 2022
CodePipeline hat mit der Verfolgung von Änderungen begonnen	CodePipeline hat mit der Verfolgung von Änderungen für seine AWS -verwalteten Richtlinien begonnen.	12. März 2021

Beispiele für vom Kunden verwaltete Richtlinien

In diesem Abschnitt finden Sie Beispiele für Benutzerrichtlinien, die Berechtigungen für verschiedene CodePipeline Aktionen gewähren. Diese Richtlinien funktionieren, wenn Sie die CodePipeline -API, AWS SDKs oder die verwenden AWS CLI. Bei Verwendung der Konsole müssen Sie zusätzliche konsolenspezifische Berechtigungen erteilen. Weitere Informationen finden Sie unter Erforderliche Berechtigungen für die Verwendung der CodePipeline -Konsole.

> **Note**
>
> In allen Beispielen werden die Region USA West (Oregon) (`us-west-2`) und fiktive Konto-IDs verwendet.

Beispiele

- Beispiel 1: Gewähren von Berechtigungen zum Abrufen des Status einer Pipeline
- Beispiel 2: Gewähren von Berechtigungen zum Aktivieren und Deaktivieren von Übergängen zwischen Stufen
- Beispiel 3: Gewähren von Berechtigungen zum Abrufen einer Liste mit allen verfügbaren Aktionstypen
- Beispiel 4: Gewähren von Berechtigungen für das Erlauben oder Ablehnen von manuellen Genehmigungsaktionen
- Beispiel 5: Gewähren von Berechtigungen zum Abfragen von Aufträgen für eine benutzerdefinierte Aktion
- Beispiel 6: Anfügen oder Bearbeiten einer Richtlinie für die Jenkins-Integration mit AWS CodePipeline
- Beispiel 7: Konfigurieren des kontoübergreifenden Zugriffs auf eine Pipeline
- Beispiel 8: Verwenden von AWS -Ressourcen, die mit einem anderen Konto in einer Pipeline verknüpft sind

Beispiel 1: Gewähren von Berechtigungen zum Abrufen des Status einer Pipeline

Im folgenden Beispiel werden Berechtigungen vergeben, um den Status der Pipeline namens `MyFirstPipeline` abzurufen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipelineState"
            ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"
```

```
        }
    ]
}
```

Beispiel 2: Gewähren von Berechtigungen zum Aktivieren und Deaktivieren von Übergängen zwischen Stufen

Im folgenden Beispiel werden Berechtigungen zum Deaktivieren und Aktivieren von Übergängen zwischen allen Stufen in der Pipeline namens `MyFirstPipeline` vergeben:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:DisableStageTransition",
                "codepipeline:EnableStageTransition"
            ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/*"
        }
    ]
}
```

Um dem Benutzer das Deaktivieren und Aktivieren von Übergängen für eine einzelne Stufe in einer Pipeline zu erlauben, müssen Sie die Stufe festlegen. So können Sie z. B. dem Benutzer das Aktivieren und Deaktivieren von Übergängen für eine Stufe mit der Bezeichnung `Staging` in einer Pipeline namens `MyFirstPipeline` erlauben:

```
"Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/Staging"
```

Beispiel 3: Gewähren von Berechtigungen zum Abrufen einer Liste mit allen verfügbaren Aktionstypen

Im folgenden Beispiel werden Berechtigungen zum Abrufen einer Liste mit allen Aktionstypen vergeben, die für Pipelines in der Region `us-west-2` verfügbar sind:

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

```
            "Effect": "Allow",
            "Action": [
                "codepipeline:ListActionTypes"
            ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:actiontype:*"
        }
    ]
}
```

Beispiel 4: Gewähren von Berechtigungen für das Erlauben oder Ablehnen von manuellen Genehmigungsaktionen

Im folgenden Beispiel werden Berechtigungen vergeben, um manuelle Genehmigungsaktionen in einer Stufe mit der Bezeichnung `Staging` in einer Pipeline namens `MyFirstPipeline` zu erlauben oder abzulehnen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PutApprovalResult"
            ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/
Staging/*"
        }
    ]
}
```

Beispiel 5: Gewähren von Berechtigungen zum Abfragen von Aufträgen für eine benutzerdefinierte Aktion

Im folgenden Beispiel werden Berechtigungen für das Pipeline-übergreifende Abfragen von Aufträgen für die benutzerdefinierte Aktion mit der Bezeichnung `TestProvider` vergeben, die ein `Test`-Aktionstyp in der ersten Version ist:

> **Note**
>
> Der Auftrags-Worker für eine benutzerdefinierte Aktion kann unter einem anderen AWS Konto konfiguriert sein oder eine bestimmte IAM-Rolle erfordern, um zu funktionieren.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PollForJobs"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111222333444:actionType:Custom/Test/TestProvider/1"
            ]
        }
    ]
}
```

Beispiel 6: Anfügen oder Bearbeiten einer Richtlinie für die Jenkins-Integration mit AWS CodePipeline

Wenn Sie eine Pipeline für die Verwendung von Jenkins zum Erstellen oder Testen konfigurieren, erstellen Sie eine separate Identität für diese Integration und fügen Sie eine IAM-Richtlinie an, die über die für die Integration zwischen Jenkins und erforderlichen Mindestberechtigungen verfügt CodePipeline. Die Richtlinie ist die gleiche wie die verwaltete Richtlinie `AWSCodePipelineCustomActionAccess`. Das folgende Beispiel zeigt eine Richtlinie für die Jenkins-Integration:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:AcknowledgeJob",
                "codepipeline:GetJobDetails",
                "codepipeline:PollForJobs",
                "codepipeline:PutJobFailureResult",
                "codepipeline:PutJobSuccessResult"
            ],
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

Beispiel 7: Konfigurieren des kontoübergreifenden Zugriffs auf eine Pipeline

Sie können den Pipeline-Zugriff für Benutzer und Gruppen in einem anderen AWS -Konto konfigurieren. Die empfohlene Methode besteht darin, eine Rolle in dem Konto zu erstellen, in dem die Pipeline erstellt wurde. Die Rolle sollte Benutzern des anderen AWS Kontos erlauben, diese Rolle zu übernehmen und auf die Pipeline zuzugreifen. Weitere Informationen finden Sie unter Anleitung: Kontenübergreifender Zugriff mithilfe von Rollen.

Das folgende Beispiel zeigt eine Richtlinie im Konto 80398EXAMPLE, die es Benutzern ermöglicht, die Pipeline mit dem Namen `MyFirstPipeline` in der CodePipeline Konsole 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 CodePipeline erstellt und verwaltet wurden. Weitere Informationen finden Sie unter Arbeiten mit verwalteten Richtlinien. Sie müssen diese Richtlinie an eine IAM-Rolle anfügen, die Sie für den Zugriff erstellen, z. B. eine Rolle mit dem Namen `CrossAccountPipelineViewers`:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:ListActionTypes",
                "codepipeline:ListPipelines",
                "iam:ListRoles",
                "s3:GetBucketPolicy",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "codedeploy:GetApplication",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "elasticbeanstalk:DescribeApplications",
                "elasticbeanstalk:DescribeEnvironments",
                "lambda:GetFunctionConfiguration",
                "lambda:ListFunctions"
            ],
```

```
            "Resource": "arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline"
        }
    ],
    "Version": "2012-10-17"
}
```

Nachdem Sie diese Richtlinie erstellt haben, erstellen Sie die IAM-Rolle im Konto 80398EXAMPLE und fügen Sie die Richtlinie dieser Rolle an. In den Vertrauensstellungen der Rolle müssen Sie das AWS Konto hinzufügen, das diese Rolle übernimmt. Das folgende Beispiel zeigt eine Richtlinie, mit der Benutzer aus dem Konto *111111111111* AWS Rollen übernehmen können, die im Konto 80398EXAMPLE definiert sind:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Das folgende Beispiel zeigt eine Richtlinie, die im Konto *111111111111* AWS erstellt wurde und es Benutzern ermöglicht, die Rolle namens `CrossAccountPipelineViewers` im Konto 80398EXAMPLE zu übernehmen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::80398EXAMPLE:role/CrossAccountPipelineViewers"
        }
    ]
}
```

Beispiel 8: Verwenden von AWS -Ressourcen, die mit einem anderen Konto in einer Pipeline verknüpft sind

Sie können Richtlinien konfigurieren, die es einem Benutzer ermöglichen, eine Pipeline zu erstellen, die Ressourcen in einem anderen AWS Konto verwendet. Dies erfordert die Konfiguration von Richtlinien und Rollen sowohl im Konto, das die Pipeline erstellen wird (AccountA), als auch im Konto, das die Ressourcen erstellt hat, die in der Pipeline verwendet werden sollen (AccountB). Sie müssen auch einen kundenverwalteten Schlüssel in erstellen AWS Key Management Service , der für den kontoübergreifenden Zugriff verwendet wird. Weitere Informationen und step-by-step Beispiele finden Sie unter Erstellen einer Pipeline in CodePipeline, die Ressourcen aus einem anderen AWS-Konto verwendet und Konfigurieren der serverseitigen Verschlüsselung für Artefakte, die in Amazon S3 für gespeichert sind CodePipeline.

Das folgende Beispiel zeigt eine Richtlinie, die von AccountA für einen S3-Bucket konfiguriert wurde, der zum Speichern von Pipeline-Artefakten verwendet wird. Die Richtlinie gewährt Zugriff auf AccountB. Im folgenden Beispiel lautet der ARN für AccountB `012ID_ACCOUNT_B`. Der ARN für den S3-Bucket lautet `codepipeline-us-east-2-1234567890`. Ersetzen Sie diese ARNs durch die ARNs für den S3-Bucket und das Konto, für die Sie Zugriff erlauben möchten:

```
{
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
```

```
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": false
                }
            }
        },
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
            },
            "Action": [
                "s3:Get*",
                "s3:Put*"
            ],
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
        },
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890"
        }
    ]
}
```

Das folgende Beispiel zeigt eine von AccountA konfigurierte Richtlinie, die von AccountB ermöglicht, eine Rolle anzunehmen. Diese Richtlinie muss auf die Servicerolle für CodePipeline angewendet werden (`CodePipeline_Service_Role`). Weitere Informationen zum Anwenden von Richtlinien auf Rollen in IAM finden Sie unter Ändern einer Rolle. Im folgenden Beispiel ist `012ID_ACCOUNT_B` der ARN für AccountB:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": [
            "arn:aws:iam::012ID_ACCOUNT_B:role/*"
```

```
        ]
    }
}
```

Das folgende Beispiel zeigt eine von AccountB konfigurierte Richtlinie, die auf die EC2-Instance-Rolle für angewendet wird CodeDeploy. Diese Richtlinie gewährt Zugriff auf den S3-Bucket, der von AccountA zum Speichern von Pipeline-Artefakten verwendet wird (*codepipeline-us-east-2-1234567890*):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*"
            ],
            "Resource": [
                "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::codepipeline-us-east-2-1234567890"
            ]
        }
    ]
}
```

Das folgende Beispiel zeigt eine Richtlinie für AWS KMS , wobei der ARN des kundenverwalteten Schlüssels ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** ist, der in AccountA erstellt und so konfiguriert wurde, dass AccountB ihn verwenden kann:

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

```
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt",
                "kms:ReEncrypt*",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:us-
east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE"
            ]
        }
    ]
}
```

Das folgende Beispiel zeigt eine Inline-Richtlinie für eine von AccountB erstellte IAM-Rolle (`CrossAccount_Role`), die den Zugriff auf CodeDeploy Aktionen ermöglicht, die von der Pipeline in AccountA benötigt werden.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:CreateDeployment",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision"
            ],
            "Resource": "*"
        }
    ]
}
```

Das folgende Beispiel zeigt eine Inline-Richtlinie für eine von AccountB erstellte IAM-Rolle (`CrossAccount_Role`), die den Zugriff auf den S3-Bucket zum Herunterladen von Eingabeartefakten und Hochladen von Ausgabeartefakten ermöglicht:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
            ]
        }
    ]
}
```

Weitere Informationen zum Bearbeiten einer Pipeline für kontenübergreifenden Zugriff auf Ressourcen finden Sie unter Schritt 2: Bearbeiten der Pipeline .

Beispiele für eine ressourcenbasierte AWS CodePipeline -Richtlinie

Andere Services, z. B. Amazon S3, unterstützen auch ressourcenbasierte Berechtigungsrichtlinien. Beispielsweise können Sie einem S3 Bucket eine Richtlinie zuweisen, um die Zugriffsberechtigungen für diesen Bucket zu verwalten. Obwohl CodePipeline keine ressourcenbasierten Richtlinien unterstützt, speichert es Artefakte, die in Pipelines in versionierten S3-Buckets verwendet werden sollen.

Example Erstellen einer Richtlinie für einen S3-Bucket zur Verwendung als Artefaktspeicher 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:

```
{
```

```
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": false
                }
            }
        }
    ]
}
```

Fehlerbehebung für AWS CodePipeline -Identität und -Zugriff

Verwenden Sie die folgenden Informationen, um häufige Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit CodePipeline und IAM auftreten können.

Themen

- Ich bin nicht autorisiert, eine Aktion in auszuführen CodePipeline
- Ich bin nicht autorisiert, iam durchzuführen:PassRole
- Ich bin Administrator und möchte anderen Zugriff auf gewähren. CodePipeline
- Ich möchte Personen außerhalb meines - AWS Kontos Zugriff auf meine CodePipeline - Ressourcen gewähren

Ich bin nicht autorisiert, eine Aktion in auszuführen CodePipeline

Wenn Ihnen AWS Management Console mitteilt, dass Sie nicht zur Ausführung einer Aktion 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.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` IAM-Benutzer versucht, die Konsole zu verwenden, um Details zu einer Pipeline anzuzeigen, jedoch keine `codepipeline:GetPipeline` Berechtigungen hat.

```
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 autorisiert, iam durchzuführen: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 vorhandene Rolle an diesen Service zu übergeben, anstatt eine neue Servicerolle oder serviceverknüpfte 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 auf gewähren. CodePipeline

Um anderen Personen oder einer Anwendung Zugriff auf zu gewähren CodePipeline, müssen Sie eine IAM-Entität (Benutzer oder Rolle) für die Person oder Anwendung erstellen, die Zugriff benötigt. Sie werden die Anmeldeinformationen für diese Einrichtung verwenden, um auf AWS zuzugreifen. Anschließend müssen Sie der Entität eine Richtlinie anfügen, die dieser die korrekten Berechtigungen in CodePipeline gewährt.

Informationen zum Einstieg finden Sie unter Erstellen Ihrer ersten delegierten IAM-Benutzer und -Gruppen im IAM-Benutzerhandbuch.

Ich möchte Personen außerhalb meines - AWS Kontos Zugriff auf meine CodePipeline -Ressourcen gewähren

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Services, die ressourcenbasierte Richtlinien oder Zugriffssteuerungslisten (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 dazu, ob diese Funktionen CodePipeline unterstützt, finden Sie unter Funktionsweise AWS CodePipeline von mit IAM.
- Informationen zum Gewähren des Zugriffs auf Ihre Ressourcen in Ihrem Besitz finden AWS-Konten Sie unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen AWS-Konto , das Sie besitzen im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre -Ressourcen gewähren AWS-Konten, finden Sie unter Gewähren von Zugriff auf im AWS-Konten Besitz von Dritten im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

CodePipeline -Berechtigungsreferenz

Verwenden Sie die folgende Tabelle als Referenz, wenn Sie die Zugriffskontrolle einrichten und Berechtigungsrichtlinien verfassen, die Sie einer IAM-Identität anfügen können (identitätsbasierte Richtlinien). In der Tabelle werden alle CodePipeline API-Operationen und die zugehörigen Aktionen aufgeführt, für die Sie Berechtigungen zur Durchführung der Aktion erteilen können. Für Operationen, die Berechtigungen auf Ressourcenebene unterstützen, listet die Tabelle die AWS Ressource auf, für die Sie die Berechtigungen erteilen können. Sie geben die Aktionen im Feld `Action` der Richtlinie an.

Berechtigungen auf Ressourcenebene sind solche, mit denen Sie angeben können, 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, basierend auf Bedingungen, die erfüllt sein müssen, oder auf den Ressourcen, die Benutzer verwenden dürfen. Beispielsweise können Sie Benutzern die Berechtigung erteilen, Ausführungsinformationen für die Pipeline aufzulisten, dies aber nur für eine oder mehrere bestimmte Pipeline bzw. Pipelines.

> **Note**
>
> In der Spalte Ressourcen werden die Ressourcen aufgeführt, die für API-Aufrufe erforderlich sind, die Berechtigungen auf Ressourcenebene unterstützen. Bei API-Aufrufen, die keine Berechtigungen auf Ressourcenebene unterstützen, können Sie den Benutzern die Berechtigung zu ihrer Verwendung erteilen, müssen aber für das Ressourcenelement in der Richtlinienanweisung einen Platzhalter (*) einfügen.

CodePipeline API-Operationen und erforderliche Berechtigungen für Aktionen

AcknowledgeJob

Aktion: `codepipeline:AcknowledgeJob`

Erforderlich, um Informationen zu einem bestimmten Auftrag anzuzeigen und zu sehen, ob dieser vom Auftragsworker erhalten wurde. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

AcknowledgeThirdPartyJob

Aktion: `codepipeline:AcknowledgeThirdPartyJob`

Erforderlich, um zu bestätigen, dass ein Auftragsworker den angegebenen Auftrag erhalten hat. Wird nur für Partneraktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

CreateCustomActionType

Aktion: `codepipeline:CreateCustomActionType`

Erforderlich zum Erstellen einer neuen benutzerdefinierten Aktion, die in allen Pipelines verwendet werden kann, die dem AWS Konto zugeordnet sind. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:actiontype:`*`owner`*`/`*`category`*`/`*`provider`*`/`*`versio`

CreatePipeline

Aktion: `codepipeline:CreatePipeline`

Erforderlich zum Erstellen einer Pipeline

Ressourcen:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`

DeleteCustomActionType

Aktion: `codepipeline:DeleteCustomActionType`

Erforderlich, um eine benutzerdefinierte Aktion als gelöscht zu markieren. `PollForJobs` für die benutzerdefinierte Aktion schlägt fehl, nachdem die Aktion zum Löschen markiert wurde. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

DeletePipeline

Aktion: `codepipeline:DeletePipeline`

Erforderlich zum Löschen einer Pipeline

Ressourcen:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

DeleteWebhook

Aktion: `codepipeline:DeleteWebhook`

Erforderlich zum Löschen eines Webhooks.

Ressourcen:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DeregisterWebhookWithThirdParty

Aktion: `codepipeline:DeregisterWebhookWithThirdParty`

Bevor ein Webhook gelöscht wird, erforderlich, um die Verbindung zwischen dem von erstellten Webhook CodePipeline und dem externen Tool mit zu erkennenden Ereignissen zu entfernen. 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 CodePipeline Aktionstypen zu generieren, die Ihrem Konto zugeordnet sind.

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 Pipeline-Namensebene 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/versior`

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 zu allen Aufträgen abzurufen CodePipeline , auf die von reagiert werden soll.

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 zu neuen Revisionen an eine Quelle zu melden.

Ressourcen:

Aktion

`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:

Aktion

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

> **Note**
>
> Dieser API-Aufruf unterstützt Berechtigungen auf Ressourcenebene. Es kann jedoch ein Fehler auftreten, wenn Sie die IAM-Konsole oder den Policy Generator nutzen, um Richtlinien mit "`codepipeline:PutApprovalResult`" zu erstellen, die einen Ressourcen-ARN angeben. Wenn ein Fehler auftritt, können Sie auf der JSON-Registerkarte in der IAM-Konsole oder mit der CLI eine Richtlinie erstellen.

PutJobFailureResult

Aktion: `codepipeline:PutJobFailureResult`

Erforderlich zum Melden des Scheiterns eines Auftrags, wie von einem Auftragsworker an die Pipeline zurückgegeben. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

PutJobSuccessResult

Aktion: `codepipeline:PutJobSuccessResult`

Erforderlich zum Melden des Erfolgs eines Auftrags, wie von einem Auftragsworker an die Pipeline zurückgegeben. Wird nur für benutzerdefinierte Aktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

PutThirdPartyJobFailureResult

Aktion: `codepipeline:PutThirdPartyJobFailureResult`

Erforderlich zum Melden des Scheiterns eines Drittanbieter-Auftrags, wie von einem Auftragsworker an die Pipeline zurückgegeben. Wird nur für Partneraktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

PutThirdPartyJobSuccessResult

Aktion: `codepipeline:PutThirdPartyJobSuccessResult`

Erforderlich zum Melden des Erfolgs eines Drittanbieter-Auftrags, wie von einem Auftragsworker an die Pipeline zurückgegeben. Wird nur für Partneraktionen verwendet.

Ressourcen: Unterstützt nur einen Platzhalter (*) im `Resource`-Richtlinienelement.

PutWebhook

Aktion: `codepipeline:PutWebhook`

Erforderlich zum Erstellen eines Webhooks.

Ressourcen:

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RegisterWebhookWithThirdParty

Aktion: `codepipeline:RegisterWebhookWithThirdParty`

Ressourcen:

Nachdem ein Webhook erstellt wurde, müssen die unterstützten Drittanbieter konfiguriert werden, um die generierte Webhook-URL aufzurufen.

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RetryStageExecution

Aktion: `codepipeline:RetryStageExecution`

Erforderlich zum Wiederaufnehmen der Pipeline-Ausführung durch Wiederholen der letzten fehlgeschlagenen Aktionen in einer Stufe.

Ressourcen:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

StartPipelineExecution

Aktion: `codepipeline:StartPipelineExecution`

Erforderlich, um die angegebene Pipeline zu starten (insbesondere, um mit der Verarbeitung des letzten Commits an den Quellspeicherort zu beginnen, der als Teil der Pipeline angegeben wurde).

Ressourcen:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

TagResource

Aktion: `codepipeline:TagResource`

Erforderlich, um die angegebene Ressource zu markieren.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UntagResource

Aktion: `codepipeline:UntagResource`

Erforderlich, um die angegebene Ressource zu markieren.

Ressourcen:

Aktionstyp

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UpdatePipeline

Aktion: `codepipeline:UpdatePipeline`

Erforderlich zum Aktualisieren einer angegebenen Pipeline mit Bearbeitungen oder Änderungen an seiner Struktur.

Ressourcen:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

Verwalten der CodePipeline Servicerolle

Die CodePipeline Servicerolle ist mit einer oder mehreren Richtlinien konfiguriert, die den Zugriff auf die von der Pipeline verwendeten AWS Ressourcen steuern. Möglicherweise möchten Sie dieser Rolle weitere Richtlinien anfügen, die der Rolle zugeordnete Richtlinie bearbeiten oder Richtlinien für andere Servicerollen in konfigurieren AWS. Sie können auch einer Rolle eine Richtlinie anfügen, wenn Sie den kontenübergreifenden Zugriff auf Ihre Pipeline konfigurieren.

Important

Das Modifizieren einer Richtlinienanweisung oder Anfügen einer weiteren Richtlinie zur Rolle kann Pipelines funktionsunfähig machen. Stellen Sie sicher, dass Sie die Auswirkungen verstehen, bevor Sie die Servicerolle für CodePipeline in irgendeiner Weise ändern. Sie sollten Ihre Pipelines nach Durchführung von Änderungen an der Servicerolle unbedingt testen.

Note

In der Konsole werden Servicerollen, die vor September 2018 erstellt wurden, mit dem Namen `oneClick_AWS-CodePipeline-Service_`*`ID-Number`* erstellt.
Servicerollen, die nach September 2018 erstellt wurden, verwenden das Namensformat der Servicerolle `AWSCodePipelineServiceRole-`*`Region`*`-`*`Pipeline_Name`*. Beispielsweise benennt `eu-west-2`die Konsole für eine Pipeline mit dem Namen `MyFirstPipeline` in die Rolle und Richtlinie `AWSCodePipelineServiceRole-eu-west-2-MyFirstPipeline`.

Entfernen von Berechtigungen aus der CodePipeline-Servicerolle

Sie können die Servicerollenanweisung entsprechend bearbeiten, um den Zugriff auf ungenutzte Ressourcen zu entfernen. Wenn beispielsweise keine Ihrer Pipelines Elastic Beanstalk enthält, können Sie die Richtlinienanweisung bearbeiten, um den Abschnitt zu entfernen, der Zugriff auf Elastic Beanstalk-Ressourcen gewährt.

Wenn keine Ihrer Pipelines enthält CodeDeploy, können Sie die Richtlinienanweisung bearbeiten, um den Abschnitt zu entfernen, der Zugriff auf - CodeDeploy Ressourcen gewährt:

```
    {
    "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetApplicationRevision",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
```

Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle

Sie müssen Ihre Servicerollenrichtlinienanweisung mit Berechtigungen für ein aktualisieren, das AWS-Service nicht bereits in der Standard-Servicerollenrichtlinienanweisung enthalten ist, bevor Sie es in Ihren Pipelines verwenden können.

Dies ist besonders wichtig, wenn die Servicerolle, die Sie für Ihre Pipelines verwenden, erstellt wurde, bevor Support CodePipeline für ein hinzugefügt wurde AWS-Service.

Die folgende Tabelle zeigt, wann Unterstützung für andere hinzugefügt wurde AWS-Services.

AWS-Service	CodePipeline Support-Datum
AWS CloudFormation StackSets -Aktionen	30. Dezember 2020
CodeCommit Format des vollständigen Klonausgabeartefakts	11. November 2020

AWS-Service	CodePipeline Support-Datum
CodeBuild Stapel-Builds	30. Juli 2020
AWS AppConfig	22. Juni 2020
AWS Step Functions	27. Mai 2020
AWS CodeStar Verbindungen	18. Dezember 2019
`CodeDeployToECS` -Aktion	27. November 2018
Amazon ECR	27. November 2018
Servicekatalog	16. Oktober 2018
AWS Device Farm	19. Juli 2018
Amazon ECS	12. Dezember 2017/Update für Opt-in für Tagging-Autorisierung am 21. Juli 2017
CodeCommit	18. April 2016
AWS OpsWorks	2. Juni 2016
AWS CloudFormation	3. November 2016
AWS CodeBuild	1. Dezember 2016
Elastic Beanstalk	Erstes Starten des Services

Gehen Sie folgendermaßen vor, um Berechtigungen für einen unterstützten Service hinzuzufügen:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der IAM-Konsole im Navigationsbereich Rollen und dann Ihre `AWS-CodePipeline-Service` Rolle aus der Rollenliste aus.
3. Wählen Sie auf der Registerkarte Berechtigungen unter Inline-Richtlinien in der Zeile für Ihre Servicerollenrichtlinie die Option Richtlinie bearbeiten aus.

4. Fügen Sie die erforderlichen Berechtigungen im Feld Richtliniendokument hinzu.

> **Note**
>
> Wenn Sie IAM-Richtlinien erstellen, befolgen Sie die Standardsicherheitshinweise zum Erteilen der geringsten Berechtigungen, d. h. nur die Berechtigungen, die zum Ausführen einer Aufgabe erforderlich sind. 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 Berechtigungen gewährt, finden Sie unter https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.

Fügen Sie beispielsweise für die - CodeCommit Unterstützung Folgendes zu Ihrer Richtlinienanweisung hinzu:

```
{
  "Effect": "Allow",
  "Action": [
      "codecommit:GetBranch",
      "codecommit:GetCommit",
      "codecommit:UploadArchive",
      "codecommit:GetUploadArchiveStatus",
      "codecommit:CancelUploadArchive"
           ],
  "Resource": "resource_ARN"
},
```

Fügen Sie zur AWS OpsWorks Unterstützung Ihrer Richtlinienanweisung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "opsworks:CreateDeployment",
        "opsworks:DescribeApps",
        "opsworks:DescribeCommands",
        "opsworks:DescribeDeployments",
```

```
        "opsworks:DescribeInstances",
        "opsworks:DescribeStacks",
        "opsworks:UpdateApp",
        "opsworks:UpdateStack"
    ],
    "Resource": "resource_ARN"
},
```

Fügen Sie zur AWS CloudFormation Unterstützung Ihrer Richtlinienanweisung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStacks",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "iam:PassRole"
    ],
    "Resource": "resource_ARN"
},
```

Beachten Sie, dass die `cloudformation:DescribeStackEvents` Berechtigung optional ist. Es ermöglicht der AWS CloudFormation Aktion, eine detailliertere Fehlermeldung anzuzeigen. Diese Berechtigung kann der IAM-Rolle entzogen werden, wenn Sie nicht möchten, dass Ressourcendetails in den Pipeline-Fehlermeldungen angezeigt werden. Weitere Informationen finden Sie unter AWS CloudFormation.

Fügen Sie zur CodeBuild Unterstützung Folgendes zu Ihrer Richtlinienanweisung hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
```

```
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Unterstützung für Batch-Builds wurde zu einem späteren Zeitpunkt hinzugefügt. In Schritt 11 finden Sie die Berechtigungen zum Hinzufügen zur Servicerolle für Batch-Builds.

Fügen Sie zur AWS Device Farm Unterstützung Ihrer Richtlinienanweisung Folgendes hinzu:

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

Fügen Sie für Service Catalog Folgendes zu Ihrer Richtlinienanweisung hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "servicecatalog:ListProvisioningArtifacts",
        "servicecatalog:CreateProvisioningArtifact",
        "servicecatalog:DescribeProvisioningArtifact",
        "servicecatalog:DeleteProvisioningArtifact",
        "servicecatalog:UpdateProduct"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
```

```
            "cloudformation:ValidateTemplate"
        ],
        "Resource": "resource_ARN"
}
```

5. Fügen Sie für Amazon-ECR-Unterstützung Folgendes zu Ihrer Richtlinienanweisung hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "ecr:DescribeImages"
    ],
    "Resource": "resource_ARN"
},
```

6. Für Amazon ECS sind die folgenden Mindestberechtigungen erforderlich, um Pipelines mit einer Amazon-ECS-Bereitstellungsaktion zu erstellen.

```
{
    "Effect": "Allow",
    "Action": [
        "ecs:DescribeServices",
        "ecs:DescribeTaskDefinition",
        "ecs:DescribeTasks",
        "ecs:ListTasks",
        "ecs:RegisterTaskDefinition",
        "ecs:TagResource",
        "ecs:UpdateService"
    ],
    "Resource": "resource_ARN"
},
```

   Sie können sich für die Verwendung der Tagging-Autorisierung in Amazon ECS anmelden. Wenn Sie sich anmelden, müssen Sie die folgenden Berechtigungen erteilen: `ecs:TagResource`. Weitere Informationen zum Anmelden und zum Ermitteln, ob die Berechtigung erforderlich ist und die Tag-Autorisierung durchgesetzt wird, finden Sie unter Markieren der Autorisierungszeitleiste im Entwicklerhandbuch für Amazon Elastic Container Service.

   Sie müssen auch die `iam:PassRole` Berechtigungen zur Verwendung von IAM-Rollen für Aufgaben hinzufügen. Weitere Informationen finden Sie unter IAM-Rolle für die Amazon-ECS-Aufgabenausführung und IAM-Rollen für Aufgaben. Verwenden Sie den folgenden Richtlinientext.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [

 "arn:aws:iam::aws_account_ID:role/ecsTaskExecutionRole_or_TaskRole_name"
            ]
        }
    ]
}
```

7. Für die -CodeDeployToECSAktion (blaue/grüne Bereitstellungen) sind die folgenden Mindestberechtigungen erforderlich, um Pipelines mit einer zu Amazon-ECS-Blau/Grün-Bereitstellungsaktion CodeDeploy zu erstellen.

```
{
    "Effect": "Allow",
    "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetDeployment",
        "codedeploy:GetApplication",
        "codedeploy:GetApplicationRevision",
        "codedeploy:RegisterApplicationRevision",
        "codedeploy:GetDeploymentConfig",
        "ecs:RegisterTaskDefinition",
        "ecs:TagResource"
    ],
    "Resource": "resource_ARN"
},
```

   Sie können sich für die Verwendung der Tagging-Autorisierung in Amazon ECS anmelden. Wenn Sie sich anmelden, müssen Sie die folgenden Berechtigungen erteilen: `ecs:TagResource`. Weitere Informationen zum Anmelden und zum Ermitteln, ob die Berechtigung erforderlich ist und die Tag-Autorisierung erzwungen wird, finden Sie unter Markieren der Autorisierungszeitleiste im Amazon Elastic Container Service-Entwicklerhandbuch.

Sie müssen auch die `iam:PassRole` Berechtigungen zur Verwendung von IAM-Rollen für Aufgaben hinzufügen. Weitere Informationen finden Sie unter IAM-Rolle für die Amazon-ECS-Aufgabenausführung und IAM-Rollen für Aufgaben. Verwenden Sie den folgenden Richtlinientext.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [

 "arn:aws:iam::aws_account_ID:role/ecsTaskExecutionRole_or_TaskRole_name"
            ]
        }
    ]
}
```

Sie können auch `ecs-tasks.amazonaws.com` zur Liste der Services unter der `iam:PassedToService` Bedingung hinzufügen, wie in diesem Beispiel gezeigt.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "resource_ARN",
            "Condition": {
                "StringEqualsIfExists": {
                    "iam:PassedToService": [
                        "cloudformation.amazonaws.com",
                        "elasticbeanstalk.amazonaws.com",
                        "ec2.amazonaws.com",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        },
```

8. Für AWS CodeStar Verbindungen ist die folgende Berechtigung erforderlich, um Pipelines mit einer Quelle zu erstellen, die eine Verbindung verwendet, z. B. Bitbucket Cloud.

```
{
    "Effect": "Allow",
    "Action": [
        "codestar-connections:UseConnection"
    ],
    "Resource": "resource_ARN"
},
```

   Weitere Informationen zu den IAM-Berechtigungen für Verbindungen finden Sie unter Referenz zu Verbindungsberechtigungen.

9. Für die `StepFunctions` Aktion sind die folgenden Mindestberechtigungen erforderlich, um Pipelines mit einer Step Functions-Aufrufaktion zu erstellen.

```
{
    "Effect": "Allow",
    "Action": [
        "states:DescribeStateMachine",
        "states:DescribeExecution",
        "states:StartExecution"
    ],
    "Resource": "resource_ARN"
},
```

10. Für die `AppConfig` Aktion sind die folgenden Mindestberechtigungen erforderlich, um Pipelines mit einer AWS AppConfig Aufrufaktion zu erstellen.

```
{
    "Effect": "Allow",
    "Action": [
        "appconfig:StartDeployment",
        "appconfig:GetDeployment",
        "appconfig:StopDeployment"
    ],
    "Resource": "resource_ARN"
},
```

11. Um CodeBuild Unterstützung für Batch-Builds zu erhalten, fügen Sie Ihrer Richtlinienanweisung Folgendes hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuildBatches",
        "codebuild:StartBuildBatch"
    ],
    "Resource": "resource_ARN"
},
```

12 Für - AWS CloudFormation StackSets Aktionen sind die folgenden Mindestberechtigungen erforderlich.

- Fügen Sie für die `CloudFormationStackSet` Aktion Folgendes zu Ihrer Richtlinienanweisung hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackSet",
        "cloudformation:UpdateStackSet",
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:DescribeStackSet",
        "cloudformation:ListStackInstances"
    ],
    "Resource": "resource_ARN"
},
```

- Fügen Sie für die `CloudFormationStackInstances` Aktion Folgendes zu Ihrer Richtlinienanweisung hinzu:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation"
    ],
    "Resource": "resource_ARN"
},
```

13 Fügen Sie Ihrer Richtlinienanweisung Folgendes hinzu, um die Option „Vollklon“ zu CodeCommit unterstützen:

```
{
    "Effect": "Allow",
    "Action": [
        "codecommit:GetRepository"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Um sicherzustellen, dass Ihre CodeBuild Aktion die vollständige Klonoption mit einer CodeCommit Quelle verwenden kann, müssen Sie auch die `-codecommit:GitPull`Berechtigung zur Richtlinienanweisung für die CodeBuild Servicerolle Ihres Projekts hinzufügen.

14 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 beschränken möchten. Weitere Informationen zum Erstellen einer Richtlinie, die den Zugriff mit den geringsten Berechtigungen gewährt, finden Sie unter https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.

15. Für eine Pipeline, die Sie für CloudWatch Protokolle konfigurieren möchten, sind die folgenden Mindestberechtigungen erforderlich, die Sie der CodePipeline Servicerolle hinzufügen müssen.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups",
        "logs:PutRetentionPolicy"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Sie sollten Platzhalter in der Ressourcenrichtlinie durch die Ressourcen für das Konto ersetzen, auf das Sie den Zugriff beschränken möchten. Weitere Informationen zum Erstellen einer Richtlinie, die den Zugriff mit den geringsten Berechtigungen gewährt, finden Sie unter https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.

16. Wählen Sie Review policy, um sicherzustellen, dass die Richtlinie keine Fehler enthält. Wenn die Richtlinie fehlerfrei ist, wählen Sie Richtlinie anwenden aus.

Protokollierung und Überwachung in CodePipeline

Sie können die Protokollierungsfunktionen in verwenden AWS , um die Aktionen zu bestimmen, die Benutzer in Ihrem Konto ausgeführt haben, und die verwendeten Ressourcen. Die Protokolldateien enthalten Folgendes:

- Datum und Uhrzeit der Aktionen.
- Die Quell-IP-Adresse für eine Aktion.
- Welche Aktionen aufgrund unzureichender Berechtigungen fehlgeschlagen sind.

Protokollierungsfunktionen sind in den folgenden verfügbar AWS-Services:

- AWS CloudTrail kann verwendet werden, um AWS -API-Aufrufe und zugehörige Ereignisse zu protokollieren, die von oder im Namen eines 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 zu überwachen, die Sie auf ausführen AWS. Sie können Warnungen in Amazon CloudWatch Events basierend auf von Ihnen definierten Metriken erstellen. Weitere Informationen finden Sie unter Überwachen von CodePipeline Ereignissen.

Compliance-Validierung für AWS CodePipeline

Informationen darüber, ob ein in den Geltungsbereich bestimmter Compliance-Programme AWS-Service fällt, finden Sie AWS-Services unter im Geltungsbereich nach Compliance-Programm und wählen Sie das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme

Sie können Auditberichte von Drittanbietern mit herunterladen AWS Artifact. Weitere Informationen finden Sie unter Herunterladen von Berichten unter AWS Artifact .

Ihre Compliance-Verantwortung bei der Verwendung von AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Unterstützung der Compliance bereit:

- Schnellstartanleitungen für Sicherheit und Compliance – In diesen Bereitstellungsleitfäden werden Überlegungen zur Architektur erörtert und Schritte für die Bereitstellung von Basisumgebungen in bereitgestellt AWS , die sich auf Sicherheit und Compliance konzentrieren.
- Architekturerstellung für HIPAA-Sicherheit und -Compliance in Amazon Web Services – In diesem Whitepaper wird beschrieben, wie Unternehmen mithilfe AWS von HIPAA-berechtigte Anwendungen erstellen können.

> **Note**
>
> Nicht alle AWS-Services sind HIPAA-berechtigt. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS Compliance-Ressourcen – Diese Sammlung von Arbeitsmappen und Leitfäden könnte für Ihre Branche und Ihren Standort gelten.
- AWS Kunden-Compliance-Leitfäden – Verstehen Sie das Modell der übergreifenden Verantwortlichkeit anhand der Konformität. Die Leitfäden fassen die bewährten Methoden zur Sicherung zusammen AWS-Services und ordnen die Leitlinien den Sicherheitskontrollen in mehreren Frameworks zu (einschließlich National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Officer (PCI) und International Organization for Standardization (ISO)).
- Bewertung von Ressourcen mit Regeln im -AWS Config Entwicklerhandbuch – Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub – Dies AWS-Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus innerhalb von AWS. Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- AWS Audit Manager – Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um den Umgang mit Risiken und die Einhaltung von Branchenstandards zu vereinfachen.

Ausfallsicherheit in AWS CodePipeline

Die AWS globale -Infrastruktur ist um - AWS Regionen und Availability Zones herum aufgebaut. - AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die mit einem Netzwerk mit niedriger Latenz, hohem Durchsatz und hoher Redundanz 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.

Infrastruktursicherheit in AWS CodePipeline

Als verwalteter Service AWS CodePipeline ist durch die AWS globale Netzwerksicherheit von geschützt. Informationen zu AWS Sicherheitsservices und wie die Infrastruktur AWS schützt, finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung mit den bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastrukturschutz in Security Pillar AWS Well-Architected Framework.

Sie verwenden durch AWS veröffentlichte API-Aufrufe, um CodePipeline über das Netzwerk auf zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Bewährte Methoden für die Gewährleistung der Sicherheit

CodePipeline bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Sie verwenden Verschlüsselung und Authentifizierung für die Quell-Repositorys, die mit Ihren Pipelines verbunden werden. Dies sind die CodePipeline bewährten Methoden für die Sicherheit:

- Wenn Sie eine Pipeline- oder Aktionskonfiguration erstellen, die Geheimnisse wie Token oder Passwörter enthalten muss, geben Sie keine Geheimnisse direkt in die Aktionskonfiguration oder

Standardwerte von Variablen ein, die auf Pipeline-Ebene oder AWS CloudFormation -Konfiguration definiert sind, da die Informationen in Protokollen angezeigt werden. Verwenden Sie Secrets Manager, um Secrets einzurichten und zu speichern, und verwenden Sie dann das referenzierte Secret in der Pipeline- und Aktionskonfiguration, wie unter beschrieben Verwenden von AWS Secrets Manager zum Nachverfolgen von Datenbankpasswörtern oder API-Schlüsseln von Drittanbietern.

- Wenn Sie eine Pipeline erstellen, die einen S3-Quell-Bucket verwendet, konfigurieren Sie die serverseitige Verschlüsselung für Artefakte, die in Amazon S3 für gespeichert sind, CodePipeline indem Sie verwalten AWS KMS keys, wie unter beschriebenKonfigurieren der serverseitigen Verschlüsselung für Artefakte, die in Amazon S3 für gespeichert sind 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 erteilt, die zum Ausführen von Aufgaben für Ihr Projekt erforderlich sind, z. B. das Abrufen von Dateien aus Amazon S3. Eine Anleitung zum Erstellen der Rolle für Ihr Jenkins-Instance-Profil finden Sie unter Erstellen Sie eine IAM-Rolle, die für die Jenkins-Integration verwendet werden soll.

AWSCodePipeline -Befehlszeilenreferenz

Verwenden Sie diese Referenz bei der Arbeit mit AWS CodePipeline-Befehlen und als Ergänzung zu den Informationen im AWS CLI-Benutzerhandbuch und der AWS CLI-Referenz.

Bevor Sie AWS CLI verwenden, sollten Sie sicherstellen, dass die in Erste Schritte mit CodePipeline genannten Voraussetzungen erfüllt sind.

Um eine Liste aller verfügbaren CodePipeline -Befehle anzuzeigen, führen Sie den folgenden Befehl aus:

```
aws codepipeline help
```

Um Informationen zu einem bestimmten CodePipeline -Befehl anzuzeigen, führen Sie den folgenden Befehl aus, wobei*befehl-name*ist der Name eines unten aufgeführten Befehls (z. B.create-pipeline):

```
aws codepipeline command-name help
```

Um Informationen zur Verwendung der Befehle in der CodePipeline -Erweiterung für dieAWS CLI, gehen Sie zu einem oder mehreren der folgenden Abschnitte:

- Erstellen einer benutzerdefinierten Aktion
- Erstellen einer Pipeline (CLI)
- Löschen einer Pipeline (CLI)
- Deaktivieren oder Aktivieren von Übergängen (CLI)
- Anzeigen von der Pipeline-Details und des Verlaufs (CLI)
- Wiederholen fehlgeschlagener Aktionen (CLI)
- Manuelles Starten einer Pipeline (CLI)
- Bearbeiten einer Pipeline (AWS CLI)

Sie können sich auch Beispiele für die Verwendung der meisten dieser Befehle unter CodePipeline - Tutorials ansehen.

CodePipeline Referenz zur Pipeline-Struktur

Standardmäßig hat jede in AWS CodePipeline erfolgreich erstellte Pipeline eine gültige Struktur. Wenn Sie jedoch eine JSON-Datei manuell erstellen oder bearbeiten, um eine Pipeline über die AWS CLI zu erstellen oder zu aktualisieren, könnten Sie unbeabsichtigt eine Struktur schaffen, die nicht gültig ist. Die folgende Referenz kann Ihnen dabei helfen, die Anforderungen hinsichtlich der Struktur Ihrer Pipeline besser zu verstehen und Probleme zu beheben. Beachten Sie die Einschränkungen in Kontingente in AWS CodePipeline, die für alle Pipelines gelten.

Themen

- Gültige Aktionstypen und Anbieter in CodePipeline
- Anforderungen an die Pipeline- und Phasenstruktur in CodePipeline
- Anforderungen an die Aktionsstruktur in CodePipeline

Gültige Aktionstypen und Anbieter in CodePipeline

Das Pipeline-Struktur-Format wird verwendet, um Aktionen und Phasen in einer Pipeline zu erstellen. Ein Aktionstyp besteht aus einer Aktionskategorie und einem Anbietertyp.

Nachfolgend sehen Sie die gültigen Aktionskategorien in CodePipeline:

- Quelle
- Entwicklung
- Test
- Bereitstellen
- Genehmigung
- Aufrufen

Jede Aktionskategorie verfügt über einen designierten Satz von Anbietern. Jeder Aktionsanbieter, wie Amazon S3, hat einen Anbieternamen, z. B. `S3`, der in dem `Provider` Feld in der Aktionskategorie in Ihrer Pipeline-Struktur verwendet werden muss.

Es gibt drei gültige Werte für das `Owner`-Feld im Abschnitt „Aktionskategorie“ in der Pipeline-Struktur: `AWS`, `ThirdParty` und `Custom`.

Informationen zum Anbieternamen und Eigentümerinformationen für Ihren Aktionsanbieter finden Sie unter Referenz der Aktionsstruktur oder Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp.

Diese Tabelle führt die gültigen Anbieter nach Aktionstyp auf.

> **Note**
>
> Informationen zu Bitbucket Cloud- GitHub, GitHub Enterprise Server- oder GitLab .com-Aktionen findest du im Referenzthema CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen Aktionen.

Gültige Aktionsanbieter nach Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
Quelle	Amazon S3	Quellaktion für Amazon S3
	Amazon ECR	Amazon ECR
	CodeCommit	CodeCommit
	CodeStarSourceConnection (für Bitbucket Cloud-GitHub, GitHub Enterprise Server- oder GitLab .com-Aktionen)	CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen
Entwicklung	CodeBuild	AWS CodeBuild
	Benutzerdefiniert CloudBees	Anzahl der Eingabe- und Ausgabe-A

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
		rtefakte für jeden Aktionstyp
	Benutzerdefiniert Jenkins	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
	Benutzerdefiniert TeamCity	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
Test	CodeBuild	AWS CodeBuild
	AWS Device Farm	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
	ThirdParty GhostInspector	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
	Benutzerdefiniert Jenkins	Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
	ThirdParty Micro Focus StormRunner Load	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	ThirdParty Nouvola	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
Bereitstellen	Amazon S3	Amazon S3 S3-Aktionen
	AWS CloudFormation	AWS CloudFormation
	AWS CloudFormation StackSets (beinhaltet die Aktionen `CloudFormationStackSet` und `CloudFormationStackInstances`)	AWS CloudFormation StackSets
	CodeDeploy	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	Amazon ECS	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
	Amazon ECS (blau/grün) (dies ist die `CodeDeployToECS` -Aktion)	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	Elastic Beanstalk	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	AWS AppConfig	AWSAppConfig
	AWS OpsWorks	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	Service Catalog	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
	Amazon Alexa	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp

Aktionskategorie	Gültige Aktionsanbieter	Aktionsreferenz
	Benutzerdefiniert XebiaLabs	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
Genehmigung	Manuell	Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
Aufrufen	AWS Lambda	AWS Lambda
	AWS Step Functions	AWS Step Functions

Einige Aktionstypen in CodePipeline sind nur in ausgewählten AWS Regionen verfügbar. Es ist möglich, dass ein Aktionstyp in einer AWS Region verfügbar ist, aber ein AWS Anbieter für diesen Aktionstyp ist nicht verfügbar.

Weitere Informationen zu den einzelnen Aktionsanbietern finden Sie unter Integrationen mit CodePipeline Aktionstypen.

In den folgenden Abschnitten finden Sie Beispiele für Anbieterinformationen und Konfigurationseigenschaften für jeden Aktionstyp.

Anforderungen an die Pipeline- und Phasenstruktur in CodePipeline

Eine zweiphasige Pipeline hat die folgende Grundstruktur:

```
{
    "roleArn": "An IAM ARN for a service role, such as arn:aws:iam::80398EXAMPLE:role/
CodePipeline_Service_Role",
    "stages": [
```

```
        {
            "name": "SourceStageName",
            "actions": [
            ... See Anforderungen an die Aktionsstruktur in  CodePipeline ...
            ]
        },
        {
            "name": "NextStageName",
            "actions": [
            ... See Anforderungen an die Aktionsstruktur in  CodePipeline ...
            ]
        }
    ],
    "artifactStore": {
        "type": "S3",
        "location": "The name of the Amazon S3 bucket automatically generated for you
 the first time you create a pipeline
            using the console, such as codepipeline-us-east-2-1234567890, or any Amazon
 S3 bucket you provision for this purpose"
    },
    "name": "YourPipelineName",
    "version": 1
}
```

Die Pipeline-Struktur hat folgende Anforderungen:

- Eine Pipeline muss mindestens zwei Phasen enthalten.
- Die erste Phase einer Pipeline muss mindestens eine Quellaktion enthalten. Sie darf nur Quellaktionen enthalten.
- Nur die erste Phase einer Pipeline kann Quellaktionen enthalten.
- Mindestens eine Phase in jeder Pipeline muss eine Aktion enthalten, die keine Quellaktion ist.
- Alle Namen von Phasen in einer Pipeline müssen eindeutig sein.
- Phasennamen können in der CodePipeline Konsole nicht bearbeitet werden. Wenn Sie einen Phasennamen mithilfe der AWS CLI bearbeiten und die Phase eine Aktion mit einem oder mehreren geheimen Parametern (z. B. ein OAuth-Token) enthält, wird der Wert dieser geheimen Parameter nicht erhalten. 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.
- Das `artifactStore` Feld enthält den Artefakt-Bucket-Typ und den Speicherort für eine Pipeline mit allen Aktionen in derselben AWS Region. Wenn Sie Aktionen in einer anderen Region als Ihrer

Pipeline hinzufügen, wird das `artifactStores` Mapping verwendet, um den Artefakt-Bucket für jede AWS Region aufzulisten, in der Aktionen ausgeführt werden. Wenn Sie eine Pipeline erstellen oder bearbeiten, müssen Sie einen Artefakt-Bucket in der Pipelineregion haben, sowie einen Artefakt-Bucket für jede Region, in der Sie eine Aktion ausführen möchten.

Das folgende Beispiel zeigt die grundlegende Struktur für eine Pipeline mit regionsübergreifenden Aktionen, die den `artifactStores`-Parameter verwendet:

```
    "pipeline": {
        "name": "YourPipelineName",
        "roleArn": "CodePipeline_Service_Role",
        "artifactStores": {
            "us-east-1": {
                "type": "S3",
                "location": "S3 artifact bucket name, such as codepipeline-us-
east-1-1234567890"
            },
            "us-west-2": {
                "type": "S3",
                "location": "S3 artifact bucket name, such as codepipeline-us-
west-2-1234567890"
            }
        },
        "stages": [
            {

...
```

- Die Pipeline-Metadatenfelder unterscheiden sich von der Pipelinestruktur und können nicht bearbeitet werden. Wenn Sie eine Pipeline aktualisieren, wird das Datum im Metadatenfeld `updated` automatisch geändert.
- Wenn Sie eine Pipeline bearbeiten oder aktualisieren, kann der Pipelinename nicht geändert werden.

> **Note**
>
> Wenn Sie eine vorhandene Pipeline umbenennen möchten, können Sie mit dem CLI-Befehl `get-pipeline` eine JSON-Datei erstellen, die die Struktur Ihrer Pipeline enthält. Anschließend können Sie mit dem CLI-Befehl `create-pipeline` eine Pipeline mit dieser Struktur erstellen und ihr einen neuen Namen geben.

Die Versionsnummer einer Pipeline wird bei jedem Pipeline-Update automatisch generiert und aktualisiert.

Anforderungen an die Aktionsstruktur in CodePipeline

Eine Aktion hat die folgende anspruchsvolle Struktur:

```
[
                {
                   "inputArtifacts": [
                        An input artifact structure, if supported for the action
 category
                    ],
                   "name": "ActionName",
                   "region": "Region",
                   "namespace": "source_namespace",
                   "actionTypeId": {
                        "category": "An action category",
                        "owner": "AWS",
                        "version": "1"
                        "provider": "A provider type for the action category",
                   },
                   "outputArtifacts": [
                        An output artifact structure, if supported for the action
 category
                   ],
                   "configuration": {
                        Configuration details appropriate to the provider type
                   },
                   "runOrder": A positive integer that indicates the run order within
 the stage,
                }
            ]
```

Eine Liste der Beispiel-`configuration` Details für den jeweiligen Anbietertyp finden Sie unter Konfigurationsdetails nach Anbietertyp.

Die Aktionsstruktur hat folgende Anforderungen:

- Alle Aktionsnamen in einer Phase müssen eindeutig sein.

- Das Eingabeartefakt einer Aktion muss exakt mit dem in einer vorherigen Aktion deklarierten Ausgabeartefakt übereinstimmen. Wenn beispielsweise eine vorherige Aktion die folgende Erklärung enthält:

  ```
  "outputArtifacts": [
      {
      "MyApp"
      }
  ],
  ```

  und keine anderen Ausgabeartefakte vorhanden sind, muss das Eingabeartefakt einer nachfolgenden Aktion folgendermaßen lauten:

  ```
  "inputArtifacts": [
      {
      "MyApp"
      }
  ],
  ```

  Dies trifft für alle Aktionen zu, unabhängig davon, ob sie in derselben Phase oder in folgenden Phasen enthalten sind. Das Eingabeartefakt muss aber nicht die nächste Aktion sein, die strikt auf die Aktion folgt, von der das Ausgabeartefakt bereitgestellt wurde. Parallele Aktionen können unterschiedliche Ausgabeartefaktpakete deklarieren, die im Gegenzug von verschiedenen Folgeaktionen genutzt werden.
- Namen von Ausgabeartefakten müssen in einer Pipeline eindeutig sein. Eine Pipeline kann z. B. eine Aktion mit einem Ausgabeartefakt namens `"MyApp"` und eine weitere Aktion mit einem Ausgabeartefakt namens `"MyBuiltApp"` enthalten. Eine Pipeline kann jedoch nicht zwei Aktionen enthalten, die beide ein Ausgabeartefakt mit dem Namen `"MyApp"` haben.
- Bei regionsübergreifenden Aktionen wird das `Region` Feld verwendet, um die AWS Region anzugeben, in der die Aktionen erstellt werden sollen. Die für diese Aktion erstellten AWS Ressourcen müssen in derselben Region erstellt werden, die `region` im Feld angegeben wurde. Für die folgenden Aktionstypen können Sie keine regionsübergreifenden Aktionen erstellen:
  - Quellaktionen
  - Aktionen von Drittanbietern
  - Aktionen von benutzerdefinierten Anbietern

- Aktionen können mit Variablen konfiguriert werden. Sie legen die Namespace- und Variableninformationen für Ausführungsvariablen im Feld `namespace` fest. Referenzinformationen zu Ausführungsvariablen und Aktionsausgabevariablen finden Sie unter Variablen.
- Für alle derzeit unterstützten Aktionstypen ist die einzig gültige BesitzerzeichenfolgeAWS,ThirdParty, oderCustom. Weitere Informationen finden Sie in der CodePipeline -API-Referenz.
- Der standardmäßige `runOrder`-Wert für eine Aktion ist 1. Der Wert muss eine positive Ganzzahl (natürliche Zahl) sein. Sie können keine Bruchzahlen, Dezimalzahlen, negative Zahlen oder Null verwenden. Für eine serielle Reihenfolge von Aktionen geben Sie die niedrigste Zahl für die erste Aktion und höhere Zahlen für jede der verbleibenden Aktionen in der Abfolge an. Parallele Aktionen geben Sie an, indem Sie dieselbe Ganzzahl für jede Aktion verwenden, die parallel ausgeführt werden soll. In der Konsole können Sie eine serielle Reihenfolge für eine Aktion angeben, indem Sie Aktionsgruppe hinzufügen auf der Ebene der Phase auswählen, auf der sie ausgeführt werden soll, oder Sie können eine parallel Reihenfolge angeben, indem Sie Aktion hinzufügen wählen. Aktionsgruppe bezieht sich auf eine Ausführungsreihenfolge einer oder mehrerer Aktionen auf derselben Ebene.

  Wenn Sie beispielsweise drei Aktionen der Reihe nach in einer Phase ausführen möchten, geben Sie der ersten Aktion den `runOrder`-Wert 1, der zweiten Aktion den `runOrder`-Wert 2 und der dritten den `runOrder`-Wert 3. Wenn Sie jedoch die zweite und dritte Aktion parallel ausführen möchten, geben Sie der ersten Aktion den `runOrder`-Wert 1 und sowohl der zweiten als auch der dritten Aktion den `runOrder`-Wert 2.

  > **Note**
  >
  > Die Nummerierung aufeinanderfolgender Aktionen muss nicht in strenger Reihenfolge sein. Wenn Sie beispielsweise drei Aktionen in einer Reihenfolge haben und die zweite Aktion entfernen möchten, müssen Sie den `runOrder`-Wert der dritten Aktion nicht neu nummerieren. Da der `runOrder`-Wert dieser Aktion (3) höher ist als der `runOrder`-Wert der ersten Aktion (1), wird sie nach der ersten Aktion in der Phase ausgeführt.

- Wenn Sie einen Amazon S3 S3-Bucket als Bereitstellungsort verwenden, geben Sie auch einen Objektschlüssel an. Ein Objektschlüssel kann ein Dateiname (Objekt) oder eine Kombination aus Präfix (Ordnerpfad) und Dateiname sein. Sie können Variablen verwenden, um den Namen des Speicherorts anzugeben, der von der Pipeline verwendet werden soll. Für Amazon S3-Bereitstellungsaktionen werden in Amazon S3-Objektschlüsseln die folgenden Variablen unterstützt.

Verwenden von Variablen in Amazon S3

Variable	Beispiel für Konsoleneingabe	Ausgang
datetime	js-application/{datetime}.zip	UTC-Zeitstempel im folgenden Format: <JJJJ>-<MM>-TT>_<HH>-<MM>-<SS> Beispiel: js-application/2019-01-10_07-39-57.zip
uuid	js-application/{uuid}.zip	Die UUID ist ein global eindeutiger Bezeichner, der garantiert mit keinem der vorhandenen Bezeichner identisch ist. Die UUID hat das Format (alle Ziffern im Hexadezimalformat): <8-Ziffern>-<4-Ziffern>-4-Ziffern>-<4-Ziffern>-<12-Ziffern> Beispiel: js-application/54a60075-b96a-4bf3-9013-db3a9EXAMPLE.zip

- Dies sind die gültigen `actionTypeId` Kategorien für CodePipeline:
  - `Source`
  - `Build`
  - `Approval`
  - `Deploy`
  - `Test`
  - `Invoke`

  Einige Anbietertypen und Konfigurationsoptionen werden hier angegeben.
- Welche Anbietertypen für eine Aktionskategorie gültig sind, hängt von der Kategorie ab. Beispielsweise ist ein gültiger Anbietertyp für einen Quellaktionstyp `S3`, `GitHub`, `CodeCommit` oder `Amazon ECR`. Dieses Beispiel zeigt die Struktur für eine Quellaktion mit einem `S3`-Anbieter:

```
"actionTypeId": {
  "category": "Source",
  "owner": "AWS",
  "version": "1",
  "provider": "S3"},
```

- Jede Aktion muss eine gültige Aktionskonfiguration aufweisen in Abhängigkeit zum Anbietertyp für diese Aktion. In der folgenden Tabelle sind die erforderlichen Aktionskonfigurationselemente für jeden gültigen Anbietertyp aufgeführt:

  Aktionskonfigurationseigenschaften für Anbietertypen

Name des Anbieters	Anbietername in Aktionstyp	Konfigurationseigenschaften	Erforderliche Eigenschaft?
Amazon S3 (Aktionsanbieter bereitstellen)	Weitere Informationen, einschließlich Beispielen zu Amazon S3 S3-Bereitstellungsaktionsparametern, finden Sie unterAmazon S3 S3-Aktionen.		
Amazon S3 (Quellaktionsanbieter)	Weitere Informationen, einschließlich Beispielen zu Amazon S3 S3-Quellaktionsparametern, finden Sie unterQuellaktion für Amazon S3.		
Amazon ECR	Weitere Informationen, einschließlich Beispielen zu Amazon ECR-Parametern, finden Sie unterAmazon ECR.		
CodeCommit	Weitere Informationen, einschließlich Beispielen zu CodeCommit Parametern, finden Sie unterCodeCommit.		
GitHub	Weitere Informationen, einschließlich Beispielen zu GitHub Parametern, finden Sie unterGitHub Referenz zur Quellaktionsstruktur der Version 1.		
AWS CloudFormation	Weitere Informationen, einschließlich Beispiele im Zusammenhang mit AWS CloudFormation-Parametern, finden Sie unter AWS CloudFormation.		

Name des Anbieters	Anbietername in Aktionstyp	Konfigurationseigenschaften	Erforderliche Eigenschaft?
CodeBuild	Weitere Beschreibungen und Beispiele zu CodeBuild Parametern finden Sie unterAWS CodeBuild.		
CodeDeploy	Weitere Beschreibungen und Beispiele zu CodeDeploy Parametern finden Sie unterAWS CodeDeploy.		
AWS Device Farm	Weitere Informationen und Beispiele für AWS Device Farm-Parameter finden Sie unter AWS Device Farm.		
AWS Elastic Beanstalk	`ElasticBeanstalk`	`ApplicationName`	Erforderlich
		`EnvironmentName`	Erforderlich
AWS Lambda	Weitere Informationen, einschließlich Beispiele im Zusammenhang mit AWS Lambda-Parametern, finden Sie unter AWS Lambda.		
AWS OpsWorks Stacks	`OpsWorks`	`Stack`	Erforderlich
		`Layer`	Optional
		App	Erforderlich
Amazon ECS	Weitere Beschreibungen und Beispiele zu Amazon ECS-Parametern finden Sie unterAmazon Elastic Container Service.		
Amazon ECS und CodeDeploy (Blau/Grün)	Weitere Beschreibungen und Beispiele zu Amazon ECS und den CodeDeploy Blau/Grün-Parametern finden Sie unter. Amazon Elastic Container Service undCodeDeploy Blau-Grün-		
Service Catalog	`ServiceCatalog`	`TemplateFilePath`	Erforderlich
		`ProductVersionName`	Erforderlich
		`ProductType`	Erforderlich

Name des Anbieters	Anbietername in Aktionstyp	Konfigurationseigenschaften	Erforderliche Eigenschaft?
		`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 (zum Beispiel) *`MyJenkinsProviderName`*	`ProjectName`	Erforderlich
Manuelle Genehmigung	`Manual`	`CustomData`	Optional
		`ExternalEntityLink`	Optional
		`NotificationArn`	Optional

Themen

- Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp
- Standardeinstellungen für den Parameter PollForSourceChanges
- Konfigurationsdetails nach Anbietertyp

Anzahl der Eingabe- und Ausgabe-Artefakte für jeden Aktionstyp

Je nach Aktionstyp können Sie die folgenden Zahlen für Eingabe- und Ausgabeartefakte verwenden:

Aktionstypbedingungen für Artefakte

Eigentümer	Aktionstyp	Anbieter	Gültige Zahl der Eingabeartefakte	Gültige Zahl der Ausgabeartefakte
AWS	Quelle	Amazon S3	0	1
AWS	Quelle	CodeCommit	0	1
AWS	Quelle	Amazon ECR	0	1
ThirdParty	Quelle	GitHub	0	1
AWS	Entwicklung	CodeBuild	1 bis 5	0 bis 5
AWS	Test	CodeBuild	1 bis 5	0 bis 5
AWS	Test	AWS Device Farm	1	0
AWS	Genehmigung	Manuell	0	0
AWS	Bereitstellen	Amazon S3	1	0
AWS	Bereitstellen	AWS CloudFormation	0 bis 10	0 bis 1
AWS	Bereitstellen	CodeDeploy	1	0
AWS	Bereitstellen	AWS Elastic Beanstalk	1	0
AWS	Bereitstellen	AWS OpsWorks Stacks	1	0
AWS	Bereitstellen	Amazon ECS	1	0
AWS	Bereitstellen	Service Catalog	1	0
AWS	Aufrufen	AWS Lambda	0 bis 5	0 bis 5

Eigentümer	Aktionstyp	Anbieter	Gültige Zahl der Eingabeartefakte	Gültige Zahl der Ausgabeartefakte
`ThirdParty`	Bereitstellen	Alexa Skills Kit	1 bis 2	0
`Custom`	Entwicklung	Jenkins	0 bis 5	0 bis 5
`Custom`	Test	Jenkins	0 bis 5	0 bis 5
`Custom`	Jede unterstützte Kategorie	Wie in der benutzerdefinierten Aktion angegeben	0 bis 5	0 bis 5

Standardeinstellungen für den Parameter PollForSourceChanges

Der Standard für den Parameter `PollForSourceChanges` wird von der Methode festgelegt, mit der die Pipeline erstellt wird, wie in der nachfolgenden Tabelle beschrieben. In vielen Fällen ist die Standardeinstellung für den Parameter `PollForSourceChanges` „true“ und muss deaktiviert werden.

Wenn der `PollForSourceChanges`-Parameter standardmäßig „true“ ist, sollten Sie wie folgt vorgehen:

- Fügen Sie den Parameter `PollForSourceChanges` der JSON-Datei oder der AWS CloudFormation-Vorlage hinzu.
- Erstellen Sie Ressourcen zur Änderungserkennung (CloudWatch Eventregel, sofern zutreffend).
- Stellen Sie den `PollForSourceChanges`-Parameter auf "false" ein.

> **Note**
>
> Wenn Sie eine CloudWatch Ereignisregel oder einen Webhook erstellen, müssen Sie den Parameter auf false setzen, um zu verhindern, dass die Pipeline mehr als einmal ausgelöst wird.

Der `PollForSourceChanges` Parameter wird nicht für Amazon ECR-Quellaktionen verwendet.

- PollForSourceChanges Standardwerte für Parameter

Quelle	Erstellungsmethode	Beispiel für „Konfiguration“ JSON-Struktur-Ausgabe
CodeCommit	Pipeline wird mit der Konsole erstellt (und die Konsole erstellt Änderungserkennung sressourcen). Der Parameter wird in der Pipeline-Strukturausgabe angezeigt und ist standardmäßig `false`.	`BranchName": "main", "PollForSourceCha nges": "false", "RepositoryName": "my- repo"`
	Die Pipeline wird mit der CLI oder erstelltA WS CloudFormation, und der `PollForSo urceChanges` Parameter wird nicht in der JSON-Ausgabe angezeigt, sondern auf `true` .[2] gesetzt.	`BranchName": "main", "RepositoryName": "my- repo"`
Amazon S3	Pipeline wird mit der Konsole erstellt (und die Konsole erstellt Änderungserkennung sressourcen). Der Parameter wird in der Pipeline-Strukturausgabe angezeigt und ist standardmäßig `false`.	`"S3Bucket": "my-bucke t", "S3ObjectKey": "object.zip", "PollForSourceChanges" : "false"`
	Die Pipeline wird mit der CLI oder erstelltA WS CloudFormation, und der `PollForSo urceChanges` Parameter wird nicht in der JSON-Ausgabe angezeigt, sondern auf `true` .[2] gesetzt.	`"S3Bucket": "my-bucke t", "S3ObjectKey": "object.zip"`

Quelle	Erstellungsmethode	Beispiel für „Konfiguration“ JSON-Struktur-Ausgabe
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 erstelltA WS CloudFormation, und der `PollForSourceChanges` Parameter wird nicht in der JSON-Ausgabe angezeigt, sondern auf `true` .[2] gesetzt.	`"Owner": "MyGitHubAccountName ", "Repo": "MyGitHubRepositoryName ", "Branch": " main", "OAuthToken": " ****"`
	[2] Wenn `PollForSourceChanges` zu irgendeinem Zeitpunkt der JSON-Struktur oder der AWS CloudFormation Vorlage hinzugefügt wurde, wird sie wie folgt angezeigt: `"PollForSourceChanges": "true",`	
	[3] Informationen zu den Ressourcen zur Änderungserkennung, die für jeden Quellanbieter gelten, finden Sie unter Methoden zur Änderungserkennung.	

Konfigurationsdetails nach Anbietertyp

In diesem Abschnitt werden gültige `configuration`-Parameter für jeden Aktionsanbieter aufgeführt.

Das folgende Beispiel zeigt eine gültige Konfiguration für eine Bereitstellungsaktion, die Service Catalog verwendet, für eine Pipeline, die in der Konsole ohne separate Konfigurationsdatei erstellt wurde:

```
"configuration": {
```

```
  "TemplateFilePath": "S3_template.json",
  "ProductVersionName": "devops S3 v2",
  "ProductType": "CLOUD_FORMATION_TEMPLATE",
  "ProductVersionDescription": "Product version description",
  "ProductId": "prod-example123456"
}
```

Das folgende Beispiel zeigt eine gültige Konfiguration für eine Bereitstellungsaktion, die Service Catalog verwendet, für eine Pipeline, die in der Konsole mit einer separaten `sample_config.json` Konfigurationsdatei erstellt wurde:

```
"configuration": {
  "ConfigurationFilePath": "sample_config.json",
  "ProductId": "prod-example123456"
}
```

Das folgende Beispiel zeigt eine gültige Konfiguration für eine Bereitstellungsaktion, für die das Alexa Skills Kit verwendet wird:

```
"configuration": {
  "ClientId": "amzn1.application-oa2-client.aadEXAMPLE",
  "ClientSecret": "****",
  "RefreshToken": "****",
  "SkillId": "amzn1.ask.skill.22649d8f-0451-4b4b-9ed9-bfb6cEXAMPLE"
}
```

Das folgende Beispiel zeigt eine gültige Konfiguration für eine manuelle Genehmigung:

```
"configuration": {
  "CustomData": "Comments on the manual approval",
  "ExternalEntityLink": "http://my-url.com",
  "NotificationArn": "arn:aws:sns:us-west-2:12345EXAMPLE:Notification"
}
```

Referenz der Aktionsstruktur

Dieser Abschnitt ist nur eine Referenz für die Aktionskonfiguration. Eine konzeptionelle Übersicht über die Pipeline-Struktur finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

Jeder Aktionsanbieter in 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 ECR
- Amazon Elastic Container Service undCodeDeploy Blau-Grün-
- Amazon Elastic Container Service
- Amazon S3 S3-Aktionen
- Quellaktion für Amazon S3
- AWSAppConfig
- AWS CloudFormation
- AWS CloudFormation StackSets
- AWS CodeBuild
- CodeCommit
- AWS CodeDeploy
- CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen
- AWS Device Farm

- AWS Lambda
- Referenz zur Snyk-Aktionsstruktur
- AWS Step Functions

Amazon ECR

Löst die Pipeline aus, wenn ein neues Image per Push in das Amazon ECR-Repository verschoben wird. Diese Aktion stellt eine Image-Definitionsdatei bereit, die auf den URI für das Image verweist, das an Amazon ECR übertragen wurde. Diese Quellaktion wird häufig in Verbindung mit einer anderen Quellaktion wie CodeCommit verwendet, um einen Quellspeicherort für alle weiteren Quellartefakte zuzulassen. Weitere Informationen finden Sie unter Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy .

Wenn Sie die Pipeline mithilfe der Konsole erstellen oder bearbeiten, CodePipeline erstellt ein CloudWatch Ereignisregel, die Ihre Pipeline startet, wenn es eine Änderung im Repository gibt.

Sie müssen ein Amazon ECR-Repository erstellen und ein Image per Push verschieben, bevor Sie die Pipeline über eine Amazon ECR-Aktion verbinden können.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Aktionsdeklaration (Beispiel für Amazon ECR)
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Source`
- Eigentümer: `AWS`
- Anbieter: `ECR`
- Version: `1`

Konfigurationsparameter

RepositoryName

Erforderlich Ja

Der Name des Amazon ECR-Repositorys, zu dem das Abbild per Push verschoben 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 Eingabeartefakte sind für diesen Aktionstyp nicht gültig.

Ausgabeartefakte

- Anzahl der Artefakte `1`
- Beschreibung Diese Aktion erzeugt ein Artefakt, das ein`imageDetail.json`Datei, die den URI für das Image enthält, das die Pipeline-Ausführung ausgelöst hat. Informationen zur `imageDetail.json`-Datei finden Sie unter imageDetail.json-Datei für Amazon ECS-Blau/Grün-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 Variablen.

RegistryId

DieAWSKonto-ID, die der Registrierung zugeordnet ist, die das Repository enthält.

RepositoryName

Der Name des Amazon ECR-Repositorys, zu dem das Abbild per Push verschoben wurde.

ImageTag

Das Tag, das für das Image verwendet wird.

ImageDigest

Der `sha256`-Digest des Image-Manifests.

ImageURI

Der URI für das Image.

Aktionsdeklaration (Beispiel für Amazon ECR)

YAML

```
Name: Source
Actions:
  - InputArtifacts: []
    ActionTypeId:
      Version: '1'
      Owner: AWS
      Category: Source
      Provider: ECR
    OutputArtifacts:
      - Name: SourceArtifact
    RunOrder: 1
    Configuration:
      ImageTag: latest
      RepositoryName: my-image-repo
```

```
    Name: ImageSource
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "InputArtifacts": [],
            "ActionTypeId": {
                "Version": "1",
                "Owner": "AWS",
                "Category": "Source",
                "Provider": "ECR"
            },
            "OutputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "RunOrder": 1,
            "Configuration": {
                "ImageTag": "latest",
                "RepositoryName": "my-image-repo"
            },
            "Name": "ImageSource"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy — Dieses Tutorial enthält eine Beispieldatei für App-Spezifikationen und -Beispiel CodeDeploy Anwendungs- und Bereitstellungsgruppe zum Erstellen einer Pipeline mit einem CodeCommit und Amazon ECR-Quelle, die auf Amazon ECS-Instanzen bereitgestellt wird.

Amazon Elastic Container Service undCodeDeploy Blau-Grün-

Sie können eine PipelineAWS CodePipeline, die Containeranwendungen mithilfe einer Blau/Grün-Bereitstellung bereitstellt. In einer Blau/Grün-Bereitstellung können Sie neben der alten Version eine neue Version Ihrer Anwendung starten, und Sie können die neue Version testen, bevor Sie den Datenverkehr auf sie umleiten. Sie können auch den Bereitstellungsprozess überwachen und bei Problemen schnell einen Rollback durchführen.

Die abgeschlossene Pipeline erkennt Änderungen an Ihren Images oder Ihrer Aufgabendefinitionsdatei und verwendetCodeDeploy sie, um den Datenverkehr an einen Amazon ECS-Cluster und Load Balancer weiterzuleiten und bereitzustellen. CodeDeployerstellt einen neuen Listener auf Ihrem Load Balancer, der Ihre neue Aufgabe über einen speziellen Port ansprechen kann. Sie können die Pipeline auch so konfigurieren, dass sie einen Quellspeicherort verwendet, z. B. einCodeCommit Repository, in dem Ihre Amazon ECS-Aufgabendefinition gespeichert ist.

Bevor Sie Ihre Pipeline erstellen, müssen Sie bereits die Amazon ECS-Ressourcen, dieCodeDeploy Ressourcen sowie den Load Balancer und die Zielgruppe erstellt haben. Sie müssen das Bild bereits markiert und in Ihrem Bild-Repository gespeichert und die Aufgabendefinition und die Datei in IhrAppSpec Datei-Repository hochgeladen haben.

> **Note**
>
> In diesem Thema wird die Bereitstellungsaktion von Amazon ECS inCodeDeploy Blau/Grün beschriebenCodePipeline. Referenzinformationen zu den standardmäßigen Bereitstellungsaktionen von Amazon ECS finden Sie unterAmazon Elastic Container Service.CodePipeline

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `CodeDeployToECS`
- Version: `1`

Konfigurationsparameter

ApplicationName

Erforderlich: Ja

Der Name der Anwendung inCodeDeploy. Bevor Sie Ihre Pipeline erstellen, müssen Sie die Anwendung bereits in erstellt habenCodeDeploy.

DeploymentGroupName

Erforderlich: Ja

Die für Amazon ECS-Aufgabensätze angegebene Bereitstellungsgruppe, die Sie für IhreCodeDeploy Anwendung erstellt haben. Bevor Sie Ihre Pipeline erstellen, müssen Sie die Bereitstellungsgruppe bereits in erstellt habenCodeDeploy.

TaskDefinitionTemplateArtifact

Erforderlich: Ja

Der Name des Eingabeartefakts, das die Aufgabendefinitionsdatei für die Bereitstellungsaktion bereitstellt. Dies ist in der Regel der Name des Ausgabe-Artefakfakts der Quell-Aktion. Wenn Sie die Konsole verwenden, lautet der Standardname für das Ausgabeartefakt der Quellaktion`SourceArtifact`.

AppSpecTemplateArtifact

Erforderlich: Ja

Der Name des Eingabeartefakts, das dieAppSpec Datei für die Bereitstellungsaktion bereitstellt. Dieser Wert wird aktualisiert, wenn Ihre Pipeline ausgeführt wird. Dies ist in der Regel der Name des Ausgabe-Artefakfakts der Quell-Aktion. Wenn Sie die Konsole verwenden, lautet der

Standardname für das Ausgabeartefakt der Quellaktion`SourceArtifact`. Für`TaskDefinition` in derAppSpec Datei können Sie den`<TASK_DEFINITION>` Platzhaltertext wie hier gezeigt beibehalten.

AppSpecTemplatePath

Required: No

Der Dateiname der Datei, die amAppSpec Speicherort der Pipeline-Quelldatei gespeichert ist, z. B. imCodeCommit Repository Ihrer Pipeline. Der Standarddateiname ist `appspec.yaml`. Wenn IhreAppSpec 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

Required: No

Der Dateiname der Aufgabendefinition, die im Quellverzeichnis der Pipeline-Datei gespeichert ist, z. B. imCodeCommit Repository Ihrer Pipeline. Der Standarddateiname ist `taskdef.json`. Wenn Ihre Aufgabendefinitionsdatei denselben Namen hat und auf der Stammebene Ihres Datei-Repositorys 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

Required: No

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 aus der Amazon ECR-Quellaktion.

Verfügbare Werte für`<Number>` sind 1 bis 4.

Bild <Number>ContainerName

Required: No

Der Name des Images, das im Bild-Repository verfügbar ist, z. B. im Amazon ECR-Quell-Repository.

Verfügbare Werte für`<Number>` sind 1 bis 4.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1 to 5`
- Beschreibung: Die`CodeDeployToECS` Aktion sucht zuerst nach der Aufgabendefinitionsdatei und derAppSpec Datei im Quelldatei-Repository, dann nach dem Image im Image-Repository, generiert dann dynamisch eine neue Revision der Aufgabendefinition und führt schließlich dieAppSpec Befehle aus, um den Tasksatz und den Container im Cluster bereitzustellen.

  Die`CodeDeployToECS` Aktion sucht nach einer`imageDetail.json` Datei, die die Bild-URI dem Bild zuordnet. Wenn Sie eine Änderung in Ihr Amazon ECR-Bild-Repository übernehmen, erstellt die Pipeline-ECR-Quellaktion ein`imageDetail.json` Datei für diesen Commit. Sie können auch manuell ein`imageDetail.json` Datei für eine Pipeline hinzufügen, in der die Aktion nicht automatisiert ist. Informationen zur `imageDetail.json`-Datei finden Sie unter imageDetail.json-Datei für Amazon ECS-Blau/Grün-Bereitstellungsaktionen.

  Die`CodeDeployToECS` Aktion generiert dynamisch eine neue Revision der Aufgabendefinition. In dieser Phase ersetzt diese Aktion Platzhalter in der Aufgabendefinitionsdatei durch Bild-URI, die aus den ImageDetail.json-Dateien abgerufen wurde. Wenn Sie beispielsweise IMAGE1_NAME alsContainerName Image1-Parameter festlegen, sollten Sie den Platzhalter <IMAGE1_NAME>als Wert des Bildfeldes in Ihrer Aufgabendefinitionsdatei angeben. In diesem Fall ersetzt dieCodeDeployTo ECS-Aktion den Platzhalter durch die <IMAGE1_NAME>tatsächliche Bild-URI, die von imageDetail.json in dem Artefakt abgerufen wurde, das Sie als Image1 angebenArtifactName.

  Für Aktualisierungen der Aufgabendefinition enthält dieCodeDeployAppSpec`.yaml` Datei die`TaskDefinition` Eigenschaft.

```
TaskDefinition: <TASK_DEFINITION>
```

  Diese Eigenschaft wird durch die`CodeDeployToECS` Aktion aktualisiert, nachdem die neue Aufgabendefinition erstellt wurde.

  Für den Wert des`TaskDefinition` Feldes muss der Platzhaltertext sein<TASK_DEFINITION>. Die`CodeDeployToECS` Aktion ersetzt diesen Platzhalter durch den tatsächlichen ARN der dynamisch generierten Aufgabendefinition.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Aktionsdeklaration

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: CodeDeployToECS
      Version: '1'
    RunOrder: 1
    Configuration:
      AppSpecTemplateArtifact: SourceArtifact
      ApplicationName: ecs-cd-application
      DeploymentGroupName: ecs-deployment-group
      Image1ArtifactName: MyImage
      Image1ContainerName: IMAGE1_NAME
      TaskDefinitionTemplatePath: taskdef.json
      AppSpecTemplatePath: appspec.yaml
      TaskDefinitionTemplateArtifact: SourceArtifact
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
      - Name: MyImage
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
```

```
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "CodeDeployToECS",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "AppSpecTemplateArtifact": "SourceArtifact",
                "ApplicationName": "ecs-cd-application",
                "DeploymentGroupName": "ecs-deployment-group",
                "Image1ArtifactName": "MyImage",
                "Image1ContainerName": "IMAGE1_NAME",
                "TaskDefinitionTemplatePath": "taskdef.json",
                "AppSpecTemplatePath": "appspec.yaml",
                "TaskDefinitionTemplateArtifact": "SourceArtifact"
            },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                },
                {
                    "Name": "MyImage"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy — Dieses Tutorial führt Sie durch die Erstellung derCodeDeploy und Amazon ECS-Ressourcen, die Sie für eine blaue/grüne Bereitstellung benötigen. Das Tutorial zeigt Ihnen, wie Sie ein Docker-Image an Amazon ECR übertragen und eine Amazon ECS-Aufgabendefinition

erstellen, die Ihren Docker-Image-Namen, Containernamen, Amazon ECS-Servicenamen und Load Balancer-Konfiguration auflistet. Das Tutorial führt Sie dann durch die Erstellung derAppSpec Datei und der Pipeline für Ihr Deployment.

> **Note**
>
> Dieses Thema und die Anleitung beschreiben dieCodeDeploy /ECS blue/green Aktion fürCodePipeline. Informationen zu ECS-Standardaktionen inCodePipeline finden Sie unter Tutorial: Kontinuierliche Bereitstellung mitCodePipeline.

- AWS CodeDeployBenutzerhandbuch — Informationen zur Verwendung des Load Balancers, des Produktions-Listeners, der Zielgruppen und Ihrer Amazon ECS-Anwendung in einer blauen/grünen Bereitstellung finden Sie unter Tutorial: Bereitstellen eines Amazon ECS-Service. Diese Referenzinformationen im AWS CodeDeployBenutzerhandbuch bieten einen Überblick über blaue/grüne Bereitstellungen mit Amazon ECS undAWS CodeDeploy.
- Amazon Elastic Container Service Developer Guide — Informationen zur Arbeit mit Docker-Images und Containern, ECS-Diensten und -Clustern sowie ECS-Aufgabensätzen finden Sie unter Was ist Amazon ECS?

Amazon Elastic Container Service

Sie können eine Amazon ECS-Aktion verwenden, um einen Amazon ECS-Service und eine Aufgabengruppe bereitzustellen. Ein Amazon ECS-Service ist eine Containeranwendung, 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 CodePipeline verwendet, um das Image bereitzustellen.

> **Important**
>
> Die Amazon ECS-Standardbereitstellungsaktion für CodePipeline erstellt eine eigene Revision der Aufgabendefinition basierend auf der vom Amazon ECS-Service verwendeten Revision. 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 mit Tags versehen und in Ihrem Image-Repository gespeichert haben und die BuildSpec -Datei in Ihr -Repository.

> **Note**
>
> Dieses Referenzthema beschreibt die Amazon ECS-Standardbereitstellungsaktion für CodePipeline. Referenzinformationen zu Amazon ECS finden Sie unter CodeDeploy -Blau/Grün-Bereitstellungsaktionen in CodePipelinefinden Sie unterAmazon Elastic Container Service undCodeDeploy Blau-Grün-.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- 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 Abbild-Definitionsdatei, die JSON-Datei, die den Namen des Containers Ihres -Service beschreibt sowie Abbild und Tag. Sie verwenden diese Datei für ECS-Standardbereitstellungen. Weitere Informationen erhalten Sie unter Input artifacts (Eingabeartefakte) und imagedefinitions.json-Datei für Amazon ECS-SStandardbereitstellungsaktionen.

DeploymentTimeout

Erforderlich: Nein

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 nach einem`imagedefinitions.json`-Datei im Quelldatei-Repository für die Pipeline. Ein Abbild-Definitionsdokument ist eine JSON-Datei, die den Namen Ihres -Amazon ECS-Containers sowie Abbild und Tag beschreibt. CodePipeline verwendet die Datei, um das Image aus Ihrem Image-Repository wie Amazon ECR abzurufen. Sie können manuell eine hinzufügen`imagedefinitions.json`-Datei für eine Pipeline, in der die Aktion nicht automatisiert ist. Informationen zur `imagedefinitions.json`-Datei finden Sie unter imagedefinitions.json-Datei für Amazon ECS-SStandardbereitstellungsaktionen.

  Die Aktion erfordert ein vorhandenes Image, das bereits in Ihr Bild-Repository übertragen wurde. Weil das Image-Mapping von`imagedefinitions.json`-Datei erfordert die Aktion nicht, dass die Amazon ECR-Quelle als Quellaktion in die Pipeline aufgenommen wird.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung Ausgabe-Artefakte sind für diesen Aktionstyp nicht gültig.

Aktionsdeklaration

YAML

```
Name: DeployECS
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: ECS
  Version: '1'
RunOrder: 2
Configuration:
  ClusterName: my-ecs-cluster
  ServiceName: sample-app-service
  FileName: imagedefinitions.json
  DeploymentTimeout: '15'
OutputArtifacts: []
InputArtifacts:
  - Name: my-image
```

JSON

```
{
    "Name": "DeployECS",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "ECS",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "ClusterName": "my-ecs-cluster",
        "ServiceName": "sample-app-service",
        "FileName": "imagedefinitions.json",
        "DeploymentTimeout": "15"
```

```
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "my-image"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Kontinuierliche Bereitstellung mit CodePipeline— In diesem Tutorial erfahren Sie, wie Sie ein Dockerfile erstellen, das Sie in einem Quelldatei-Repository wie CodeCommit. Als nächstes erfahren Sie, wie Sie eine CodeBuild BuildSpec -Datei, die Ihr Docker-Image erstellt und an Amazon ECR überträgt und Ihre imagedefinitions.json-Datei erstellt. Schließlich erstellen Sie einen Amazon ECS-Service und eine Aufgabendefinition und dann erstellen Sie Ihre Pipeline mit einer Amazon ECS-Bereitstellungsaktion.

  > **Note**
  >
  > Dieses Thema und Tutorial beschreiben die Amazon ECS-Standardbereitstellungsaktion für CodePipeline. Weitere Informationen zu Amazon ECS finden Sie unter CodeDeploy - Blau/Grün-Bereitstellungsaktionen in CodePipelinefinden Sie unterTutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy .

- Amazon Elastic Container Service Entwicklerhandbuch— Weitere Informationen zum Arbeiten mit Docker-Images und Containern, Amazon ECS-Services und -Clustern sowie Amazon ECS-Tasksets finden Sie unterWas ist Amazon ECS?

Amazon S3 S3-Aktionen

Sie verwenden eine Amazon S3 S3-Bereitstellungsaktion, um Dateien in einem Amazon S3 S3-Bucket bereitzustellen, um statische Websites zu hosten oder zu archivieren. Sie können angeben, ob Bereitstellungsdateien vor dem Hochladen in Ihren Bucket extrahiert werden sollen.

> **Note**
>
> In diesem Referenzthema wird die Amazon S3 S3-Bereitstellungsaktion beschrieben, CodePipeline bei der die Bereitstellungsplattform ein für das Hosting konfigurierter Amazon S3 S3-Bucket ist. Referenzinformationen zur Amazon S3 S3-Quellaktion in CodePipeline finden Sie unterQuellaktion für Amazon S3.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- 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

Falls wahr, gibt dies an, dass Dateien vor dem Upload extrahiert werden sollen. Andernfalls bleiben die Anwendungsdateien für den Upload komprimiert, z. B. im Fall einer gehosteten statischen Website. Wenn falsch, `ObjectKey` ist der erforderlich.

ObjectKey

Bedingt. Erforderlich, wenn `Extract` = false

Der Name des Amazon-S3-Objektschlüssels, der das Objekt im S3-Bucket eindeutig identifiziert.

KM ARN EncryptionKey

Required: No

Der ARN des AWS KMS Verschlüsselungsschlüssels für den Host-Bucket. Der `KMSEncryptionKeyARN` Parameter verschlüsselt hochgeladene Artefakte mit den bereitgestellten. AWS KMS key Für einen -Schlüssel können Sie die Schlüssel-ID, den Schlüssel-ARN oder den Alias-ARN verwenden.

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

> **Important**
>
> CodePipelineunterstützt nur symmetrische KMS-Schlüssel. Verwenden Sie keinen asymmetrischen KMS-Schlüssel, um die Daten in Ihrem S3-Bucket zu verschlüsseln.

CannedACL

Required: No

Der `CannedACL` Parameter wendet die angegebene vordefinierte ACL auf Objekte an, die in Amazon S3 bereitgestellt werden. Dadurch werden alle auf das Objekt angewandten vorhandenen ACLs überschrieben.

CacheControl

Required: No

Der `CacheControl` Parameter steuert das Caching-Verhalten für Anforderungen/Antworten für Objekte im Bucket. Eine Liste der gültigen Werte finden Sie im `Cache-Control`-Header-Feld für HTTP-Operationen. Um mehrere Werte in `CacheControl` einzugeben, verwenden Sie ein Komma zwischen den einzelnen Werten. Sie können nach jedem Komma ein Leerzeichen hinzufügen (optional), wie in diesem Beispiel für die CLI gezeigt:

```
"CacheControl": "public, max-age=0, no-transform"
```

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `1`
- Beschreibung: Die Dateien für die Bereitstellung oder das Archiv werden aus dem Quell-Repository abgerufen, komprimiert und von CodePipeline hochgeladen.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten für diesen Aktionstyp nicht.

Beispielaktionskonfiguration

Im Folgenden werden Beispiele für die Aktionskonfiguration gezeigt.

Beispielkonfiguration, wenn eingestellt **`Extract`** ist auf **`false`**

Das folgende Beispiel zeigt die Standardaktionskonfiguration, wenn die Aktion erstellt wird und das `Extract` Feld auf gesetzt ist`false`.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
```

```
      Category: Deploy
      Owner: AWS
      Provider: S3
      Version: '1'
    RunOrder: 1
    Configuration:
      BucketName: website-bucket
      Extract: 'false'
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "S3",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BucketName": "website-bucket",
                "Extract": "false"
                },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

Beispielkonfiguration, wenn eingestellt **`Extract`** ist auf **`true`**

Das folgende Beispiel zeigt die Standardaktionskonfiguration, wenn die Aktion erstellt wird und das `Extract` Feld auf gesetzt ist`true`.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: S3
      Version: '1'
    RunOrder: 1
    Configuration:
      BucketName: website-bucket
      Extract: 'true'
      ObjectKey: MyWebsite
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "S3",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BucketName": "website-bucket",
                "Extract": "true",
```

```
                "ObjectKey": "MyWebsite"
                },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen Sie eine Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet— Dieses Tutorial führt Sie durch zwei Beispiele für die Erstellung einer Pipeline mit einer S3-Bereitstellungsaktion. Sie laden Beispieldateien herunter, laden die Dateien in Ihr CodeCommit Repository hoch, erstellen Ihren S3-Bucket und konfigurieren Ihren Bucket für das Hosting. Als Nächstes verwenden Sie die CodePipeline Konsole, um Ihre Pipeline zu erstellen und eine Amazon S3 S3-Bereitstellungskonfiguration anzugeben.
- Quellaktion für Amazon S3— Diese Aktionsreferenz enthält Referenzinformationen und Beispiele für Amazon S3 S3-Quellaktionen inCodePipeline.

Quellaktion für Amazon S3

Löst die Pipeline aus, wenn ein neues Objekt zum konfigurierten Bucket und Objektschlüssel hochgeladen wird.

> **Note**
>
> In diesem Referenzthema wird die Amazon S3 S3-Quellaktion für CodePipeline Wenn der Quellspeicherort ein Amazon S3 S3-Bucket ist, der für die Versioning konfiguriert wurde.

Weitere Informationen zur Amazon S3 S3-Bereitstellungsaktion in CodePipeline finden Sie unterAmazon S3 S3-Aktionenaus.

Sie können einen Amazon S3 S3-Bucket erstellen, der als Quellspeicherort für Ihre Anwendungsdateien verwendet werden soll.

> **Note**
>
> Wenn Sie den Quell-Bucket erstellen, müssen Sie für den Bucket das Versioning aktivieren. Wenn Sie einen vorhandenen Amazon S3 S3-Bucket verwenden möchten, finden Sie unterVerwenden von Versioningum die Versioning für einen vorhandenen Bucket zu aktivieren.

Wenn Sie die Pipeline über die Konsole erstellen oder bearbeiten, CodePipeline erstellt ein CloudWatch Ereignisregel, mit der die Pipeline gestartet wird, wenn eine Änderung im S3-Quell-Bucket auftritt.

Sie müssen einen Amazon S3 S3-Quell-Bucket erstellen und die Quelldateien als einzelne ZIP-Datei hochladen, bevor Sie die Pipeline über eine Amazon S3 S3-Aktion verbinden können.

> **Note**
>
> Wenn Amazon S3 der Quellanbieter für Ihre Pipeline ist, können Sie die Quelldatei (en) in eine einzelne ZIP-Datei komprimieren und die ZIP-Datei in Ihren Quell-Bucket hochladen. Sie können auch eine einzelne Datei ungezippt hochladen, aber nachgelagerte Aktionen, die eine ZIP-Datei erwarten, schlagen dann fehl.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- 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 S3-Buckets, in dem Quelländerungen erkannt werden sollen.

S3ObjectKey

: Erforderlich Ja

Der Name des Amazon S3 S3-Objektschlüssels, in dem Quelländerungen erkannt werden sollen.

PollForSourceChanges

: Erforderlich Nein

`PollForSourceChanges`steuert, ob CodePipeline fragt den Quell-Bucket für Amazon S3 nach Quelländerungen ab. Wir empfehlen Ihnen, CloudWatch Stattdessen und CloudTrail, um Quelländerungen zu erkennen. Weitere Informationen zur Konfigurierung CloudWatch Ereignisse, sieheMigrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (CLI)oderMigrieren Sie Polling-Pipelines mit einer S3-Quelle und einem CloudTrail S3-Trail (Vorlage) AWS CloudFormationaus.

> **Important**
>
> Wenn Sie beabsichtigen zu konfigurieren CloudWatch Ereignisse, die Sie einstellen müssen`PollForSourceChanges`zu`false`um doppelte Pipeline-Ausführungen zu vermeiden.

Gültige Werte für diesen Parameter sind:

- `true`: Falls gesetzt, CodePipeline fragt den Quellspeicherort nach Quelländerungen ab.

> **Note**
>
> Wenn du weglässt`PollForSourceChanges`, CodePipeline fragt den Quellspeicherort standardmäßig nach Quelländerungen ab. Dieses Verhalten ist das gleiche, als ob `PollForSourceChanges` enthalten wäre und auf `true` festgelegt würde.

- `false`: Falls gesetzt, CodePipeline fragt den Quellspeicherort nicht nach Quelländerungen ab. Sie verwenden diese Einstellung, wenn Sie eine CloudWatch Ereignisregel zum Erkennen von Quelländerungen.

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 Ausgabe-Artefakte für die Amazon S3 S3-Aktion. Die Amazon-S3-Objektmetadaten (ETag und Versions-ID) werden in CodePipeline als Quellrevision für die ausgelöste Pipeline-Ausführung.

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 unterVariablenaus.

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.

VersionId

Die Versions-ID für die Version des Objekts, die sich auf die Quelländerung bezieht, von der die Pipeline ausgelöst wurde.

Aktionsdeklaration

YAML

```
Name: Source
Actions:
  - RunOrder: 1
    OutputArtifacts:
      - Name: SourceArtifact
    ActionTypeId:
      Provider: S3
      Owner: AWS
      Version: '1'
      Category: Source
    Region: us-west-2
    Name: Source
    Configuration:
      S3Bucket: my-bucket-oregon
      S3ObjectKey: my-application.zip
      PollForSourceChanges: 'false'
    InputArtifacts: []
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "RunOrder": 1,
            "OutputArtifacts": [
                {
```

```
                    "Name": "SourceArtifact"
                }
            ],
            "ActionTypeId": {
                "Provider": "S3",
                "Owner": "AWS",
                "Version": "1",
                "Category": "Source"
            },
            "Region": "us-west-2",
            "Name": "Source",
            "Configuration": {
                "S3Bucket": "my-bucket-oregon",
                "S3ObjectKey": "my-application.zip",
                "PollForSourceChanges": "false"
            },
            "InputArtifacts": []
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket)- Dieses Tutorial enthält eine Beispieldatei für App-Spezifikationen und eine Beispieldatei für CodeDeploy. Sie verwenden dieses Tutorial, um eine Pipeline mit einer Amazon S3 S3-Quelle zu erstellen, die auf Amazon EC2 EC2-Instances bereitgestellt wird.

AWSAppConfig

AWS AppConfig ist eine -Funktion vonAWS Systems Manageraus. AppConfig unterstützt kontrollierte Bereitstellungen für Anwendungen jeder Größe und umfasst integrierte Validierungen sowie Überwachungsfunktionen. Sie können AppConfig mit Anwendungen, die auf Amazon EC2 EC2-Instances gehostet werden,AWS Lambda, Containern, mobilen Anwendungen oder IoT-Geräten.

DieAppConfigDeploy-Aktion ist einAWS CodePipelineAktion, die Konfigurationen bereitstellt, die in Ihrem Pipeline-Quellspeicherort gespeichert sind, an einem bestimmten AppConfig

Anwendung,Umwelt, undAufbau-Profil. Es verwendet die Einstellungen, die in einem AppConfig Bereitstellungsstrategieaus.

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `AppConfig`
- Version: `1`

Konfigurationsparameter

Anwendung

: Erforderlich Ja

Die ID derAWS AppConfig Anwendung mit den Details für Ihre Konfiguration und Bereitstellung.

Umgebung

: Erforderlich Ja

Die ID derAWS AppConfig Umgebung, in der die Konfiguration bereitgestellt wird.

ConfigurationProfile

: Erforderlich Ja

Die ID derAWS AppConfig -Konfigurationsprofil, das bereitgestellt werden soll.

InputArtifactConfigurationPath

: Erforderlich Ja

Der Dateipfad der Konfigurationsdaten innerhalb des bereitzustellenden Eingabeartefakts.

DeploymentStrategy

: Erforderlich Nein

DieAWS AppConfig Bereitstellungsstrategie, die für die Bereitstellung verwendet werden soll.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung Das Eingabe-Artefakt für die Bereitstellungsaktion.

Ausgabeartefakte

Nicht zutreffend.

Beispielaktionskonfiguration

YAML

```
name: Deploy
actions:
  - name: Deploy
    actionTypeId:
      category: Deploy
      owner: AWS
      provider: AppConfig
      version: '1'
    runOrder: 1
    configuration:
      Application: 2s2qv57
      ConfigurationProfile: PvjrpU
      DeploymentStrategy: frqt7ir
      Environment: 9tm27yd
      InputArtifactConfigurationPath: /
    outputArtifacts: []
    inputArtifacts:
      - name: SourceArtifact
    region: us-west-2
    namespace: DeployVariables
```

JSON

```
{
    "name": "Deploy",
    "actions": [
        {
```

```
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "AppConfig",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "Application": "2s2qv57",
                "ConfigurationProfile": "PvjrpU",
                "DeploymentStrategy": "frqt7ir",
                "Environment": "9tm27yd",
                "InputArtifactConfigurationPath": "/"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-west-2",
            "namespace": "DeployVariables"
        }
    ]
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWSAppConfig— Für weitere Informationen zuAWS AppConfig Bereitstellungen sieheAWS Systems Manager-Benutzerhandbuchaus.
- Tutorial: Erstellen einer Pipeline mit AWS AppConfig als Bereitstellungsanbieter— In diesem Tutorial können Sie mit der Einrichtung einfacher Bereitstellungskonfigurationsdateien beginnen und AppConfig -Ressourcen und zeigt Ihnen, wie Sie die -Konsole verwenden, um eine Pipeline mit einemAWS AppConfig Deployment-Aktion.

AWS CloudFormation

Führt eine Operation für einen AWS CloudFormation-Stack aus. Bei einem Stack handelt es sich um eine Sammlung von AWS-Ressourcen, die Sie als einzelne Einheit verwalten können. Die Ressourcen in einem Stack werden durch die AWS CloudFormation-Vorlage des Stacks definiert. Ein Änderungssatz erstellt einen Vergleich, der angezeigt werden kann, ohne den ursprünglichen Stack zu ändern. Weitere Informationen zu den Arten von AWS CloudFormation-Aktionen, die für Stacks und Änderungssätze ausgeführt werden können, finden Sie unter dem `ActionMode`-Parameter.

CodePipeline Ruft die AWS CloudFormation `DescribeStackEvents` API auf, um eine Fehlermeldung für eine AWS CloudFormation Aktion zu erstellen, bei der ein Stack-Vorgang fehlgeschlagen ist. Wenn eine Aktions-IAM-Rolle berechtigt ist, auf diese API zuzugreifen, werden die Details zur ersten ausgefallenen Ressource in die CodePipeline Fehlermeldung aufgenommen. Andernfalls ignoriert die Rollenrichtlinie den Zugriff auf die API und zeigt stattdessen eine allgemeine Fehlermeldung an, CodePipeline wenn die Rollenrichtlinie nicht über die entsprechende Berechtigung verfügt. Dazu muss die `cloudformation:DescribeStackEvents` Berechtigung der Servicerolle oder anderen IAM-Rollen für die Pipeline hinzugefügt werden.

Wenn Sie nicht möchten, dass die Ressourcendetails in den Pipeline-Fehlermeldungen auftauchen, können Sie diese Berechtigung für die Aktions-IAM-Rolle widerrufen, indem Sie die Berechtigung entfernen. `cloudformation:DescribeStackEvents`

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`

- Anbieter: `CloudFormation`
- Version: `1`

Konfigurationsparameter

ActionMode

Erforderlich: Ja

`ActionMode` ist der Name der Aktion, die AWS CloudFormation auf einem Stack oder Änderungssatz ausführt. Die folgenden Aktionsmodi sind verfügbar:

- `CHANGE_SET_EXECUTE` führt einen Änderungssatz für den Ressourcen-Stack aus, der auf einer Reihe von angegebenen Ressourcenaktualisierungen basiert. Mit dieser Aktion beginnt AWS CloudFormation, den Stack zu ändern.
- `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 bereits vorhanden ist, wird er von AWS CloudFormation gelöscht und ein neuer erstellt.
- `CREATE_UPDATE` erstellt den Stack, wenn er nicht vorhanden ist. Wenn der Stack bereits vorhanden ist, wird er von AWS CloudFormation aktualisiert. Verwenden Sie diese Aktion zum Aktualisieren von bestehenden Stacks. Im Gegensatz `REPLACE_ON_FAILURE` dazu, wenn der Stack vorhanden ist 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. Ist der Stack vorhanden und befindet sich in einem fehlerhaften Zustand, löscht AWS CloudFormation den Stack und erstellt dann einen neuen. Befindet sich der Stack nicht in einem fehlerhaften Zustand, wird er von AWS CloudFormation aktualisiert.

  Der Stack befindet sich in einem fehlerhaften Zustand, wenn einer der folgenden Statustypen in AWS CloudFormation angezeigt wird:

  - `ROLLBACK_FAILED`
  - `CREATE_FAILED`
  - `DELETE_FAILED`
  - `UPDATE_ROLLBACK_FAILED`

Verwenden Sie diese Aktion, um fehlerhafte Stacks automatisch ohne Wiederherstellung oder Fehlerbehebung zu ersetzen.

> **Important**
>
> Wir empfehlen Ihnen, `REPLACE_ON_FAILURE` nur für Testzwecke zu verwenden, da Ihr Stack dadurch möglicherweise gelöscht wird.

StackName

Erforderlich: Ja

`StackName` ist der Name eines vorhandenen Stacks oder eines Stacks, den Sie erstellen möchten.

Funktionen

Required: Conditional

Die Verwendung von `Capabilities` bestätigt, dass die Vorlage möglicherweise über die Funktionen verfügt, einige Ressourcen selbst zu erstellen und zu aktualisieren, und dass diese Funktionen basierend auf den Ressourcentypen in der Vorlage bestimmt werden.

Diese Eigenschaft ist erforderlich, wenn Sie über IAM-Ressourcen in Ihrer Stack-Vorlage verfügen oder einen Stack direkt aus einer Vorlage mit Makros erstellen. Damit die AWS CloudFormation-Aktion erfolgreich auf diese Weise ausgeführt wird, müssen Sie explizit bestätigen, dass Sie dies mit einer der folgenden Funktionen tun möchten:

- `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 `Capabilities` dazu finden Sie UpdateStackin den Eigenschaften unter der AWS CloudFormationAPI-Referenz.

ChangeSetName

Required: Conditional

`ChangeSetName` ist der Name eines vorhandenen Änderungssatzes oder eines neuen Änderungssatzes, den Sie für den angegebenen Stack erstellen möchten.

Diese Eigenschaft ist für die folgenden Aktionsmodi erforderlich: CHANGE_SET_REPLACE und CHANGE_SET_EXECUTE. Bei allen anderen Aktionsmodi wird die Eigenschaft ignoriert.

RoleArn

Required: Conditional

`RoleArn` ist der ARN der IAM-Servicerolle, die AWS CloudFormation annimmt, wenn es mit Ressourcen im angegebenen Stack arbeitet. `RoleArn` wird nicht angewendet, wenn ein Änderungssatz ausgeführt wird. Wenn Sie den Änderungssatz nicht CodePipeline zur Erstellung verwenden, stellen Sie sicher, dass dem Änderungssatz oder Stack eine zugeordnete Rolle zugewiesen ist.

> **Note**
>
> Diese Rolle muss sich in demselben Konto befinden wie die Rolle für die Aktion, die gerade ausgeführt wird, wie in der Aktionsdeklaration konfiguriert`RoleArn`.

Die Eigenschaft ist für die folgenden Aktionsmodi erforderlich:

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- DELETE_ONLY
- CHANGE_SET_REPLACE

> **Note**
>
> AWS CloudFormationerhält 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` repräsentiert die AWS CloudFormation-Vorlagendatei. Sie schließen die Datei in ein Eingabeartefakt für diese Aktion ein. Der Dateiname folgt diesem Format:

`Artifactname``::`*`TemplateFileName`*

`Artifactname` ist der Name des Eingabeartefakts, wie er in CodePipeline 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 AWS CloudFormation Vorlagendatei, die den Vorlagentext enthält, hat eine Mindestlänge von 1 Byte und eine Höchstlänge von 1 MB. Für AWS CloudFormation Bereitstellungsaktionen in CodePipeline beträgt die maximale Größe des Eingabeartefakts immer 256 MB. Weitere Informationen finden Sie unter Kontingente in AWS CodePipeline und AWS CloudFormation-Limits.

OutputFileName

Erforderlich: Nein

Verwenden Sie diese Option, `OutputFileName` um einen Namen für die Ausgabedatei anzugeben, z. B.`CreateStackOutput.json`, der dem Pipeline-Ausgabeartefakt für diese Aktion CodePipeline hinzugefügt wird. Die JSON-Datei enthält den Inhalt des `Outputs`-Abschnitts aus dem AWS CloudFormation-Stack.

Wenn Sie keinen Namen angeben, wird CodePipeline keine Ausgabedatei oder kein Ausgabeartefakt 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 CloudFormationBenutzerhandbuch 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 CloudFormationArtifacts.

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 AWS CloudFormation-Aktion optional Artefakte für folgende Zwecke:
  - Zur Bereitstellung der auszuführenden Stack-Vorlagendatei. (Weitere Informationen finden Sie unter dem `TemplatePath`-Parameter.)
  - Zur Bereitstellung der zu verwendenden Vorlagenkonfigurationsdatei. (Weitere Informationen finden Sie unter dem `TemplateConfiguration`-Parameter.) Weitere Informationen zum Format der Vorlagenkonfigurationsdatei finden Sie unter AWS CloudFormationArtefakte.
  - Zum Bereitstellen des Artefakts für eine Lambda-Funktion, die als Teil des AWS CloudFormation-Stacks bereitgestellt werden soll.

Ausgabeartefakte

- Anzahl der Artefakte: `0 to 1`
- Beschreibung: Wenn der `OutputFileName`-Parameter angegeben ist, wird durch diese Aktion ein Ausgabeartefakt erzeugt, das eine JSON-Datei mit dem angegebenen Namen enthält. Die JSON-Datei enthält den Inhalt des Ausgabeabschnitts aus dem AWS CloudFormation-Stack.

  Weitere Informationen zum Ausgabeabschnitt, den Sie für Ihre AWS CloudFormation-Aktion erstellen können, finden Sie unter Ausgaben.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

Für AWS CloudFormation-Aktionen werden Variablen aus beliebigen Werten erzeugt, die im `Outputs`-Abschnitt einer Stack-Vorlage angegeben sind. Beachten Sie, dass die einzigen CloudFormation Aktionsmodi, die Ausgaben generieren, diejenigen sind, die zur Erstellung oder Aktualisierung eines Stacks führen, wie z. B. die Erstellung von Stacks, Stack-Aktualisierungen und die Ausführung von Änderungssätzen. Die entsprechenden Aktionsmodi, die Variablen generieren, sind:

- CHANGE_SET_EXECUTE
- CHANGE_SET_REPLACE
- CREATE_UPDATE
- REPLACE_ON_FAILURE

Weitere Informationen finden Sie unter Variablen. Ein Tutorial, das Ihnen zeigt, wie Sie eine Pipeline mit einer CloudFormation Bereitstellungsaktion in einer Pipeline erstellen, die CloudFormation Ausgabevariablen verwendet, finden Sie unterTutorial: Erstellen einer Pipeline, die Variablen aus AWS CloudFormation-Bereitstellungsaktionen verwendet.

Aktionsdeklaration

YAML

```
Name: ExecuteChangeSet
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormation
  Version: '1'
RunOrder: 2
Configuration:
  ActionMode: CHANGE_SET_EXECUTE
  Capabilities: CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND
  ChangeSetName: pipeline-changeset
  ParameterOverrides: '{"ProjectId": "my-project","CodeDeployRole":
 "CodeDeploy_Role_ARN"}'
  RoleArn: CloudFormation_Role_ARN
  StackName: my-project--lambda
  TemplateConfiguration: 'my-project--BuildArtifact::template-configuration.json'
  TemplatePath: 'my-project--BuildArtifact::template-export.yml'
OutputArtifacts: []
InputArtifacts:
  - Name: my-project-BuildArtifact
```

JSON

```
{
    "Name": "ExecuteChangeSet",
    "ActionTypeId": {
```

```
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormation",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "ActionMode": "CHANGE_SET_EXECUTE",
        "Capabilities": "CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND",
        "ChangeSetName": "pipeline-changeset",
        "ParameterOverrides": "{\"ProjectId\": \"my-project\",\"CodeDeployRole\":
 \"CodeDeploy_Role_ARN\"}",
        "RoleArn": "CloudFormation_Role_ARN",
        "StackName": "my-project--lambda",
        "TemplateConfiguration": "my-project--BuildArtifact::template-
configuration.json",
        "TemplatePath": "my-project--BuildArtifact::template-export.yml"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
             "Name": "my-project-BuildArtifact"
        }
    ]
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Referenz zu den Konfigurationseigenschaften — Dieses Referenzkapitel im AWS CloudFormationBenutzerhandbuch enthält weitere Beschreibungen und Beispiele für diese CodePipeline Parameter.
- AWS CloudFormationAPI-Referenz — Der CreateStackParameter in der AWS CloudFormationAPI-Referenz beschreibt Stack-Parameter für AWS CloudFormation Vorlagen.

AWS CloudFormation StackSets

CodePipeline bietet die Möglichkeit, AWS CloudFormation StackSets Operationen als Teil Ihres CI/CD-Prozesses durchzuführen. Sie verwenden ein Stack-Set, um mithilfe einer einzigen Vorlage Stapel in AWS Konten in verschiedenen AWS Regionen zu erstellen. AWS CloudFormation Alle in jedem Stack enthaltenen Ressourcen werden durch die AWS CloudFormation Vorlage des Stack-Sets definiert. Wenn Sie das Stack-Set erstellen, geben Sie die zu verwendende Vorlage sowie alle Parameter und Funktionen an, die für die Vorlage erforderlich sind.

> **Note**
>
> Für die Bereitstellung müssen Sie das Verwaltungskonto für AWS Organizations verwenden AWS CloudFormation StackSets. Sie können für diese Aktion kein delegiertes Administratorkonto verwenden.

Weitere Informationen zu Konzepten für AWS CloudFormation StackSets finden Sie unter StackSets Konzepte im AWS CloudFormationBenutzerhandbuch.

Sie integrieren Ihre Pipeline AWS CloudFormation StackSets mithilfe von zwei unterschiedlichen Aktionstypen, die Sie zusammen verwenden:

- Die `CloudFormationStackSet` Aktion erstellt oder aktualisiert ein Stack-Set oder Stack-Instances anhand der Vorlage, die im Quellverzeichnis der Pipeline gespeichert ist. Jedes Mal, wenn ein Stack-Set erstellt oder aktualisiert wird, initiiert es eine Bereitstellung dieser Änderungen für bestimmte Instances. In der Konsole können Sie den CloudFormation Stack Set-Aktionsanbieter auswählen, wenn Sie Ihre Pipeline erstellen oder bearbeiten.
- Die `CloudFormationStackInstances` Aktion stellt Änderungen an der `CloudFormationStackSet` Aktion auf bestimmte Instanzen bereit, erstellt neue Stack-Instances und definiert Parameterüberschreibungen für bestimmte Instanzen. In der Konsole können Sie den CloudFormation Stack Instances-Aktionsanbieter auswählen, wenn Sie eine bestehende Pipeline bearbeiten.

> **Note**
>
> Die `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und sind in den Regionen Asien-Pazifik (Hongkong), Europa (Zürich), Europa (Mailand), Afrika

(Kapstadt) und Naher Osten (Bahrain) nicht verfügbar. Weitere verfügbare Aktionen finden Sie unterProdukt- und Serviceintegrationen mit CodePipeline.

Sie können diese Aktionen verwenden, um sie für AWS Zielkonten oder Organisationseinheiten-IDs von AWS Zielorganisationen bereitzustellen.

> **Note**
>
> Um die Bereitstellung für Konten oder Organisationseinheiten-IDs von AWS Zielorganisationen durchzuführen und das vom Service verwaltete Berechtigungsmodell zu verwenden, müssen Sie den vertrauenswürdigen Zugriff zwischen AWS Organizations AWS CloudFormation StackSets und Organizations aktivieren. Weitere Informationen finden Sie unter Vertrauenswürdigen Zugriff mit AWS CloudFormation Stacksets aktivieren.

Themen

- Wie funktionieren Aktionen AWS CloudFormation StackSets
- Wie strukturiert man StackSets Aktionen in einer Pipeline
- CloudFormationStackSet-Aktion
- CloudFormationStackInstances-Aktion
- Berechtigungsmodelle für Stack-Set-Operationen
- Datentypen von Vorlagenparametern
- Weitere Informationen finden Sie auch unter

Wie funktionieren Aktionen AWS CloudFormation StackSets

Eine `CloudFormationStackSet` Aktion erstellt oder aktualisiert Ressourcen, je nachdem, ob die Aktion zum ersten Mal ausgeführt wird.

Die `CloudFormationStackSet` Aktion erstellt oder aktualisiert das Stack-Set und stellt diese Änderungen auf bestimmten Instanzen bereit.

> **Note**
>
> Wenn Sie diese Aktion verwenden, um ein Update durchzuführen, das das Hinzufügen von Stack-Instances beinhaltet, werden die neuen Instanzen zuerst bereitgestellt und das Update

wird zuletzt abgeschlossen. Die neuen Instanzen erhalten zuerst die alte Version, und dann wird das Update auf alle Instanzen angewendet.

- Erstellen: Wenn keine Instanzen angegeben sind und das Stack-Set nicht existiert, erstellt die CloudFormationStackSetAktion das Stack-Set, ohne dass Instanzen erstellt werden.
- Update: Wenn die CloudFormationStackSetAktion für ein Stack-Set ausgeführt wird, das bereits erstellt wurde, aktualisiert die Aktion das Stack-Set. Wenn keine Instanzen angegeben sind und das Stack-Set bereits existiert, werden alle Instanzen aktualisiert. Wenn diese Aktion verwendet wird, um bestimmte Instanzen zu aktualisieren, werden alle verbleibenden Instanzen in den Status VERALTET versetzt.

  Sie können die CloudFormationStackSetAktion verwenden, um das Stack-Set auf folgende Weise zu aktualisieren.

  - Aktualisieren Sie die Vorlage auf einigen oder allen Instanzen.
  - Aktualisieren Sie die Parameter für einige oder alle Instanzen.
  - Aktualisieren Sie die Ausführungsrolle für das Stack-Set (diese muss mit der Ausführungsrolle übereinstimmen, die in der Administratorrolle angegeben ist).
  - Ändern Sie das Berechtigungsmodell (nur wenn keine Instanzen erstellt wurden).
  - Aktivieren/Deaktivieren, `AutoDeployment` wenn das Stack-Set-Berechtigungsmodell aktiviert ist. `Service Managed`
  - Aktualisieren Sie die Administratorrolle.
  - Aktualisieren Sie die Beschreibung auf dem Stack-Set.
  - Fügen Sie dem Stack-Set-Update Bereitstellungsziele hinzu, um neue Stack-Instances zu erstellen.

Die `CloudFormationStackInstances` Aktion erstellt neue Stack-Instances oder aktualisiert veraltete Stack-Instances. Eine Instanz wird veraltet, wenn ein Stack-Set aktualisiert wird, aber nicht alle Instances darin aktualisiert werden.

- Erstellen: Wenn der Stack bereits existiert, aktualisiert die `CloudFormationStackInstances` Aktion nur Instanzen und erstellt keine Stack-Instanzen.
- Update: Wenn die Vorlage oder die Parameter nach Ausführung der `CloudFormationStackSet` Aktion nur in einigen Fällen aktualisiert wurden, wird der Rest markiert`OUTDATED`. `CloudFormationStackInstances`Aktualisiert in späteren Pipeline-Phasen die restlichen

Instanzen im Stack in Wellen, sodass alle Instanzen markiert sindCURRENT. Diese Aktion kann auch verwendet werden, um zusätzliche Instanzen hinzuzufügen oder Parameter für neue oder bestehende Instanzen zu überschreiben.

Im Rahmen eines Updates können mit den `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und neue Bereitstellungsziele angegeben werden, wodurch neue Stack-Instances erstellt werden.

Im Rahmen eines Updates löschen die `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und keine Stack-Sets, Instanzen oder Ressourcen. Wenn die Aktion einen Stack aktualisiert, aber nicht alle zu aktualisierenden Instanzen angibt, werden die Instanzen, die nicht für die Aktualisierung angegeben wurden, aus dem Update entfernt und auf den Status von gesetzt`OUTDATED`.

Während einer Bereitstellung können Stack-Instances auch den Status anzeigen, `OUTDATED` ob die Bereitstellung für Instances fehlgeschlagen ist.

Wie strukturiert man StackSets Aktionen in einer Pipeline

Als bewährte Methode sollten Sie Ihre Pipeline so aufbauen, dass das Stack-Set erstellt und zunächst für eine Teilmenge oder eine einzelne Instanz bereitgestellt wird. Nachdem Sie Ihre Bereitstellung getestet und das generierte Stack-Set angesehen haben, fügen Sie die `CloudFormationStackInstances` Aktion hinzu, sodass die verbleibenden Instanzen erstellt und aktualisiert werden.

Verwenden Sie die Konsole oder die CLI, um die empfohlene Pipeline-Struktur wie folgt zu erstellen:

1. Erstellen Sie eine Pipeline mit einer Quellaktion (erforderlich) und der `CloudFormationStackSet` Aktion als Bereitstellungsaktion. Führen Sie Ihre Pipeline aus.
2. Wenn Ihre Pipeline zum ersten Mal ausgeführt wird, erstellt die `CloudFormationStackSet` Aktion Ihr Stack-Set und mindestens eine erste Instanz. Überprüfen Sie die Erstellung des Stack-Sets und die Bereitstellung auf Ihrer ersten Instance. Bei der ersten Erstellung eines Stack-Sets für das Konto Account-A, bei dem `us-east-1` es sich um die angegebene Region handelt, wird die Stack-Instance beispielsweise mit dem Stack-Set erstellt:

Stack-Instanz	Region	Status
StackInstanceID-1	us-east-1	CURRENT

3. Bearbeiten Sie Ihre Pipeline, um sie `CloudFormationStackInstances` als zweite Bereitstellungsaktion hinzuzufügen, um Stack-Instances für die von Ihnen angegebenen Ziele zu erstellen/zu aktualisieren. Beispielsweise werden bei der Erstellung von Stack-Instanzen für ein Konto, `Account-A` in dem die `eu-central-1` Regionen `us-east-2` und die Regionen angegeben sind, die verbleibenden Stack-Instances erstellt und die ursprüngliche Instanz bleibt wie folgt aktualisiert:

Stack-Instanz	Region	Status
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	CURRENT
StackInstanceID-3	eu-central-1	CURRENT

4. Führen Sie Ihre Pipeline nach Bedarf aus, um Ihr Stack-Set zu aktualisieren und Stack-Instances zu aktualisieren oder zu erstellen.

Wenn Sie ein Stack-Update initiieren, bei dem Sie Bereitstellungsziele aus der Aktionskonfiguration entfernt haben, werden die Stack-Instances, die nicht für das Update vorgesehen waren, aus der Bereitstellung entfernt und erhalten den Status VERALTET. Bei einem Stack-Instance-Update für ein Konto, `Account-A` bei dem die `us-east-2` Region aus der Aktionskonfiguration entfernt wurde, werden beispielsweise die verbleibenden Stack-Instances erstellt und die entfernte Instanz wird wie folgt auf VERALTET gesetzt:

Stack-Instanz	Region	Status
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	VERALTET
StackInstanceID-3	eu-central-1	CURRENT

Weitere Informationen zu bewährten Methoden für die Bereitstellung von Stack-Sets finden Sie unter Bewährte Methoden für StackSets im AWS CloudFormationBenutzerhandbuch.

`CloudFormationStackSet`-Aktion

Diese Aktion erstellt oder aktualisiert ein Stack-Set anhand der Vorlage, die im Quellverzeichnis der Pipeline gespeichert ist.

Nachdem Sie ein Stack-Set definiert haben, können Sie Stacks in den in den Konfigurationsparametern angegebenen Zielkonten und Regionen erstellen, aktualisieren oder löschen. Beim Erstellen, Aktualisieren und Löschen von Stacks können Sie weitere Einstellungen angeben, z. B. die Reihenfolge der Regionen für auszuführende Operationen, den Prozentsatz der Fehlertoleranz, ab dem Stackoperationen beendet werden, und die Anzahl der Konten, in denen Operationen an Stacks gleichzeitig ausgeführt werden.

Ein Stack-Set ist eine regionale Ressource. Wenn Sie ein Stack-Set in einer AWS Region erstellen, können Sie von anderen Regionen aus nicht darauf zugreifen.

Wenn diese Aktion als Aktualisierungsaktion für das Stack-Set verwendet wird, sind Aktualisierungen des Stacks nur zulässig, wenn mindestens eine Stack-Instance bereitgestellt wird.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Beispiel für CloudFormationStackSeteine 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 maximal 128 Zeichen lang sein.

Beschreibung

Erforderlich: Nein

Eine Beschreibung des StackSets. Sie können dies verwenden, um den Zweck des Stack-Sets oder andere relevante Informationen zu beschreiben.

TemplatePath

Erforderlich: Ja

Der Speicherort der Vorlage, die die Ressourcen im Stack-Set definiert. Dies muss auf eine Vorlage mit einer maximalen Größe von 460.800 Byte verweisen.

Geben Sie den Pfad zum Quellartefaktnamen und zur Vorlagendatei im Format ein`"InputArtifactName::TemplateFileName"`, wie im folgenden Beispiel gezeigt.

```
SourceArtifact::template.txt
```

Parameter

Erforderlich: Nein

Eine Liste von Vorlagenparametern für Ihr Stack-Set, die während einer Bereitstellung aktualisiert werden.

Sie können Parameter als wörtliche Liste oder als Dateipfad angeben:

- Sie können Parameter im folgenden Kurzsyntaxformat eingeben:.
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`

Weitere Informationen zu diesen Datentypen finden Sie unter. Datentypen von Vorlagenparametern

Das folgende Beispiel zeigt einen Parameter, der `BucketName` mit dem Wert my- benannt ist`bucket`.

```
ParameterKey=BucketName,ParameterValue=my-bucket
```

Das folgende Beispiel zeigt einen Eintrag mit mehreren Parametern:

```
ParameterKey=BucketName,ParameterValue=my-bucket
ParameterKey=Asset1,ParameterValue=true
ParameterKey=Asset2,ParameterValue=true
```

- Sie können den Speicherort der Datei eingeben, die eine Liste der im Format eingegebenen Überschreibungen von Vorlagenparametern enthält`"InputArtifactName::ParametersFileName"`, wie im folgenden Beispiel gezeigt.

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

Das folgende Beispiel zeigt den Dateiinhalt für`parameters.txt`.

```
[
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    },
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    }
]
```

Funktionen

Erforderlich: Nein

Zeigt an, dass die Vorlage je nach Ressourcentyp in der Vorlage Ressourcen erstellen und aktualisieren kann.

Sie müssen diese Eigenschaft verwenden, wenn Ihre Stack-Vorlage IAM-Ressourcen enthält oder wenn Sie einen Stack direkt aus einer Vorlage erstellen, die Makros enthält. Damit die AWS CloudFormation Aktion auf diese Weise erfolgreich ausgeführt werden kann, müssen Sie eine der folgenden Funktionen verwenden:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`

Sie können mehr als eine Fähigkeit angeben, indem Sie ein Komma und keine Leerzeichen zwischen den Funktionen verwenden. Das Beispiel in Beispiel für CloudFormationStackSeteine Aktionskonfiguration zeigt einen Eintrag mit mehreren Funktionen.

PermissionModel

Erforderlich: Nein

Legt fest, wie IAM-Rollen erstellt und verwaltet werden. Wenn das Feld nicht angegeben ist, wird der Standard verwendet. Weitere Informationen finden Sie unter Berechtigungsmodelle für Stack-Set-Operationen.

Gültige Werte für sind:

- `SELF_MANAGED`(Standard): Sie müssen Administrator- und Ausführungsrollen für die Bereitstellung auf Zielkonten erstellen.
- `SERVICE_MANAGED`: erstellt AWS CloudFormation StackSets automatisch die IAM-Rollen, die für die Bereitstellung auf Konten erforderlich sind, die von AWS Organizations verwaltet werden. Dazu muss ein Konto Mitglied einer Organisation sein.

> **Note**
>
> Dieser Parameter kann nur geändert werden, wenn keine Stack-Instances im Stack-Set vorhanden sind.

AdministrationRoleArn

> **Note**
>
> Da AWS CloudFormation StackSets Operationen über mehrere Konten hinweg ausgeführt werden, müssen Sie die erforderlichen Berechtigungen für diese Konten definieren, bevor Sie das Stack-Set erstellen können.

Erforderlich: Nein

> **Note**
>
> Dieser Parameter ist optional für das SELF_MANAGED-Berechtigungsmodell und wird nicht für das SERVICE_MANAGED-Berechtigungsmodell verwendet.

Der ARN der IAM-Rolle im Administratorkonto, das zur Ausführung von Stack-Set-Vorgängen verwendet wird.

Der Name kann alphanumerische Zeichen und eines der folgenden Zeichen enthalten: _ +=, .@- und keine Leerzeichen. Beim Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden. Dieser Rollenname muss eine Mindestlänge von 20 Zeichen und eine maximale Länge von 2048 Zeichen haben. Rollennamen müssen innerhalb des Kontos eindeutig sein. Der hier angegebene Rollenname muss ein vorhandener Rollenname sein. Wenn Sie den Rollennamen nicht angeben, ist er auf gesetzt AWSCloudFormationStackSetAdministrationRole. Wenn Sie angeben ServiceManaged, dürfen Sie keinen Rollennamen definieren.

ExecutionRoleName

> **Note**
>
> Da AWS CloudFormation StackSets Operationen über mehrere Konten hinweg ausgeführt werden, müssen Sie die erforderlichen Berechtigungen für diese Konten definieren, bevor Sie das Stack-Set erstellen können.

Erforderlich: Nein

> **Note**
>
> Dieser Parameter ist optional für das SELF_MANAGED-Berechtigungsmodell und wird nicht für das SERVICE_MANAGED-Berechtigungsmodell verwendet.

Der Name der IAM-Rolle in den Zielkonten, die zur Ausführung von Stack-Set-Vorgängen verwendet wird. Der Name kann alphanumerische Zeichen und eines der folgenden Zeichen enthalten: _+=, .@- und keine Leerzeichen. Beim Namen wird nicht zwischen

Groß- und Kleinschreibung unterschieden. Dieser Rollenname muss eine Mindestlänge von 1 Zeichen und eine maximale Länge von 64 Zeichen haben. Rollennamen müssen innerhalb des Kontos eindeutig sein. Der hier angegebene Rollenname muss ein vorhandener Rollenname sein. Geben Sie diese Rolle nicht an, wenn Sie benutzerdefinierte Ausführungsrollen verwenden. Wenn Sie den Rollennamen nicht angeben, ist er auf gesetzt`AWSCloudFormationStackSetExecutionRole`. Wenn Sie Service_Managed auf true setzen, dürfen Sie keinen Rollennamen definieren.

OrganizationsAutoDeployment

Erforderlich: Nein

> **Note**
>
> Dieser Parameter ist optional für das SERVICE_MANAGED-Berechtigungsmodell und wird nicht für das SELF_MANAGED-Berechtigungsmodell verwendet.

Beschreibt, ob Konten, die einer Zielorganisation oder Organisationseinheit (OU) hinzugefügt werden, AWS CloudFormation StackSets automatisch für AWS Organizations bereitgestellt werden. Wenn `OrganizationsAutoDeployment` angegeben, geben Sie `DeploymentTargets` und `Regions` nicht an.

> **Note**
>
> Wenn keine Eingabe angegeben ist`OrganizationsAutoDeployment`, ist der Standardwert`Disabled`.

Gültige Werte für sind:

- `Enabled`. Erforderlich: Nein.

  StackSets stellt automatisch zusätzliche Stack-Instances für AWS Unternehmenskonten bereit, die einer Zielorganisation oder Organisationseinheit (OU) in den angegebenen Regionen hinzugefügt werden. Wenn ein Konto aus einer Zielorganisation oder Organisationseinheit entfernt wird, werden Stack-Instances aus dem Konto in den angegebenen Regionen AWS CloudFormation StackSets gelöscht.
- `Disabled`. Erforderlich: Nein.

StackSets stellt nicht automatisch zusätzliche Stack-Instances für AWS Organisationskonten bereit, die einer Zielorganisation oder Organisationseinheit (OU) in den angegebenen Regionen hinzugefügt werden.

- `EnabledWithStackRetention`. Erforderlich: Nein.

  Stack-Ressourcen bleiben erhalten, wenn ein Konto aus einer Zielorganisation oder Organisationseinheit entfernt wird.

DeploymentTargets

Erforderlich: Nein

> **Note**
>
> Für das Berechtigungsmodell SERVICE_MANAGED können Sie entweder die Stamm-ID der Organisation oder die IDs der Organisationseinheiten für Bereitstellungsziele angeben. Für das SELF_MANAGED-Berechtigungsmodell können Sie nur Konten angeben.

> **Note**
>
> Wenn dieser Parameter ausgewählt ist, müssen Sie auch Regionen auswählen.

Eine Liste von AWS Konten oder Organisationseinheiten-IDs, für die Stackset-Instanzen erstellt/aktualisiert werden sollen.

- Konten:

  Sie können Konten als wörtliche Liste oder als Dateipfad angeben:

  - Literal: Geben Sie Parameter im Kurzsyntaxformat ein`account_ID,account_ID`, wie im folgenden Beispiel gezeigt.

    ```
    111111222222,333333444444
    ```

  - Dateipfad: Der Speicherort der Datei, die eine Liste von AWS Konten enthält, für die Stackset-Instanzen erstellt/aktualisiert werden sollen, eingegeben im Format. `InputArtifactName::AccountsFileName` Wenn Sie den Dateipfad verwenden, um

entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat in JSON sein, wie im folgenden Beispiel gezeigt.

```
SourceArtifact::accounts.txt
```

Das folgende Beispiel zeigt den Dateiinhalt für`accounts.txt`.

```
[
    "111111222222"
]
```

Das folgende Beispiel zeigt den Dateiinhalt für den Fall`accounts.txt`, dass mehr als ein Konto aufgeführt wird:

```
[
    "111111222222","333333444444"
]
```

- OrganizationalUnitIds:

> **Note**
>
> Dieser Parameter ist optional für das SERVICE_MANAGED-Berechtigungsmodell und wird nicht für das SELF_MANAGED-Berechtigungsmodell verwendet. Verwenden Sie diesen Wert nicht, wenn Sie auswählen. OrganizationsAutoDeployment

Die AWS Organisationseinheiten, in denen die zugehörigen Stack-Instances aktualisiert werden sollen.

Sie können die IDs der Organisationseinheiten als wörtliche Liste oder als Dateipfad angeben:

- Literal: Geben Sie eine Reihe von durch Kommas getrennten Zeichenfolgen ein, wie im folgenden Beispiel gezeigt.

```
ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
```

- Dateipfad: Der Speicherort der Datei, die eine Liste enthält, OrganizationalUnitIds in der Stackset-Instanzen erstellt oder aktualisiert werden sollen. Wenn Sie den Dateipfad

verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat in JSON sein, wie im folgenden Beispiel gezeigt.

Geben Sie einen Pfad zur Datei im Format ein`InputArtifactName::OrganizationalUnitIdsFileName`.

```
SourceArtifact::OU-IDs.txt
```

Das folgende Beispiel zeigt den Dateiinhalt für`OU-IDs.txt`:

```
[
    "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
]
```

Regionen

Erforderlich: Nein

> **Note**
>
> Wenn dieser Parameter ausgewählt ist, müssen Sie auch auswählen DeploymentTargets.

Eine Liste von AWS Regionen, in denen Stackset-Instanzen erstellt oder aktualisiert werden. Regionen werden in der Reihenfolge aktualisiert, in der sie eingegeben wurden.

Geben Sie eine Liste gültiger AWS Regionen in dem Format ein`Region1,Region2`, wie im folgenden Beispiel gezeigt.

```
us-west-2,us-east-1
```

FailureTolerancePercentage

Erforderlich: Nein

Der Prozentsatz der Konten pro Region, bei denen dieser Stack-Vorgang fehlschlagen kann, bevor der Vorgang in dieser Region AWS CloudFormation beendet wird. Wenn der Vorgang in einer Region gestoppt wird, wird der Vorgang in nachfolgenden Regionen AWS CloudFormation nicht versucht. Bei der Berechnung der Anzahl der Konten auf der Grundlage des angegebenen Prozentsatzes wird von AWS CloudFormation auf die nächste ganze Zahl 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 von AWS CloudFormation auf die nächste ganze Zahl abgerundet. Wenn das Abrunden zu Null führen würde, wird AWS CloudFormation die Zahl stattdessen auf Eins gesetzt. Sie verwenden diese Einstellung zwar, um das Maximum anzugeben, aber bei großen Bereitstellungen kann die tatsächliche Anzahl der Konten, auf die gleichzeitig zugegriffen wird, aufgrund der Dienstdrosselung geringer sein.

RegionConcurrencyType

Erforderlich: Nein

Sie können angeben, ob das Stack-Set AWS-Regionen sequentiell oder parallel bereitgestellt werden soll, indem Sie den Bereitstellungsparameter Region Concurrency konfigurieren. Wenn die Regions-Parallelität so angegeben ist, dass Stacks über mehrere gleichzeitig AWS-Regionen bereitgestellt werden, kann dies zu kürzeren Gesamtbereitstellungszeiten führen.

- Parallel: Stack-Set-Bereitstellungen werden zur gleichen Zeit durchgeführt, sofern die Bereitstellungsfehler einer Region eine festgelegte Fehlertoleranz nicht überschreiten.
- Sequentiell: Stack-Set-Bereitstellungen werden nacheinander durchgeführt, sofern die Bereitstellungsfehler einer Region eine festgelegte Fehlertoleranz nicht überschreiten. Die sequenzielle Bereitstellung ist die Standardauswahl.

ConcurrencyMode

Erforderlich: Nein

Im Parallelitätsmodus können Sie wählen, wie sich die Parallelitätsebene bei Stack-Set-Vorgängen verhält, unabhängig davon, ob es sich um eine strikte oder eine weiche Fehlertoleranz handelt. Strikte Fehlertoleranz senkt die Bereitstellungsgeschwindigkeit bei fehlerhaften Stack-Set-Vorgängen, da die Parallelität bei jedem Fehler abnimmt. Lose Fehlertoleranz priorisiert die Bereitstellungsgeschwindigkeit und nutzt dennoch die Sicherheitsfunktionen von AWS CloudFormation.

- `STRICT_FAILURE_TOLERANCE`: Diese Option senkt die Parallelitätsstufe dynamisch, um sicherzustellen, dass die Anzahl ausgefallener Konten niemals eine bestimmte Fehlertoleranz überschreitet. Dies ist das Standardverhalten.

- `SOFT_FAILURE_TOLERANCE`: Diese Option entkoppelt die Fehlertoleranz von der tatsächlichen Parallelität. Dadurch können Stack-Set-Operationen unabhängig von der Anzahl der Fehler auf einer bestimmten Parallelitätsebene ausgeführt werden.

Input artifacts (Eingabeartefakte)

Sie müssen mindestens ein Eingabeartefakt, das die Vorlage für das Stack-Set enthält, in eine `CloudFormationStackSet` Aktion einbeziehen. Sie können weitere Eingabeartefakte für Listen mit Bereitstellungszielen, Konten und Parametern hinzufügen.

- Anzahl der Artefakte: `1 to 3`
- Beschreibung: Sie können Artefakte hinzufügen, um Folgendes bereitzustellen:
  - Die Stack-Vorlagendatei. (Weitere Informationen finden Sie unter dem `TemplatePath`-Parameter.)
  - Die Parameterdatei. (Weitere Informationen finden Sie unter dem `Parameters`-Parameter.)
  - Die Kontendatei. (Weitere Informationen finden Sie unter dem `DeploymentTargets`-Parameter.)

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Ausgabevariablen

Wenn Sie diese Aktion konfigurieren, erzeugt sie Variablen, auf die in der Aktionskonfiguration einer Downstream-Aktion in der Pipeline verwiesen werden kann. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

- StackSetId: Die ID des Stack-Sets.
- OperationId: Die ID der Stack-Set-Operation.

Weitere Informationen finden Sie unter Variablen.

Beispiel für CloudFormationStackSeteine Aktionskonfiguration

Die folgenden Beispiele zeigen die Aktionskonfiguration für die CloudFormationStackSetAktion.

Beispiel für das selbstverwaltete Berechtigungsmodell

Das folgende Beispiel zeigt eine CloudFormationStackSetAktion, bei der das eingegebene Bereitstellungsziel eine AWS Konto-ID ist.

YAML

```
Name: CreateStackSet
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackSet
  Version: '1'
RunOrder: 1
Configuration:
  DeploymentTargets: '111111222222'
  FailureTolerancePercentage: '20'
  MaxConcurrentPercentage: '25'
  PermissionModel: SELF_MANAGED
  Regions: us-east-1
  StackSetName: my-stackset
  TemplatePath: 'SourceArtifact::template.json'
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: us-west-2
Namespace: DeployVariables
```

JSON

```
{
    "Name": "CreateStackSet",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackSet",
        "Version": "1"
    },
    "RunOrder": 1,
```

```
    "Configuration": {
        "DeploymentTargets": "111111222222",
        "FailureTolerancePercentage": "20",
        "MaxConcurrentPercentage": "25",
        "PermissionModel": "SELF_MANAGED",
        "Regions": "us-east-1",
        "StackSetName": "my-stackset",
        "TemplatePath": "SourceArtifact::template.json"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "us-west-2",
    "Namespace": "DeployVariables"
}
```

Beispiel für das Modell der vom Service verwalteten Berechtigungen

Das folgende Beispiel zeigt eine CloudFormationStackSetAktion für das vom Service verwaltete Berechtigungsmodell, bei dem die Option für die auto Bereitstellung für AWS Organizations mit Stack-Aufbewahrung aktiviert ist.

YAML

```
Name: Deploy
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackSet
  Version: '1'
RunOrder: 1
Configuration:
  Capabilities: 'CAPABILITY_IAM,CAPABILITY_NAMED_IAM'
  OrganizationsAutoDeployment: EnabledWithStackRetention
  PermissionModel: SERVICE_MANAGED
  StackSetName: stacks-orgs
  TemplatePath: 'SourceArtifact::template.json'
OutputArtifacts: []
InputArtifacts:
```

```
  - Name: SourceArtifact
Region: eu-central-1
Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackSet",
        "Version": "1"
    },
    "RunOrder": 1,
    "Configuration": {
        "Capabilities": "CAPABILITY_IAM,CAPABILITY_NAMED_IAM",
        "OrganizationsAutoDeployment": "EnabledWithStackRetention",
        "PermissionModel": "SERVICE_MANAGED",
        "StackSetName": "stacks-orgs",
        "TemplatePath": "SourceArtifact::template.json"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "eu-central-1",
    "Namespace": "DeployVariables"
}
```

CloudFormationStackInstances-Aktion

Diese Aktion erstellt neue Instanzen und stellt Stack-Sets für bestimmte Instanzen bereit. Eine Stack-Instance ist eine Referenz auf einen Stack in einem Zielkonto innerhalb einer Region. Eine Stack-Instance kann auch ohne Stack existieren. Wenn die Stack-Erstellung beispielsweise nicht erfolgreich ist, zeigt die Stack-Instance den Grund für das Fehlschlagen der Stack-Erstellung an. Ein Stack-Instance ist nur einem Stack-Set zugeordnet.

Nach der ersten Erstellung eines Stack-Sets können Sie neue Stack-Instances hinzufügen, indem `CloudFormationStackInstances` Sie Vorlagenparameterwerte können auf Stack-Instance-Ebene bei der Erstellung oder Aktualisierung von Stack-Set-Instance-Vorgängen überschrieben werden.

Jedes Stack-Set hat eine Vorlage und einen Satz von Vorlagenparametern. Wenn Sie die Vorlage oder die Vorlagenparameter aktualisieren, aktualisieren Sie sie für den gesamten Satz. Dann werden alle Instanzstatus auf gesetzt, `OUTDATED` bis die Änderungen für diese Instanz bereitgestellt werden.

Um Parameterwerte für bestimmte Instanzen zu überschreiben, z. B. wenn die Vorlage einen Parameter für `stage` mit dem Wert von enthält`prod`, können Sie den Wert dieses Parameters auf `beta` oder `gamma` überschreiben.

Themen

- Aktionstyp
- 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 maximal 128 Zeichen lang sein.

DeploymentTargets

Erforderlich: Nein

> **Note**
>
> Für das SERVICE_MANAGED-Berechtigungsmodell können Sie entweder die Stamm-ID der Organisation oder die IDs der Organisationseinheiten für Bereitstellungsziele angeben. Für das SELF_MANAGED-Berechtigungsmodell können Sie nur Konten angeben.

> **Note**
>
> Wenn dieser Parameter ausgewählt ist, müssen Sie auch Regionen auswählen.

Eine Liste von AWS Konten oder Organisationseinheiten-IDs, für die Stackset-Instanzen erstellt/aktualisiert werden sollen.

- Konten:

  Sie können Konten als wörtliche Liste oder als Dateipfad angeben:

  - Literal: Geben Sie Parameter im Kurzsyntaxformat ein`account_ID,account_ID`, wie im folgenden Beispiel gezeigt.

    ```
    111111222222,333333444444
    ```

  - Dateipfad: Der Speicherort der Datei, die eine Liste von AWS Konten enthält, für die Stackset-Instanzen erstellt/aktualisiert werden sollen, eingegeben im Format. `InputArtifactName::AccountsFileName` Wenn Sie den Dateipfad verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat in JSON sein, wie im folgenden Beispiel gezeigt.

    ```
    SourceArtifact::accounts.txt
    ```

    Das folgende Beispiel zeigt den Dateiinhalt für`accounts.txt`:

```
[
    "111111222222"
]
```

Das folgende Beispiel zeigt den Dateiinhalt für den Fall`accounts.txt`, dass mehr als ein Konto aufgeführt wird:

```
[
    "111111222222","333333444444"
]
```

- OrganizationalUnitIds:

> **Note**
>
> Dieser Parameter ist optional für das SERVICE_MANAGED-Berechtigungsmodell und wird nicht für das SELF_MANAGED-Berechtigungsmodell verwendet. Verwenden Sie diesen Wert nicht, wenn Sie auswählen. OrganizationsAutoDeployment

Die AWS Organisationseinheiten, in denen die zugehörigen Stack-Instances aktualisiert werden sollen.

Sie können die IDs der Organisationseinheiten als wörtliche Liste oder als Dateipfad angeben.

- Literal: Geben Sie eine Reihe von durch Kommas getrennten Zeichenfolgen ein, wie im folgenden Beispiel gezeigt.

```
ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
```

- Dateipfad: Der Speicherort der Datei, die eine Liste enthält, OrganizationalUnitIds in der Stackset-Instanzen erstellt oder aktualisiert werden sollen. Wenn Sie den Dateipfad verwenden, um entweder Konten oder anzugeben OrganizationalUnitIds, muss das Dateiformat in JSON sein, wie im folgenden Beispiel gezeigt.

  Geben Sie einen Pfad zur Datei im Format einInputArtifactName::OrganizationalUnitIdsFileName.

```
SourceArtifact::OU-IDs.txt
```

Das folgende Beispiel zeigt den Dateiinhalt 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 Stackset-Instanzen erstellt oder aktualisiert werden. Regionen werden in der Reihenfolge aktualisiert, in der sie eingegeben wurden.

Geben Sie eine Liste gültiger AWS Regionen im Format: ein`Region1,Region2`, wie im folgenden Beispiel gezeigt.

```
us-west-2,us-east-1
```

ParameterOverrides

Erforderlich: Nein

Eine Liste von Stack-Set-Parametern, die Sie in den ausgewählten Stack-Instances überschreiben möchten. Überschriebene Parameterwerte werden auf alle Stack-Instances in den angegebenen Konten und Regionen angewendet.

Sie können Parameter als Literalliste oder als Dateipfad angeben:

- Sie können Parameter im folgenden Kurzsyntaxformat eingeben:. `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  Weitere Informationen zu diesen Datentypen finden Sie unter. Datentypen von Vorlagenparametern

  Das folgende Beispiel zeigt einen Parameter, der `BucketName` mit dem Wert my- benannt ist`bucket`.

```
ParameterKey=BucketName,ParameterValue=my-bucket
```

Das folgende Beispiel zeigt einen Eintrag mit mehreren Parametern.

```
ParameterKey=BucketName,ParameterValue=my-bucket
ParameterKey=Asset1,ParameterValue=true
ParameterKey=Asset2,ParameterValue=true
```

- Sie können den Speicherort der Datei eingeben, die eine Liste der im Format eingegebenen Überschreibungen von Vorlagenparametern enthält`InputArtifactName::ParameterOverridessFileName`, wie im folgenden Beispiel gezeigt.

  ```
  SourceArtifact::parameter-overrides.txt
  ```

  Das folgende Beispiel zeigt den Dateiinhalt für`parameter-overrides.txt`.

  ```
  [
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      },
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      }
  ]
  ```

FailureTolerancePercentage

Erforderlich: Nein

Der Prozentsatz der Konten pro Region, für die dieser Stack-Vorgang fehlschlagen kann, bevor der Vorgang in dieser Region AWS CloudFormation beendet wird. Wenn der Vorgang in einer Region gestoppt wird, wird der Vorgang in nachfolgenden Regionen AWS CloudFormation nicht versucht. Bei der Berechnung der Anzahl der Konten auf der Grundlage des angegebenen Prozentsatzes wird von AWS CloudFormation auf die nächste ganze Zahl abgerundet.

MaxConcurrentPercentage

Erforderlich: Nein

Der maximale Prozentsatz der Konten, für die dieser Vorgang gleichzeitig ausgeführt werden soll. Bei der Berechnung der Anzahl der Konten auf der Grundlage des angegebenen Prozentsatzes wird von AWS CloudFormation auf die nächste ganze Zahl abgerundet. Wenn das Abrunden zu Null führen würde, wird AWS CloudFormation die Zahl stattdessen auf Eins gesetzt. Sie geben zwar das Maximum an, aber bei großen Bereitstellungen kann die tatsächliche Anzahl der Konten, auf die gleichzeitig zugegriffen wird, aufgrund der Dienstdrosselung geringer sein.

RegionConcurrencyType

Erforderlich: Nein

Sie können angeben, ob das Stack-Set AWS-Regionen sequentiell oder parallel bereitgestellt werden soll, indem Sie den Bereitstellungsparameter Region Concurrency konfigurieren. Wenn die Regions-Parallelität so angegeben ist, dass Stacks über mehrere gleichzeitig AWS-Regionen bereitgestellt werden, kann dies zu kürzeren Gesamtbereitstellungszeiten führen.

- Parallel: Stack-Set-Bereitstellungen werden zur gleichen Zeit durchgeführt, sofern die Bereitstellungsfehler einer Region eine festgelegte Fehlertoleranz nicht überschreiten.
- Sequentiell: Stack-Set-Bereitstellungen werden nacheinander durchgeführt, sofern die Bereitstellungsfehler einer Region eine festgelegte Fehlertoleranz nicht überschreiten. Die sequenzielle Bereitstellung ist die Standardauswahl.

ConcurrencyMode

Erforderlich: Nein

Im Parallelitätsmodus können Sie wählen, wie sich die Parallelitätsebene bei Stack-Set-Vorgängen verhält, unabhängig davon, ob es sich um eine strikte oder eine weiche Fehlertoleranz handelt. Strikte Fehlertoleranz senkt die Bereitstellungsgeschwindigkeit bei fehlerhaften Stack-Set-Vorgängen, da die Parallelität bei jedem Fehler abnimmt. Lose Fehlertoleranz priorisiert die Bereitstellungsgeschwindigkeit und nutzt dennoch die Sicherheitsfunktionen von AWS CloudFormation.

- `STRICT_FAILURE_TOLERANCE`: Diese Option senkt die Parallelitätsstufe dynamisch, um sicherzustellen, dass die Anzahl ausgefallener Konten niemals eine bestimmte Fehlertoleranz überschreitet. Dies ist das Standardverhalten.

- `SOFT_FAILURE_TOLERANCE`: Diese Option entkoppelt die Fehlertoleranz von der tatsächlichen Parallelität. Dadurch können Stack-Set-Operationen unabhängig von der Anzahl der Fehler auf einer bestimmten Parallelitätsebene ausgeführt werden.

Input artifacts (Eingabeartefakte)

`CloudFormationStackInstances`kann Artefakte enthalten, die Bereitstellungsziele und -parameter auflisten.

- Anzahl der Artefakte: `0 to 2`
- Beschreibung: Als Eingabe akzeptiert die Stack-Set-Aktion optional Artefakte für folgende Zwecke:
  - Um die zu verwendende Parameterdatei bereitzustellen. (Weitere Informationen finden Sie unter dem `ParameterOverrides`-Parameter.)
  - Um die zu verwendende Zielkontendatei bereitzustellen. (Weitere Informationen finden Sie unter dem `DeploymentTargets`-Parameter.)

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

- StackSetId: Die ID des Stack-Sets.
- OperationId: Die ID der Stack-Set-Operation.

Weitere Informationen finden Sie unter Variablen.

Beispielaktionskonfiguration

Die folgenden Beispiele zeigen die Aktionskonfiguration für die CloudFormationStackInstancesAktion.

Beispiel für das selbstverwaltete Berechtigungsmodell

Das folgende Beispiel zeigt eine CloudFormationStackInstancesAktion, bei der das eingegebene Bereitstellungsziel eine AWS-Konto ID 111111222222 ist.

YAML

```
Name: my-instances
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackInstances
  Version: '1'
RunOrder: 2
Configuration:
  DeploymentTargets: '111111222222'
  Regions: 'us-east-1,us-east-2,us-west-1,us-west-2'
  StackSetName: my-stackset
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: us-west-2
```

JSON

```
{
    "Name": "my-instances",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackInstances",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "DeploymentTargets": "111111222222",
        "Regions": "us-east-1,us-east-2,us-west-1,us-west-2",
        "StackSetName": "my-stackset"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
```

```
        }
    ],
    "Region": "us-west-2"
}
```

Beispiel für das Modell der vom Service verwalteten Berechtigungen

Das folgende Beispiel zeigt eine CloudFormationStackInstancesAktion für das Modell mit vom Dienst verwalteten Berechtigungen, bei dem das Bereitstellungsziel eine AWS Organisationseinheit-ID `ou-1111-1example` einer Organisation ist.

YAML

```
Name: Instances
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackInstances
  Version: '1'
RunOrder: 2
Configuration:
  DeploymentTargets: ou-1111-1example
  Regions: us-east-1
  StackSetName: my-stackset
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: eu-central-1
```

JSON

```
{
    "Name": "Instances",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackInstances",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "DeploymentTargets": "ou-1111-1example",
```

```
        "Regions": "us-east-1",
        "StackSetName": "my-stackset"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "eu-central-1"
}
```

Berechtigungsmodelle für Stack-Set-Operationen

Da AWS CloudFormation StackSets Operationen über mehrere Konten hinweg ausgeführt werden, müssen Sie die erforderlichen Berechtigungen für diese Konten definieren, bevor Sie das Stack-Set erstellen können. Sie können Berechtigungen über selbstverwaltete oder vom Service verwaltete Berechtigungen definieren.

Mit selbstverwalteten Berechtigungen erstellen Sie die beiden IAM-Rollen, die für erforderlich sind: eine Administratorrolle wie die AWSCloudFormationStackSetAdministrationRole in dem Konto, in dem Sie das Stack-Set definieren, und eine Ausführungsrolle wie die AWSCloudFormationStackSetExecutionRole in jedem der Konten, in denen Sie Stackset-Instanzen bereitstellen. StackSets Mit diesem Berechtigungsmodell StackSets können Sie die Bereitstellung für jedes AWS Konto durchführen, für das der Benutzer über Berechtigungen zum Erstellen einer IAM-Rolle verfügt. Weitere Informationen finden Sie im AWS CloudFormationBenutzerhandbuch unter Gewähren von selbstverwalteten Berechtigungen.

> **Note**
>
> Da AWS CloudFormation StackSets Operationen über mehrere Konten hinweg ausgeführt werden, müssen Sie die erforderlichen Berechtigungen für diese Konten definieren, bevor Sie das Stack-Set erstellen können.

Mit vom Service verwalteten Berechtigungen können Sie Stack-Instances für Konten bereitstellen, die von AWS Organizations verwaltet werden. Mit diesem Berechtigungsmodell müssen Sie nicht die erforderlichen IAM-Rollen erstellen, da die IAM-Rollen in Ihrem Namen StackSets erstellt werden. Mit diesem Modell können Sie auch automatische Bereitstellungen für Konten aktivieren, die der

Organisation in future hinzugefügt werden. Weitere Informationen finden Sie unter Aktivieren des vertrauenswürdigen Zugriffs mit AWS Organizations im AWS CloudFormationBenutzerhandbuch.

Datentypen von Vorlagenparametern

Zu den bei Stack-Set-Operationen verwendeten Vorlagenparametern gehören die folgenden Datentypen. Weitere Informationen finden Sie unter DescribeStackSet.

ParameterKey

- Beschreibung: Der dem Parameter zugeordnete Schlüssel. Wenn Sie keinen Schlüssel und keinen Wert für einen bestimmten Parameter angeben, wird der Standardwert AWS CloudFormation verwendet, der in der Vorlage angegeben ist.
- Beispiel:

```
"ParameterKey=BucketName,ParameterValue=my-bucket"
```

ParameterValue

- Beschreibung: Der dem Parameter zugeordnete Eingabewert.
- Beispiel:

```
"ParameterKey=BucketName,ParameterValue=my-bucket"
```

UsePreviousValue

- Verwenden Sie während einer Stack-Aktualisierung den vorhandenen Parameterwert, den der Stack für einen bestimmten Parameterschlüssel verwendet. Wenn Sie angeben`true`, geben Sie keinen Parameterwert an.
- Beispiel:

```
"ParameterKey=Asset1,UsePreviousValue=true"
```

Jeder Stacksatz hat eine Vorlage und einen Satz von Vorlagenparametern. Wenn Sie die Vorlage oder die Vorlagenparameter aktualisieren, aktualisieren Sie sie für den gesamten Satz. Dann werden alle Instanzstatus auf VERALTET gesetzt, bis die Änderungen für diese Instanz bereitgestellt werden.

Um Parameterwerte für bestimmte Instanzen zu überschreiben, z. B. wenn die Vorlage einen Parameter für `stage` mit dem Wert von enthält`prod`, können Sie den Wert dieses Parameters auf `beta` oder `gamma` überschreiben.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Parametertypen — Dieses Referenzkapitel im AWS CloudFormationBenutzerhandbuch enthält weitere Beschreibungen und Beispiele für CloudFormation Vorlagenparameter.
- Bewährte Methoden — Weitere Informationen zu bewährten Methoden für die Bereitstellung von Stack-Sets finden Sie https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-bestpractices.html im AWS CloudFormationBenutzerhandbuch.
- AWS CloudFormationAPI-Referenz — Weitere Informationen zu den bei Stack-Set-Vorgängen verwendeten Parametern finden Sie in der AWS CloudFormationAPI-Referenz auf die folgenden CloudFormation Aktionen:

  - Die CreateStackSetAktion erstellt ein Stack-Set.
  - Die UpdateStackSetAktion aktualisiert das Stack-Set und die zugehörigen Stack-Instances in den angegebenen Konten und Regionen. Selbst wenn der durch die Aktualisierung des Stack-Sets erzeugte Stack-Set-Vorgang fehlschlägt (ganz oder teilweise, unter oder über einer bestimmten Fehlertoleranz), wird das Stack-Set mit diesen Änderungen aktualisiert. Nachfolgende CreateStackInstances Aufrufe des angegebenen Stack-Sets verwenden das aktualisierte Stack-Set.
  - Die CreateStackInstancesAktion erstellt eine Stack-Instanz für alle angegebenen Regionen innerhalb aller angegebenen Konten bei einem selbstverwalteten Berechtigungsmodell oder innerhalb aller angegebenen Bereitstellungsziele bei einem vom Service verwalteten Berechtigungsmodell. Sie können die Parameter für die durch diese Aktion erstellten Instanzen überschreiben. Wenn die Instanzen bereits existieren, werden CreateStackInstances Aufrufe UpdateStackInstances mit denselben Eingabeparametern ausgeführt. Wenn Sie diese Aktion verwenden, um Instanzen zu erstellen, ändert sich dadurch nicht der Status anderer Stack-Instances.
  - Durch die UpdateStackInstancesAktion werden Stack-Instances mit dem Stack-Set für alle angegebenen Regionen innerhalb aller angegebenen Konten bei einem selbstverwalteten Berechtigungsmodell oder innerhalb aller angegebenen Bereitstellungsziele bei einem vom Service verwalteten Berechtigungsmodell auf den neuesten Stand gebracht. Sie können die Parameter für die durch diese Aktion aktualisierten Instanzen überschreiben. Wenn Sie diese Aktion verwenden, um eine Teilmenge von Instances zu aktualisieren, ändert sich dadurch nicht der Status anderer Stack-Instances.

- Die DescribeStackSetOperationAktion gibt die Beschreibung des angegebenen Stack-Set-Vorgangs zurück.
- Die DescribeStackSetAktion gibt die Beschreibung des angegebenen Stack-Sets zurück.

AWS CodeBuild

Ermöglicht die Ausführung von Builds und Tests als Teil Ihrer Pipeline. Wenn Sie eine CodeBuild Build- oder Testaktion ausführen, werden die in der Build-Spezifikation angegebenen Befehle in einem CodeBuild Container ausgeführt. Alle Artefakte, die als Eingabeartefakte für eine CodeBuild Aktion angegeben sind, sind innerhalb des Containers verfügbar, der die Befehle ausführt. CodeBuild kann entweder eine Build- oder eine Testaktion bereitstellen. Weitere Informationen finden Sie im AWS CodeBuild-Benutzerhandbuch.

Wenn Sie den CodePipeline Assistenten in der Konsole verwenden, um ein Build-Projekt zu erstellen, zeigt das CodeBuild Build-Projekt an, welcher Quellanbieter es ist CodePipeline. Wenn Sie ein Build-Projekt in der CodeBuild Konsole erstellen, können Sie es nicht CodePipeline als Quellanbieter angeben. Wenn Sie jedoch 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
- Aktionsdeklaration (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 für die Aktion sein. CodeBuild sucht nach der Build-Spezifikationsdatei und führt die Build-Spezifikationsbefehle in dem Verzeichnis aus, das die entpackte Version dieses Artefakts enthält.

Dieser Parameter ist erforderlich, wenn 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

Required: No

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

Required: No

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

Required: No

Der Wert dieses Parameters wird verwendet, um Umgebungsvariablen für die CodeBuild -Aktion in Ihrer Pipeline festzulegen. Der Wert für den Parameter `EnvironmentVariables` hat das Format eines JSON-Arrays von Umgebungsvariablenobjekten. Einen Beispielparameter finden Sie unter Aktionsdeklaration (CodeBuild-Beispiel).

Jedes Objekt besteht aus drei Teilen. Alle Teile sind Zeichenfolgen:

- `name`: Der Name oder Schlüssel der Umgebungsvariable.
- `value`: Der Wert der Umgebungsvariable. Wenn Sie den`SECRETS_MANAGER` Typ`PARAMETER_STORE` oder verwenden, muss dieser Wert der Name eines Parameters sein, den Sie bereits imAWS Systems Manager Parameter Store gespeichert haben, oder eines geheimen Parameters, das Sie bereits inAWS Secrets Manager gespeichert haben.

  > **Note**
  >
  > Wir raten dringend von der Verwendung von Umgebungsvariablen zum Speichern sensibler Werte, insbesondere vonAWS Anmeldeinformationen, ab. Wenn Sie die CodeBuild Konsole oder dieAWS 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`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.

Weitere Informationen finden Sie EnvironmentVariableunterAWS CodeBuild API-Referenz. Eine CodeBuild Beispielaktion mit einer Umgebungsvariablen, die in den GitHub Zweignamen aufgelöst wird, finden Sie unterBeispiel: Verwenden Sie eine BranchName Variable mit CodeBuild Umgebungsvariablen.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte:`1 to 5`
- Beschreibung: CodeBuild sucht nach der Build-Spezifikationsdatei und führt die Build-Spezifikationsbefehle aus dem Verzeichnis des primären Quellartefakts aus. Wenn mehr als eine Eingabequelle für die CodeBuild Aktion angegeben ist, muss dieses Artefakt mithilfe des`PrimarySource` Aktionskonfigurationsparameters in festgelegt werden CodePipeline.

  Jedes Eingabeartefakt wird in ein eigenes Verzeichnis extrahiert, dessen Speicherorte in Umgebungsvariablen gespeichert sind. Das Verzeichnis für das primäre Quellartefakt wird mit `$CODEBUILD_SRC_DIR` zur Verfügung gestellt. Die Verzeichnisse für alle anderen Eingabeartefakte werden mit `$CODEBUILD_SRC_DIR_yourInputArtifactName` zur Verfügung gestellt.

  > **Note**
  >
  > Das in Ihrem CodeBuild Projekt konfigurierte Artefakt wird zum Eingabeartefakt, das von der CodeBuild Aktion in Ihrer Pipeline verwendet wird.

Ausgabeartefakte

- Anzahl der Artefakte:`0 to 5`
- Beschreibung: Diese können verwendet werden, um die in der CodeBuild Build-Spezifikationsdatei definierten Artefakte für nachfolgende Aktionen in der Pipeline verfügbar zu machen. Wenn nur ein Ausgabeartefakt definiert ist, kann dieses Artefakt direkt im `artifacts`-Abschnitt der Build-Spezifikationsdatei definiert werden. Wenn mehr als ein Ausgabe-Artefakt angegeben ist, müssen in der Buildspezifikationsdatei alle Artefakte, auf die verwiesen wird, als sekundäre Artefakte definiert werden. Die Namen der Ausgabeartefakte in CodePipeline müssen mit den Artefakt-IDs in der Build-Spezifikationsdatei übereinstimmen.

> **Note**
>
> Das in Ihrem CodeBuild Projekt konfigurierte Artefakt wird zum CodePipeline Eingabeartefakt in Ihrer Pipeline-Aktion.

Wenn der`CombineArtifacts` Parameter für Batch-Builds ausgewählt ist, enthält der Speicherort des Ausgabe-Artefakts die kombinierten Artefakte aus mehreren Builds, die in derselben Ausführung ausgeführt wurden.

Ausgabevariablen

Diese Aktion erzeugt als Variablen alle Umgebungsvariablen, die als Teil des Builds exportiert wurden. Weitere Informationen zum Exportieren von Umgebungsvariablen finden Sie EnvironmentVariableim AWS CodeBuildAPI-Handbuch.

Weitere Informationen zur Verwendung von CodeBuild Umgebungsvariablen in CodePipeline finden Sie in den Beispielen inCodeBuild Aktionsausgabevariablen. Eine Liste der Umgebungsvariablen, die Sie verwenden können CodeBuild, finden Sie im AWS CodeBuildBenutzerhandbuch unter Umgebungsvariablen in Build-Umgebungen.

Aktionsdeklaration (CodeBuild-Beispiel)

YAML

```
Name: Build
Actions:
  - Name: PackageExport
    ActionTypeId:
      Category: Build
      Owner: AWS
      Provider: CodeBuild
      Version: '1'
    RunOrder: 1
    Configuration:
      BatchEnabled: 'true'
      CombineArtifacts: 'true'
      ProjectName: my-build-project
      PrimarySource: MyApplicationSource1
```

```
      EnvironmentVariables:
 '[{"name":"TEST_VARIABLE","value":"TEST_VALUE","type":"PLAINTEXT"},
{"name":"ParamStoreTest","value":"PARAMETER_NAME","type":"PARAMETER_STORE"}]'
    OutputArtifacts:
      - Name: MyPipeline-BuildArtifact
    InputArtifacts:
      - Name: MyApplicationSource1
      - Name: MyApplicationSource2
```

JSON

```
{
    "Name": "Build",
    "Actions": [
        {
            "Name": "PackageExport",
            "ActionTypeId": {
                "Category": "Build",
                "Owner": "AWS",
                "Provider": "CodeBuild",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BatchEnabled": "true",
                "CombineArtifacts": "true",
                "ProjectName": "my-build-project",
                "PrimarySource": "MyApplicationSource1",
                "EnvironmentVariables": "[{\"name\":\"TEST_VARIABLE\",\"value\":
\"TEST_VALUE\",\"type\":\"PLAINTEXT\"},{\"name\":\"ParamStoreTest\",\"value\":
\"PARAMETER_NAME\",\"type\":\"PARAMETER_STORE\"}]"
            },
            "OutputArtifacts": [
                {
                    "Name": "MyPipeline-BuildArtifact"
                }
            ],
            "InputArtifacts": [
                {
                    "Name": "MyApplicationSource1"
                },
                {
```

```
                    "Name": "MyApplicationSource2"
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWS CodeBuildBenutzerhandbuch — Ein Beispiel für eine Pipeline mit einer CodeBuild Aktion finden Sie unter Verwenden CodePipeline von, CodeBuild um Code zu testen und Builds auszuführen. Beispiele für Projekte mit mehreren Eingabe- und CodeBuild Ausgabeartefakten finden Sie unter Beispiel CodePipelineIntegration mit CodeBuild und mehreren Eingabequellen und Ausgabeartefakten und Beispiel für mehrere Eingabequellen und Ausgabeartefakte.
- Tutorial: Erstellen einer Pipeline, die Ihre Android-App mit erstellt und testet AWS Device Farm— Dieses Tutorial enthält ein Beispiel für eine Build-Spezifikationsdatei und eine Beispielanwendung zum Erstellen einer Pipeline mit einer GitHub Quelle, die eine Android-App mit und erstellt CodeBuild und testetAWS Device Farm.
- Referenz zur Build-Spezifikation für CodeBuild — Dieses Referenzthema enthält Definitionen und Beispiele zum Verständnis von CodeBuild Build-Spezifikationsdateien. Eine Liste der Umgebungsvariablen, die Sie verwenden können CodeBuild, finden Sie im AWS CodeBuildBenutzerhandbuch unter Umgebungsvariablen in Build-Umgebungen.

CodeCommit

Startet die Pipeline, wenn ein neuer Commit für das konfigurierte CodeCommit Repository und die Verzweigung vorgenommen wird.

Wenn Sie die Konsole verwenden, um die Pipeline zu erstellen oder zu bearbeiten, CodePipeline erstellt eine CodeCommit CloudWatch Ereignisregel, die Ihre Pipeline startet, wenn eine Änderung im Repository auftritt.

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 für die Ausgabe einer ZIP-Datei mit einer beliebigen Kopie Ihres Commit.
- Vollständiger Klon – Konfiguriert die Quellaktion so, dass ein Git-URL-Referenz zum Repository für nachfolgende Aktionen ausgegeben wird.

  Derzeit kann die Git-URL-Referenz nur von nachgelagerten CodeBuild Aktionen verwendet werden, um das Repo und die zugehörigen Git-Metadaten zu klonen. Der Versuch, eine Git-URL-Referenz an Nicht-CodeBuild Aktionen zu übergeben, führt zu einem Fehler.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Beispielaktionskonfiguration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Source`
- Eigentümer: `AWS`
- Anbieter: `CodeCommit`
- Version: `1`

Konfigurationsparameter

RepositoryName

Erforderlich: Ja

Der Name des Repositorys, in dem Quelländerungen erkannt werden sollen.

BranchName

Erforderlich: Ja

Der Name des Zweigs, in dem Quelländerungen erkannt werden sollen.

PollForSourceChanges

Erforderlich: Nein

`PollForSourceChanges` steuert, ob das CodeCommit Repository nach Quelländerungen CodePipeline abfragt. Wir empfehlen, stattdessen CloudWatch Ereignisse zu verwenden, um Quelländerungen zu erkennen. Weitere Informationen zum Konfigurieren von CloudWatch Ereignissen finden Sie unter Migrieren von Abfrage-Pipelines (CodeCommit Quelle) (CLI) oder Migrieren Sie Abfrage-Pipelines (CodeCommit Quelle) (AWS CloudFormationVorlage).

> **Important**
>
> Wenn Sie eine CloudWatch Ereignisregel konfigurieren möchten, müssen Sie `PollForSourceChanges` auf setzen, `false` um doppelte Pipeline-Ausführungen zu vermeiden.

Gültige Werte für diesen Parameter sind:

- `true`: Wenn diese Option festgelegt ist, CodePipeline werden die Quelländerungen in Ihrem Repository erfasst.

  > **Note**
  >
  > Wenn Sie weglassen`PollForSourceChanges`, fragt CodePipeline standardmäßig Ihr Repository nach Quelländerungen ab. Dieses Verhalten ist das gleiche, als ob `PollForSourceChanges` enthalten wäre und auf `true` festgelegt würde.

- `false`: Wenn diese Option festgelegt ist, CodePipeline fragt Ihr Repository nicht nach Quelländerungen ab. Verwenden Sie diese Einstellung, wenn Sie eine CloudWatch Ereignisregel konfigurieren möchten, um Quelländerungen zu erkennen.

OutputArtifactFormat

Erforderlich: Nein

Das Ausgabeartefaktformat. Die Werte können entweder `CODEBUILD_CLONE_REF` oder sein`CODE_ZIP`. 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 in gezeigtHinzufügen von CodeBuild GitClone Berechtigungen für CodeCommit Quellaktionen. Sie müssen Ihrer Servicerolle auch die `-codecommit:GetRepository`Berechtigung hinzufügen, CodePipeline wie in gezeigtHinzufügen von Berechtigungen zur CodePipeline-Servicerolle. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiger Klon verwenden, finden Sie unter Tutorial: Vollständigen Klon mit einer CodeCommit Pipeline-Quelle verwenden.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0`
- Beschreibung: Eingabe-Artefakte sind für diesen Aktionstyp nicht gültig.

Ausgabeartefakte

- Anzahl der Artefakte: `1`
- Beschreibung: Das Ausgabe-Artefakt dieser Aktion ist eine ZIP-Datei, die den Inhalt des konfigurierten Repositorys und Zweigs beim Commit enthält, der als Quellrevision für die Pipeline-Ausführung angegeben wurde. Die aus dem Repository generierten Artefakte sind die Ausgabeartefakte für die CodeCommit Aktion. Die Quellcode-Commit-ID wird in CodePipeline als Quellrevision für die ausgelöste Pipeline-Ausführung angezeigt.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Diese Aktion erzeugt Variablen, die als Ausgabevariablen angezeigt werden können, auch wenn die Aktion

keinen Namespace hat. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

Weitere Informationen finden Sie unter Variablen.

CommitId

Die CodeCommit Commit-ID, die die Pipeline-Ausführung ausgelöst hat. Commit-IDs sind der vollständige SHA des Commits.

CommitMessage

Die Beschreibungsmeldung (wenn vorhanden), die dem Commit zugeordnet ist, der die Pipeline-Ausführung ausgelöst hat.

RepositoryName

Der Name des CodeCommit Repositorys, in dem der Commit durchgeführt wurde, der die Pipeline ausgelöst hat.

BranchName

Der Name der Verzweigung für das CodeCommit Repository, in dem die Quelländerung vorgenommen wurde.

AuthorDate

Das Datum im Zeitstempelformat, an dem der Commit erstellt wurde.

Weitere Informationen zum Unterschied zwischen einem Autor und einem Committer in Git finden Sie unter Viewing the Commit History in Pro Git von Scott Chacon und Ben Straub.

CommitterDate

Das Datum im Zeitstempelformat, an dem der Commit durchgeführt wurde.

Weitere Informationen zum Unterschied zwischen einem Autor und einem Committer in Git finden Sie unter Viewing the Commit History in Pro Git von Scott Chacon und Ben Straub.

Beispielaktionskonfiguration

Beispiel für das Standardformat des Ausgabeartefakts

YAML

```
Actions:
  - OutputArtifacts:
      - Name: Artifact_MyWebsiteStack
    InputArtifacts: []
    Name: source
    Configuration:
      RepositoryName: MyWebsite
      BranchName: main
      PollForSourceChanges: 'false'
    RunOrder: 1
    ActionTypeId:
      Version: '1'
      Provider: CodeCommit
      Category: Source
      Owner: AWS
  Name: Source
```

JSON

```
{
    "Actions": [
        {
            "OutputArtifacts": [
                {
                    "Name": "Artifact_MyWebsiteStack"
                }
            ],
            "InputArtifacts": [],
            "Name": "source",
            "Configuration": {
                "RepositoryName": "MyWebsite",
                "BranchName": "main",
                "PollForSourceChanges": "false"
            },
            "RunOrder": 1,
            "ActionTypeId": {
                "Version": "1",
```

```
                "Provider": "CodeCommit",
                "Category": "Source",
                "Owner": "AWS"
            }
        }
    ],
    "Name": "Source"
},
```

Beispiel für das vollständige Klonausgabeartefaktformat

YAML

```
name: Source
actionTypeId:
  category: Source
  owner: AWS
  provider: CodeCommit
  version: '1'
runOrder: 1
configuration:
  BranchName: main
  OutputArtifactFormat: CODEBUILD_CLONE_REF
  PollForSourceChanges: 'false'
  RepositoryName: MyWebsite
outputArtifacts:
  - name: SourceArtifact
inputArtifacts: []
region: us-west-2
namespace: SourceVariables
```

JSON

```
{
    "name": "Source",
    "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "provider": "CodeCommit",
        "version": "1"
    },
    "runOrder": 1,
```

```
    "configuration": {
        "BranchName": "main",
        "OutputArtifactFormat": "CODEBUILD_CLONE_REF",
        "PollForSourceChanges": "false",
        "RepositoryName": "MyWebsite"
    },
    "outputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "inputArtifacts": [],
    "region": "us-west-2",
    "namespace": "SourceVariables"
}
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository) – Dieses Tutorial enthält eine Beispiel-App-Spezifikationsdatei sowie eine CodeDeploy Beispielanwendung und Bereitstellungsgruppe. Verwenden Sie dieses Tutorial, um eine Pipeline mit einer CodeCommit Quelle zu erstellen, die auf Amazon EC2 bereitgestellt wird.

AWS CodeDeploy

Du benutzt einAWS CodeDeployAktion zum Bereitstellen von Anwendungscode für Ihre Bereitstellungsflotte. Ihre Bereitstellungsflotte kann aus Amazon EC2 EC2-Instances, lokalen Instances oder beiden bestehen.

> **Note**
>
> In diesem Referenzthema wird die CodeDeploy Bereitstellungsaktion für CodePipeline wo die Bereitstellungsplattform Amazon EC2 ist. Weitere Informationen zu Amazon Elastic Container Service zu CodeDeploy blau/grün-Bereitstellungsaktionen in CodePipeline finden Sie unterAmazon Elastic Container Service undCodeDeploy Blau-Grün-aus.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Deploy`
- Eigentümer: `AWS`
- Anbieter: `CodeDeploy`
- Version: `1`

Konfigurationsparameter

ApplicationName

Erforderlich: Ja

Der Name der Anwendung, die Sie in CodeDeploy erstellt haben.

DeploymentGroupName

Erforderlich: Ja

Die Bereitstellungsgruppe, die Sie in CodeDeploy erstellt haben.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung Die AppSpec datei das CodeDeploy verwendet, um zu bestimmen:
  - Was auf Ihren Instances aus der Anwendungsrevision in Amazon S3 oder GitHub installiert werden soll.

- Welche Lebenszyklusereignis-Hooks als Reaktion auf Bereitstellungslebenszyklusereignisse ausgeführt werden sollen.

Weitere Informationen zu AppSpec -Datei, siehe dieCodeDeploy AppSpec Datei-Referenzaus.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung Ausgabe-Artefakte sind für diesen Aktionstyp nicht gültig.

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 durch die Erstellung eines Quell-Buckets, EC2-Instanzen und CodeDeploy Ressourcen zum Bereitstellen einer Beispielanwendung Dann baust du deine Pipeline mit einem CodeDeploy Bereitstellungsaktion, die den in Ihrem S3-Bucket verwalteten Code auf Ihrer Amazon EC2 EC2-Instance bereitstellt.
- Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository)— Dieses Tutorial führt Sie durch die Erstellung Ihres CodeCommit Quell-Repository, EC2-Instanzen und CodeDeploy Ressourcen zum Bereitstellen einer Beispielanwendung Dann baust du deine Pipeline mit einem CodeDeploy Bereitstellungsaktion, die Code von Ihrem bereitstellt CodeCommit Repository für Ihre Amazon EC2 EC2-Instance.

- CodeDeploy AppSpec Datei-Referenz— Dieses Referenzkapitel imAWS CodeDeploy-Benutzerhandbuchbietet Referenzinformationen und Beispiele für CodeDeploy AppSpec-Dateien.

CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen

Startet eine Pipeline, wenn ein neuer Commit für ein Quellcode-Repository eines Drittanbieters durchgeführt 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 eine Git-Konfiguration verwendet wird, mit der Sie Ihre Pipeline mit Auslösern starten können. Weitere Informationen zum Konfigurieren der Pipeline-Auslöserkonfiguration zum Filtern mit Auslösern finden Sie unter Filtern von Auslösern bei Code-Push- oder Pull-Anforderungen.

> **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 (USA-West) nicht verfügbar. Informationen zum Verweisen auf andere verfügbare Aktionen finden Sie unter Produkt- und Serviceintegrationen mit CodePipeline. Überlegungen zu dieser Aktion in der Region Europa (Mailand) finden Sie im Hinweis unter CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen.

Verbindungen können Ihre AWS-Ressourcen mit den folgenden Drittanbieter-Repositorys verknüpfen:

- Bitbucket Cloud (über die Bitbucket-Anbieteroption 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 Sie einen Bitbucket-Workspace verwenden, müssen Sie über Administratorzugriff verfügen, um die Verbindung zu erstellen.

- GitHub und GitHub Enterprise Cloud (über die Anbieteroption GitHub (Version 2) in der CodePipeline Konsole oder den `GitHub` Anbieter in der CLI)

> **Note**
>
> Wenn sich Ihr Repository in einer GitHub Organisation befindet, müssen Sie der Eigentümer der Organisation sein, um die Verbindung herzustellen. Wenn Sie ein Repository verwenden, das sich nicht in einer Organisation befindet, müssen Sie der Eigentümer des Repositorys sein.

- GitHub Enterprise Server (über die GitHub Enterprise Server-Anbieteroption in der CodePipeline Konsole oder den `GitHub Enterprise Server` Anbieter in der CLI)
- GitLab.com (über die GitLab Anbieteroption in der CodePipeline Konsole oder den `GitLab` Anbieter in der CLI)

> **Note**
>
> Sie können Verbindungen zu einem Repository erstellen, in dem Sie die Rolle Besitzer in haben GitLab. Anschließend kann die Verbindung mit dem Repository mit Ressourcen wie verwendet werden CodePipeline. Bei Repositorys in Gruppen müssen Sie nicht der Gruppenbesitzer sein.

- Selbstverwaltete Installation für GitLab (Enterprise Edition oder Community Edition) (über die Option für GitLab selbstverwaltete Anbieter in der CodePipeline Konsole oder den `GitLabSelfManaged` Anbieter in der CLI)

> **Note**
>
> Jede Verbindung unterstützt alle Repositorys, die Sie mit diesem Anbieter haben. Sie müssen nur für jeden Anbietertyp eine neue Verbindung erstellen.

Verbindungen ermöglichen es Ihrer Pipeline, Quelländerungen über die Installations-App des Drittanbieters zu erkennen. Webhooks werden beispielsweise verwendet, um GitHub Ereignistypen zu abonnieren, und können in einer Organisation, einem Repository oder einer GitHub App installiert werden. Ihre Verbindung installiert einen Repository-Webhook auf Ihrer - GitHub App, der GitHub Push-Typ-Ereignisse abonniert.

Nachdem eine Codeänderung erkannt wurde, haben Sie die folgenden Optionen, um den Code an nachfolgende Aktionen zu übergeben:

- Standard: Wie andere vorhandene CodePipeline Quellaktionen `CodeStarSourceConnection` kann eine ZIP-Datei mit einer zulässigen Kopie Ihres Commit ausgeben.
- Vollständiger Klon: `CodeStarSourceConnection` kann auch so konfiguriert werden, dass ein URL-Referenz auf das Repo für nachfolgende Aktionen ausgegeben wird.

  Derzeit kann die Git-URL-Referenz nur von nachgelagerten CodeBuild Aktionen verwendet werden, um das Repo und die zugehörigen Git-Metadaten zu klonen. Der Versuch, einen Git-URL-Referenz auf Nicht-CodeBuild Aktionen zu übergeben, führt zu einem Fehler.

CodePipeline fordert Sie auf, die AWS Connector-Installations-App zu Ihrem Drittanbieterkonto hinzuzufügen, wenn Sie eine Verbindung erstellen. Sie müssen bereits Ihr Drittanbieterkonto und Repository erstellt haben, bevor Sie über die `CodeStarSourceConnection` Aktion eine Verbindung herstellen können.

> **Note**
>
> Informationen zum Erstellen oder Anfügen einer Richtlinie an Ihre Rolle mit den Berechtigungen, die für die Verwendung von AWS CodeStar Verbindungen erforderlich sind, finden Sie unter Referenz zu Verbindungsberechtigungen. Je nachdem, wann Ihre CodePipeline Servicerolle erstellt wurde, müssen Sie möglicherweise ihre Berechtigungen aktualisieren, um AWS CodeStar Verbindungen zu unterstützen. Anweisungen finden Sie unter Hinzufügen von Berechtigungen zur CodePipeline-Servicerolle.

> **Note**
>
> Voraussetzungen, um Verbindungen in der AWS-Region Europa (Mailand) nutzen zu können:
>
> 1. Regionsspezifische App installieren
> 2. Region aktivieren
>
> Diese regionsspezifische App unterstützt Verbindungen in der Region Europa (Mailand). Sie ist auf der Website des Drittanbieters veröffentlicht und von der bestehenden App getrennt, die Verbindungen für andere Regionen unterstützt. Durch die Installation dieser App autorisieren Sie Drittanbieter, Ihre Daten nur für diese Region an den Dienst weiterzugeben, und Sie können diese Autorisierung jederzeit widerrufen, indem Sie die App deinstallieren. Der Dienst verarbeitet oder speichert Ihre Daten nicht, es sei denn, Sie aktivieren die Region. Durch die Aktivierung dieser Region gewähren Sie unserem Dienst die Erlaubnis, Ihre Daten zu verarbeiten und zu speichern.
> Auch wenn die Region nicht aktiviert ist, können Drittanbieter Ihre Daten trotzdem mit unserem Dienst teilen, wenn die regionsspezifische App installiert bleibt. Achten Sie also darauf, die App zu deinstallieren, sobald Sie die Region deaktivieren. Weitere Informationen finden Sie unter Aktivieren einer Region.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Aktionsdeklaration
- Installieren der Installations-App und Erstellen 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 richtige Schreibweise für den FullRepositoryId Wert beibehalten. Wenn Ihr Benutzername beispielsweise `some-user` und der Reponame lautet`My-Repo`, FullRepositoryId ist der empfohlene Wert von `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 Projektservicerolle aktualisieren, wie in gezeigtHinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder GitLab.com. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiger Klon verwenden, finden Sie unter Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

DetectChanges

Erforderlich: Nein

Steuert automatisch das Starten Ihrer Pipeline, wenn ein neuer Commit für das konfigurierte Repository und die Verzweigung vorgenommen wird. Wenn nicht angegeben, ist der Standardwert und `true`das Feld wird standardmäßig nicht angezeigt. Gültige Werte für diesen Parameter sind:

- `true`: startet Ihre Pipeline bei neuen Commits CodePipeline automatisch.
- `false`: startet Ihre Pipeline CodePipeline 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 in 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 nachgelagerten Aktionen verwendet werden. Wenn Sie diese Option wählen, müssen Sie die Berechtigungen für Ihre CodeBuild Projektservicerolle aktualisieren, wie in gezeigtFehlerbehebung CodePipeline. Ein Tutorial, das Ihnen zeigt, wie Sie die Option Vollständiger Klon verwenden, finden Sie unter Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

Ausgabevariablen

Wenn dies konfiguriert ist, werden durch diese Aktion Variablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können. Diese Aktion erzeugt Variablen, die als Ausgabevariablen angezeigt werden können, auch wenn die Aktion keinen Namespace hat. Sie konfigurieren eine Aktion mit einem Namespace, um diese Variablen für die Konfiguration nachgeschalteter Aktionen zur Verfügung zu stellen.

Weitere Informationen finden Sie unter Variablen.

AuthorDate

Das Datum im Zeitstempelformat, an dem der Commit erstellt wurde.

BranchName

Der Name des Zweigs für das -Repository, in dem die Quelländerung ausgeführt wurde.

CommitId

Die -Commit-ID, die die Pipeline-Ausführung ausgelöst hat.

CommitMessage

Die Beschreibungsmeldung (wenn vorhanden), die dem Commit zugeordnet ist, der die Pipeline-Ausführung ausgelöst hat.

ConnectionArn

Der Verbindungs-ARN, der für den Quellanbieter konfiguriert und authentifiziert ist.

FullRepositoryName

Der Name des -Repositorys, in dem der Commit, der die Pipeline ausgelöst hat, ausgeführt wurde.

Aktionsdeklaration

Im folgenden Beispiel ist das Ausgabeartefakt auf das Standard-ZIP-Format von `CODE_ZIP` für die Verbindung mit ARN festgelegt`arn:aws:codestar-connections:region:account-id:connection/connection-id`.

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"
        }
    ]
},
```

Installieren der Installations-App und Erstellen einer Verbindung

Wenn Sie zum ersten Mal die Konsole verwenden, um eine neue Verbindung zu einem Drittanbieter-Repository hinzuzufügen, müssen Sie den CodePipeline Zugriff auf Ihre Repositorys autorisieren. Sie wählen oder erstellen eine Installations-App, mit der Sie eine Verbindung mit dem Konto herstellen können, in dem Sie das Code-Repository eines Drittanbieters erstellt haben.

Wenn Sie die AWS CLI oder eine -AWS CloudFormationVorlage verwenden, müssen Sie den Verbindungs-ARN einer Verbindung angeben, die bereits den Installations-Handshake durchlaufen hat. Andernfalls wird die Pipeline nicht ausgelöst.

> **Note**
>
> Für eine `CodeStarSourceConnection` Quellaktion müssen Sie keinen Webhook einrichten oder standardmäßig abfragen. Die Verbindungsaktion verwaltet die Erkennung von Quelländerungen für Sie.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWS::CodeStarConnections::Connection – Die AWS CloudFormation Vorlagenreferenz für die AWS CodeStar Connections-Ressource enthält Parameter und Beispiele für Verbindungen in AWS CloudFormation Vorlagen.
- AWS CodeStar Connections API Reference – Die AWS CodeStar Connections API Reference enthält Referenzinformationen für die verfügbaren Verbindungsaktionen.
- Die Schritte zum Erstellen einer Pipeline mit Quellaktionen, die von Verbindungen unterstützt werden, finden Sie im Folgenden:
  - Verwenden Sie für Bitbucket Cloud die Bitbucket-Option in der Konsole oder die `-CodestarSourceConnection`Aktion in der CLI. Siehe Bitbucket-Cloud-Verbindungen.
  - Verwenden Sie für GitHub und GitHub Enterprise Cloud die GitHub Anbieteroption in der Konsole oder die `CodestarSourceConnection` Aktion in der CLI. Siehe GitHub Verbindungen.
  - Verwenden Sie für GitHub Enterprise Server die Option GitHub Enterprise Server-Anbieter in der Konsole oder die `CodestarSourceConnection` Aktion in der CLI. Siehe GitHub Enterprise Server-Verbindungen.
  - Verwenden Sie für GitLab.com die GitLab Anbieteroption in der Konsole oder die `CodestarSourceConnection` Aktion mit dem `GitLab` Anbieter in der CLI. Siehe GitLab.com-Verbindungen.
- Ein Tutorial für die ersten Schritte, das eine Pipeline mit einer Bitbucket-Quelle und einer -CodeBuild Aktion erstellt, finden Sie unter Erste Schritte mit Verbindungen.
- Ein Tutorial, das Ihnen zeigt, wie Sie eine Verbindung zu einem GitHub Repository herstellen und die Option Vollständiger Klon mit einer Downstream- CodeBuild Aktion verwenden, finden Sie unter Tutorial: Vollständigen Klon mit einer GitHub Pipeline-Quelle verwenden.

AWS Device Farm

In Ihrer Pipeline können Sie eine Testaktion konfigurieren, mit der AWS Device Farm Ihre Anwendung auf Geräten ausgeführt und getestet wird. Device Farm verwendet Testpools von Geräten und Test-Frameworks, um Anwendungen auf bestimmten Geräten zu testen. Informationen

zu den Typen von Testframeworks, die von der Aktion Device Farm unterstützt werden, finden Sie unter Arbeiten mit Testtypen in AWS Device Farm.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Aktionsdeklaration
- Weitere Informationen finden Sie auch unter

Aktionstyp

- Kategorie: `Test`
- Eigentümer: `AWS`
- Anbieter: `DeviceFarm`
- Version: `1`

Konfigurationsparameter

AppType

Erforderlich: Ja

Das Betriebssystem und die Art der Anwendung, die Sie testen. Im Folgenden finden Sie eine Liste gültiger Werte:

- `iOS`
- `Android`
- `Web`

ProjectId

Erforderlich: Ja

Die Device Farm Farm-Projekt-ID.

Um Ihre Projekt-ID zu finden, wählen Sie in der Device Farm Farm-Konsole Ihr Projekt aus. Kopieren Sie im Browser die URL Ihres neuen Projekts. Die URL enthält die Projekt-ID. Die Projekt-ID ist der Wert in der URL danach`projects/`. Im folgenden Beispiel lautet die Projekt-ID`eec4905f-98f8-40aa-9afc-4c1cfexample`.

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

App

Erforderlich: Ja

Der Name und der Speicherort der Anwendungsdatei in Ihrem Eingabeartefakt. Beispiel: `s3-ios-test-1.ipa`

TestSpec

Bedingt: Ja

Der Speicherort der Definitionsdatei der Testspezifikation in Ihrem Eingabeartefakt. Dies ist für den Test im benutzerdefinierten Modus erforderlich.

DevicePoolArn

Erforderlich: Ja

Der Gerätefarm-Gerätepool-ARN.

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

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

TestType

Erforderlich: Ja

Gibt das unterstützte Test-Framework für Ihren Test an. Im Folgenden finden Sie eine Liste gültiger Werte für`TestType`:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG

- APPIUM_KNOTEN
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- EINGEBAUTETER_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> Die folgenden Testtypen werden von der Aktion in CodePipeline:`WEB_PERFORMANCE_PROFILE`, und nicht unterstützt. `REMOTE_ACCESS_RECORD` `REMOTE_ACCESS_REPLAY`

Informationen zu Device Farm-Testtypen finden Sie unter Arbeiten mit Testtypen in AWS Device Farm.

RadioBluetoothEnabled

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob Bluetooth zu Beginn des Tests aktiviert werden soll.

RecordAppPerformanceData

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob während des Tests Geräteleistungsdaten wie CPU-, FPS- und Speicherleistung aufgezeichnet werden sollen.

RecordVideo

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob während des Tests Videos aufgenommen werden sollen.

RadioWifiEnabled

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob Wi-Fi zu Beginn des Tests aktiviert werden soll.

RadioNfcEnabled

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob NFC zu Beginn des Tests aktiviert werden soll.

RadioGpsEnabled

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob GPS zu Beginn des Tests aktiviert werden soll.

Test

Erforderlich: Nein

Der Name und der Pfad der Testspezifikationsdatei an Ihrem Quellspeicherort. Der Pfad ist relativ zum Stamm des Eingabeartefakts für Ihren Test.

FuzzEventCount

Erforderlich: Nein

Die Anzahl der Benutzeroberflächenereignisse, die der Fuzz-Test durchführen soll, zwischen 1 und 10.000.

FuzzEventThrottle

Erforderlich: Nein

Die Anzahl der Millisekunden, die der Fuzz-Test warten muss, bevor das nächste Benutzeroberflächenereignis ausgeführt wird, zwischen 1 und 1.000.

FuzzRandomizerSeed

Erforderlich: Nein

Ein Startwert für den Fuzz-Test, der für die Randomisierung von Benutzeroberflächenereignissen verwendet wird. Die Verwendung derselben Zahl für nachfolgende Fuzz-Tests führt zu identischen Ereignissequenzen.

CustomHostMachineArtifacts

Erforderlich: Nein

Der Speicherort auf dem Host-Computer, an dem benutzerdefinierte Artefakte gespeichert werden.

CustomDeviceArtifacts

Erforderlich: Nein

Der Ort auf dem Gerät, an dem benutzerdefinierte Artefakte gespeichert werden.

UnmeteredDevicesOnly

Erforderlich: Nein

Ein boolescher Wert, der angibt, ob Sie bei der Ausführung von Tests in diesem Schritt nur Ihre Geräte ohne Zähler verwenden möchten.

JobTimeoutMinutes

Erforderlich: Nein

Die Anzahl der Minuten, die ein Testlauf pro Gerät ausgeführt wird, bevor das Timeout überschritten wird.

Breitengrand

Erforderlich: Nein

Der Breitengrad des Geräts, ausgedrückt in Grad des geographischen Koordinatensystems.

Längengrad

Erforderlich: Nein

Der Längengrad des Geräts, ausgedrückt in Grad des geographischen Koordinatensystems.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Die Gruppe von Artefakten, die für die Testaktion verfügbar gemacht werden sollen. Device Farm sucht nach der erstellten Anwendung und den zu verwendenden Testdefinitionen.

Ausgabeartefakte

- Anzahl der Artefakte: `0`
- Beschreibung: Ausgabeartefakte gelten nicht für diesen Aktionstyp.

Aktionsdeklaration

YAML

```
Name: Test
Actions:
  - Name: TestDeviceFarm
    ActionTypeId: null
    category: Test
    owner: AWS
    provider: DeviceFarm
    version: '1'
RunOrder: 1
Configuration:
  App: s3-ios-test-1.ipa
  AppType: iOS
  DevicePoolArn: >-
    arn:aws:devicefarm:us-west-2::devicepool:0EXAMPLE-d7d7-48a5-ba5c-b33d66efa1f5
  ProjectId: eec4905f-98f8-40aa-9afc-4c1cfEXAMPLE
  TestType: APPIUM_PYTHON
  TestSpec: example-spec.yml
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: us-west-2
```

JSON

```
{
    "Name": "Test",
    "Actions": [
        {
            "Name": "TestDeviceFarm",
            "ActionTypeId": null,
            "category": "Test",
            "owner": "AWS",
```

```
            "provider": "DeviceFarm",
            "version": "1"
        }
    ],
    "RunOrder": 1,
    "Configuration": {
        "App": "s3-ios-test-1.ipa",
        "AppType": "iOS",
        "DevicePoolArn": "arn:aws:devicefarm:us-west-2::devicepool:0EXAMPLE-
d7d7-48a5-ba5c-b33d66efa1f5",
        "ProjectId": "eec4905f-98f8-40aa-9afc-4c1cfEXAMPLE",
        "TestType": "APPIUM_PYTHON",
        "TestSpec": "example-spec.yml"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "us-west-2"
},
```

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Arbeiten mit Testtypen in Device Farm — Dieses Referenzkapitel im Device Farm Developer Guide enthält eine genauere Beschreibung der von Device Farm unterstützten Test-Frameworks für Android, iOS und Webanwendungen.
- Aktionen in Device Farm — Die API-Aufrufe und Parameter in der Device Farm API-Referenz können Ihnen bei der Arbeit mit Device Farm Farm-Projekten helfen.
- Tutorial: Erstellen einer Pipeline, die Ihre Android-App mit erstellt und testet AWS Device Farm— Dieses Tutorial enthält eine Beispieldatei mit einer Build-Spezifikationsdatei und eine Beispielanwendung zum Erstellen einer Pipeline mit einer GitHub Quelle, die eine Android-App mit einer Device Farm erstellt CodeBuild und testet.

- Tutorial: Erstellen Sie eine Pipeline, die Ihre iOS-App testet mit AWS Device Farm— Dieses Tutorial enthält eine Beispielanwendung zum Erstellen einer Pipeline mit einer Amazon S3 S3-Quelle, die eine erstellte iOS-App mit Device Farm testet.

AWS Lambda

Ermöglicht es Ihnen, eine Lambda-Funktion als Aktion in Ihrer Pipeline auszuführen. Mit dem Ereignisobjekt, das eine Eingabe für diese Funktion ist, hat die Funktion Zugriff auf die Aktionskonfiguration, Speicherorte der Eingabeartefakte, Speicherorte der Ausgabeartefakte und andere Informationen, die für den Zugriff auf die Artefakte erforderlich sind. Ein Beispielereignis, das an eine Lambda-Aufruffunktion übergeben wurde, finden Sie unterJSON-Beispielereignis. Im Rahmen der Implementierung der Lambda-Funktion muss entweder `PutJobSuccessResult API` oder `PutJobFailureResult API` aufgerufen werden. Andernfalls hängt die Ausführung dieser Aktion, bis die Aktion abläuft. Wenn Sie Ausgabeartefakte für die Aktion angeben, müssen sie als Teil der Funktionsimplementierung in den S3-Bucket hochgeladen werden.

> **Important**
>
> Protokollieren Sie nicht das JSON-Ereignis, das CodePipeline an Lambda gesendet wird, da dies dazu führen kann, dass Benutzeranmeldeinformationen in CloudWatch Logs protokolliert werden. Die CodePipeline Rolle verwendet ein JSON-Ereignis, um temporäre Anmeldeinformationen an Lambda im`artifactCredentials` Feld zu übergeben. Ein Beispiel für ein Ereignis finden Sie unter JSON-Beispielereignis.

Aktionstyp

- Kategorie: `Invoke`
- Eigentümer: `AWS`
- Anbieter: `Lambda`
- Version: `1`

Konfigurationsparameter

FunctionName

Erforderlich: Ja

`FunctionName`ist der Name der in Lambda erstellten Funktion.

UserParameters

Required: No

Eine Zeichenfolge, die als Eingabe durch die Lambda-Funktion 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: Die Menge 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-Anfrage enthalten sind.

Weitere Informationen zu Variablen in CodePipeline finden Sie unterVariablen.

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 Job-Worker 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`: DieAWS Konto-ID, die dem Job zugeordnet ist.
- `data`: Sonstige Informationen, die für einen Auftragsworker erforderlich sind, um den Auftrag abzuschließen.
  - `actionConfiguration`: Die Aktionsparameter für die Lambda-Aktion. Definitionen finden Sie unter Konfigurationsparameter .
  - `inputArtifacts`: Das für die Aktion bereitgestellte Artefakt.
    - `location`: Der Speicherort des Artefaktspeichers.
      - `s3Location`: Die Informationen zum Speicherort des Eingabeartefakts für die Aktion.
        - `bucketName`: Der Name des Pipeline-Artefaktspeichers für die Aktion (z. B. ein Amazon S3 S3-Bucket mit dem Namen codepipeline-us-east -2-1234567890).
        - `objectKey`: Der Name der Anwendung (z. B. `CodePipelineDemoApplication.zip`).
      - `type`: Der Typ des Artefakts am Speicherort. Derzeit ist `S3` der einzige gültige Artefakttyp.
    - `revision`: Die Revisions-ID des Artefakts. Je nach Objekttyp kann dies eine Commit-ID (GitHub) oder eine Revision-ID (Amazon Simple Storage Service) sein. Weitere Informationen finden Sie unter ArtifactRevision.
    - `name`: Der Name des zu bearbeitenden Artefakts, z. B. `MyApp`.
  - `outputArtifacts`: Die Ausgabe der Aktion.
    - `location`: Der Speicherort des Artefaktspeichers.
      - `s3Location`: Die Speicherortinformationen des Ausgabeartefakts für die Aktion.
        - `bucketName`: Der Name des Pipeline-Artefaktspeichers für die Aktion (z. B. ein Amazon S3 S3-Bucket mit dem Namen codepipeline-us-east -2-1234567890).
        - `objectKey`: Der Name der Anwendung (z. B. `CodePipelineDemoApplication.zip`).
      - `type`: Der Typ des Artefakts am Speicherort. Derzeit ist `S3` der einzige gültige Artefakttyp.
    - `revision`: Die Revisions-ID des Artefakts. Je nach Objekttyp kann dies eine Commit-ID (GitHub) oder eine Revision-ID (Amazon Simple Storage Service) sein. Weitere Informationen finden Sie unter ArtifactRevision.

    - `name`: Der Name der Ausgabe eines Artefakts, z. B. `MyApp`.
- `artifactCredentials`: DieAWS Sitzungsanmeldeinformationen, die für den Zugriff auf Eingabe- und Ausgabeartefakte im Amazon S3 S3-Bucket verwendet werden. Diese Anmeldeinformationen sind temporäre Anmeldeinformationen, die von AWS Security Token Service (AWS STS) ausgestellt werden.
    - `secretAccessKey`: Der geheime Zugriffsschlüssel für die Sitzung.
    - `sessionToken`: Das Token für die Sitzung.
    - `accessKeyId`: Der geheime Zugriffsschlüssel für die Sitzung.
- `continuationToken`: Ein von der Aktion generiertes Token. Zukünftige Aktionen verwenden dieses Token, um die laufende Instance der Aktion zu identifizieren. Wenn die Aktion abgeschlossen ist, sollte kein Fortsetzungs-Token übermittelt werden.
- `encryptionKey`: Der Verschlüsselungsschlüssel, der zum Verschlüsseln der Daten im Artefaktspeicher verwendet wird, etwa ein AWS KMS-Schlüssel. Wenn dies 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 CloudFormationBenutzerhandbuch — Weitere Informationen zu Lambda-Aktionen undAWS CloudFormation Artefakten für Pipelines finden Sie unter Verwenden von Parameter-Override-Funktionen mit CodePipeline Pipelines, Automatisieren der Bereitstellung von Lambda-basierten Anwendungen und AWS CloudFormationArtefakten.
- Aufrufen einer AWS Lambda-Funktion in einer Pipeline in CodePipeline— Dieses Verfahren bietet eine Lambda-Beispielfunktion und zeigt Ihnen, wie Sie die Konsole verwenden, um eine Pipeline mit einer Lambda-Aufrufaktion zu erstellen.

Referenz zur Snyk-Aktionsstruktur

Die Snyk-Aktion in CodePipeline automatisiert die Erkennung und Behebung von Sicherheitslücken in Ihrem Open-Source-Code. Du kannst Snyk mit Anwendungsquellcode in deinem Drittanbieter-Repository wie Bitbucket Cloud GitHub oder mit Bildern für Containeranwendungen verwenden. Deine Aktion scannt und meldet die von dir konfigurierten Sicherheitslücken und Warnmeldungen.

> **Note**

Themen

- ID des Aktionstyps
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Weitere Informationen finden Sie auch unter

ID des Aktionstyps

- Kategorie: `Invoke`
- Eigentümer: `ThirdParty`
- Anbieter: `Snyk`
- Version: `1`

Beispiel:

```
{
    "Category": "Invoke",
    "Owner": "ThirdParty",
    "Provider": "Snyk",
    "Version": "1"
},
```

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: 1
- Beschreibung: Die Dateien, aus denen das Eingabeartefakt für die Aufrufaktion besteht.

Ausgabeartefakte

- Anzahl der Artefakte: 1
- Beschreibung: Die Dateien, aus denen das Ausgabeartefakt für die Aufrufaktion besteht.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- Weitere Informationen zur Verwendung von Snyk-Aktionen in CodePipeline finden Sie unter Automatisieren Sie das Scannen von Sicherheitslücken mit Snyk. CodePipeline

AWS Step Functions

Eine AWS CodePipeline-Aktion, die Folgendes ausführt:

- Startet die Ausführung eines AWS Step Functions-Zustandsautomaten über Ihre Pipeline.
- 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**
>
> In den Regionen Asien-Pazifik (Hongkong) und Europa (Mailand) ist diese Funktion nicht verfügbar. Weitere Informationen zu anderen verfügbaren Aktionen finden Sie unterProdukt- und Serviceintegrationen mit CodePipelineaus.

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(default): Wenn angegeben, ist der Wert in derEingabeDas Feld wird direkt an die Eingabe des Zustandsautomaten übergeben.

  Beispieleintrag für das Feld Input (Eingabe), wenn Literal ausgewählt ist:

  ```
  {"action": "test"}
  ```

- FilePath: Der Inhalt einer Datei in dem Eingastartefakt, das imEingabeDas Feld wird als Eingabe für die Ausführung des Zustandsautomaten verwendet. Ein Eingabeartefakt ist erforderlich, wenn InputType auf FilePath festgelegt ist.

  Beispieleintrag für das Feld Input (Eingabe) wenn FilePath ausgewählt ist:

  ```
  assets/input.json
  ```

Eingabe

Erforderlich Bedingt

- Literal: WannInputTypewird auf gesetztLiteral(Standard) 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: WannInputTypewird auf gesetztFilePathist dieses Feld ein Pflichtfeld.

  Ein Eingabe-Artefakt ist auch erforderlich, wenn InputType auf FilePathfestgelegt ist.

  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 WennInputTypewird auf gesetztFilePath, ist dieses Artefakt erforderlich und wird verwendet, um die Eingabe für die Ausführung des Zustandsautomaten zu erhalten.

Ausgabeartefakte

- Anzahl der Artefakte `0 to 1`
- Beschreibung
  - Standard-Zustandsautomaten: Wenn angegeben, wird das Ausgabe-Artefakt mit der Ausgabe des Zustandsautomaten gefüllt. Dies wird von der`output`Eigentum derSchrittfunktionen DescribeExecution API-Antwort, nachdem die Ausführung des Zustandsautomaten erfolgreich abgeschlossen wurde.

- Express-Zustandsautomaten: Nicht unterstützt

Ausgabevariablen

Durch diese Aktion werden Ausgabevariablen erzeugt, die von der Aktionskonfiguration einer nachgeschalteten Aktion in der Pipeline referenziert werden können.

Weitere Informationen finden Sie unter Variablen.

StateMachineArn

Der ARN des Zustandsautomaten.

ExecutionArn

Der ARN der Ausführung des Zustandsautomaten. Nur Standard-Zustandsautomaten.

Beispielaktionskonfiguration

Beispiel für Standardeingabe

YAML

```
Name: ActionName
ActionTypeId:
  Category: Invoke
  Owner: AWS
  Version: 1
  Provider: StepFunctions
OutputArtifacts:
  - Name: myOutputArtifact
Configuration:
  StateMachineArn: arn:aws:states:us-east-1:111122223333:stateMachine:HelloWorld-
StateMachine
  ExecutionNamePrefix: my-prefix
```

JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Invoke",
```

```
        "Owner": "AWS",
        "Version": 1,
        "Provider": "StepFunctions"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "StateMachineArn": "arn:aws:states:us-
east-1:111122223333:stateMachine:HelloWorld-StateMachine",
        "ExecutionNamePrefix": "my-prefix"
    }
}
```

Beispiel für Literaleingabe

YAML

```
Name: ActionName
ActionTypeId:
  Category: Invoke
  Owner: AWS
  Version: 1
  Provider: StepFunctions
OutputArtifacts:
  - Name: myOutputArtifact
Configuration:
  StateMachineArn: arn:aws:states:us-east-1:111122223333:stateMachine:HelloWorld-
StateMachine
  ExecutionNamePrefix: my-prefix
  Input: '{"action": "test"}'
```

JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Invoke",
        "Owner": "AWS",
        "Version": 1,
```

```
        "Provider": "StepFunctions"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "StateMachineArn": "arn:aws:states:us-
east-1:111122223333:stateMachine:HelloWorld-StateMachine",
        "ExecutionNamePrefix": "my-prefix",
        "Input": "{\"action\": \"test\"}"
    }
}
```

Beispiel für Eingabedatei

YAML

```
Name: ActionName
InputArtifacts:
  - Name: myInputArtifact
ActionTypeId:
  Category: Invoke
  Owner: AWS
  Version: 1
  Provider: StepFunctions
OutputArtifacts:
  - Name: myOutputArtifact
Configuration:
  StateMachineArn: 'arn:aws:states:us-east-1:111122223333:stateMachine:HelloWorld-
StateMachine'
  ExecutionNamePrefix: my-prefix
  InputType: FilePath
  Input: assets/input.json
```

JSON

```
{
    "Name": "ActionName",
    "InputArtifacts": [
        {
```

```
            "Name": "myInputArtifact"
        }
    ],
    "ActionTypeId": {
        "Category": "Invoke",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "StepFunctions"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "StateMachineArn": "arn:aws:states:us-
east-1:111122223333:stateMachine:HelloWorld-StateMachine",
        "ExecutionNamePrefix": "my-prefix",
        "InputType": "FilePath",
        "Input": "assets/input.json"
    }
}
```

Behavior

Während einer Veröffentlichung CodePipeline führt den konfigurierten Zustandsautomaten unter Verwendung der Eingabe aus, die in der Aktionskonfiguration angegeben ist.

Wenn InputType auf Literal, festgelegt ist, wird der Inhalt des Konfigurationsfelds der Aktion, Input (Eingabe), als Eingabe für den Zustandsautomaten verwendet. Wenn keine Literaleingabe bereitgestellt wird, wird für die Ausführung des Zustandsautomaten ein leeres JSON-Objekt ({}) verwendet. Weitere Informationen zum Ausführen eines Zustandsautomaten ohne Eingabe finden Sie in der Step Functions-StartExecution API.

Wenn InputType auf FilePath festgelegt ist, entpackt die Aktion das Eingabeartefakt und verwendet den Inhalt der Datei, die im Konfigurationsfeld der Aktion, Input (Eingabe), angegeben ist, als Eingabe für den Zustandsautomaten. Wenn FilePath angegeben wird, ist das Feld Input (Eingabe) 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 des Standard-Zustandsautomaten erfolgreich gestartet wurde, CodePipeline fragt`DescribeExecution`-API, 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 von der`output`Eigentum derSchrittfunktionen DescribeExecution API-Antwort, nachdem die Ausführung des Zustandsautomaten erfolgreich abgeschlossen wurde. Beachten Sie, dass für diese API Einschränkungen für die Ausgabelänge zwingend gelten.

Fehlerbehandlung

- Wenn der Start einer Ausführung für einen Zustandsautomaten fehlschlägt, schlägt die Ausführung der Aktion ebenfalls fehl.
- Wenn die Ausführung des Zustandsautomaten nicht beendet ist CodePipeline Step Functions Functions-Aktion erreicht ihren Timeout (Standard 7 Tage), die Ausführung der Aktion schlägt fehl. Der Zustandsautomat läuft möglicherweise trotz dieses Fehlers weiter. Weitere Informationen zu Timeouts für die Ausführung von Zustandsautomaten in Step Functions-Standard-Workflows gegen Express-Workflowsaus.

  > **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 in CodePipeline gestartet wurde, ist die Ausführung der Aktion auch dann erfolgreich, wenn die Ausführung des Zustandsautomaten fehlschlägt.

Fehlerbehandlung

- Wenn CodePipeline Starten der Ausführung eines Zustandsautomaten nicht, die Ausführung der Aktion schlägt fehl. Andernfalls gilt die Aktion sofort ebenfalls als erfolgreich ausgeführt. Die Aktion gelingt CodePipeline unabhängig davon, wie lange die Ausführung des Zustandsautomaten dauert oder deren Ergebnis dauert.

Weitere Informationen finden Sie auch unter

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit dieser Aktion.

- AWS Step FunctionsEntwicklerhandbuch— Weitere Informationen zu Zustandsautomaten sowie deren Ausführung und Eingaben finden Sie imAWS Step FunctionsEntwicklerhandbuchaus.
- Tutorial: Verwenden einerAWS Step FunctionsAufrufen einer Aktion in einer Pipeline- In diesem Tutorial beginnen Sie mit einem Beispiel-Standard-Zustandsautomaten. Das Tutorial veranschaulicht, wie Sie mithilfe der Konsole eine Pipeline aktualisieren, indem Sie eine - Aufrufaktion Step Functions Functions-Aufrufen verwenden.

Integration-Subtyreferenz

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.

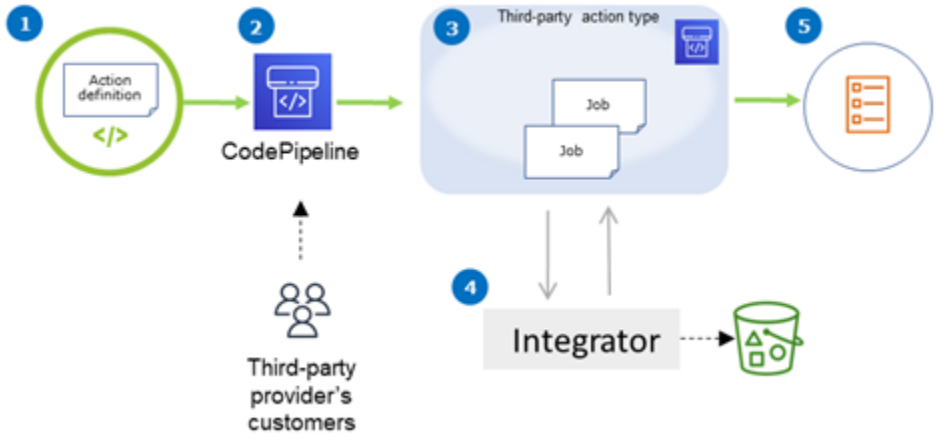
Um Ihren Aktionstyp eines Drittanbieters als Partnerintegration mit zu zertifizieren CodePipeline, verweisen Sie auf dasAWS Partnernetzwerk (APN). Diese Information ist eine Ergänzung zur AWS CLIReferenz.

Themen

- So funktionieren Aktionstypen von Drittanbietern mit dem Integrator
- Konzepte
- Unterstützte Integrationmodelle
- Lambda-Integrationmodell
- Modell zur Integration von Arbeitnehmern

So funktionieren Aktionstypen von Drittanbietern mit dem Integrator

Sie können den Kunden-Pipelines Aktionstypen von Drittanbietern hinzufügen, um Aufgaben in Kundenressourcen zu erledigen. Der Integrator verwaltet Jobanfragen und führt die Aktion mit aus CodePipeline. Das folgende Diagramm zeigt einen Aktionstyp eines Drittanbieters, der für Kunden zur Verwendung in ihrer Pipeline erstellt wurde. Nachdem der Kunde die Aktion konfiguriert hat, wird die Aktion ausgeführt und erstellt Jobanfragen, die von der Action Engine des Integrators verarbeitet werden.



Das Diagramm zeigt die folgenden Schritte:

1. Die Aktionsdefinition ist registriert und verfügbar in CodePipeline. Die Aktion „Drittanbieter“ steht Kunden des Drittanbieters zur Verfügung.
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 Jobanfrage gesendet.
4. Der Integrator (der Job Worker für Abfrage-APIs von Drittanbietern oder die Lambda-Funktion) nimmt die Jobanfrage entgegen, gibt eine Bestätigung zurück und arbeitet an den Artefakten für die Aktionen.
5. Der Integrator gibt die Erfolgs-/Fehlerausgabe zurück (der Job-Worker verwendet Erfolgs-/Fehler-APIs oder die Lambda-Funktion sendet eine Erfolgs-/Fehlerausgabe) mit einem Auftragsergebnis und einem Fortführungstoken.

Informationen zu den Schritten, die Sie verwenden können, um einen Aktionstyp anzufordern, anzuzeigen und zu aktualisieren, finden Sie unterArbeiten mit Aktionstypen.

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 Workloads für die kontinuierliche Bereitstellung ausführen. Aktionstypen werden durch ein`OwnerCategoryProvider`, und gekennzeichnet`Version`. Beispiel:

```
        {

            "Category": "Deploy",
            "Owner": "AWS",
            "Provider": "CodeDeploy",
            "Version": "1"
        },
```

Alle Aktionen desselben Typs haben dieselbe Implementierung.

Action

Eine einzelne Instanz eines Aktionstyps, eines 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.

Aktionsdefinition

Das Schema für einen Aktionstyp, das die Eigenschaften definiert, die für die Konfiguration der Aktion und der Eingabe-/Ausgabeartefakte erforderlich sind.

Aktionsausführung

Eine Sammlung von Jobs, 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 von einem Aktionstyp verwendeten Integrationstyp definiert. Gültige Werte sind `JobWorker` und `Lambda`.

Integration

Beschreibt eine Software, die von einem Integrator ausgeführt wird, um einen Aktionstyp zu implementieren. CodePipeline unterstützt zwei Integrationstypen, die den beiden unterstützten Action Engines entsprechen`JobWorker` und`Lambda`.

Integrator

Die Person, die für die Implementierung eines Aktionstyps verantwortlich ist.

Aufgabe

Eine Arbeit mit Pipeline und Kundenkontext zur Ausführung einer Integration. Eine Aktionsausführung besteht aus einem oder mehreren Aufträgen.

Arbeitsarbeiter

Der Dienst, der die Kundeneingaben verarbeitet und einen Job ausführt.

Unterstützte Integrationmodelle

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 Auftragsanforderungen zu verarbeiten, wenn Ihre Aktion ausgeführt wird.
- Integrationsmodell für Arbeitnehmer: Das Integrationsmodell für Job-Worker ist das zuvor verwendete Modell für Integrationen von Drittanbietern. Das Job-Worker-Integrationsmodell verwendet einen Job Worker, der so konfiguriert ist, dass er die CodePipeline APIs kontaktiert, um Jobanfragen zu verarbeiten, wenn Ihre Aktion ausgeführt wird.

Zum Vergleich werden in der folgenden Tabelle die Funktionen der beiden Modelle beschrieben:

	Lambda-Integrationsmodell	Modell zur Integration von Arbeitnehmern
Beschreibung	Der Integrator schreibt die Integration als Lambda-Funktion, die von CodePipeline immer dann aufgerufen wird, wenn ein Job für die Aktion verfügbar ist. Die Lambda-Funktion fragt nicht nach verfügbaren Jobs ab, sondern wartet, bis die nächste Jobanfrage eingeht.	Der Integrator schreibt die Integration als Jobworker, der ständig nach verfügbaren Jobs in den Pipelines des Kunden fragt. Der Jobworker führt dann den Job aus und sendet das Auftragsergebnis mithilfe CodePipeline von CodePipeline APIs zurück.
Infrastruktur	AWS Lambda	Stellen Sie Job-Worker-Code für die Infrastruktur des Integrators bereit, z. B. Amazon EC2 EC2-Instances.

	Lambda-Integrationsmodell	Modell zur Integration von Arbeitnehmern
Entwicklungsaufwand	Die Integration enthält nur die Geschäftslogik.	Die Integration muss zusätzlich zur Geschäftslogik mit CodePipeline APIs interagieren.
Operativer Aufwand	Geringerer Betriebsaufwand, da Infrastruktur nur ausAWS Ressourcen besteht.	Höherer Betriebsaufwand, da der Mitarbeiter seine eigenständige Hardware benötigt.
Max. Auftragslaufzeit	Wenn die Integration länger als 15 Minuten aktiv laufen 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 am Ende ein Ergebnis zurückgeben müssen. Wir empfehlen dem Integrator nicht, weiter auf den Abschluss des Builds zu warten. Geben Sie stattdessen eine Fortsetzung zurück. CodePipelineerstellt in weiteren 30 Sekunden einen neuen Job, wenn der Code des Integrators eine Fortsetzung erhält, um den Job zu überprüfen, bis er abgeschlossen ist.	Mit diesem Modell können Aufträge mit sehr langer Laufzeit (Stunden/Tage) aufrechterhalten werden.

Lambda-Integrationmodell

Das unterstützte Lambda-Integrationsmodell umfasst das Erstellen der Lambda-Funktion und das Definieren 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 ausgeführt wird, wenn in Ihrer Pipeline ein Job für Ihren Aktionstyp verfügbar ist. In Anbetracht des Kundenkontextes und der Pipeline möchten Sie beispielsweise mit der Erweiterung Ihres Services für den Kunden beginnen.

Verwenden Sie die folgenden Parameter, um Ihre Lambda-Funktion zu aktualisieren, um die Eingabe von zu verarbeiten CodePipeline.

Format:

- `jobId`:
  - Die eindeutige, systemgenerierte ID des Auftrags.
  - 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 desAWS Kundenkontos, das bei der Ausführung des Auftrags verwendet werden soll.
  - Typ: Zeichenfolge
  - Muster: [0-9] {12}
- `data`:
  - Weitere Informationen zu einem Job, die eine Integration verwendet, um den Job abzuschließen.
  - Enthält eine Karte der folgenden Elemente:
    - `actionConfiguration`:
      - Die Konfigurationsdaten für die Aktion. Die Aktionskonfigurationsfelder sind eine Zuordnung von Schlüssel-Wert-Paaren, in die Ihr Kunde Werte eingeben kann. Die Schlüssel werden durch die Schlüsselparameter in der Definitionsdatei für den Aktionstyp 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: Abbildung einer Zeichenfolge auf eine Zeichenfolge

        Beispiel:

```
"configuration": {
```

```
        "Username": "MyUser",
        "Password": "MyPassword"
    },
```

- `encryptionKey`:
  - Stellt Informationen zum Verschlüsseln von Daten im Artefaktspeicher dar, z. B. einAWS KMS Schlüssel.
  - Inhalt: Typ des Datentyps`encryptionKey`, optional vorhanden
- `inputArtifacts`:
  - Liste mit Informationen zu einem Artefakt, das die Bearbeitung, z. B. ein Test- oder Build-Artefakt.
  - Inhalt: Liste des Datentyps`Artifact`, optional vorhanden
- `outputArtifacts`:
  - Liste der Informationen zum Ergebnis einer Aktion.
  - Inhalt: Liste des Datentyps`Artifact`, optional vorhanden
- `actionCredentials`:
  - Stellt ein Objekt mitAWS Sitzungsanmeldeinformationen dar. Bei diesen Anmeldeinformationen handelt es sich um temporäre Anmeldeinformationen, die von ausgestellt werdenAWS STS. Sie können verwendet werden, um auf Eingabe- und Ausgabeartefakte im S3-Bucket zuzugreifen, in dem Artefakte für die Pipeline gespeichert werden CodePipeline.

    Diese Anmeldeinformationen haben auch dieselben Berechtigungen wie die angegebene Vorlage für Richtlinienanweisungen in der Definitionsdatei des Aktionstyps.
  - 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. einAWS KMS Schlüssel.
    - Typ: Zeichenfolge
    - Zulässige Werte: `KMS`
- `Artifact`:
  - `name`:
    - Der Name des Artefakts.
    - Typ: Zeichenfolge, optional vorhanden
  - `revision`:
    - Die Revisions-ID des Artefakts. Je nach Objekttyp kann dies eine Commit-ID (GitHub) oder eine Revision-ID (Amazon S3) sein.
    - Typ: Zeichenfolge, optional vorhanden
  - `location`:
    - Der Standort eines Artefakts.
    - Inhalt: Typ des Datentyps`ArtifactLocation`, optional vorhanden
- `ArtifactLocation`:
  - `type`:
    - Die Art des Artefakts an dem Ort.
    - Typ: Zeichenfolge, optional vorhanden
    - Zulässige Werte: `S3`
  - `s3Location`:
    - Der Standort 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 in dem 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 aus der Lambda-Funktion zurück CodePipeline

Die Jobworker-Ressource des Integrators muss bei Erfolg, Misserfolg oder Fortsetzung eine gültige Payload zurückgeben.

Format:

- `result`: Das Ergebnis des Jobs.
  - Erforderlich
  - Gültige Werte (Groß- und Kleinschreibung wird nicht beachtet):
    - `Success`: Zeigt an, dass ein Job erfolgreich und beendet ist.
    - `Continue`: Zeigt an, dass ein Job erfolgreich ist und fortgesetzt werden muss, z. B. wenn der Job Worker erneut aufgerufen wird, um dieselbe Aktion auszuführen.
    - `Fail`: Zeigt an, dass ein Auftrag fehlgeschlagen und beendet wurde.

- `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 zurückgegeben wird CodePipeline.

  - Optional. Erforderlich, wenn das Ergebnis vorliegt`Fail`.
  - Muss null sein, wenn`result` ist`Success` oder`Continue`
  - Zulässige Werte:
    - ConfigurationError
    - JobFailed
    - PermissionsError
    - RevisionOutOfSync
    - RevisionUnavailable
    - SystemUnavailable
- `continuation`: Fortsetzungsstatus, der innerhalb der aktuellen Aktionsausführung an den nächsten Job übergeben werden soll.
  - Optional. Erforderlich, wenn das Ergebnis vorliegt`Continue`.
  - Muss null sein, wenn`result` es`Success` oder ist`Fail`.
  - Eigenschaften:
    - `State`: Ein Hash des Staates, 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 zur Fehlermeldung, die der Benutzer sieht.
- `outputVariables`: Eine Reihe von Schlüssel/Wert-Paaren, die an die nächste Aktionsausführung übergeben werden.
  - Optional.
  - Muss null sein, wenn`result` es`Continue` oder 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 Beschäftigungsstatus zu übergeben CodePipeline und darauf hinzuweisen, dass die Tätigkeit fortgesetzt werden muss. Nachdem ein Integrator beispielsweise einen Build für den Kunden auf seiner Ressource gestartet hat, wartet er nicht darauf, dass der Build abgeschlossen ist, sondern gibt an, CodePipeline dass er kein terminales Ergebnis hat, indem er das`result` as`continue` und die eindeutige ID des Builds an CodePipeline as`continuation` token zurückgibt.

> **Note**
>
> Lambda-Funktionen können nur bis zu 15 Minuten ausgeführt werden. Wenn der Job länger laufen muss, können Sie Continuation Tokens verwenden.

Das CodePipeline Team ruft den Integrator nach 30 Sekunden mit demselben`continuation` Token in seiner Payload auf, damit er überprüfen kann, ob er abgeschlossen ist. Wenn der Build abgeschlossen ist, gibt der Integrator das Ergebnis „Erfolgreich/Fehlgeschlagen“ im Terminal zurück, andernfalls wird der Vorgang fortgesetzt.

Geben Sie CodePipeline die Berechtigungen zum Aufrufen der Integrator-Lambda-Funktion zur Laufzeit an

Sie fügen Ihrer Integrator-Lambda-Funktion Berechtigungen hinzu, um dem CodePipeline Dienst Berechtigungen zu geben, ihn mithilfe des CodePipeline Dienstprinzips aufzurufen:`codepipeline.amazonaws.com`. Sie können Berechtigungen mithilfe derAWS CloudFormation oder der Befehlszeile hinzufügen. Ein Beispiel finden Sie unter Arbeiten mit Aktionstypen.

Modell zur Integration von Arbeitnehmern

Nachdem Sie Ihren übergeordneten Arbeitsablauf entworfen haben, können Sie Ihren Job Worker erstellen. Obwohl die Besonderheiten der Drittanbieteraktion festlegen, was für den Jobworker erforderlich ist, verfügen die meisten Jobworker für Aktionen Dritter über die folgenden Funktionen:

- Umfragen für Jobs aus der CodePipeline Nutzung`PollForThirdPartyJobs`.
- Anerkennung von Aufträgen und Rückgabe der Ergebnisse an die CodePipeline Verwendung von`AcknowledgeThirdPartyJob`,`PutThirdPartyJobSuccessResult`, und`PutThirdPartyJobFailureResult`.
- Abrufen von Artefakten aus dem Amazon S3-Bucket und/oder Einfügen von Artefakten in den Amazon S3 S3-Bucket für die Pipeline. Um Artefakte aus dem Amazon S3 S3-Bucket herunterzuladen, müssen Sie einen Amazon S3 S3-Client erstellen, der Signature Version 4-Signatur (Sig V4) verwendet. Sig V4 ist erforderlich fürAWS KMS.

  Um Artefakte in den Amazon S3 S3-Bucket hochzuladen, müssen Sie auch die Amazon S3`PutObject` S3-Anfrage so konfigurieren, dass sie die Verschlüsselung überAWS Key Management Service (AWS KMS) verwendet. AWS KMSNutzungenAWS KMS keys. Um zu wissen, ob der vom Kunden verwaltete SchlüsselVon 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 verwendenAWS KMS. Wenn die Schlüsseleigenschaft Null ist, verwenden Sie dieVon AWS verwalteter Schlüssel. CodePipeline verwendet die,Von AWS verwalteter Schlüssel sofern nicht anders konfiguriert.

  Ein Beispiel, das zeigt, wie dieAWS KMS Parameter in Java oder.NET erstellt werden, finden Sie unter Angeben derAWS Key Management Service in Amazon S3 mithilfe derAWS SDKs.

Weitere Informationen zum Amazon S3 S3-Bucket für CodePipeline finden Sie unterCodePipeline - Konzepte.

Wählen und Konfigurieren einer Strategie für die Berechtigungsverwaltung Ihres Auftragsworkers

Um einen Job Worker für Ihre Drittanbieteraktivitäten zu entwickeln CodePipeline, benötigen Sie eine Strategie für die Integration von Benutzer- und Rechteverwaltung.

Die einfachste Strategie besteht darin, die Infrastruktur hinzuzufügen, die Sie für Ihren Job Worker benötigen, indem Sie Amazon EC2 EC2-Instances mit einerAWS Identity and Access Management (IAM) -Instance-Rolle erstellen, sodass Sie die Ressourcen, die Sie für Ihre Integration benötigen, einfach skalieren können. Sie können die integrierte Integration mit verwendenAWS, um die Interaktion zwischen Ihrem Jobworker 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 Identitätsanbietersystem und Ihre Ressourcen zu integrieren. Diese Strategie ist nützlich, wenn Sie bereits über einen Corporate Identity Provider verfügen oder bereits so konfiguriert sind, dass sie Benutzer unterstützen, die Web Identity Provider verwenden. Mit dem Identitätsverbund können Sie sicheren Zugriff aufAWS Ressourcen gewähren CodePipeline, auch ohne IAM-Benutzer erstellen oder verwalten zu müssen. Sie können Funktionen und Richtlinien für die Sicherheitsanforderungen bezüglich Passwörtern und die Rotation von Anmeldeinformationen nutzen. Sie können Beispielanwendungen als Vorlage für Ihre selbsterstellten Anwendungen verwenden. Weitere Informationen finden Sie unter Manage Federation.

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

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center-Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

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

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

Im Folgenden finden Sie ein Beispiel für eine Richtlinie, die Sie für die Verwendung mit Ihrem externen Jobworker erstellen können. Diese Richtlinie dient lediglich als Beispiel und wird unverändert bereitgestellt.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:PollForThirdPartyJobs",
        "codepipeline:AcknowledgeThirdPartyJob",
        "codepipeline:GetThirdPartyJobDetails",
        "codepipeline:PutThirdPartyJobSuccessResult",
        "codepipeline:PutThirdPartyJobFailureResult"
      ],
      "Resource": [
        "arn:aws:codepipeline:us-east-2::actionType:ThirdParty/Build/Provider/1/"
      ]
    }
  ]
}
```

Referenz zu Abbild-Definitionsdateien

Dieser Abschnitt dient nur als Referenz. Informationen zum Erstellen einer Pipeline mit Quell- oder Bereitstellungsaktionen für Container finden Sie unter Erstellen einer Pipeline in CodePipeline.

AWS CodePipeline-Aufgaben-Worker für Container-Aktionen wie -Amazon ECR-Quellaktionen oder -Bereitstellungsaktionen verwenden Definitionsdateien, um den Abbild-URI und Container-Namen der Aufgabendefinition zuzuordnen. Jede Definitionsdatei ist eine JSON-formatierte Datei, die vom Aktionsanbieter wie folgt verwendet wird:

- Amazon ECS-Standardbereitstellungen erfordern eine`imagedefinitions.json`-Datei als Eingabe für die Bereitstellungsaktion.
- Amazon ECS-Blau/Grün-Bereitstellungen erfordern eine`imageDetail.json`-Datei als Eingabe für die Bereitstellungsaktion.
  - Amazon ECR-Quellaktionen generieren eine `imageDetail.json`-Datei, die als Ausgabe aus der Quellaktion bereitgestellt wird.

Themen

- imagedefinitions.json-Datei für Amazon ECS-SStandardbereitstellungsaktionen
- imageDetail.json-Datei für Amazon ECS-Blau/Grün-Bereitstellungsaktionen

imagedefinitions.json-Datei für Amazon ECS-SStandardbereitstellungsaktionen

Ein Abbild-Definitionsdokument ist eine JSON-Datei, die den Namen Ihres Amazon ECSContainers sowie Abbild und Tag beschreibt. Wenn Sie Container-basierte Anwendungen bereitstellen, müssen Sie eine Abbild-Definitionsdatei erstellen, um die CodePipeline Arbeitsmitarbeiter mit dem Amazon ECS-Container und der Bildidentifikation zum Abrufen aus dem Bild-Repository, 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 UTF-8-Kodierung verwenden.
- Die maximale Dateigröße für die Abbild-Definitionsdatei ist 100 KB.
- Sie müssen die Datei als Quell- oder Build-Artefakt erstellen, damit sie ein Eingabeartefakt für die Bereitstellungsaktion ist. Sie stellen also sicher, dass die Datei entweder an Ihren Quellspeicherort hochgeladen wird, z. B. CodeCommit -Repository oder als erstelltes Ausgabeartefakt generiert.

Die Datei `imagedefinitions.json` stellt den Containernamen und den Abbild-URI bereit. Dieser muss mit dem folgenden Satz von Schlüssel-Wert-Paaren konstruiert werden.

Schlüssel	Wert
Name	*container_name*
imageUri	*imageUri*

Dies ist die JSON-Struktur, wobei der Containername `sample-app`, der Abbild-URI `ecs-repo` und der Tag `latest` ist:

```
[
  {
    "name": "sample-app",
    "imageUri": "11111EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest"
  }
]
```

Sie können die Datei auch so konstruieren, dass mehrere Container-Abbild-Paare aufgelistet werden.

JSON-Struktur:

```
[
  {
    "name": "simple-app",
    "imageUri": "httpd:2.4"
  },
  {
    "name": "simple-app-1",
```

```
    "imageUri": "mysql"
  },
  {
    "name": "simple-app-2",
    "imageUri": "java1.8"
  }
]
```

Führen Sie vor Erstellung Ihrer Pipeline die die folgenden Schritte aus, um die `imagedefinitions.json` einzurichten.

1. Planen Sie während der Planung der Bereitstellung Container-basierter Anwendungen für Ihre Pipeline die Quell- und die Build-Phase, sofern vorhanden.
2. Wählen Sie eine der folgenden Optionen aus:

   a. Wenn Ihre Pipeline so erstellt wird, dass die Build-Phase übersprungen wird, müssen Sie die JSON-Datei manuell erstellen und in Ihr Quell-Repository laden, 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 Ihr Quell-Repository ein Amazon S3 S3-Bucket ist, denken Sie daran, die JSON-Datei komprimieren.

   b. Wenn Ihre Pipeline eine Build-Phase enthält, fügen Sie Ihrer Build Spec-Datei einen Befehl hinzu, der die Abbild-Definitionsdatei während der Build-Phase in Ihr Quell-Repository ausgibt. Im folgenden Beispiel wird der Befehl printf verwendet, um eine `imagedefinitions.json`-Datei zu erstellen. Listen Sie diesen Befehl im Abschnitt `post_build` der Datei `buildspec.yml` auf:

   ```
   printf '[{"name":"container_name","imageUri":"image_URI"}]' >
   imagedefinitions.json
   ```

   Sie müssen die Abbild-Definitionsdatei als Ausgabeartefakt in die Datei `buildspec.yml` einfügen.

3. Wenn Sie Ihre Pipeline in der Konsole erstellen, geben Sie auf der Seite Deploy (Bereitstellen) des Assistenten Create Pipeline (Pipeline erstellen) in Image Filename (Name der Abbilddatei) den Namen der Abbild-Definitionsdatei ein.

Für ein step-by-step -Tutorial zum Erstellen einer Pipeline mit Amazon ECS als Bereitstellungsanbieter finden Sie unterTutorial: Kontinuierliche Bereitstellung mit CodePipeline.

imageDetail.json-Datei für Amazon ECS-Blau/Grün-Bereitstellungsaktionen

Importieren in &S3;`imageDetail.json`-Dokument ist eine JSON-Datei, die Ihren Amazon ECSAbbild-URI beschreibt. Wenn Sie Container-basierte Anwendungen für eine Blau/Grün-Bereitstellung bereitstellen, müssen Sie die`imageDetail.json`-Datei für die Bereitstellung des Amazon ECS-Systems und CodeDeployArbeitsarbeiter mit der Bild-Identifikation, die aus dem Bild-Repository abgerufen werden soll, z. B. Amazon ECR.

> **Note**
>
> Der Name der Datei muss `imageDetail.json` sein.

Eine Beschreibung der Aktion und ihrer Parameter finden Sie unterAmazon Elastic Container Service undCodeDeploy Blau-Grün-.

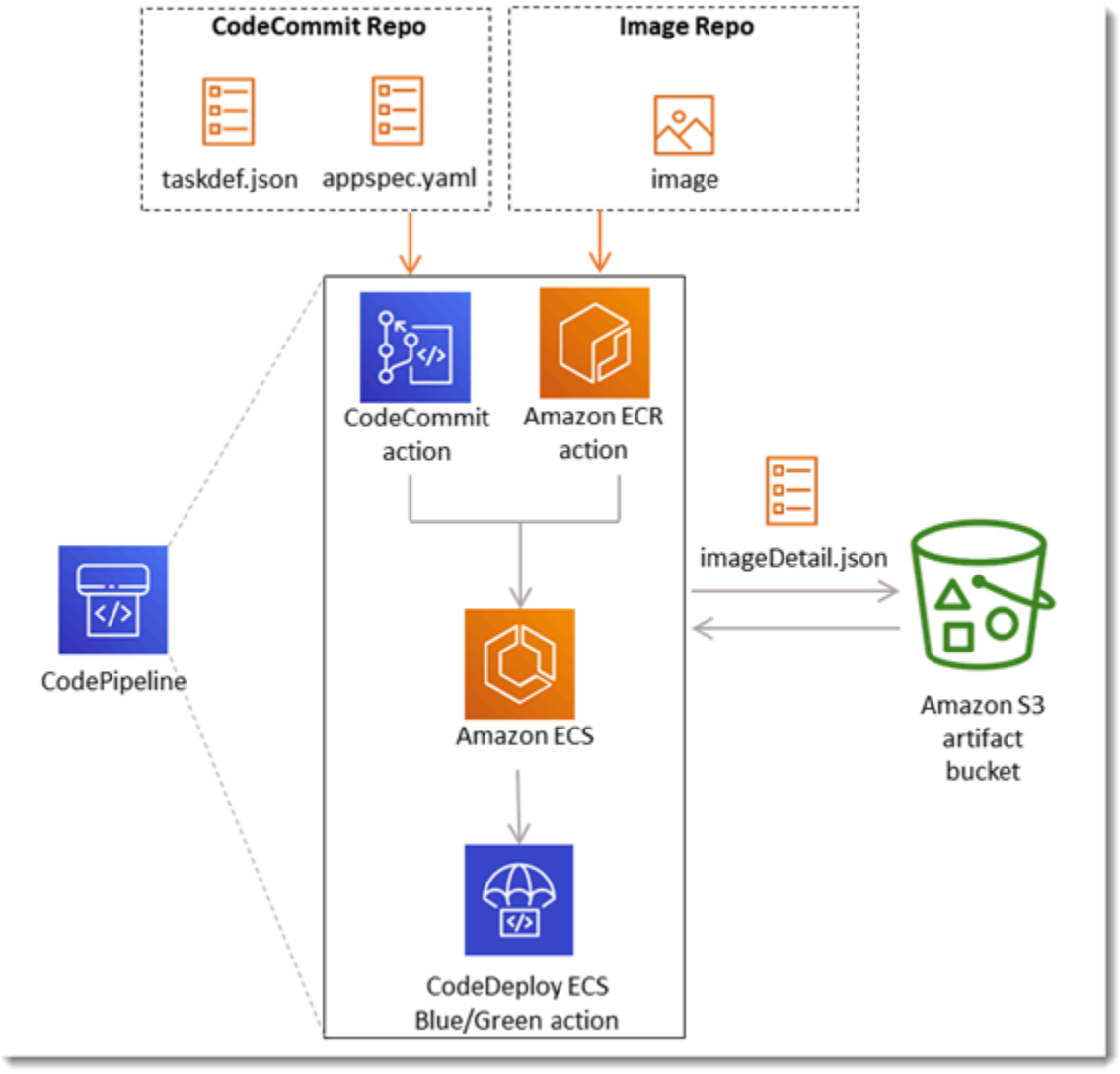
Sie müssen die Datei `imageDetail.json` als Quell- oder Build-Artefakt erstellen, damit sie ein Eingabeartefakt für die Bereitstellungsaktion ist. Sie können eine dieser Methoden verwenden, um die `imageDetail.json`-Datei in der Pipeline bereitzustellen:

- Schließen Sie das mit ein`imageDetail.json`-Datei an Ihrem Quellspeicherort, sodass sie in der Pipeline als Eingabe für Ihre Amazon ECS-Blau/Grün-Bereitstellungsaktion bereitgestellt wird.

  > **Note**
  >
  > Wenn Ihr Quell-Repository ein Amazon S3 S3-Bucket ist, denken Sie daran, die JSON-Datei komprimieren.

- Amazon ECR-Quellaktionen generieren automatisch ein`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 manuell`imageDetail.json`file.
> Ein Lernprogramm zum Erstellen einer Pipeline mit einem Amazon ECR-Quell-Stagingbereich finden Sie unter.Tutorial: Eine Pipeline mit einer Amazon ECR-Quelle und ECS-TO-Bereitstellung erstellen CodeDeploy .



Die `imageDetail.json`-Datei stellt den Abbild-URI bereit. Er muss mit dem Schlüssel-Wert-Paar konstruiert werden.

Schlüssel	Wert
ImageURI	*image_URI*

imageDetail.json

Dies ist die JSON-Struktur, wobei der Abbild-URI `ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-image-repo@sha256:example3` ist:

```
{
"ImageURI": "ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-image-
repo@sha256:example3"
}
```

imageDetail.json (generated by ECR)

Importieren in &S3;`imageDetail.json`-Datei wird automatisch jedes Mal von der Amazon ECR-Quellaktion generiert, wenn eine Änderung per Push an das Abbild-Repository übertragen wird. Die`imageDetail.json`-Quellaktionen, die von Amazon ECR-Quellaktionen generiert werden, werden als Ausgabe-Artefakt von der Quellaktion bis 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 Abbild-URI und den Container-Namen der Amazon ECS-Taskdefinition wie folgt zu:

- `ImageSizeInBytes`: Die Größe des Abbilds im Repository in Bytes.
- `ImageDigest`: Der `sha256`-Digest des Abbildmanifests.
- `Version`: Die Abbildversion.
- `ImagePushedAt`: Datum und Uhrzeit der Push-Übertragung des neuesten Abbilds an das Repository.
- `RegistryId`: DasAWS-Konto-ID, die der Registrierung zugeordnet ist, die das Repository enthält.
- `RepositoryName`: Der Name des Amazon ECR-Repositorys, zu dem das Image per Push übertragen wurde
- `ImageURI`: Der URI für das Abbild.
- `ImageTags`: Das Tag für das Abbild.

Führen Sie vor Erstellung Ihrer Pipeline die die folgenden Schritte aus, um die `imageDetail.json` einzurichten.

1. Planen Sie während der Planung der Container-basierten Blau/Grün-Anwendungsbereitstellung für Ihre Pipeline die Quell- und Build-Phase, wenn zutreffend.
2. Wählen Sie eine der folgenden Optionen aus:

   a. Wenn Ihre Pipeline die Build-Phase übersprungen hat, müssen Sie die JSON-Datei manuell erstelle und in Ihr Quell-Repository laden, z. B. in Ihr Quell-Repository laden. CodeCommit, sodass die Quellaktion das Artefakt bereitstellen kann. Erstellen Sie die Datei mit einem Texteditor und geben Sie der Datei einen Namen oder verwenden Sie den Standarddateinamen `imageDetail.json`. Übertragen Sie die `imageDetail.json`-Datei per Push zu Ihrem Quell-Repository.

   b. Wenn Ihre Pipeline eine Build-Phase enthält, führen Sie die folgenden Aktionen aus:

      i. Fügen Sie Ihrer Build-Spezifikationsdatei einen Befehl hinzu, der die Abbild-Definitionsdatei während der Build-Phase in Ihr Quell-Repository ausgibt. Im folgenden Beispiel wird der Befehl printf verwendet, um eine `imageDetail.json`-Datei zu erstellen. Listen Sie diesen Befehl im Abschnitt `post_build` der Datei buildspec.yml auf:

```
printf '{"ImageURI":"image_URI"}' > imageDetail.json
```

Sie müssen die Datei `imageDetail.json` als Ausgabeartefakt in die Datei `buildspec.yml` einfügen.

ii. Fügen Sie die Datei `imageDetail.json` als Artefakt zur Datei `buildspec.yml` hinzu.

```
artifacts:
  files:
    - imageDetail.json
```

Variablen

Dieser Abschnitt dient nur als Referenz. Informationen zum Erstellen von Variablen finden Sie unter Arbeiten mit Variablen.

Mit Variablen können Sie Ihre Pipeline-Aktionen mit Werten konfigurieren, die zum Zeitpunkt der Pipeline-Ausführung oder der Aktionsausführung bestimmt werden.

Einige Aktionsanbieter erzeugen einen definierten Satz von Variablen. Sie wählen aus Standardvariablenschlüsseln für diesen Aktionsanbieter aus, z. B. eine Commit-ID.

> **Important**
>
> Geben Sie 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 Secrets. Es wird dringend empfohlen, AWS Secrets Manager zum Speichern von Secrets zu verwenden.

Beispiele step-by-step für die Verwendung von Variablen finden Sie hier:

- 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.
- Ein Tutorial mit einer Lambda-Aktion, die Variablen aus einer Upstream-Aktion (CodeCommit) verwendet und Ausgabevariablen generiert, finden Sie unter Tutorial: Verwenden von Variablen mit Lambda-Aufrufaktionen.
- Ein Tutorial mit einer AWS CloudFormation Aktion, die auf Stack-Ausgabevariablen aus einer Upstream- CloudFormation Aktion verweist, finden Sie unter Tutorial: Erstellen einer Pipeline, die Variablen aus AWS CloudFormation-Bereitstellungsaktionen verwendet.
- Ein Beispiel für eine manuelle Genehmigungsaktion mit Nachrichtentext, der auf Ausgabevariablen verweist, die in die CodeCommit Commit-ID und Commit-Nachricht aufgelöst werden, finden Sie unter Beispiel: Variablen in manuellen Genehmigungen verwenden.
- Eine CodeBuild Beispielaktion mit einer Umgebungsvariablen, die in den GitHub Zweignamen aufgelöst wird, finden Sie unter Beispiel: 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 Aktionsausgabevariablen.

Variablengrenzwerte

Informationen zu Grenzwerten finden Sie unter Kontingente in AWS CodePipeline.

> **Note**
>
> Wenn Sie Variablensyntax in die Aktionskonfigurationsfelder eingeben, überschreiten Sie das Limit von 1000 Zeichen für die Konfigurationsfelder nicht. 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, wie diese Variablen verfügbar gemacht werden:

- 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 zur Laufzeit der Pipeline erstellt und 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 von einer Aktion erzeugten Variablen sehen, indem Sie das `outputVariables` Feld überprüfen, das Teil der ListActionExecutions API ist. Die Liste der verfügbaren Schlüsselnamen nach Aktionsanbieter finden Sie unter Für Pipeline-Aktionen verfügbare Variablen. Informationen dazu, welche Variablen jeder Aktionstyp erzeugt, finden Sie unter CodePipeline Referenz der Aktionsstruktur.

Um in Ihrer Aktionskonfiguration auf diese Variablen zu verweisen, müssen Sie die Variablenreferenzsyntax mit dem richtigen Namespace verwenden.

Ein Beispiel für einen Variablen-Workflow finden Sie unter Konfigurieren von Variablen .

Namespaces

Um sicherzustellen, dass Variablen eindeutig referenziert werden können, müssen sie einem Namespace zugewiesen werden. Nachdem Sie einen Satz von Variablen einem Namespace zugewiesen haben, können sie in einer Aktionskonfiguration durch Verwendung des Namespace und Variablenschlüssels mit der folgenden Syntax referenziert werden:

```
#{namespace.variable_key}
```

Es gibt drei Arten von Namespaces, 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 finden Sie einige der häufigsten Anwendungsfälle für Variablen auf Pipeline-Ebene, mit denen Sie feststellen können, wie Sie Variablen für Ihre spezifischen Anforderungen verwenden können.

- Variablen auf Pipeline-Ebene sind für CodePipeline Kunden gedacht, die jedes Mal dieselbe Pipeline mit geringfügigen Variationen in den Eingaben für die Aktionskonfiguration verwenden möchten. Jeder Entwickler, der eine Pipeline startet, fügt den Variablenwert in der Benutzeroberfläche hinzu, wenn die Pipeline gestartet wird. Mit dieser Konfiguration übergeben Sie Parameter nur 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, mit denen Sie eine Pipeline bei jeder Ausführung wiederverwenden können, z. B. wenn Sie angeben möchten, welche Version Sie in einer Produktionsumgebung bereitstellen möchten, sodass Sie Pipelines nicht duplizieren müssen.
- Sie können eine einzelne Pipeline verwenden, um Ressourcen in mehreren Build- und Bereitstellungsumgebungen bereitzustellen. Für eine Pipeline mit einem CodeCommit Repository kann beispielsweise die Bereitstellung von einer bestimmten Zweig- und Zielbereitstellungsumgebung aus mit den Parametern CodeBuild und CodeDeploy erfolgen, die auf Pipeline-Ebene übergeben werden.

Konfigurieren von Variablen

Sie können Variablen auf Pipeline-Ebene oder auf Aktionsebene in der Pipeline-Struktur konfigurieren.

Konfigurieren von Variablen auf Pipeline-Ebene

Sie können eine oder mehrere Variablen auf Pipeline-Ebene hinzufügen. Sie können diesen Wert in der Konfiguration von CodePipeline Aktionen referenzieren. Sie können die Variablennamen, Standardwerte und Beschreibungen beim Erstellen der Pipeline hinzufügen. Variablen werden zum Zeitpunkt der Ausführung aufgelöst.

> **Note**
>
> Wenn kein Standardwert für eine Variable auf Pipeline-Ebene 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 bereit, indem Sie das Variablenattribut in der Pipeline-Struktur verwenden. Im folgenden Beispiel `Variable1` hat die Variable den Wert `Value1`.

```
"variables": [
     {
         "name": "Variable1",
         "defaultValue": "Value1",
         "description": "description"
     }
 ]
```

Ein Beispiel in der Pipeline-JSON-Struktur finden Sie unter Erstellen einer Pipeline in CodePipeline.

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 jeder Art von Quellaktion nicht unterstützt wird.

> **Note**
>
> Wenn der `variables` Namespace bereits in einigen Aktionen innerhalb der Pipeline verwendet wird, müssen Sie die Aktionsdefinition aktualisieren und einen anderen Namespace für die widersprüchliche Aktion auswählen.

Konfigurieren von Variablen auf Aktionsebene

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 mit dem Namen zur CodeCommit Quellaktion 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.

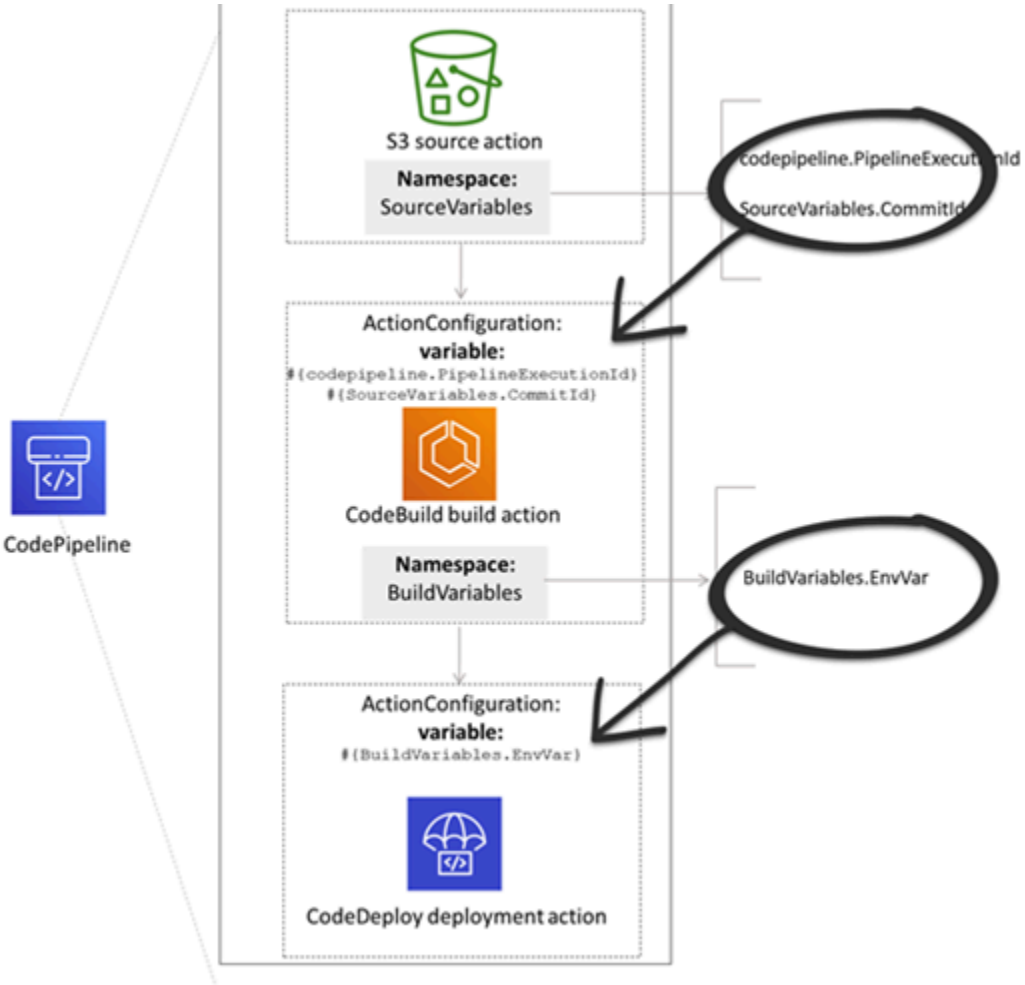
In diesem Beispiel zeigt das Konfigurationsfeld der Build-Aktion Umgebungsvariablen an, die bei Aktionsausführung aktualisiert werden. Das Beispiel gibt den Namespace und die Variable für die Ausführungs-ID mit `#{codepipeline.PipelineExecutionId}` sowie den Namespace und die Variable für die Commit-ID 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. Bevor eine verbrauchende Aktion ausgeführt wird, CodePipeline löst alle Variablenreferenzen auf, die in der Konfiguration vorhanden sind, 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.

step-by-step 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 Variablenschlüssel, 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 von generierten Pipeline CodePipeline-Variablen, die Daten über die Ausführung enthalten, z. B. die Pipeline-Release-ID. Diese Variablen können von jeder Aktion in der Pipeline verbraucht werden.

Themen

- CodePipeline Ausführungs-ID-Variable
- Ausgabevariablen für Amazon-ECR-Aktionen
- AWS CloudFormation StackSets Aktionsausgabevariablen
- CodeCommit Aktionsausgabevariablen
- CodeStarSourceConnection Aktionsausgabevariablen
- GitHub Aktionsausgabevariablen (GitHub Aktionsversion 1)
- Ausgabevariablen der S3-Aktion

CodePipeline Ausführungs-ID-Variable

CodePipeline Ausführungs-ID-Variable

Anbieter	Variablenschlüssel	Beispielwert	Beispiel für eine Variablensyntax
codepipeline	`PipelineExecutionId`	8abc75f0-fbf8-4f4c-bfEXAMPLE	`#{codepipeline.PipelineExecutionId}`

Ausgabevariablen für Amazon-ECR-Aktionen

Amazon-ECR-Variablen

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`ImageDigest`	sha256:EXAMPLE1122334455	`#{SourceVariables.ImageDigest}`
`ImageTag`	brandneue	`#{SourceVariables.ImageTag}`
`ImageURI`	11111EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest	`#{SourceVariables.ImageURI}`
`RegistryId`	EXAMPLE12233	`#{SourceVariables.RegistryId}`
`RepositoryName`	my-image-repo	`#{SourceVariables.RepositoryName}`

AWS CloudFormation StackSets Aktionsausgabevariablen

AWS CloudFormation StackSets -Variablen

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`OperationId`	111111111–2bbb-111–2bbb-1111Beispiel	`#{DeployVariables.OperationId}`
`StackSetId`	my-stackset:111aaaa-1111-2222-2bbb-1111Beispiel	`#{DeployVariables.StackSetId}`

CodeCommit Aktionsausgabevariablen

CodeCommit -Variablen

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`AuthorDate`	2019-10-29T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	Entwicklung	`#{SourceVariables.BranchName}`
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Behebung eines Fehlers (maximale Größe 100 KB)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	2019-10-29T03:32:21Z	`#{SourceVariables.CommitterDate}`
`RepositoryName`	myCodeCommitRepo	`#{SourceVariables.RepositoryName}`

CodeStarSourceConnection Aktionsausgabevariablen

`CodeStarSourceConnection` -Variablen (Bitbucket Cloud, GitHub, GitHub Enterprise Repository und GitLab.com)

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`AuthorDate`	2019-10-29T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	Entwicklung	`#{SourceVariables.BranchName}`

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Behebung eines Fehlers (maximale Größe 100 KB)	`#{SourceVariables.CommitMessage}`
`ConnectionArn`	arn:aws:codestar-connections:region:*account-id* :connection/*connection-id*	`#{SourceVariables.ConnectionArn}`
`FullRepositoryName`	Benutzername/GitHubRepo	`#{SourceVariables.FullRepositoryName}`

GitHub Aktionsausgabevariablen (GitHub Aktionsversion 1)

GitHub -Variablen (GitHub Aktionsversion 1)

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`AuthorDate`	2019-10-29T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	Haupt	`#{SourceVariables.BranchName}`
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Behebung eines Fehlers (maximale Größe 100 KB)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	2019-10-29T03:32:21Z	`#{SourceVariables.CommitterDate}`

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`CommitUrl`		`#{SourceVariables.CommitUrl}`
`RepositoryName`	myGitHubRepo	`#{SourceVariables.RepositoryName}`

Ausgabevariablen der S3-Aktion

S3-Variablen

Variablenschlüssel	Beispielwert	Beispiel für eine Variablen syntax
`ETag`	example28be1c3	`#{SourceVariables.ETag}`
`VersionId`	exampleta_IUQCv	`#{SourceVariables.VersionId}`

Aktionen mit benutzerdefinierten Variablenschlüsseln

Für CodeBuild- AWS CloudFormation, - und Lambda-Aktionen werden die Variablenschlüssel vom Benutzer konfiguriert.

Themen

- CloudFormation Aktionsausgabevariablen
- CodeBuild Aktionsausgabevariablen
- Ausgabevariablen für Lambda-Aktionen

CloudFormation Aktionsausgabevariablen

AWS CloudFormation -Variablen

Variablenschlüssel	Beispiel für eine Variablensyntax
Bei - AWS CloudFormation Aktionen werden Variablen aus allen Werten erstellt, die im `Outputs` Abschnitt einer Stack-Vorlage angegeben sind. Beachten Sie, dass die einzigen CloudFormation Aktionsmodi, die Ausgaben generieren, diejenigen sind, die zum Erstellen oder Aktualisieren eines Stacks führen, z. B. Stack-Erstellung, Stack-Aktualisierungen und 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 unter AWS CloudFormation. Ein Tutorial, das Ihnen zeigt, wie Sie eine Pipeline mit einer - AWS CloudFormation Bereitste llungsaktion in einer Pipeline erstellen, die AWS CloudForm ation Ausgabevariablen verwendet, finden Sie unter Tutorial: Erstellen einer Pipeline, die Variablen aus AWS CloudForm ation-Bereitstellungsaktionen verwendet.	`#{DeployVariables. StackName}`

CodeBuild Aktionsausgabevariablen

CodeBuild -Variablen

Variablenschlüssel	Beispiel für eine Variablensyntax
Für - CodeBuild Aktionen werden Variablen aus Werten erzeugt, die von exportierten Umgebungsvariablen generiert werden. Richten Sie eine CodeBuild Umgebungsvariable ein,	`#{BuildVariables.EnvVar}`

Variablenschlüssel	Beispiel für eine Variablensyntax
indem Sie Ihre CodeBuild Aktion in bearbeiten CodePipeline oder die Umgebungsvariable zur Build-Spezifikation hinzufüge n. Fügen Sie Ihrer CodeBuild Build-Spezifikation Anweisung en hinzu, um die Umgebungsvariable im Abschnitt Exportier te Variablen hinzuzufügen. Siehe env/exported-variables im AWS CodeBuild -Benutzerhandbuch.	

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 PutJobSuc cessResult 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: Verwenden von Variablen mit Lambda-Aufrufaktionen.	`#{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. Beispiele:

- `*.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 Sie`my-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 aktualisieren, um die empfohlene Erkennungsmethode zu verwenden. Ein Migrationsleitfaden mit Anweisungen zur Aktualisierung Ihrer Polling-Pipelines zur Verwendung der empfohlenen ereignisbasierten Methode zur Erkennung von Änderungen finden Sie unterMigrieren Sie Abfrage-Pipelines, um die ereignisbasierte Änderungserkennung zu nutzen.

Aktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2

AWS CodePipelineIn gibt es zwei unterstützte Versionen der GitHub Quellaktion:

- Empfohlen: Die Aktion GitHub Version 2 verwendet die auf der Github-App basierende Authentifizierung, die von einer CodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen Ressource unterstützt wird. Sie installiert eine AWS CodeStar Connections-Anwendung in Ihrer GitHub Organisation, sodass Sie den Zugriff verwalten können. GitHub
- Nicht empfohlen: Die Aktion GitHub Version 1 verwendet OAuth-Token zur Authentifizierung GitHub und verwendet einen separaten Webhook, um Änderungen zu erkennen. Dies ist nicht mehr die empfohlene Methode.

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

Die Verwendung der Aktion GitHub Version 2 gegenüber der Aktion GitHub Version 1 bietet einige wichtige Vorteile:

- Bei Verbindungen sind für den Zugriff auf Ihr Repository CodePipeline keine OAuth-Apps oder persönlichen Zugriffstoken mehr erforderlich. Wenn Sie eine Verbindung herstellen, installieren Sie eine GitHub App, die die Authentifizierung für Ihr GitHub Repository verwaltet und Berechtigungen auf Organisationsebene gewährt. Sie müssen OAuth-Token als Benutzer autorisieren, um auf das Repository zuzugreifen. Weitere Informationen zum OAuth-basierten GitHub Zugriff im Gegensatz

zum App-basierten Zugriff finden Sie unter. GitHub https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps

- Wenn Sie Aktionen der GitHub Version 2 in der CLI oder verwalten CloudFormation, müssen Sie Ihr persönliches Zugriffstoken nicht mehr als Geheimnis in Secrets Manager speichern. Sie müssen in Ihrer CodePipeline Aktionskonfiguration nicht mehr dynamisch auf das gespeicherte Geheimnis verweisen. Stattdessen fügen Sie den Verbindungs-ARN zu Ihrer Aktionskonfiguration hinzu. Ein Beispiel für eine Aktionskonfiguration finden Sie unterCodeStarSourceConnection für Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com und GitLab selbstverwaltete Aktionen.
- Wenn Sie eine Verbindungsressource erstellen, die Sie mit Ihrer Aktion in GitHub Version 2 verwenden möchten CodePipeline, können Sie dieselbe Verbindungsressource verwenden, um Ihrem Repository andere unterstützte Dienste wie CodeGuru Reviewer zuzuordnen.
- In Github Version 2 kannst du Repositorys klonen, um bei nachfolgenden CodeBuild Aktionen auf Git-Metadaten zuzugreifen, während du in Github Version 1 nur die Quelle herunterladen kannst.
- Ein Administrator installiert die App für die Repositorys Ihrer Organisation. Sie müssen keine OAuth-Token mehr nachverfolgen, die von der Person abhängen, die das Token erstellt hat.

Alle in einer Organisation installierten Apps haben Zugriff auf dieselben Repositorys. Um zu ändern, wer auf jedes Repository zugreifen kann, ändern Sie die IAM-Richtlinie für jede Verbindung. Ein Beispiel finden Sie unter Beispiel: Eine nach unten abgegrenzte Richtlinie für die Verwendung von Verbindungen mit einem bestimmten Repository.

Sie können die Schritte in diesem Thema verwenden, um Ihre Quellaktion für GitHub Version 1 zu löschen und eine Quellaktion für GitHub Version 2 aus der Konsole hinzuzufügen. CodePipeline

Themen

- Schritt 1: Ersetzen Sie Ihre GitHub Aktion für Version 1
- Schritt 2: Stellen Sie eine Verbindung her zu GitHub
- Schritt 3: Speichern Sie Ihre GitHub Quellaktion

Schritt 1: Ersetzen Sie Ihre GitHub Aktion für Version 1

Verwenden Sie die Pipeline-Bearbeitungsseite, um Ihre Aktion aus Version 1 durch eine GitHub Aktion aus Version 2 GitHub zu ersetzen.

Um Ihre GitHub Aktion aus Version 1 zu ersetzen

1. Melden Sie sich bei der CodePipeline Konsole an.
2. Wählen Sie Ihre Pipeline und dann Bearbeiten aus. Wählen Sie in Ihrer Quellstufe die Option Phase bearbeiten aus. Es wird eine Meldung angezeigt, in der empfohlen wird, Ihre Aktion zu aktualisieren.
3. Wählen Sie unter Aktionsanbieter die Option GitHub (Version 2) aus.
4. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie noch keine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Option Connect aus GitHub. Fahren Sie mit Schritt 2 fort: Verbindung herstellen zu GitHub.
   - Wenn Sie bereits eine Verbindung zu Ihrem Anbieter hergestellt haben, wählen Sie unter Verbindung die Verbindung aus. Fahren Sie mit Schritt 3 fort: Speichern Sie die Quellaktion für Ihre Verbindung.

Schritt 2: Stellen Sie eine Verbindung her zu GitHub

Nachdem Sie sich entschieden haben, die Verbindung herzustellen, wird die GitHub Seite Connect angezeigt.

Um eine Verbindung herzustellen zu GitHub

1. Unter GitHub Verbindungseinstellungen wird Ihr Verbindungsname unter Verbindungsname angezeigt.

   Wählen Sie unter GitHub Apps eine App-Installation aus oder wählen Sie Neue App installieren, um eine zu erstellen.

   > **Note**
   >
   > Sie installieren eine App für alle Verbindungen mit einem bestimmten Anbieter. Wenn Sie die GitHub App bereits installiert haben, wählen Sie sie aus und überspringen Sie diesen Schritt.

2. Wenn die Autorisierungsseite für GitHub angezeigt wird, melden Sie sich mit Ihren Anmeldeinformationen an und wählen Sie dann, ob Sie fortfahren möchten.

3. Auf der App-Installationsseite wird eine Meldung angezeigt, dass die AWS CodeStar App versucht, eine Verbindung zu Ihrem GitHub Konto herzustellen.

> **Note**
>
> Sie installieren die App nur einmal für jedes GitHub Konto. Wenn Sie die App schon einmal installiert haben, können Sie Configure (Konfiguration) wählen und mit einer Änderungsseite für die App-Installation fortfahren. Alternativ kommen Sie über die Schaltfläche „Back“ (Zurück) zur Konsole zurück.

4. Wählen Sie auf der AWS CodeStar Seite „Installieren“ die Option „Installieren“.
5. Auf der GitHub Seite Connect wird die Verbindungs-ID für Ihre neue Installation angezeigt. Wählen Sie Connect aus.

Schritt 3: Speichern Sie Ihre GitHub Quellaktion

Vervollständigen Sie Ihre Aktualisierungen auf der Seite Aktion bearbeiten, um Ihre neue Quellaktion zu speichern.

Um Ihre GitHub Quellaktion zu speichern

1. Geben Sie unter Repository den Namen Ihres Drittanbieter-Repositorys ein. Geben Sie im Feld Branch den Branch ein, in dem Ihre Pipeline Quelländerungen erkennen soll.

> **Note**
>
> Geben Sie im Feld Repository `owner-name/repository-name` wie in diesem Beispiel gezeigt ein:
>
> ```
> my-account/my-repository
> ```

2. Wählen Sie unter Ausgabeartefaktformat das Format für Ihre Artefakte aus.

   - Um die Ausgabeartefakte der GitHub Aktion mit der Standardmethode zu speichern, wählen Sie 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 beschrieben. Hinzufügen von CodeBuild GitClone Berechtigungen für Verbindungen zu Bitbucket, GitHub, GitHub Enterprise Server oder 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.

3. Unter Ausgabeartefakte können Sie den Namen des Ausgabeartefakts für diese Aktion beibehalten, z. B. `SourceArtifact` Wählen Sie Fertig, um die Aktionsseite Bearbeiten zu schließen.
4. Wählen Sie „Fertig“, um die Seite zur Bearbeitung der Phase zu schließen. Wählen Sie Speichern, um die Seite zur Bearbeitung der Pipeline zu schließen.

Kontingente in AWS CodePipeline

CodePipeline verfügt über Kontingente für die Anzahl der Pipelines, Phasen, Aktionen und Webhooks, die ein AWS Konto in jeder AWS Region haben kann. Diese Kontingente gelten pro Region und können erhöht werden. Um eine Erhöhung anzufordern, verwenden Sie die Support-Center-Konsole.

Die Verarbeitung von Anträgen zur Kontingenterhöhung kann bis zu zwei Wochen dauern.

Ressource	Standard
Dauer bis zum Timeout einer Aktion (Dies sind konfigurierbare Timeouts. In der folgenden Tabelle finden Sie nicht konfigurierbare Timeouts)	AWS CloudFormation-Bereitstellungsaktion : 3 Tage CodeDeploy und CodeDeploy ECS-Bereitstellungsaktionen (blau/grün): 5 Tage AWS Lambda-Aufrufaktion: 24 Stunden **Note** Während die Aktion ausgeführt wird, wendet sich CodePipeline regelmäßig an Lambda, um einen -Status zu erhalten. Die Lambda-Funktion antwortet mit einem Status, bei dem die Aktionsausführung entweder erfolgreich, fehlgeschlagen oder in Bearbeitung ist. Wenn die Lambda-Funktion nach 20 Minuten keine Antwort gesendet hat, wird die Aktion beendet. Wenn die Lambda-Funktion während der 20 Minuten darauf geantwortet hat, dass die Aktion noch ausgeführt wird, CodePipeline startet den 20-minütigen Timer neu und versucht es erneut. Wenn nach 24 Stunden

Ressource	Standard
	nicht erfolgreich, CodePipeline setzt den Status der Lambda-Aufrufaktion auf Fehlgeschlagen. Lambda hat ein separates Timeout für Lambda-Funktionen, das nicht mit dem CodePipeline Aktions-Timeout zusammenhängt. Amazon S3-Bereitstellungsaktion: 90 Minuten **Note** Wenn der Upload in S3 während der Bereitstellung einer großen ZIP-Datei abläuft, schlägt die Aktion mit einem Timeout-Fehler fehl. Versuchen Sie, die ZIP-Datei in kleinere Dateien aufzuteilen. Standard-Timeout auf Kontoebene für manuelle Genehmigungsaktionen: 7 Tage **Note** Das Standard-Timeout für die manuelle Genehmigungsaktion kann für eine bestimmte Aktion in der Pipeline überschrieben und bis zu 86 400 Minuten (60 Tage) mit einem Mindestwert von 5 Minuten konfiguriert werden. Weitere Informationen finden Sie

Ressource	Standard
	unter ActionDeclaration in der API CodePipeline -Referenz zu . Bei der Konfiguration wird dieses Timeout für die Aktion angewende t. Andernfalls wird der Standardwert auf Kontoebene verwendet. Alle anderen Aktionen; 1 Stunde **Note** Das Amazon-ECS-Bereitstellungsa ktions-Timeout ist bis zu einer Stunde konfigurierbar (das Standard-Timeout).
Maximale Gesamtanzahl der Pipelines pro Region in einem AWS-Konto	1000 **Note** Pipelines, die für die Polling- oder ereignisbasierte Änderungs erkennung konfiguriert sind, werden auf dieses Kontingent angerechnet.

Ressource	Standard
Maximale Anzahl von Pipelines, die für die Abfrage von Quelländerungen festgelegt sind, pro AWS Region	300 **Note** Dieses Kontingent ist festgelegt und kann nicht geändert werden. Wenn Sie das Limit für die Abfrage von Pipelines erreichen, können Sie trotzdem zusätzliche Pipelines konfigurieren, die die ereignisb asierte Änderungserkennung verwenden. Weitere Informationen finden Sie unter Quellaktionen und Methoden zur Erkennung von Änderungen.[1]
Maximale Anzahl der Webhooks pro Region in einem AWS-Konto	300
Anzahl der benutzerdefinierten Aktionen pro Region in einem AWS-Konto	50

[1]Führen Sie basierend auf Ihrem Quellanbieter die folgenden Anweisungen aus, um Ihre Polling-P ipelines zu aktualisieren, um die ereignisbasierte Änderungserkennung zu verwenden:

- Informationen zum Aktualisieren einer CodeCommit Quellaktion finden Sie unter Migrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole).
- Informationen zum Aktualisieren einer Amazon S3-Quellaktion finden Sie unter Migrieren von Abfrage-Pipelines (CodeCommit oder Amazon S3 S3-Quelle) (Konsole).
- Informationen zum Aktualisieren einer GitHub Quellaktion finden Sie unter Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (Konsole).

Die folgenden Kontingente in AWS CodePipeline gelten für die Verfügbarkeit von Regionen, Namensbeschränkungen und zulässigen Artefaktgrößen. Diese Kontingente stehen fest und können nicht geändert werden.

Eine Liste der CodePipeline Service-Endpunkte für jede Region finden Sie unter -AWS CodePipelineEndpunkte und -Kontingente in der AWS Allgemeinen Referenz zu .

Weitere Informationen zu strukturellen Anforderungen finden Sie unter CodePipeline Referenz zur Pipeline-Struktur.

AWS Regionen, in denen Sie eine Pipeline erstellen können	US East (Ohio) USA Ost (Nord-Virginia) USA West (Nordkalifornien) USA West (Oregon) Kanada (Zentral) Europa (Frankfurt) Europa (Zürich)* Israel (Tel Aviv) Europa (Irland) Europe (London) Europa (Mailand)* Europa (Paris) Europa (Spain) Europa (Stockholm) Afrika (Kapstadt)* Asien-Pazifik (Hongkong)* Asien-Pazifik (Hyderabad)

	Asien-Pazifik (Mumbai) Asien-Pazifik (Tokio) Asien-Pazifik (Seoul) Asien-Pazifik (Osaka) Asien-Pazifik (Singapur) Asien-Pazifik (Sydney) Asien-Pazifik (Jakarta) Asien-Pazifik (Melbourne) Südamerika (São Paulo) Naher Osten (Bahrain)* Naher Osten (VAE) AWS GovCloud (USA-West) AWS GovCloud (USA-Ost)
Zulässige Zeichen in einem Aktionsnamen	Aktionsnamen dürfen nicht mehr als 100 Zeichen enthalten. Zulässige Zeichen sind: Kleinbuchstaben `a` bis `z`, einschließlich. Großbuchstaben `A` bis `Z`, einschließlich. Zahlen `0` bis `9`, einschließlich. Sonderzeichen `.` (Punkt), `@` (At-Zeichen), `-` (Minuszeichen) und `_` (Unterstrich). Andere Zeichen, wie beispielsweise Leerzeichen, sind nicht zulässig.

Zulässige Zeichen in Aktionstypnamen	Aktionstypnamen dürfen nicht mehr als 25 Zeichen enthalten. Zulässige Zeichen sind: Kleinbuchstaben a bis z, einschließlich. Großbuchstaben A bis Z, einschließlich. Zahlen 0 bis 9, inklusive. Sonderzeichen . (Punkt), @ (At-Zeichen), - (Minuszeichen) und _ (Unterstrich). Andere Zeichen, wie beispielsweise Leerzeichen, sind nicht zulässig.
Zulässige Zeichen in Artefaktnamen	Artefaktnamen dürfen 100 Zeichen nicht überschreiten. Zulässige Zeichen sind: Kleinbuchstaben a bis z, einschließlich. Großbuchstaben A bis Z, einschließlich. Zahlen 0 bis 9, einschließlich. Sonderzeichen - (Minuszeichen) und _ (Unterstrich). Andere Zeichen, wie beispielsweise Leerzeichen, sind nicht zulässig.

Zulässige Zeichen in Partneraktionsnamen	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 Build-Aktion und Testaktion: 8 Stunden Benutzerdefinierte Aktionen: 24 Stunden Step Functions-Aufrufaktion: 7 Tage
Maximallänge des Aktionskonfigurationsschlüssels (z. B. die CodeBuild-Konfigurationsschlüssel `ProjectName` , `PrimarySource` und `EnvironmentVariables`)	50 Zeichen
Die maximale Länge des Aktionskonfigurationswerts (z. B. sollte der Wert der `RepositoryName` Konfiguration in der CodeCommit Aktionskonfiguration weniger als 1000 Zeichen betragen: `"RepositoryName": "my-repo-name-less-than-1000-characters"`)	1 000 Zeichen
Maximale Anzahl von Aktionen pro Pipeline	500

Maximale Anzahl gleichzeitiger Pipeline-Ausführungen pro Pipeline (QUEUED|PARALLEL-Modus)	50
Maximale Anzahl gleichzeitiger Aktionsausführungen pro Pipeline-Ausführung im PARALLEL-Modus	5
Maximale Anzahl von Dateien für ein Amazon S3-Objekt	100 000
Maximalanzahl der Monate, für die die Informationen im Pipeline-Ausführungsverlauf aufbewahrt werden	12
Maximalanzahl paralleler Aktionen in einer Phase	50
Maximalanzahl der sequenziellen Aktionen in einer Phase	50

Maximalgröße der Artefakte in einer Quellstufe	In Amazon S3-Buckets gespeicherte Artefakte: 7 GB In - CodeCommit oder - GitHub Repositorys gespeicherte Artefakte: 1 GB Ausnahme: Wenn Sie AWS Elastic Beanstalk für die Bereitstellung von Anwendungen verwenden, beträgt die maximal zulässige Größe von Artefakten stets 512 MB. Ausnahme: Wenn Sie AWS CloudFormation für die Bereitstellung von Anwendungen verwenden, beträgt die maximal zulässige Artefaktgröße stets 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 für Image-Definitionen, die in Pipelines verwendet wird, die Amazon-ECS-Container und -Images bereitstellen	100 KB
Maximale Größe der Eingabeartefakte für AWS CloudFormation-Aktionen	256 MB
Maximale Größe der Eingabeartefakte für die `CodeDeployToECS` -Aktion	3 MB

Maximale Größe des JSON-Objekts, das in der Eigenschaft `Parameter0verrides` gespeichert werden kann	Bei einer CodePipeline Bereitstellungsaktion mit AWS CloudFormation als Anbieter wird die -`Parameter0verrides` Eigenschaft verwendet, um ein JSON-Objekt zu speichern , das Werte für die AWS CloudFormation Vorlagenkonfigurationsdatei angibt. Für das JSON-Objekt, das in der Eigenschaft `Parameter0verrides` gespeichert werden kann, gilt eine Maximalgröße von 1 Kilobyte.
Anzahl der Aktionen in einer Phase	Mindestens 1, maximal 50
Anzahl der zulässigen Artefakte für jede Aktion	Die Anzahl der Eingabe- und Ausgabe-A rtefakte, die für jede Aktion zulässig sind, finden Sie unter Anzahl der Eingabe- und Ausgabe-A rtefakte für jeden Aktionstyp
Anzahl der Phasen in einer Pipeline	Mindestens 2, maximal 50
Pipeline-Tags	Bei Tags muss die Groß- und Kleinschreibung beachtet werden. Maximal 50 pro Ressource.

Pipeline-Tag-Schlüsselnamen	Beliebige Kombination aus Unicode-Buchstaben, Zahlen, Leerstellen und zulässigen UTF-8-Zeichen mit einer Länge zwischen 1 und 128 Zeichen. Zulässige Zeichen sind + - = . _ : / @ Tag-Schlüsselnamen müssen eindeutig sein, und jeder Schlüssel darf nur einen Wert besitzen. Ein Tag darf nicht: • beginnen mit AWS: • nur aus Leerstellen bestehen • mit einem Leerzeichen enden • aus Emojis oder beliebigen der folgenden Zeichen bestehen: ? ^ * [\ ~ ! # $ % & * ( ) > < | " '
Pipeline-Tag-Werte	Beliebige Kombination aus Unicode-Buchstaben, Zahlen, Leerstellen und zulässigen UTF-8-Zeichen mit einer Länge zwischen 1 und 256 Zeichen. Zulässige Zeichen sind + - = . _ : / @ Ein Schlüssel kann nur einen Wert haben, aber viele Schlüssel können den gleichen Wert aufweisen. Ein Tag darf nicht: • beginnen mit AWS: • nur aus Leerstellen bestehen • mit einem Leerzeichen enden • aus Emojis oder beliebigen der folgenden Zeichen bestehen: ? ^ * [\ ~ ! # $ % & * ( ) > < | " '

Auslöser	Es gibt maximal 50 Auslöser in einer Pipeline-Definition über die `pull request` Konfiguration `push` und . Es gibt maximal drei Filter pro Push-Auslöser und Pull-Anforderungsauslöser. **Note** Duplikate für Filter im selben Ereignistyp-Array sind nicht zulässig. Sie können bis zu 8 Einschließen- und 8 Ausschlussmuster, Verzweigungen und Dateipfade für jeden Ereignistyp hinzufügen (Push, Pull-Anforderung). Zulässige Zeichen in Musterwerten enthalten alle Zeichentypen. Für Ein- und Ausschlussmuster gibt es eine maximale Länge von 255 Zeichen. Für Tag-Namen ist eine maximale Länge von 255 Zeichen zulässig. Die maximale Größe des `triggers`Arrays sollte 200 KB nicht überschreiten

Auslöserfilter	Dateipfade: • Anzahl der Muster: Sie können bis zu 8 Einschließen- und 8 Ausschließen-Muster hinzufügen. • Mustergröße: Jede Ein- oder Ausschließen-Mustergröße kann bis zu 255 Zeichen lang sein. Verzweigungen: • Anzahl der Muster: Sie können bis zu 8 Einschließen- und 8 Ausschließen-Muster hinzufügen. • Mustergröße: Jede Ein- oder Ausschließen-Mustergröße kann bis zu 255 Zeichen lang sein. Pull-Anforderungen: Verzweigungen: • Anzahl der Muster: Sie können bis zu 8 Einschließen- und 8 Ausschließen-Muster hinzufügen. • Mustergröße: Jede Ein- oder Ausschließen-Mustergröße kann bis zu 255 Zeichen lang sein.

Eindeutigkeit von Namen	Innerhalb eines einzelnen AWS Kontos muss jede Pipeline, die Sie in einer -AWSRegion 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: • Maximale Länge von 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 variable Werte gibt es eine maximale Länge von 1000 Zeichen Bei Variablenwerten sind alle Zeichen zulässig. Bei Variablenbeschreibungen ist eine maximale Länge von 200 Zeichen zulässig.

* Sie müssen diese Region aktivieren, bevor Sie sie verwenden können.

Anhang A: Quellaktionen der GitHub Version 1

Dieser Anhang enthält Informationen zu Version 1 der GitHub Aktion in CodePipeline.

> **Note**
>
> Obwohl wir nicht empfehlen, die Aktion GitHub Version 1 zu verwenden, funktionieren vorhandene Pipelines mit der Aktion GitHub Version 1 weiterhin ohne Auswirkungen. Für eine Pipeline mit einer GitHub Aktion der Version 1 CodePipeline verwendet OAuth-basierte Token, um eine Verbindung zu Ihrem GitHub Repository herzustellen. Im Gegensatz dazu verwendet die GitHub Aktion (Version 2) eine Verbindungsressource, um AWS Ihrem GitHub Repository Ressourcen zuzuordnen. Die Verbindungsressource verwendet App-basierte Token, um eine Verbindung herzustellen. Weitere Informationen zum Aktualisieren Ihrer Pipeline auf die empfohlene GitHub Aktion, die eine Verbindung verwendet, finden Sie unter Aktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2. Weitere Informationen zum OAuth-basierten GitHub Zugriff im Gegensatz zum anwendungsbasierten GitHub Zugriff finden Sie unter https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps.

Zur Integration in CodePipeline verwendet eine GitHub OAuth-Anwendung für Ihre Pipeline. CodePipeline verwendet Webhooks GitHub, um die Änderungserkennung für Ihre Pipeline mit der Quellaktion der GitHub Version 1 zu verwalten.

> **Note**
>
> Wenn Sie eine GitHub Version-2-Quellaktion in konfigurieren AWS CloudFormation, fügen Sie keine GitHub Token-Informationen hinzu oder fügen eine Webhook-Ressource hinzu. Sie konfigurieren eine Verbindungsressource wie AWS::CodeStarConnections::Connection unter im AWS CloudFormation -Benutzerhandbuch gezeigt.

Diese Referenz enthält die folgenden Abschnitte für die Aktion der GitHub Version 1:

- Informationen zum Hinzufügen einer Quellaktion der GitHub Version 1 und eines Webhooks zu einer Pipeline finden Sie unter Hinzufügen einer GitHub Version 1-Quellaktion.

- Informationen zu den Konfigurationsparametern und Beispiel-YAML/JSON-Snippets für eine Quellaktion der GitHub Version 1 finden Sie unter GitHub Referenz zur Quellaktionsstruktur der Version 1.

Themen

- Hinzufügen einer GitHub Version 1-Quellaktion
- GitHub Referenz zur Quellaktionsstruktur der Version 1

Hinzufügen einer GitHub Version 1-Quellaktion

Sie fügen Quellaktionen der GitHub Version 1 zu hinzu, CodePipeline indem Sie:

- Wählen Sie auf der Seite Pipeline erstellen (Erstellen einer Pipeline (Konsole)) oder Aktion bearbeiten der CodePipeline Konsole die GitHub Option Anbieter aus. Die Konsole erstellt einen Webhook, der Ihre Pipeline startet, wenn sich die Quelle ändert.
- Verwenden der CLI zum Hinzufügen der Aktionskonfiguration für die `GitHub` Aktion und Erstellen zusätzlicher Ressourcen wie folgt:
  - Verwenden der `GitHub` Beispielaktionskonfiguration in GitHub Referenz zur Quellaktionsstruktur der Version 1 , um die Aktion zu erstellen, wie in gezeigtErstellen einer Pipeline (CLI).
  - Deaktivieren regelmäßiger Prüfungen und manuelles Erstellen der Änderungserkennung, da die Methode zur Änderungserkennung standardmäßig die Pipeline startet, indem die Quelle abgefragt wird. Sie migrieren Ihre Abfrage-Pipeline zu Webhooks für Aktionen der GitHub Version 1.

GitHub Referenz zur Quellaktionsstruktur der Version 1

> **Note**
>
> Obwohl wir nicht empfehlen, die Aktion der GitHub Version 1 zu verwenden, funktionieren vorhandene Pipelines mit der Aktion der GitHub Version 1 weiterhin ohne Auswirkungen. Für eine Pipeline mit einer Quellaktion der GitHub GitHub Version 1 CodePipeline verwendet OAuth-basierte Token, um eine Verbindung zu Ihrem GitHub Repository herzustellen. Im Gegensatz dazu verwendet die neue GitHub Aktion (Version 2) eine Verbindungsressource, um Ihrem GitHub Repository AWS Ressourcen zuzuordnen. Die Verbindungsressource verwendet App-basierte Token, um eine Verbindung herzustellen. Weitere Informationen

zum Aktualisieren Ihrer Pipeline auf die empfohlene GitHub Aktion, die eine Verbindung verwendet, finden Sie unter Aktualisieren Sie eine Quellaktion von GitHub Version 1 auf eine Quellaktion GitHub von Version 2.

Löst die Pipeline aus, wenn ein neuer Commit für das konfigurierte GitHub Repository und die Verzweigung ausgeführt wird.

Zur Integration mit GitHub CodePipeline verwendet 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 einen GitHub Webhook, der Ihre Pipeline startet, wenn eine Änderung im Repository auftritt.

Sie müssen bereits ein GitHub Konto und ein Repository erstellt haben, bevor Sie die Pipeline über eine GitHub Aktion verbinden.

Wenn Sie den Zugriff CodePipeline auf Repositorys einschränken möchten, erstellen Sie ein GitHub Konto und gewähren Sie dem Konto nur Zugriff auf die Repositorys, die Sie in integrieren möchten CodePipeline. Verwenden Sie dieses Konto, wenn Sie so konfigurieren CodePipeline , dass GitHub Repositorys für Quellphasen in Pipelines verwendet werden.

Weitere Informationen finden Sie in der GitHub Entwicklerdokumentation auf der - GitHub Website.

Themen

- Aktionstyp
- Konfigurationsparameter
- Input artifacts (Eingabeartefakte)
- Ausgabeartefakte
- Ausgabevariablen
- Aktionsdeklaration (GitHub-Beispiel)
- Herstellen einer Verbindung mit GitHub (OAuth)
- 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 dem Vorgänge CodePipeline in Ihrem GitHub Repository ausführen kann. Die Eingabe wird stets als Maske mit vier Sternchen angezeigt. Sie stellt einen der folgenden Werte dar:

- Wenn Sie die Pipeline mit der Konsole 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 Ihr GitHub persönliches Zugriffstoken in diesem Feld übergeben. Ersetzen Sie die Sternchen (****) durch Ihr persönliches Zugriffstoken, das aus kopiert wurde GitHub. Wenn Sie `get-pipeline` ausführen, um die Aktionskonfiguration anzuzeigen, wird dieser Wert durch vier Sternchen maskiert angezeigt.
- Wenn Sie eine - AWS CloudFormation Vorlage zum Erstellen der Pipeline verwenden, müssen Sie zuerst das Token als Secret in speichern AWS Secrets Manager. Sie fügen den Wert für dieses Feld als dynamischen Verweis auf das in Secrets Manager gespeicherte Secret ein, z. B. `{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}`.

Weitere Informationen zu GitHub Bereichen finden Sie in der GitHub Entwickler-API-Referenz auf der - GitHub Website.

PollForSourceChanges

Erforderlich: Nein

`PollForSourceChanges` steuert, ob das GitHub Repository nach Quelländerungen CodePipeline abfragt. Es wird empfohlen, stattdessen Webhooks zu verwenden, um Quelländerungen zu erkennen. Weitere Informationen zum Konfigurieren von Webhooks finden Sie unter Migrieren Sie Polling-Pipelines zu Webhooks (Quellaktionen der GitHub Version 1) (CLI) oder Aktualisierungspipelines für Push-Ereignisse (Quellaktionen der GitHub Version 1) (AWS CloudFormationVorlage).

> **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`: Wenn diese Option festgelegt ist, CodePipeline wird Ihr Repository nach Quelländerungen sortiert.

  > **Note**
  >
  > Wenn Sie weglassen`PollForSourceChanges`, fragt CodePipeline standardmäßig Ihr Repository nach Quelländerungen ab. Dieses Verhalten ist das gleiche, als ob `PollForSourceChanges` auf `true` festgelegt würde.

- `False`: Wenn diese Option festgelegt ist, CodePipeline fragt Ihr Repository nicht nach Quelländerungen ab. Sie verwenden diese Einstellung, wenn Sie einen Webhook konfigurieren möchten, um Quelländerungen zu erkennen.

Input artifacts (Eingabeartefakte)

- Anzahl der Artefakte: `0`
- Beschreibung: Eingabe-Artefakte sind für diesen Aktionstyp nicht gültig.

Ausgabeartefakte

- Anzahl der Artefakte: 1
- Beschreibung: Das Ausgabe-Artefakt dieser Aktion ist eine ZIP-Datei, die den Inhalt des konfigurierten Repositorys und Zweigs beim Commit enthält, der als Quellrevision für die Pipeline-Ausführung angegeben wurde. Die aus dem Repository generierten Artefakte sind die Ausgabeartefakte für die GitHub Aktion. Die Quellcode-Commit-ID wird in 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 CodePipelinefinden Sie unter Variablen.

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 durchgeführt wurde, der die Pipeline ausgelöst hat.

BranchName

Der Name der Verzweigung für das GitHub Repository, in dem die Quelländerung vorgenommen wurde.

AuthorDate

Das Datum im Zeitstempelformat, an dem der Commit erstellt wurde.

Weitere Informationen zum Unterschied zwischen einem Autor und einem Committer in Git finden Sie unter Viewing the Commit History in Pro Git von Scott Chacon und Ben Straub.

CommitterDate

Das Datum im Zeitstempelformat, an dem der Commit durchgeführt wurde.

Weitere Informationen zum Unterschied zwischen einem Autor und einem Committer in Git finden Sie unter Viewing the Commit History in Pro Git von Scott Chacon und Ben Straub.

Aktionsdeklaration (GitHub-Beispiel)

YAML

```
Name: Source
Actions:
  - InputArtifacts: []
    ActionTypeId:
      Version: '1'
      Owner: ThirdParty
      Category: Source
      Provider: GitHub
    OutputArtifacts:
      - Name: SourceArtifact
    RunOrder: 1
    Configuration:
      Owner: MyGitHubAccountName
      Repo: MyGitHubRepositoryName
      PollForSourceChanges: 'false'
      Branch: main
      OAuthToken: '{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}'
    Name: ApplicationSource
```

JSON

```
{
    "Name": "Source",
    "Actions": [
```

```
        {
            "InputArtifacts": [],
            "ActionTypeId": {
                "Version": "1",
                "Owner": "ThirdParty",
                "Category": "Source",
                "Provider": "GitHub"
            },
            "OutputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "RunOrder": 1,
            "Configuration": {
                "Owner": "MyGitHubAccountName",
                "Repo": "MyGitHubRepositoryName",
                "PollForSourceChanges": "false",
                "Branch": "main",
                "OAuthToken":
 "{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}"
            },
            "Name": "ApplicationSource"
        }
    ]
},
```

Herstellen einer Verbindung mit GitHub (OAuth)

Wenn Sie zum ersten Mal die Konsole verwenden, um einer Pipeline ein GitHub Repository hinzuzufügen, werden Sie aufgefordert, den CodePipeline Zugriff auf Ihre Repositorys zu autorisieren. Das Token erfordert die folgenden GitHub Bereiche:

- Den `repo`- Umfang, der für die volle Kontrolle verwendet wird, um Artefakte aus öffentlichen und privaten Repositorys in eine Pipeline zu lesen und einzufügen.
- Den `admin:repo_hook` Umfang, der für die volle Kontrolle über Repository-Hooks verwendet wird.

Wenn Sie die CLI oder eine - AWS CloudFormation Vorlage verwenden, müssen Sie den Wert für ein persönliches Zugriffstoken angeben, das Sie bereits in 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 – Dies umfasst Felddefinitionen, Beispiele und Ausschnitte für die Ressource in AWS CloudFormation.
- Ressourcenreferenz für das AWS CloudFormation -Benutzerhandbuch- AWS::CodeStar::GitHubRepository – Dies umfasst Felddefinitionen, Beispiele und Ausschnitte für die Ressource in AWS CloudFormation.
- Tutorial: Erstellen einer Pipeline, die Ihre Android-App mit erstellt und testet AWS Device Farm – Dieses Tutorial enthält eine Beispieldatei für Build-Spezifikationen und eine Beispielanwendung zum Erstellen einer Pipeline mit einer GitHub Quelle. Es erstellt und testet eine Android-App mit CodeBuild und AWS Device Farm.

AWS CodePipeline Dokumentverlauf des Benutzerhandbuchs

In der folgenden Tabelle werden die wichtigen Änderungen in jeder Version 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: 15. März 2024

Änderung	Beschreibung	Datum
Aktualisierungen der von verwalteten Richtlinie	Die AWS verwaltete Richtlinie `AWSCodePipeline_FullAccess` wurde aktualisiert. Siehe -AWS verwaltete Richtlinien für AWS CodePipeline.	15. März 2024
Unterstützung für konfigurierbares Timeout für manuelle Genehmigungsaktionen	Kontingentinformationen für das neue konfigurierbare Timeout-Feld für manuelle Genehmigungsaktionen hinzugefügt. Weitere Informationen finden Sie unter Kontingente.	15. Februar 2024
Unterstützung für die Auslöserfilterung nach Verzweigungen und Dateipfaden	Unterstützung für Auslöserkonfiguration hinzugefügt, die Filterung nach Pull-Anforderungsstatus, Verzweigungen und Dateipfaden für Pipelines vom Typ V2 ermöglicht. Weitere Informationen finden Sie unter Filtern	8. Februar 2024

	von Auslösern bei Code-Push - oder Pull-Anforderungen , Auslöser und Filtern bei Feature-Zweigen zum Starten Ihrer Pipeline und Kontingente.	
Unterstützung für neue Pipeline-Ausführungsmodi	Unterstützung für PARALLEL- und QUEUED-Pipeline-Ausführungsmodi hinzugefügt. Weitere Informationen finden Sie unter Festlegen des Pipeline-Ausführungsmodus, So werden Ausführungen im QUEUED-Modus verarbeitet, So werden Ausführungen im PARALLEL-Modus verarbeitet und Kontingente.	8. Februar 2024
Aktualisierungen der Konsolenseiten zum Anzeigen von Aktionsdetails, Überprüfen manueller Genehmigungsaktionen und Auflisten der Pipelines	Konsolenaktualisierungen sind für die neue Schaltfläche und das Dialogfeld Details anzeigen, für das neue Dialogfeld für die manuelle Genehmigung und für neue Spalten für die letzten Ausführungen auf der Seite Pipelines auflisten dokumentiert. Weitere Informationen finden Sie unter Pipelines anzeigen (Konsole), Aktionsdetails in einer Pipeline anzeigen und Genehmigungsaktionen in Pipelines verwalten.	10. Januar 2024

Unterstützung für GitLab selbstverwaltet	Unterstützung für die Konfiguration von Verbindungen für AWS Ressourcen zur Interaktion mit GitLab selbstverwalteten hinzugefügt. Weitere Informationen finden Sie unter Verbindungen für GitLab selbstverwaltete .	28. Dezember 2023
Aktualisierungen der `CloudFormationStackInstances` Aktionen `CloudFormationStackSet` und	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 unter -AWS Device Farm Aktionsreferenz .	18. Dezember 2023
Unterstützung für detaillierte Fehlermeldungen für die AWS CloudFormation Aktion in hinzugefügt CodePipeline	AWS CloudFormation - Aktionsfehlermeldungen können jetzt Details zu Ressourcen anzeigen, die fehlgeschlagen sind. Weitere Informationen finden Sie unter -AWS CloudFormation Aktionsreferenz .	15. Dezember 2023

Aktualisierungen zum Starten 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 Bucket-Referenzthema für Ereignisplatzhalter und AWS-Service Endpunkte wurden aktualisiert.	13. November 2023
Aktualisierungen für Ereignisfelder in Amazon EventBridge	Sie können jetzt aktualisierte Ereignisfelder in Amazon anzeigen EventBridge. Weitere Informationen finden Sie unter Überwachen CodePipeline von Ereignissen.	9. November 2023

Updates für neue Pipeline-Typ-V2-Pipelines, -Auslöser für Git-Tags und Pipeline-Variablen in CodePipeline	Sie können jetzt einen Pipeline-Typ in auswählen CodePipeline. Für eine Pipeline vom Typ V2 können Sie jetzt eine Auslöserk onfiguration verwenden, um Ihre Pipeline auf Git-Tags zu starten. Mit Pipelines vom Typ V2 können Sie auch Variablen auf Pipeline-Ebene verwenden, um Eingabepa rameter für eine Pipeline-Ausführung zu übergeben. Weitere Informationen finden Sie unter Variablen , Tutorial: Verwenden von Variablen auf Pipeline-Ebene und Tutorial: Verwenden von Git-Tags, um Ihre Pipeline zu starten. Weitere Informationen zu Pipeline-Typen finden Sie unter Pipeline-Typen.	24. Oktober 2023

CodePipeline ermöglicht das Wiederholen aller Aktionen in einer fehlgeschlagenen Phase	Bei einer fehlgeschlagenen Phase in können Sie die Stufe wiederholen CodePipeline, ohne die Pipeline erneut auszuführen. Dazu versuchen Sie entweder die fehlgeschlagenen Aktionen in einer Phase erneut oder alle Aktionen in der Phase ab der ersten Aktion in der Phase erneut. Weitere Informationen finden Sie unter Wiederholen einer fehlgeschlagenen Phase oder fehlgeschlagener Aktionen in einer Phase.	17. Oktober 2023
Unterstützung für - GitLab Gruppen	Unterstützung 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 zu GitLab.com	Sie können Verbindungen verwenden, um AWS Ressourcen für die Interaktion mit GitLab.com zu konfigurieren. Sie können auch die vollständige Klonoption für die Verwendung von Git-Befehlen und Metadaten für nachgelagerte Aktionen auswählen. Weitere Informationen finden Sie unter GitLab Verbindungen und im CodeStarSourceConnection Referenzthema zur Aktionsstruktur.	10. August 2023
Aktualisierung der `CloudFormationStackInstances` Aktion	Der `RegionConcurrencyType` Parameter wurde für die `CloudFormationStackInstances` Aktion hinzugefügt. Informationen zur Aktion finden Sie auf der Seite mit der Aktionsreferenz`CloudFormationStackInstances` .	08. August 2023
Aktualisierung der `CloudFormationStackSet` Aktion	Der `RegionConcurrencyType` Parameter wurde für die `CloudFormationStackSet` Aktion hinzugefügt. Informationen zur Aktion finden Sie auf der Aktionsreferenzseite`CloudFormationStackSet` .	24. Juli 2023

Aktualisierungen der von verwalteten Richtlinie	Die AWS verwaltete Richtlinie `AWSCodePipeline_FullAccess` wurde aktualisiert. Siehe -AWS verwaltete Richtlinien für AWS CodePipeline.	21. Juni 2023
Aktualisierungen der Migrationsverfahren für die Abfrage von Pipelines	Die Verfahren zum Migrieren (Aktualisieren) von Abfrage-Pipelines zur Verwendung der ereignisbasierten Änderungserkennung wurden mit den Schritten für Pipelines aktualisiert, die einen Amazon S3-Bucket verwenden, der für Benachrichtigungen an aktiviert ist EventBridge. Weitere Informationen finden Sie unter Migrieren von Abfragepipelines zur Verwendung der ereignisbasierten Änderungserkennung.	12. Juni 2023
Aktualisierungen für -verwaltete Richtlinien	Die AWS verwalteten Richtlinien `AWSCodePipeline_FullAccess` und `AWSCodePipeline_ReadOnlyAccess` wurden mit einer zusätzlichen Berechtigung aktualisiert. Weitere Informationen finden Sie unter AWS CodePipeline Aktualisierungen der von AWS verwalteten Richtlinien.	16. Mai 2023

Aktualisierungen für -verwaltete Richtlinien	Die von AWS verwalteten Richtlinien `AWSCodePipelineFullAccess` und `AWSCodePipelineReadOnlyAccess` sind veraltet. Verwenden Sie die `AWSCodePipeline_ReadOnlyAccess` Richtlinien `AWSCodePipeline_FullAccess` und . Siehe AWS CodePipeline Aktualisierungen für AWS -verwaltete Richtlinien.	17. November 2022
Aktualisierungen der Verfahren, die verwenden CloudTrail	Alle Konsolenprozeduren, CLI-Beispielbefehle und AWS CloudFormation Beispielausschnitte und Vorlagen für eine Pipeline mit einer S3-Quelle wurden mit der Option Schreiben und Falsch für Verwaltungsereignisse in aktualisiert CloudTrail. Weitere Informationen finden Sie in den aktualisierten Beispielen unter Starten einer Pipeline, Tutorial: Erstellen einer Pipeline mit AWS CloudFormation, Bearbeiten von Pipelines zur Verwendung von Push-Ereignissen und Aktualisieren von Abfragepipelines.	27. April 2022

Neue unterstützte Integration mit Snyk	Sie können die Snyk-Aufr ufaktion in verwenden CodePipeline , um das Sicherheitsscannen für Ihren Open-Source-Code 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. Das Thema Limits und AWS-Service Endpunkte wurden aktualisiert.	27. Januar 2021
Die Änderungserkennung kann für Quellaktionen mit Verbindungen deaktiviert werden	Sie können die CLI oder das SDK verwenden, um eine `CodeStarSourceConnection` Quellaktion zu aktualisieren, um die automatische Änderungserkennung für das Quell-Repository zu deaktivieren. Das CodeStarSourceConnection Referenzthema zur Aktionsstruktur wurde mit einer Beschreibung für den `DetectChanges` Parameter aktualisiert.	08. Januar 2021

CodePipeline unterstützt jetzt AWS CloudFormation StackSets Bereitstellungsaktionen	Ein neues Tutorial, Tutorial: Erstellen einer Pipeline, die AWS CloudFormation StackSets als Bereitstellungsanbieter verwendet, enthält Schritte zum AWS CloudFormation StackSets Erstellen und Aktualisieren Ihrer Stack-Sets und Stack-Instances mit Ihrer Pipeline. Das AWS CloudFormation StackSets Referenzthema zur Aktionsstruktur wurde ebenfalls hinzugefügt.	30. Dezember 2020
Neue unterstützte Region Asien-Pazifik (Hongkong)	CodePipeline ist jetzt in Asien-Pazifik (Hongkong) verfügbar. Das Thema Limits und AWS-Service Endpunkte wurden aktualisiert.	22. Dezember 2020
Anzeigen aktualisierter EventBridge Ereignismuster in CodePipeline	Aktualisierte Ereignismuster und -status für Ereignisse auf Pipeline-, Stufen- und Aktionsebene wurden zur Überwachung von CodePipeline Ereignissen hinzugefügt.	21. Dezember 2020

Anzeigen eingehender Pipeline-Ausführungen in CodePipeline	Sie können die Konsole oder die CLI verwenden, um eingehende Ausführungen anzuzeigen. Weitere Informationen finden Sie unter Anzeigen einer eingehenden Ausführung (Konsole) und Anzeigen des Status der eingehenden Ausführung (CLI).	16. November 2020
Die CodeCommit Quellaktion in CodePipeline unterstützt die vollständige Klonoption	Wenn Sie eine CodeCommit Quellaktion verwenden, können Sie die vollständige Klonoption für die Verwendung von Git-Befehlen und Metadaten für nachgelagerte CodeBuild Aktionen auswählen. Weitere Informationen finden Sie in der -CodeCommit Aktionsreferenz und im Tutorial: Verwenden des vollständigen Klons mit einer CodeCommit Pipeline-Quelle.	11. November 2020

CodePipeline unterstützt Verbindungen zu GitHub und 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 vollständige Klonoption für die Verwendung von Git-Befehlen und Metadaten für nachgelagerte Aktionen auswählen. 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 vorhandene Pipeline mit einer GitHub Quellaktion verfügen, finden Sie weitere Informationen unter Aktualisieren einer Quellaktion der GitHub Version 1 auf eine Quellaktion der GitHub Version 2.	30. September 2020

Die CodeBuild Aktion unterstützt die Aktivierung von Batch-Builds in AWS CodePipeline	Für CodeBuild Aktionen in Ihrer Pipeline können Sie Batch-Builds aktivieren, um mehrere Builds in einer einzigen Ausführung auszuführen. Weitere Informationen finden Sie unter CodeBuild Referenz zur Aktionsstruktur und Erstellen einer Pipeline (Konsole).	30. Juli 2020
AWS CodePipeline unterstützt jetzt AWS AppConfig Bereitstellungsaktionen	Ein neues Tutorial, Tutorial: Erstellen einer Pipeline, die AWS AppConfig als Bereitstellungsanbieter verwendet, enthält Schritte zur Bereitstellung AWS AppConfig von Konfigurationsdateien mit Ihrer Pipeline. Das AWS AppConfig Referenzthema zur Aktionsstruktur wurde ebenfalls hinzugefügt.	25. Juni 2020
AWS CodePipeline unterstützt jetzt Amazon VPC in AWS GovCloud (USA-West)	Sie können jetzt AWS CodePipeline über einen privaten Amazon-VPC-Endpunkt in AWS GovCloud (USA-West) eine direkte Verbindung zu herstellen. Weitere Informationen finden Sie unter Verwenden von CodePipeline mit Amazon Virtual Private Cloud.	2. Juni 2020

AWS CodePipeline unterstützt jetzt AWS Step Functions Aufrufaktionen	Sie können jetzt eine Pipeline in erstellen CodePipeline , die AWS Step Functions als Aufrufaktionsanbieter verwendet. Ein neues Tutorial, Tutorial: Verwenden einer AWS Step Functions Aufrufaktion in einer Pipeline enthält Schritte zum Starten einer Zustandsautomaten-Ausführung aus Ihrer Pipeline. Das Thema AWS Step Functions – Aktionsstrukturreferenz wurde ebenfalls hinzugefügt.	28. Mai 2020
Anzeigen, Auflisten und Aktualisieren von Verbindungen	Sie können Verbindungen in der Konsole auflisten, löschen und aktualisieren. Siehe Auflisten von Verbindungen in CodePipeline.	21. Mai 2020
Verbindungen unterstützen das Markieren von Verbindungsressourcen in der CLI	Die Verbindungsressourcen unterstützen jetzt das Markieren 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 (USA-West) verfügbar	Sie können jetzt CodePipeline in AWS GovCloud (USA-West) verwenden. Weitere Informationen finden Sie unter Kontingente.	8. April 2020

Das Thema Kontingente zeigt, welche CodePipeline Servicekontingente konfigurierbar sind	Das CodePipeline Kontingenttthema wurde neu formatiert. Die Dokumentation zeigt, welche Servicekontingente konfigurierbar sind und welche nicht. Siehe Kontingente in AWS CodePipeline.	12. März 2020
Das Timeout der Amazon-ECS-Bereitstellungsaktion ist konfigurierbar	Das Amazon-ECS-Bereitstellungsaktions-Timeout ist bis zu einer Stunde konfigurierbar (das Standard-Timeout). Siehe Kontingente in AWS CodePipeline.	5. Februar 2020
Neue Themen beschreiben, wie Sie eine Pipeline-Ausführung 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 Services wie zum CodePipeline Herstellen einer Verbindung mit einem Drittanbieter-Repository wie Bitbucket Cloud verwendet werden kann. Weitere Informationen finden Sie unter Arbeiten mit Verbindungen in CodePipeline.	18. Dezember 2019
Aktualisierte Themen zu Sicherheit, Authentifizierung und Zugriffskontrolle	Die Sicherheits-, Authentifizierungs- und Zugriffskontrollinformationen für CodePipeline wurden in ein neues Kapitel Sicherheit organisiert. Weitere Informationen finden Sie unter Sicherheit.	17. Dezember 2019

Neue Themen beschreiben, 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
Neue Themen beschreiben, 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 für die 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 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. Das Thema Limits und AWS-Service Endpunkte wurden aktualisiert.	5. September 2019

Vordefinierte ACLs und Cache-Kontrolle für Amazon S3-Bereitstellungsaktionen angeben	Sie können jetzt vordefinierte ACL- und Cache-Kontrolloptionen angeben, wenn Sie eine Amazon S3-Bereitstellungsaktion in erstellen CodePipeline. Die folgenden Themen wurden aktualisiert: Erstellen einer Pipeline (Konsole), CodePipeline Pipeline-Strukturreferenz und Tutorial: Erstellen einer Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet.	27. Juni 2019

Sie können jetzt Tags zu Ressourcen in hinzufügen 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 hinzugefügt: Markieren von Ressourcen, Verwenden von Tags zur Steuerung des Zugriffs auf CodePipeline Ressourcen, Markieren einer Pipeline in CodePipeline, Markieren einer benutzerdefinierten Aktion in CodePipeline und Markieren eines Webhooks in CodePipeline. Die folgenden Themen wurden aktualisiert, um zu zeigen, wie die CLI zum Markieren 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 jetzt den Verlauf der Aktionsausführung in anzeigen. 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 in der AWS Serverless Application Repository	Sie können jetzt eine Pipeline in erstellen CodePipeline , die Ihre Serverless-Anwendung in veröffentlicht AWS Serverless Application Repository. Ein neues Tutorial, Tutorial: Veröffentlichen von Anwendungen in der AWS Serverless Application Repository, enthält Schritte zum Erstellen und Konfigurieren einer Pipeline, um Ihre Serverless-Anwendung kontinuierlich an die zu liefern 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. Eine regionsübergreifende Aktion wurde mit den Schritten zum Hinzufügen, Bearbeiten oder Löschen einer Aktion aktualisiert, die sich in einer anderen AWS Region als Ihre 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-Bereitstellungen	Sie können jetzt eine Pipeline in erstellen CodePipeline , die Amazon S3 als Bereitstellungsaktionsanbieter verwendet. Ein neues Tutorial, Tutorial: Erstellen einer Pipeline, die Amazon S3 als Bereitstellungsanbieter verwendet, enthält Schritte zum Bereitstellen von Beispieldateien in Ihrem Amazon S3-Bucket mit CodePipeline. Das Thema Referenz der CodePipeline-Pipeline-Struktur wurde ebenfalls aktualisiert.	16. Januar 2019

AWS CodePipeline unterstützt jetzt Alexa Skills Kit-Bereitstellungen	Sie können jetzt CodePipeline und Alexa Skills Kit für die kontinuierliche Bereitstellung von Alexa-Fähigkeiten verwenden. Ein neues Tutorial, Tutorial: Erstellen einer Pipeline, die einen Amazon Alexa Skill bereitstellt, enthält Schritte zum Erstellen von Anmeldeinformationen, die es ermöglichen, eine AWS CodePipeline Verbindung zu Ihrem Alexa Skills Kit-Entwicklerkonto herzustellen, 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 unterstützt werden AWS PrivateLink	Sie können jetzt AWS CodePipeline über einen privaten Endpunkt in Ihrer VPC eine direkte Verbindung zu herstellen, sodass der gesamte Datenverkehr innerhalb Ihrer VPC und des AWS Netzwerks beibehalten wird. Weitere Informationen finden Sie unter Verwenden von CodePipeline mit Amazon Virtual Private Cloud.	6. Dezember 2018

AWS CodePipeline unterstützt jetzt Amazon-ECR-Quellaktionen und ECS-to-CodeDeploy Deployment-Aktionen	Sie können jetzt CodePipeline und CodeDeploy mit Amazon ECR und Amazon ECS für die kontinuierliche Bereitstellung von containerbasierten Anwendungen verwenden. Ein neues Tutorial, Erstellen einer Pipeline mit einer Amazon-ECR-Quelle und ECS-zu-CodeDeploy Bereitstellung, enthält Schritte zur Verwendung der Konsole zum Erstellen einer Pipeline, die Containeranwendungen, die in einem Image-Repository gespeichert sind, auf 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, Hinzufügen einer regionsübergreifenden Aktion , enthält Schritte zur Verwendung der AWS CLI oder AWS CloudFormation zum Hinzufügen einer Aktion, die sich in einer anderen Region als Ihre Pipeline befindet. Die Themen zum Erstellen einer Pipeline, zum Bearbeiten einer Pipeline und zur CodePipeline -Pipeline -Strukturreferenz wurden aktualisiert.	12. November 2018
AWS CodePipeline ist jetzt in Service Catalog integriert	Sie können jetzt Service Catalog als Bereitstellungsaktion zu Ihrer Pipeline hinzufügen. Auf diese Weise können Sie eine Pipeline einrichten, um Produktaktualisierungen im Service Catalog zu veröffentlichen, wenn Sie eine Änderung an Ihrem Quell-Repository vornehmen. Das Thema Integrationen wurde aktualisiert, um diese Unterstützung für Service Catalog widerzuspiegeln. Zwei Service-Catalog-Tutorials wurden dem Abschnitt AWS CodePipeline Tutorials hinzugefügt.	16. Oktober 2018

AWS CodePipeline ist jetzt in integriert AWS Device Farm	Sie können jetzt AWS Device Farm als Testaktion zu Ihrer Pipeline hinzufügen. Auf diese Weise können Sie eine Pipeline zum Testen von mobilen Anwendung en einrichten. Das Thema Integrationen wurde aktualisi ert, um diese Unterstützung für widerzuspiegeln AWS Device Farm. Zwei AWS Device Farm -Tutorials wurden dem Abschnitt AWS CodePipeline - Tutorials hinzugefügt.	19. Juli 2018
AWS CodePipeline -Benutzer handbuch – Aktualisi erungsbenachrichtigungen jetzt über RSS verfügbar	Die HTML-Version des CodePipeline -Benutzer handbuchs unterstützt jetzt einen RSS-Feed mit Aktualisi erungen, die auf der Seite Aktualisierungsverlauf der Dokumentation 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 jeder Version des CodePipeline - Benutzerhandbuchs am 30. Juni 2018 und früher beschrieben.

Änderung	Beschreibung	Datum geändert
Verwenden von Webhooks, um Quelländerungen in GitHub Pipelines zu erkennen	Wenn Sie eine Pipeline in der Konsole erstellen oder bearbeiten, erstellt CodePipeline jetzt einen Webhook, der Änderungen an Ihrem GitHub Quell-Repository erkennt und dann Ihre Pipeline startet. Informationen zur Migration Ihrer Pipeline finden Sie unter Konfigurieren Ihrer GitHub Pipelines für die Verwendung von Webhooks für die Änderungserkennung. Weitere Informationen finden Sie unter Starten einer Pipeline-Ausführung in CodePipeline.	1. Mai 2018
Aktualisierte Themen	Wenn Sie eine Pipeline in der Konsole erstellen oder bearbeiten, erstellt CodePipeline jetzt eine Amazon CloudWatch Events-Regel und einen - AWS CloudTrail Trail, der Änderungen an Ihrem Amazon S3-Quell-Bucket erkennt und dann Ihre Pipeline startet. Weitere Informationen zur Migration Ihrer Pipeline finden Sie unter Quellaktionen und Methoden zur Erkennung von Änderungen. Die Tutorial: Erstellen einer einfachen Pipeline (S3-Bucket) wurde aktualisiert, um zu zeigen, wie die Amazon CloudWatch -Events-Regel und der Trail erstellt werden, wenn Sie eine Amazon S3-Quelle auswählen. Erstellen einer Pipeline in CodePipeline und Bearbeiten einer Pipeline in CodePipeline wurden ebenfalls aktualisiert. Weitere Informationen finden Sie unter Starten einer Pipeline in CodePipeline.	22. März 2018
Aktualisiertes Thema	CodePipeline ist jetzt in Europa (Paris) verfügbar. Das Thema Kontingente in AWS CodePipeline wurde aktualisiert.	21. Februar 2018

Änderung	Beschreibung	Datum geändert
Aktualisierte Themen	Sie können jetzt CodePipeline und Amazon ECS für die kontinuierliche Bereitstellung von containerbasierten Anwendungen verwenden. Wenn Sie eine Pipeline erstellen, können Sie Amazon ECS als Bereitstellungsanbieter auswählen. Eine Codeänderung in Ihrem Quellcode verwaltungs-Repository veranlasst Ihre Pipeline, ein neues Docker-Image zu erstellen, es in Ihre Container-Registry zu verschieben und dann das aktualisierte Image auf einem Amazon-ECS-Service bereitzustellen. Die Themen Produkt- und Serviceintegrationen mit CodePipeline, und CodePipeline Referenz zur Pipeline-Struktur wurden aktualisiertErstellen einer Pipeline in CodePipeline, um diese Unterstützung für Amazon ECS widerzuspiegeln.	12. Dezember 2017
Aktualisierte Themen	Wenn Sie eine Pipeline in der Konsole erstellen oder bearbeiten, erstellt CodePipeline jetzt eine Amazon CloudWatch Events-Regel, die Änderungen an Ihrem CodeCommit Repository erkennt und dann Ihre Pipeline automatisch startet. Weitere Informationen zur Migration Ihrer bestehenden Pipeline finden Sie unter Quellaktionen und Methoden zur Erkennung von Änderungen. Die Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository) wurde aktualisiert, um zu zeigen, wie die Amazon CloudWatch Events-Regel und -Rolle erstellt werden, wenn Sie ein CodeCommit Repository und einen Branch auswählen. Erstellen einer Pipeline in CodePipeline und Bearbeiten einer Pipeline in CodePipeline wurden ebenfalls aktualisiert. Weitere Informationen finden Sie unter Starten einer Pipeline in CodePipeline.	11. Oktober 2017

Änderung	Beschreibung	Datum geändert
Neue und aktualisierte Themen	CodePipeline bietet jetzt integrierte Unterstützung für Benachrichtigungen über Pipeline-Statusänderungen über Amazon CloudWatch Events und Amazon Simple Notification Service (Amazon SNS). Es wurde ein neues Tutorial mit dem Titel Tutorial: Einrichten einer CloudWatch Events-Regel, um E-Mail-Benachrichtigungen zu Änderungen am Pipelinestatus zu empfangen hinzugefügt. Weitere Informationen finden Sie unter Überwachen 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 Quelländerungen erkennen, sodass die Pipeline gestartet wird, sobald diese Änderungen auftreten, oder sie können so eingerichtet werden, dass geplante Pipeline-Ausführungen ausgeführt werden. Für die Konfigurationsoption der PollForSourceChanges Quellaktion wurden Informationen hinzugefügt. Weitere Informationen finden Sie unter Starten einer Pipeline in CodePipeline.	5. September 2017
Neue -Regionen	CodePipeline ist jetzt in Asien-Pazifik (Seoul) und Asien-Pazifik (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 USA West (Nordkalifornien), Kanada (Zentral) und Europa (London) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	29. Juni 2017

Änderung	Beschreibung	Datum geändert
Aktualisierte Themen	Sie können jetzt Details zu allen letzten Ausführungen einer Pipeline statt nur zur letzten Ausführung anzeigen. Diese Details umfassen die Start- und Endzeiten, die Dauer und die Ausführungs-ID. Die Details stehen für maximal 100 Pipeline-Ausführungen während der letzten 12 Monate zur Verfügung. Die Themen Pipelines und Details anzeigen in CodePipeline, CodePipeline -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 Schritt 4: Beta in Schritt 4: Bereitstellen umbenannt. Der Standardname der in diesem Schritt erstellten Phase wurde von „Beta“ in „Staging“ geändert. Zahlreiche Themen und Screenshots wurden aktualisiert, um diese Änderungen widerzuspiegeln.	7. April 2017
Aktualisierte Themen	Sie können jetzt AWS CodeBuild als Testaktion zu jeder Phase einer Pipeline hinzufügen. Auf diese Weise können Sie einfacher verwenden, AWS CodeBuild um Einheiten tests für Ihren Code durchzuführen. Vor dieser Version könnten Sie verwenden, AWS CodeBuild um Einheiten tests nur als Teil einer Build-Aktion auszuführen. Eine Build-Aktion erfordert ein Build-Ausgabeartefakt. Einheiten tests produzieren diese in der Regel nicht. Die Themen Produkt- und Serviceintegrationen mit CodePipeline, und CodePipeline Referenz zur Pipeline-Struktur wurden aktualisiertBearbeiten einer Pipeline in CodePipeline, um diese Unterstützung für widerzuspiegeln AWS CodeBuild.	8. März 2017

Änderung	Beschreibung	Datum geändert
Neue und aktualisierte Themen	Das Inhaltsverzeichnis wurde neu organisiert, um Abschnitte für Pipelines, Aktionen und Phasenübergänge einzuschließen. Für CodePipeline Tutorials wurde ein neuer Abschnitt hinzugefügt. Produkt- und Serviceintegrationen mit CodePipeline wurde in kürzere Themen unterteilt, um die Benutzerfreundlichkeit zu verbessern. Ein neuer Abschnitt, Autorisierung und Zugriffskontrolle, enthält umfassende Informationen zur Verwendung von AWS Identity and Access Management (IAM) und CodePipeline zur Sicherstellung des Zugriffs auf Ihre -Ressourcen durch die Verwendung von -Anmeldeinformationen. Diese Anmeldeinformationen bieten die erforderlichen Berechtigungen für den Zugriff auf - AWS Ressourcen, wie das Einfügen und Abrufen von Artefakten aus Amazon S3-Buckets und das Integrieren von AWS OpsWorks Stacks in Ihre Pipelines.	8. Februar 2017
Neue -Region	CodePipeline ist jetzt in Asien-Pazifik (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) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	7. Dezember 2016

Änderung	Beschreibung	Datum geändert
Aktualisierte Themen	Sie können 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, Einheitentests ausführt und Artefakte erzeugt, die bereitgestellt werden können. 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, Erstellen einer Pipeline in CodePipeline, Authentication and Access Control und CodePipeline Referenz zur Pipeline-Struktur wurden aktualisiert, um diese Unterstützung für widerzuspiegeln AWS CodeBuild. Sie können jetzt CodePipeline mit AWS CloudFormation und dem AWS Serverless Application Model verwenden, um Ihre Serverless-Anwendungen kontinuierlich bereitzustellen. Das Thema Produkt- und Serviceintegrationen mit CodePipeline wurde aktualisiert, um diese Unterstützung widerzuspiegeln. Produkt- und Serviceintegrationen mit CodePipeline wurde umorganisiert, um Angebote nach Aktionstyp zu gruppieren AWS und zu partnerisieren.	1. Dezember 2016
Neue -Region	CodePipeline ist jetzt in Europa (Frankfurt) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	16. November 2016

Änderung	Beschreibung	Datum geändert
Aktualisierte Themen	AWS CloudFormation kann jetzt als Bereitstellungsanbieter in Pipelines ausgewählt werden, sodass Sie im Rahmen einer Pipeline-Ausführung Maßnahmen für AWS CloudFormation Stacks und Änderungssätze ergreifen können. Die Themen Produkt- und Serviceintegrationen mit CodePipeline, Erstellen einer Pipeline in CodePipeline, Authentication and Access Control und CodePipeline Referenz zur Pipeline-Struktur wurden aktualisiert, um diese Unterstützung für widerzuspiegeln 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 in Asien-Pazifik (Singapur) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	20. Oktober 2016
Neue -Region	CodePipeline ist jetzt in der Region USA Ost (Ohio) verfügbar. Das Thema Kontingente in AWS CodePipeline und das Thema Regionen und Endpunkte wurden aktualisiert.	17. Oktober 2016
Aktualisiertes Thema	Erstellen einer Pipeline in CodePipeline wurde aktualisiert, um die Unterstützung für die Anzeige von Versions-IDs für benutzerdefinierte Aktionen in den Listen Quellanbieter provider und Build-Anbieter widerzuspiegeln.	22. September 2016

Änderung	Beschreibung	Datum geändert
Aktualisiertes Thema	Der Abschnitt Genehmigungsaktionen verwalten in CodePipeline wurde aktualisiert, um eine Optimierung widerzuspiegeln, mit deren Hilfe Personen, die Genehmigungsaktionen prüfen, das Formular Approve or reject the revision direkt aus einer E-Mail-Benachrichtigung heraus öffnen können.	14. September 2016
Neue und aktualisierte Themen	Ein neues Thema, das beschreibt, wie Sie Details zu Codeänderungen anzeigen, die derzeit durch Ihre Software-Release-Pipeline erfolgen. Der schnelle Zugriff auf diese Informationen kann bei der Prüfung manueller Genehmigungsaktionen oder der Behebung von Fehlern in Ihrer Pipeline nützlich sein. Ein neuer Abschnitt mit dem Titel Überwachung von Pipelines stellt eine zentrale Stelle für alle Themen im Zusammenhang mit der Überwachung von Status und Fortschritt Ihrer Pipelines dar.	08. September 2016
Neue und aktualisierte Themen	Ein neuer Abschnitt mit dem Titel Genehmigungsaktionen verwalten in CodePipeline stellt Informationen zur Konfiguration und Verwendung manueller Genehmigungsaktionen in Pipelines bereit. Die Themen in diesem Abschnitt enthalten konzeptionelle Informationen zum Genehmigungsprozess, Anweisungen zum Einrichten der erforderlichen IAM-Berechtigungen, zum Erstellen von Genehmigungsaktionen und zum Genehmigen oder Ablehnen von Genehmigungsaktionen sowie Beispiele für JSON-Daten, die generiert werden, wenn eine Genehmigungsaktion in einer Pipeline erreicht wird.	06. Juli 2016
Neue -Region	CodePipeline ist jetzt in der Region Europa (Irland) verfügbar. Die Themen Kontingente in AWS CodePipeline und Regionen und Endpunkte wurden aktualisiert.	23. Juni 2016

Änderung	Beschreibung	Datum geändert
Neues Thema	Ein neues Thema mit dem Titel Eine fehlgeschlagene Aktion in einer Phase wiederholen wurde hinzugefügt, um zu beschreiben, wie eine fehlgeschlagene Aktion oder eine Gruppe fehlgeschlagener paralleler Aktionen in einer Phase wiederholt werden können.	22. Juni 2016
Aktualisierte Themen	Eine Reihe von Themen, darunter Erstellen einer Pipeline in CodePipeline, Authentication and Access Control, und Produkt- und Serviceintegrationen mit CodePipeline, wurden aktualisiertCodePipeline Referenz zur Pipeline-Struktur, um die Unterstützung für die Konfiguration einer Pipeline für die Bereitstellung von Code in Verbindung mit benutzerdefinierten Chef-Cookbooks und Anwendungen widerzuspiegeln, die in erstellt wurden AWS OpsWorks. - CodePipeline Support für AWS OpsWorks ist derzeit nur in der Region USA Ost (Nord-Virginia) (us-east-1) verfügbar.	2. Juni 2016
Neue und aktualisierte Themen	Es wurde ein neues Thema mit dem Titel Tutorial: Erstellen einer einfachen Pipeline (CodeCommit-Repository) hinzugefügt. Dieses Thema enthält eine Beispielanleitung, die zeigt, wie ein CodeCommit Repository und ein Branch als Quellspeicherort für eine Quellaktion in einer Pipeline verwendet werden. Mehrere andere Themen wurden aktualisiert, um diese Integration mit widerzuspiegeln CodeCommit, darunter Authentifizierung und Zugriffsk ontrolle, Produkt- und Serviceintegrationen mit CodePipel ineTutorial: Erstellen einer vierstufigen Pipeline, und Fehlerbehebung CodePipeline.	18. April 2016
Neues Thema	Es wurde ein neues Thema mit dem Titel Aufrufen einer 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

Änderung	Beschreibung	Datum geändert
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 AWS-Services wurden in 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
Neues Thema	Es wurden zwei neue Themen mit den Titeln Konfiguri eren der serverseitigen Verschlüsselung für Artefakte, die in Amazon S3 für gespeichert sind CodePipeline und Erstellen einer Pipeline in CodePipeline, die Ressourcen aus einem anderen AWS-Konto verwendet hinzugefügt. Zur Authentifizierung und Zugriffskontrolle wurde ein neuer Abschnitt Beispiel 8: Verwenden von AWS -Ressourcen, die mit einem anderen Konto in einer Pipeline verknüpft sind hinzugefügt.	25. August 2015

Änderung	Beschreibung	Datum geändert
Aktualisiertes Thema	Das Thema Erstellen und Hinzufügen einer benutzerdefinierten Aktion in CodePipeline wurde aktualisiert, um Änderungen der Struktur widerzuspiegeln, darunter `inputArtifactDetails` und `outputArtifactDetails` .	17. August 2015
Aktualisiertes Thema	Das Fehlerbehebung 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 Fehlerbehebung CodePipeline wurde hinzugefügt. Aktualisierte Schritte wurden für IAM-Rollen und Jenkins in hinzugefügtTutorial: 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
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

Änderung	Beschreibung	Datum geändert
Erste veröffentlichte Version	Dies ist die erste öffentliche Version des CodePipeline - Benutzerhandbuchs.	9. Juli 2015

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im AWS-Glossar in der AWS-Glossar-Referenz.

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